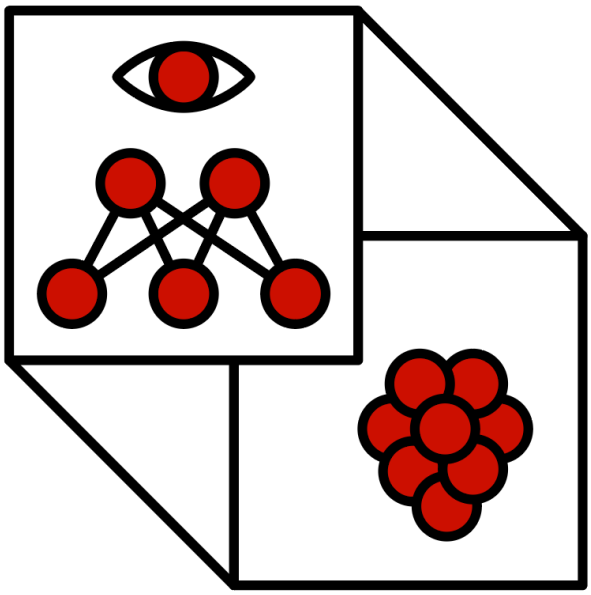
ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ
ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ & ΗΛΕΚΤΡΟΝΙΚΩΝ ΜΗΧΑΝΙΚΩΝ

Διπλωματική Εργασία

Μελέτη και ανάπτυξη λογισμικού ανίχνευσης αντικειμένων με χρήση του Raspberry Pi 4



Φοιτητής: Μαρτάκης Χαράλαμπος
ΑΜ: 50106534

Επιβλέπων:

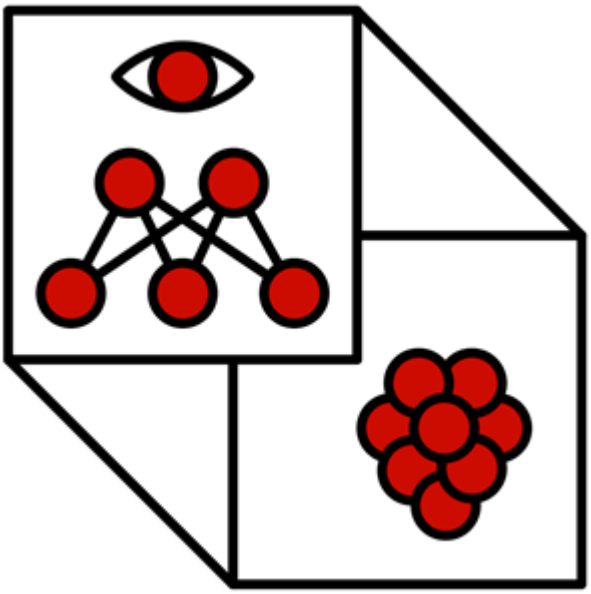
Τσακιρίδης Οδυσσέας, Καθηγητής

ΑΘΗΝΑ-ΑΙΓΑΛΕΩ, ΙΟΥΛΙΟΣ 2023

UNIVERSITY OF WEST ATTICA
FACULTY OF ENGINEERING
DEPARTMENT OF ELECTRICAL & ELECTRONICS ENGINEERING

Diploma Thesis

Study and development of object detection software using Raspberry Pi 4



Student: Martakis Charalampos
Registration Number: 50106534

Supervisor

Tsakiridis Odysseas, Professor

ATHENS-EGALEO, JULY 2023

Η Διπλωματική Εργασία έγινε αποδεκτή και βαθμολογήθηκε από την εξής τριμελή επιτροπή:

Τσακιρίδης Οδυσσεύς Επίκουρος Καθηγητής	Φωτόπουλος Παναγιώτης Αναπληρωτής Καθηγητής	Γαλατά Σωτηρία Επίκουρη Καθηγήτρια
Odyssefs Tsakiridis Digitally signed by Odyssefs Tsakiridis Date: 2023.07.20 07:46:47 +03'00'	Panagiotis Fotopoulos Digitally signed by Panagiotis Fotopoulos Date: 2023.07.20 15:44:50 +03'00'	Sotiria Galata Digitally signed by Sotiria Galata Date: 2023.07.20 11:23:19 +03'00'

ΔΗΛΩΣΗ ΣΥΓΓΡΑΦΕΑ ΔΙΠΛΩΜΑΤΙΚΗΣ ΕΡΓΑΣΙΑΣ

Ο κάτωθι υπογεγραμμένος Μαρτάκης Χαράλαμπος του Γεωργίου, με αριθμό μητρώου 50106534 φοιτητής του Πανεπιστημίου Δυτικής Αττικής της Σχολής ΜΗΧΑΝΙΚΩΝ του Τμήματος ΗΛΕΚΤΡΟΛΟΓΩΝ ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΜΗΧΑΝΙΚΩΝ,

δηλώνω υπεύθυνα ότι:

«Είμαι συγγραφέας αυτής της διπλωματικής εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, οι όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών ή λέξεων, είτε ακριβώς είτε παραφρασμένες, αναφέρονται στο σύνολό τους, με πλήρη αναφορά στους συγγραφείς, τον εκδοτικό οίκο ή το περιοδικό, συμπεριλαμβανομένων και των πηγών που ενδεχομένως χρησιμοποιήθηκαν από το διαδίκτυο. Επίσης, βεβαιώνω ότι αυτή η εργασία έχει συγγραφεί από μένα αποκλειστικά και αποτελεί προϊόν πνευματικής ιδιοκτησίας τόσο δικής μου, όσο και του Ιδρύματος.
Παράβαση της ανωτέρω ακαδημαϊκής μου ευθύνης αποτελεί ουσιώδη λόγο για την ανάκληση του διπλώματός μου.»

Ο Δηλών
Μαρτάκης Χαράλαμπος

Ευχαριστίες

Με τη διπλωματική εργασία αυτή ένα ωραίο ταξίδι γεμάτο εμπειρίες και γνώσεις φτάνει στο τέλος του, καθώς ολοκληρώνω τις προπτυχιακές σπουδές μου.

Θα ήθελα να ευχαριστήσω πρώτα απ' όλα τον υπεύθυνο καθηγητή μου κ. Τσακιρίδη Οδυσσέα ο οποίος μου έδωσε την ευκαιρία να γνωρίσω ένα πεδίο τρομερά ενδιαφέρον και με μέλλον. Τον ευχαριστώ για τις συμβουλές του, την καθοδήγησή του, την συνεχή διαθεσιμότητά του για συζήτηση περί της Διπλωματικής εργασίας και την αισιοδοξία του που με ενέπνευσε.

Επίσης θα ήθελα να ευχαριστήσω τα μέλη της οικογενείας μου που ήταν πραγματικά δίπλα μου και με στήριξαν ο καθένας με τον δικό του τρόπο όλη αυτήν την περίοδο.

Τέλος ευχαριστώ όλα τα υπόλοιπα κοντινά μου άτομα που γνώριζαν όλον αυτόν τον καιρό τις δυσκολίες που αντιμετώπιζα και τους στόχους που έθετα αλλά ήταν πάντα εκεί εάν τους χρειαζόμουν.

Περίληψη

Η παρούσα διπλωματική εργασία μελετάει το πεδίο της τεχνητής νοημοσύνης και πιο συγκεκριμένα τα υποπεδία του, δηλαδή την μηχανική και βαθιά μάθηση. Τα δύο πεδία αυτά εφαρμόζονται σήμερα σε πολλούς τομείς και έχουν βοηθήσει με πολλούς τρόπους είτε την καθημερινότητα των ανθρώπων είτε όσον αφορά την μελέτη και την έρευνα συγκεκριμένων θεμάτων. Επειδή τα πεδία αυτά εκτός από το παρόν θα αποτελέσουν και το μέλλον, επικεντρωθήκαμε στο να δημιουργήσουμε μια εργασία που να επεξηγεί τόσο το θεωρητικό κομμάτι ενός μοντέλου μηχανικής μάθησης, όσο και την διαδικασία της κατασκευής του. Πιο συγκεκριμένα επιλέχθηκε η δημιουργία ενός μοντέλου που έχει σχέση με την επικαιρότητα και θα ήταν χρήσιμο για την κοινωνία. Το μοντέλο αυτό έχει έμμεση σχέση με την υγεία, που είναι από τους σημαντικότερους τομείς τους οποίους μπορεί να βοηθήσει η μηχανική μάθηση και συγκεκριμένα η επιλογή του επηρεάστηκε από τα πρόσφατα γεγονότα της διασποράς του ιού covid-19. Ειδικότερα γίνεται ανάλυση και συγγραφή κώδικα που αφορά την εκπαίδευση ενός βέλτιστου μοντέλου ανίχνευσης χειρουργικής μάσκας ή μη στα πρόσωπα ανθρώπων. Το μοντέλο εκπαιδεύεται σε ένα σύνολο δεδομένων από πρόσωπα που φορούν ή δεν φορούν μάσκα και έπειτα χρησιμοποιείται σε μια εφαρμογή, για την πρακτική εμφάνιση άμεσων αποτελεσμάτων σε βίντεο ζωντανής μετάδοσης. Έτσι η διπλωματική εργασία συνεχίζει αναλύοντας τον δεύτερο κώδικα που κατασκευάστηκε για την χρήση του μοντέλου ανίχνευσης χειρουργικής μάσκας και επεξηγείται η εκτέλεση του μοντέλου όχι σε έναν σταθερό ή φορητό υπολογιστή αλλά σε μια συσκευή αιχμής και συγκεκριμένα το Raspberry Pi 4. Οι συσκευές αιχμής είναι μικρές, βολικές, μπορούν να τοποθετηθούν σε μέρη που ένας υπολογιστής δεν είναι εφικτό να τοποθετηθεί και δεν καταναλώνουν πολλούς πόρους, άρα και ενέργεια.

Λέξεις – κλειδιά

Τεχνητή Νοημοσύνη, Μηχανική Μάθηση, Βαθιά Μάθηση, Ανίχνευση Αντικειμένων, Όραση Υπολογιστή, Συνελικτικά, Νευρωνικά Δίκτυα, Raspberry Pi 4, Python, Tensorflow, OpenCV

Abstract

This thesis studies the field of artificial intelligence and more specifically its subfields, machine learning and deep learning. These two fields are used today in many areas and have helped in many ways either in the daily life of people or in terms of the study and research of specific subjects. Regarding the fact that these fields are not only important for the present but will also be important for the future, we focused on creating a work that would explain both the theoretical part of a machine learning model, as well as the process of its construction. More specifically, we chose to create a model that is relevant to the current global situation and would be useful for society. This model has an indirect relationship with health, which is one of the most important areas that machine learning can help, and in particular the choice of the model was influenced by the recent events of the spread of the covid-19 virus. In particular, an analysis and writing of code is done that concerns the training of an optimal model for mask or non-mask detection on human faces. The model is trained on a dataset of masked and non-masked faces and then used in an application to practically display immediate results on a live streaming video. Thus, the thesis continues by analyzing the second code that was built to use the mask detection model and by explaining the use of the model not on a PC or Laptop but on an edge device and more specifically the Raspberry Pi 4. Edge devices are small, convenient, they can be placed in spots that a PC cannot and they do not consume a lot of resources and thus energy.

Keywords

Artificial Intelligence, Machine Learning, Deep Learning, Object Detection, Computer Vision, Convolutional Neural Networks, Raspberry Pi 4, Python, Tensorflow, OpenCV

Περιεχόμενα

Κατάλογος Πινάκων

Κατάλογος Εικόνων

Αλφαβητικό Ευρετήριο

CNN: Convolutional Neural Networks (συνελικτικά νευρωνικά δίκτυα)

ReLU: Rectified Linear Units (Διορθωμένες Γραμμικές Μονάδες)

Tahn: Υπερβολική συνάρτηση εφαπτομένης

ReLU: Διορθωμένη γραμμική μονάδα

Leaky ReLU: Διορθωμένη γραμμική μονάδα Leaky

Adam: Προσαρμοστική εκτίμηση ροπών

rpi4: Raspberry Pi 4

INIT_LR: Ρυθμός εκπαίδευσης

EPOCHS: Εποχές

BS: Μέγεθος υποσυνόλου δεδομένων

Λεξικό Ελληνικών - Αγγλικών

Ελληνικοί όροι	**Αγγλικοί όροι**
Ακέραιες	Integers
Ακρίβεια	Precision
Αλγόριθμοι παλινδρόμησης	Regression algorithms
Αλγόριθμοι ταξινόμησης	Classification algorithms
Αλφαριθμητικό	String
Αναλογία	Aspect ratio
Αντικείμενο	Object
Αντίληπτρο	Perceptron
Αποθετήριο	Repository
Απώλεια	Loss
Αύξηση δεδομένων (μέθοδος τακτοποίησης)	Data augmentation (regularization method)
Βαθιά μάθηση	Deep learning
Βαθμολογία F1	F1-score
Βάρος	Weight
Βελτιστοποίηση	Optimization
Βιβλιοθήκες	Libraries
Γενικά οπτικά μοτίβα	General visual patterns
Γραμμή εντολών	Command line
Δεδομένα δοκιμής/επικύρωσης	Testing/Validation data/set/subset
Δεδομένα εκπαίδευσης	Training data/set/subset
Δεύτερη στιγμή	Second moment
Διακλάδωση	Branch
Διανομή	Distribution
Διεπαφή χρήστη	User interface
Διορθωμένη γραμμική μονάδα (συνάρτηση ενεργοποίησης)	Rectified linear unit (activation function)
Διορθωμένη γραμμική μονάδα Leaky (συνάρτηση ενεργοποίησης)	Rectified linear unit Leaky (activation function)
Δυαδική διασταυρούμενη εντροπία	Binary cross entropy
Δυαδοποίηση	Binarizing
Εμπρόσθια διάδοση	Forward propagation
Ενισχυτική Μάθηση	Reinforcement learning
Ενότητες	Modules
Ενσωματωμένα συστήματα	Embedded devices
Επιβλεπόμενη Μάθηση	Supervised learning
Επίπεδο	Layer
Επίπεδο διορθωμένων γραμμικών μονάδων	Rectified Linear Units layer
Επίπεδο εγκατάλειψης	Dropout layer
Επίπεδο εισόδου	Input layer
Επίπεδο ισοπέδωσης	Flatten layer
Επίπεδο συγκέντρωσης	Pooling layer

Επίπεδο συνέλιξης	Convolution layer
Επιστήμη δεδομένων	Data science
Επιστήμονες δεδομένων	Data scientists
Επιτάχυνση υλικού	Hardware acceleration
Εποχές/Επαναλήψεις	Epochs
Εργασίες	Project
Εσοχή	Identation
Ετικέτα	Label
Ικανότητα γενίκευσης	Generalization abillity
Κάθοδος βασισμένη στην κλίση	Gradient Descent
Καμπύλη λειτουργικού χαρακτηριστικού δέκτη	Receiver operating characteristic curve
Κανάλι	Channel
Κανονικοποίηση υποσυνόλου δεδομένων (μέθοδος τακτοποίησης)	Batch normalization (regularization method)
Κατηγορικές ετικέτες	Categorical labels
Κατώφλι	Threshold
Κλάση	Class
Κορυφή (μέθοδος τακτοποίησης)	Ridge/L2 (regularization method)
Κόστος	Cost
Κρυφά επίπεδα	Hidden layers
Λάθος αρνητικά	False negatives/FN
Λάθος θετικά	False positives/FP
Λάθος θετικός ρυθμός	False positive rate/FPR
Λάθος/Σφάλμα	Error
Λάσο (μέθοδος τακτοποίησης)	Lasso/L1 (regularization method)
Λεξικό	Dictionary
Μακρύς μέσος όρος	Macro average
Μέγεθος υποσυνόλου δεδομένων	Batch size
Μέθοδοι	Methods
Μέθοδοι τακτοποίησης	Regularization methods
Μέση τετραγωνική διάδοση ρίζας (αλγόριθμος)	Root mean square propagation algorithm / RMSProp
Μέσο τετραγωνικό σφάλμα	Root Mean Squared Error
Μεταγλωτίζονται	Compile
Μεταφορά μάθησης (μέθοδος τακτοποίησης)	Transfer learning (regularization method)
Μετρικές αξιολόγησης	Evaluation Metrics
Μη επιβλεπόμενη μάθηση	Unsupervised learning
Μηχανική μάθηση	Machine learning
Μοντέλα μηχανικής μάθησης	Machine learning models
Νευρωνικό δίκτυο	Neural network
Ολικό ελάχιστο	Global minima
Ολοκληρωμένο εργαλείο ανάπτυξης	Integrated development tool
Ολοκληρωμένο περιβάλλον ανάπτυξης	Integrated development environment/IDE
Ομαδοποίηση	Clustering
Οπισθοδιάδοση /Οπισθοδρόμηση	Backpropagation

Όραση υπολογιστή	Computer vision
Ορθότητα	Accuracy
Περιοχή κάτω από την καμπύλη λειτουργικού χαρακτηριστικού δέκτη	Area under the ROC curve /AUC score
Πίνακας σύγχυσης	Confusion matrix
Πλαίσιο	Framework
Πλειάδα	Tuple
Πλήρως συνδεδεμένο επίπεδο	Fully Connected layer
Πόλωση	Bias
Προσαρμοστική εκτίμηση ροπών	Adam (adaptive moment estimation)
Προσαρμοστική κλίση (αλγόριθμος)	Adaptive gradient algorithm / AdaGrad
Πρόωρο σταμάτημα (μέθοδος τακτοποίησης)	Early stopping (regularization method)
Πρώτη στιγμή	First moment
Ρυθμός εκπαίδευσης	Learning rate
Σιγμοειδής συνάρτηση ενεργοποίησης	Sigmoid activation function
Σταθμισμένος μέσος όρος	Weighted average
Στιγμιότυπα	Frames
Συγκεντρωτικός χάρτης χαρακτηριστικών	Pooled feature map
Συναρτήσεις	Functions
Συνάρτηση απωλειών	Loss function
Συνάρτηση ενεργοποίησης	Activation function
Συνάρτηση κόστους	Cost function
Σύναψη	Synapse
Συνελικτικό νευρωνικό δίκτυο	Convolutional neural network
Σύννεφο	Cloud
Σύνολο δεδομένων εκπαίδευσης	Training dataset
Σύνολο δεδομένων	Dataset
Συσκευές αιχμής	Edge devices
Συσχέτιση	Association
Σωστά αρνητικά	True negatives/TN
Σωστά θετικά	True positives/TP
Σωστός θετικός ρυθμός	True positive rate/TPR
Τεχνητή νοημοσύνη	Artificial Intelligence
Τοπικό ελάχιστο	Local minima
Τοπικό μέγιστο	Local maxima
Υλικολογισμικό	Firmware
Υπερβολική συνάρτηση εφαπτομένης (συνάρτηση ενεργοποίησης)	Tanh (activation function)
Υπερπαράμετροι	Hyperparameters
Υπερπροσαρμογή	Overfitting
Υποπροσαρμογή	Underfitting
Υποσύνολα	Sets
Χαρακτηριστικό	Attribute
Χάρτης χαρακτηριστικών	Feature map
Χωρικές διαστάσεις	Spatial dimensions

ΕΙΣΑΓΩΓΗ

Στην παρούσα διπλωματική εργασία αναλύεται εκτενώς η εκπαίδευση και η εφαρμογή ενός μοντέλου μηχανικής μάθησης και συγκεκριμένα της ανίχνευσης χειρουργικής μάσκας στα πρόσωπα ανθρώπων. Πιο αναλυτικά, η εργασία αρχίζει με την ανάλυση του αλγορίθμου που χρησιμοποιήθηκε με ονομασία «επιβλεπόμενη μάθηση» που μαζί με την τεχνική της «μεταφοράς μάθησης» αποτέλεσαν τα «θεμέλια» του πρακτικού μέρους της εργασίας. Στη συνέχεια επεξηγείται η αρχιτεκτονική ενός μοντέλου, που στην ουσία είναι ένα συνελικτικό νευρωνικό δίκτυο, και αναλύονται τα επιμέρους κομμάτια του δικτύου που ονομάζονται επίπεδα. Μετέπειτα, παρουσιάζονται οι διαδικασίες εκπαίδευσης ενός συνελικτικού νευρωνικού δικτύου όπως η οπισθοδιάδοση, ο υπολογισμός των απωλειών του καθώς και η βελτιστοποίησή του. Με βάση την θεωρία εκπαίδευσης ενός συνελικτικού νευρωνικού δικτύου, γίνεται εξειδικευμένη εφαρμογή των τεχνικών πάνω σε ένα συγκεκριμένο μοντέλο, αυτό της ανίχνευσης χειρουργικής μάσκας πάνω σε ανθρώπινα πρόσωπα. μέσω της εισαγωγής εικόνων ανθρώπινων προσώπων με ή χωρίς χειρουργική μάσκα. Τέλος με σκοπό την επιτυχή υλοποίηση και τη λειτουργικότητα του συγκεκριμένου μοντέλου, δημιουργήθηκε μια εφαρμογή στην οποία εισάγονται στιγμιότυπα από ζωντανή μετάδοση βίντεο και το μοντέλο τα ταξινομεί εμφανίζοντας σε ένα γραφικό παράθυρο τα αποτελέσματα σε μορφή βίντεο περί της χρήσης ή μη μάσκας από τους εκάστοτε ανθρώπους.

Αντικείμενο της διπλωματικής εργασίας

Με αφορμή την ραγδαία εξέλιξη της τεχνητής νοημοσύνης στον χώρο της τεχνολογίας στη σημερινή εποχή, αντικείμενο αυτής της διπλωματικής εργασίας αποτελεί η μελέτη και η ανάπτυξη λογισμικού ανίχνευσης αντικειμένων και συγκεκριμένα η ανίχνευση χειρουργικής μάσκας στα ανθρώπινα πρόσωπα από ένα συνελικτικό νευρωνικό δίκτυο. Το αντικείμενο αυτό αποτελεί ένα πολύ ενδιαφέρον και επίκαιρο θέμα, καθώς λόγω της πρόσφατης πανδημίας που έπληξε τον παγκόσμιο πληθυσμό τα τελευταία τρία χρόνια, μία τέτοια εφαρμογή θα ήταν χρήσιμη τόσο για την πρόληψη του ιού όσο και για την διευκόλυνση και εξυπηρέτηση της κοινωνίας και της ιατρικής γενικότερα.

Σκοπός και στόχοι

Σκοπός της εργασίας αυτής είναι η δημιουργία μιας ολοκληρωμένης εφαρμογής για την αναγνώριση χειρουργικής μάσκας σε ανθρώπινα πρόσωπα η οποία θα μπορεί να εκτελεστεί από την φορητή συσκευή, Rapsberry Pi 4. Για την ανάπτυξη του κώδικα αυτής της εφαρμογής, πρώτος στόχος είναι η εκπαίδευση ενός μοντέλου ταξινόμησης εικόνων σχετικά με το αν φοράει κάποιος μάσκα ή όχι. Ο δεύτερος στόχος είναι η δοκιμή αυτού του μοντέλου στην βασική εφαρμογή και η προσαρμογή της στο λογισμικό του Raspberry Pi 4.

Μεθοδολογία

Το πρώτο κομμάτι της εργασίας αφορά τη δημιουργία μοντέλου μηχανικής μάθησης με τη μέθοδο μεταφοράς μάθησης που ανιχνεύει την ύπαρξη ή μη μάσκας covid-19 πάνω στα πρόσωπα των ανθρώπων με τη χρήση της γλώσσας προγραμματισμού Python. Πραγματοποιείται προεπεξεργασία των εικόνων που θα εκπαιδευτεί το μοντέλο, κατασκευάζεται το πλήρως συνδεδεμένο νευρωνικό δίκτυο για την ταξινόμηση των αποτελεσμάτων του μοντέλου και γίνεται σύνδεση με το συνελικτικό νευρωνικό δίκτυο της αρχιτεκτονικής “MobileNetV2”. Στη συνέχεια το μοντέλο

αποθηκεύεται τοπικά στον υπολογιστή σε μορφή ".h5" με την βιβλιοθήκη "tensorflow" και μετατρέπεται σε μορφή ".tflite" για να μπορεί να χρησιμοποιηθεί σε μία συσκευή αιχμής και συγκεκριμένα το Raspberry Pi 4. Έπειτα το μοντέλο αξιολογείται με εικόνες που δεν έχει ξαναδεί και γίνεται εμφάνιση αποτελεσμάτων σχετικά με την απόδοσή του, τις απώλειες και την ακρίβεια στις προβλέψεις που κάνει. Ακόμα, δημιουργούνται γραφικές παραστάσεις για την εμφάνιση των μετρήσεων που υπολογίστηκαν μετά το πέρας της εκπαίδευσης του μοντέλου. Για την επιλογή του βέλτιστου μοντέλου, πραγματοποιείται εκπαίδευση διαφορετικών μοντέλων με διαφορετικές υπερπαραμέτρους και στο τέλος γίνεται η σύγκρισή τους σχετικά με την απόδοση, την ακρίβεια και τις απώλειες.
Το δεύτερο κομμάτι, αφορά τη δημιουργία του κώδικα ανίχνευσης μάσκας στα πρόσωπα των ανθρώπων και της εκτέλεσής του στο Raspberry Pi 4. Ο κώδικας αυτός χρησιμοποιεί το μοντέλο της μορφής ".tflite". Αρχικά γίνεται εύρεση των ανθρώπινων προσώπων, μέσω της κάμερας του υπολογιστή, με τη χρήση ενός προεκπαιδευμένου μοντέλου που δημιουργήθηκε με τη βοήθεια του αλγορίθμου Single Shot MultiBox Detector (SSD). Έτσι ελέγχεται η ύπαρξη ή μη μάσκας πάνω στα πρόσωπα και δημιουργούνται τα περιγράμματα που θα τοποθετηθούν γύρω από τα εντοπισμένα πρόσωπα. Τέλος εμφανίζονται τα αποτελέσματα σε ζωντανή μετάδοση η οποία προβάλλεται μέσω παραθύρου στον υπολογιστή.

Καινοτομία

Το αντικείμενο της παρούσας διπλωματικής εργασίας μπορεί να θεωρηθεί ιδιαίτερα καινοτόμο, όχι μόνο για τον τομέα της τεχνολογίας αλλά και για την επιστήμη της Ιατρικής. Συγκεκριμένα η ανίχνευση μάσκας με τη χρήση του Raspberry Pi 4 αποτελεί ένα χρήσιμο εργαλείο για την υποστήριξη της Ιατρικής καθώς μπορεί να τοποθετηθεί σε χώρους όπως νοσοκομεία, φαρμακεία, ιατρεία αλλά και γενικότερα σε κοινόχρηστους χώρους όπως τα σούπερ μάρκετ ή τα εμπορικά κέντρα με σκοπό την πρόληψη της διασποράς του ιού covid-19 αλλά και για κάθε μελλοντικό ιό που απαιτεί αντίστοιχη πρόληψη.
Επιπλέον ένα άλλο καινοτόμο στοιχείο είναι η υλοποίηση αυτής της εφαρμογής στη συσκευή Rapsberry Pi 4, καθώς μέχρι στιγμής ο συχνότερος τρόπος εφαρμογής της ανίχνευσης μάσκας πραγματοποιείται μέσω υπολογιστών.
Τέλος, στην εργασία έχουν δοθεί οδηγίες και πληροφορίες που μπορούν να λειτουργήσουν ως πρότυπο για την δημιουργία παρόμοιων μοντέλων μηχανικής μάθησης

Δομή

Η δομή της διπλωματικής αυτής εργασίας χωρίζεται σε πέντε βασικά κεφάλαια. Το πρώτο κεφάλαιο αφορά το θεωρητικό υπόβαθρο το οποίο περιέχει πληροφορίες σχετικά με την τεχνητή νοημοσύνη, το υποπεδίο της «τεχνική μάθηση», τις βασικές αρχές της βαθιάς μάθησης, την αρχιτεκτονική των νευρωνικών δικτύων, τα συνελικτικά νευρωνικά δίκτυα, την εκπαίδευση του μοντέλου μηχανικής μάθησης και την αξιολόγησή του.
Το δεύτερο κεφάλαιο παρουσιάζει το τεχνικό υπόβαθρο το οποίο περιλαμβάνει όλα τα εργαλεία, τις βιβλιοθήκες, τα προγράμματα και τις συσκευές που χρησιμοποιήθηκαν για το πρακτικό μέρος της εργασίας. Επιπλέον στο παρών κεφάλαιο αναλύεται η προετοιμασία του Raspberry Pi 4 για την εκτέλεση του κώδικα ανίχνευσης μάσκας.
Στο τρίτο κεφάλαιο αναλύεται ο κώδικας της εκπαίδευσης του μοντέλου ανίχνευσης μάσκας και συγκρίνονται τα αποτελέσματα του κώδικα αυτού, με σκοπό την εύρεση του βέλτιστου μοντέλου.

Το τέταρτο κεφάλαιο αφορά την ανάλυση του κώδικα ανίχνευσης μάσκας σε ανθρώπινα πρόσωπα και γίνεται παράθεση των αποτελεσμάτων του.
Το πέμπτο και τελευταίο κεφάλαιο αποτελεί τον επίλογο και περιέχει τα συμπεράσματα μετά το πέρας αυτής της διπλωματικής εργασίας καθώς και κάποιες προτάσεις για μελλοντικές βελτιώσεις και ιδέες.

1 ΚΕΦΑΛΑΙΟ 1^{ο} : Θεωρητικό υπόβαθρο

Σε αυτό το κεφάλαιο παρουσιάζονται και αναλύονται οι γνώσεις που χρειάζεται κάποιος για να κατανοήσει την διπλωματική εργασία αρχικά σε θεωρητικό επίπεδο.

Στην ενότητα 1.1 αρχικά περιγράφεται το πεδίο της τεχνητής νοημοσύνης και έπειτα αναλύεται το υποπεδίο της η μηχανική μάθηση. Συγκεκριμένα γίνεται ανάλυση των διάφορων μεθόδων μηχανικής μάθησης και των αλγορίθμων που χρησιμοποιούνται σε αυτή.

Στην ενότητα 1.2 γίνεται ανάλυση της βαθιάς μάθησης και των αρχών της, η οποία είναι επίσης υποπεδίο της τεχνητής νοημοσύνης. Η ενότητα ξεκινά με την περιγραφή του νευρώνα Perceptron και την εισαγωγή στην βασική αρχιτεκτονική ενός απλού νευρωνικού δικτύου. Γίνεται εμβάθυνση στα συνελικτικά νευρωνικά δίκτυα, τα επίπεδα τους και αναφέρονται διάφορες μέθοδοι που χρησιμοποιούν όπως οι συναρτήσεις ενεργοποίησης. Έπειτα γίνεται επεξήγηση της καθόδου βασισμένη στην κλίση και της οπισθοδιάδοσης μαζί με την εμπρόσθια διάδοση που είναι πολύ σημαντικά κομμάτια των συνελικτικών νευρωνικών δικτύων. Επίσης, γίνεται αναφορά σε διάφορα προβλήματα που μπορεί να υπάρξουν στα νευρωνικά δίκτυα, όπως η υπερπροσαρμογή, η υποπροσαρμογή καθώς παρουσιάζονται και λύσεις σε αυτά που ονομάζονται μέθοδοι τακτοποίησης.

Η ενότητα 1.3 ασχολείται με την εκπαίδευση ενός μοντέλου μηχανικής ή και βαθιάς μάθησης όπου αναφέρονται με τη σειρά κάποιες από τις πιο βασικές ενέργειες ή ρυθμίσεις που εκτελούνται για την σωστή λειτουργία του. Γίνεται εισαγωγή στις πιο σημαντικές υπερπαραμέτρους που χρησιμοποιεί ένα μοντέλο και συγκεκριμένα το δικό μας και αναλύεται η τακτική της αύξησης των δεδομένων για την εκπαίδευση του μοντέλου. Στη συνέχεια, περιγράφεται η δυαδική διασταυρούμενη εντροπία που χρησιμοποιεί το μοντέλο μας για την εύρεση των απωλειών και ο βελτιστοποιητής Adam για την μείωση των απωλειών αυτών. Ακόμα γίνεται παρουσίαση της αρχιτεκτονικής του μοντέλου που δημιουργήθηκε η οποία είναι βασισμένη στην αρχιτεκτονική του προεκπαιδευμένου μοντέλου MobileNetV2.

Η ενότητα 1.4 επεξηγεί την διαδικασία αξιολόγησης του μοντέλου και τις μετρικές που χρησιμοποιούνται για τον υπολογισμό της απόδοσής του.

1.1 Εισαγωγή στην Τεχνητή Νοημοσύνη και τη Μηχανική Μάθηση

1.1.1 Ορισμός και επισκόπηση της τεχνητής νοημοσύνης και της μηχανικής μάθησης

Η τεχνητή νοημοσύνη (Artificial Intelligence) είναι ένας τομέας της επιστήμης των υπολογιστών που εστιάζει στη μεταφορά ανθρώπινης νοημοσύνης και γνωστικών ικανοτήτων σε μηχανές, δίνοντάς τους τη δυνατότητα να βοηθούν τους ανθρώπους με διάφορους τρόπους. Επινοήθηκε από τον John McCarthy το 1956 και κέρδισε σταδιακά δυναμική σε κλάδους όπως η μηχανική, τα μαθηματικά, η φυσική και η τεχνολογία, οδηγώντας στον αξιοσημείωτο μετασχηματισμό που βλέπουμε αυτήν τη στιγμή [1].

Η τεχνητή νοημοσύνη περιλαμβάνει την ιδέα ότι οι μηχανές μπορούν να αποκτήσουν νοημοσύνη, επιτρέποντάς τους να μαθαίνουν αυτόνομα, να προσαρμόζονται σε συγκεκριμένες καταστάσεις και να διορθώνουν τα δικά τους λάθη. Ουσιαστικά, οι μηχανές μπορούν να συμμετέχουν με ανεξάρτητη σκέψη χωρίς να βασίζονται αποκλειστικά σε ρητά προγραμματισμένες οδηγίες.

Η βάση της τεχνητής νοημοσύνης βρίσκεται στην ενοποίηση της επιστήμης των υπολογιστών και των γνωστικών διαδικασιών. Η νοημοσύνη, με απλά λόγια, περιλαμβάνει υπολογιστικές δυνατότητες που είναι απαραίτητες για την επίτευξη στόχων στον πραγματικό κόσμο. Περιλαμβάνει δραστηριότητες όπως η σκέψη, η φαντασία, η δημιουργικότητα, η μνήμη, η κατανόηση, η αναγνώριση προτύπων, η λήψη αποφάσεων, η προσαρμοστικότητα και η βιωματική μάθηση. Ο στόχος της τεχνητής νοημοσύνης είναι να κάνει τους υπολογιστές να συμπεριφέρονται με τρόπο που μοιάζει με την ανθρώπινη συμπεριφορά, εκτελώντας εργασίες σε σημαντικά μικρότερα χρονικά πλαίσια από τους ανθρώπους [2].

Σύμφωνα με τον Arthur Samuel, η μηχανική μάθηση (Machine Learning) είναι το πεδίο μελέτης που παρέχει στους υπολογιστές τη δυνατότητα να μαθαίνουν χωρίς ρητό προγραμματισμό. Ο Samuel έγινε γνωστός για το πρόγραμμα που έπαιζε το παιχνίδι «ντάμα». Αρχικά, αυτός κατάφερε να ξεπεράσει το πρόγραμμα, αλλά μέσω της εξάσκησης, το πρόγραμμα σταδιακά έμαθε να εντοπίζει καλές και κακές θέσεις στο ταμπλό.

Ο Tom Mitchell έδωσε έναν πιο επίσημο ορισμό, δηλώνοντας ότι ένα πρόγραμμα υπολογιστή μαθαίνει από την εμπειρία (Ε) σε σχέση με μια συγκεκριμένη εργασία (ΣΕ) και το μέτρο απόδοσης (ΜΑ). Εάν η απόδοση του προγράμματος στην εργασία ΣΕ, όπως μετράται με το ΜΑ, βελτιώνεται με την εμπειρία Ε, τότε το πρόγραμμα αυτό ταξινομείται ως πρόγραμμα μηχανικής εκμάθησης.

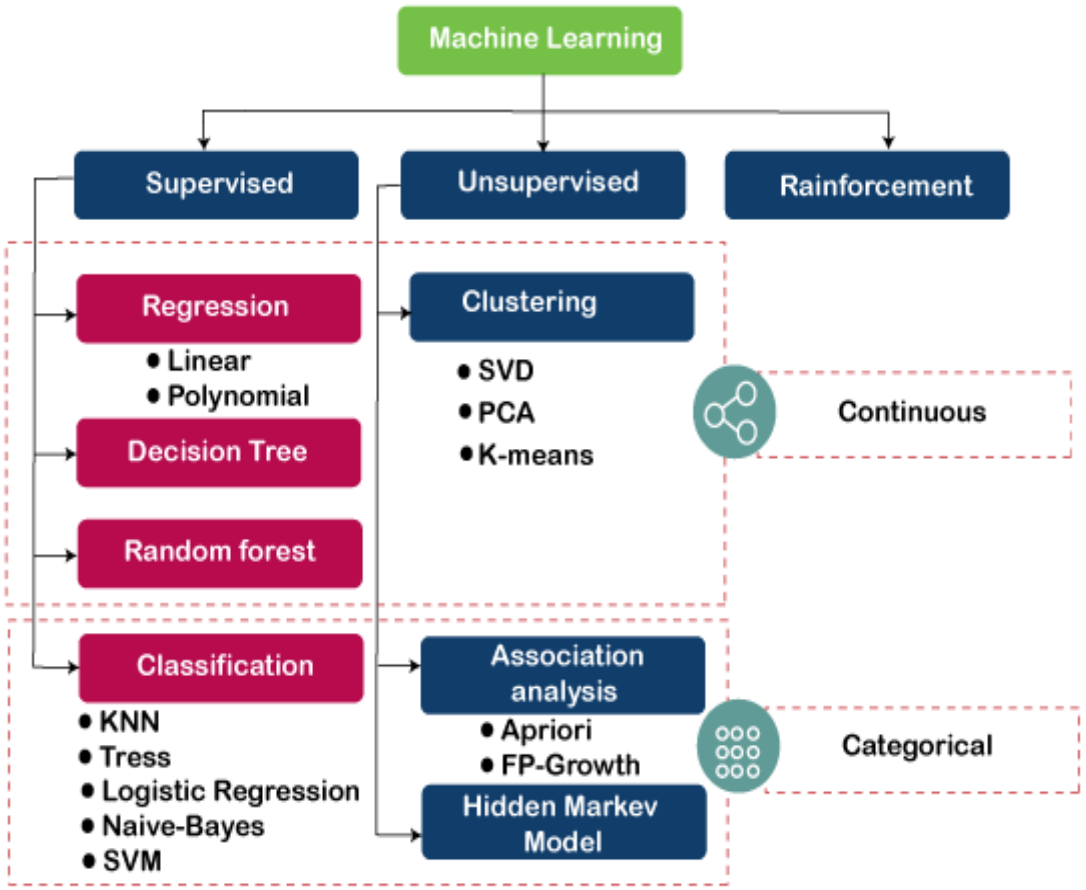
Στο παράδειγμα του προγράμματος που παίζει «ντάμα», η εμπειρία Ε περιλάμβανε το πρόγραμμα να παίζει παιχνίδια εναντίον του εαυτού του. Η εργασία ΣΕ ήταν το παιχνίδι «ντάμα» και το μέτρο απόδοσης ΜΑ ήταν η πιθανότητα νίκης του επόμενου παιχνιδιού εναντίον ενός νέου αντιπάλου [3].

Η μάθηση, με απλά λόγια, περιλαμβάνει την απόκτηση νέων γνώσεων ή την ενίσχυση και ενημέρωση των ατομικών δεξιοτήτων. Η απόκτηση νέας γνώσης περιλαμβάνει διαδικασίες όπως η κατανόηση σημαντικών εννοιών, η κατανόηση των νοημάτων και των σχέσεών τους και η συνάφειά τους με τον τομέα. Η βελτίωση των δεξιοτήτων μπορεί να εξηγηθεί βιολογικά ως η ενίσχυση των νευρωνικών συνδέσεων για την εκτέλεση μιας επιθυμητής λειτουργίας.

Η μηχανική μάθηση, που θεωρείται υποσύνολο της τεχνητής νοημοσύνης, είναι η επιστημονική μελέτη αλγορίθμων και στατιστικών μοντέλων που χρησιμοποιούνται από συστήματα υπολογιστών για την εκτέλεση συγκεκριμένων εργασιών. Οι αλγόριθμοι μηχανικής μάθησης κατασκευάζουν μαθηματικά μοντέλα βασισμένα σε δεδομένα εκπαίδευσης (training data), επιτρέποντάς τα να κάνουν προβλέψεις (predictions) ή να παίρνουν αποφάσεις χωρίς ρητό προγραμματισμό.
Σε διάφορα πεδία μηχανικής, οι αλγόριθμοι μάθησης χρησιμοποιούνται όλο και περισσότερο για την αποτελεσματική κατανόηση μεγαλύτερων συνόλων δεδομένων[2].

1.1.2 Τύποι αλγορίθμων μηχανικής μάθησης

Σε αυτήν την ενότητα περιγράφονται οι τρείς βασικές κατηγορίες αλγορίθμων μηχανικής μάθησης μεταξύ των οποίων είναι και η «supervised» που χρησιμοποιήθηκε για την εκπαίδευση του μοντέλου βαθιάς μάθησης αυτής της διπλωματικής εργασίας.



Εικόνα 1.1 Κατηγορίες αλγορίθμων μηχανικής μάθησης [4]

1.1.2.1 Επιβλεπόμενη μάθηση

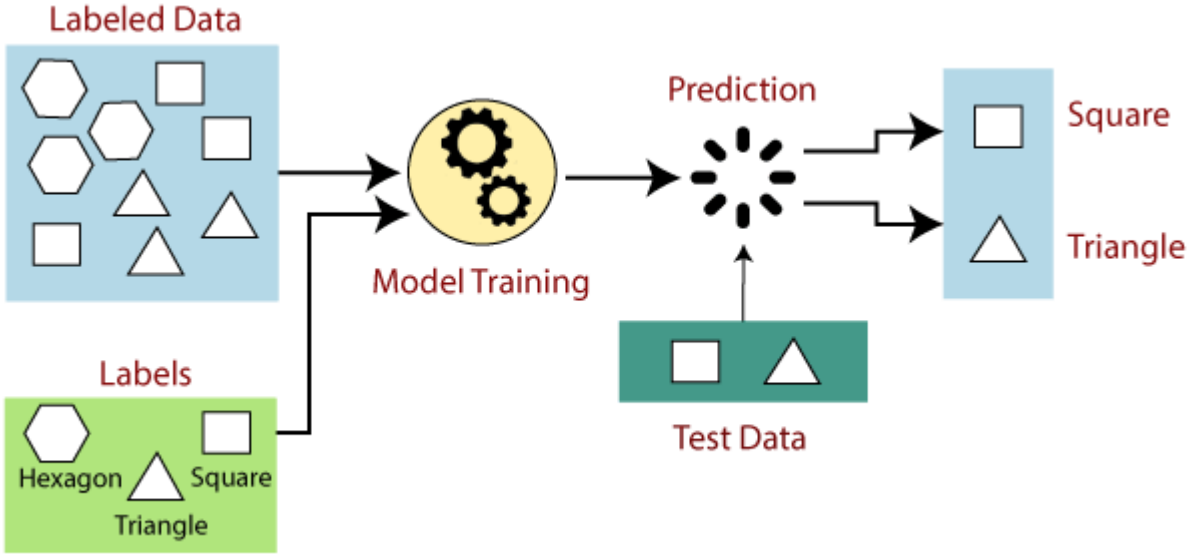
Η επιβλεπόμενη μάθηση (supervised learning) είναι μία κατηγορία μηχανικής εκμάθησης όπου οι μηχανές εκπαιδεύονται χρησιμοποιώντας δεδομένα εκπαίδευσης με ετικέτες (labels), επιτρέποντάς τους να προβλέψουν την έξοδο με βάση αυτά τα δεδομένα. Τα δεδομένα με ετικέτα αποτελούνται από δεδομένα εισόδου που έχουν ήδη συσχετιστεί με τη σωστή έξοδο.

Στην επιβλεπόμενη μάθηση, τα δεδομένα εκπαίδευσης χρησιμεύουν ως ο επόπτης που καθοδηγεί τις μηχανές να κάνουν ακριβείς προβλέψεις. Λειτουργεί με την ίδια λογική που ένας μαθητής μαθαίνει υπό την καθοδήγηση ενός δασκάλου.
Η διαδικασία της επιβλεπόμενης μάθησης περιλαμβάνει την παροχή δεδομένων εισόδου μαζί με τα αντίστοιχα σωστά δεδομένα εξόδου στο μοντέλο μηχανικής εκμάθησης. Ο στόχος ενός αλγορίθμου εποπτευόμενης μάθησης είναι να ανακαλύψει μία συνάρτηση χαρτογράφησης που μπορεί να συσχετίσει αποτελεσματικά τη μεταβλητή εισόδου (x) με τη μεταβλητή εξόδου (y)

Σε πρακτικές εφαρμογές, η επιβλεπόμενη μάθηση μπορεί να χρησιμοποιηθεί για εργασίες όπως η αξιολόγηση κινδύνου, η ταξινόμηση εικόνων, ο εντοπισμός απάτης, το φιλτράρισμα ανεπιθύμητων μηνυμάτων και άλλα. Τα μοντέλα εκπαιδεύονται χρησιμοποιώντας σύνολα δεδομένων με ετικέτα, επιτρέποντας στο μοντέλο να μάθει για διάφορους τύπους δεδομένων. Μόλις ολοκληρωθεί η διαδικασία εκπαίδευσης, το μοντέλο ελέγχεται χρησιμοποιώντας ένα ξεχωριστό υποσύνολο δεδομένων που ονομάζεται σύνολο δοκιμών (test set) και στη συνέχεια κάνει προβλέψεις με βάση αυτή την αξιολόγηση.

Για παράδειγμα, ας εξετάσουμε ένα σύνολο δεδομένων που αποτελείται από διαφορετικά σχήματα, συμπεριλαμβανομένων τετραγώνων, ορθογωνίων, τριγώνων και πολυγώνων. Για να εκπαιδεύσουμε το μοντέλο, ορίζουμε πρώτα κανόνες για κάθε σχήμα. Για παράδειγμα, εάν ένα σχήμα έχει τέσσερις ίσες πλευρές, χαρακτηρίζεται ως τετράγωνο, ενώ ένα σχήμα με τρεις πλευρές χαρακτηρίζεται ως τρίγωνο. Ομοίως, ένα σχήμα με έξι ίσες πλευρές χαρακτηρίζεται ως εξάγωνο. Μετά την εκπαίδευση του μοντέλου, μπορούμε να δοκιμάσουμε την ικανότητά του να αναγνωρίζει σχήματα με βάση τον αριθμό των πλευρών.

Το εκπαιδευμένο μηχάνημα, έχοντας μάθει για διάφορα σχήματα, μπορεί να ταξινομήσει νέα σχήματα εξετάζοντας τον αριθμό των πλευρών και προβλέποντας την αντίστοιχη έξοδο. Στην εικόνα 1.2 φαίνεται οπτικά το παραπάνω παράδειγμα.



Εικόνα 1.2 Παράδειγμα Επιβλεπόμενης Μάθησης [5]

Τα ακόλουθα βήματα συνήθως εμπλέκονται στην επιβλεπόμενη μάθηση:

1. Προσδιορισμός του συνόλου δεδομένων εκπαίδευσης (training dataset).

2. Συλλογή των δεδομένων εκπαίδευσης με ετικέτες.
3. Διαχωρισμός του συνόλου δεδομένων εκπαίδευσης σε υποσύνολα (sets) για την εκπαίδευση, δοκιμή και επικύρωση.
4. Προσδιορισμός των χαρακτηριστικών εισόδου από το υποσύνολο δεδομένων εκπαίδευσης που παρέχουν επαρκείς πληροφορίες για ακριβείς προβλέψεις.
5. Επιλογή κατάλληλου αλγορίθμου ταξινόμησης.
6. Εφαρμογή του επιλεγμένου αλγορίθμου στο σύνολο δεδομένων εκπαίδευσης, χρησιμοποιώντας το υποσύνολο δεδομένων επικύρωσης (validation data)για βελτίωση του μοντέλου.
7. Αξιολόγηση της ακρίβειας του μοντέλου χρησιμοποιώντας το υποσύνολο δοκιμής. Εάν το μοντέλο παράγει σωστές προβλέψεις, τότε υποδεικνύει την ακρίβειά του.

Η επιβλεπόμενη μάθηση χωρίζεται σε δύο τύπους προβλημάτων:

I. Αλγόριθμοι παλινδρόμησης (regression algorithms) που χρησιμοποιούνται όταν υπάρχει σχέση μεταξύ των μεταβλητών εισόδου και εξόδου, προβλέποντας συνεχείς μεταβλητές όπως προγνώσεις καιρού ή τάσεις της αγοράς.

II. Αλγόριθμοι ταξινόμησης (classification algorithms) που χρησιμοποιούνται όταν η μεταβλητή εξόδου είναι κατηγορική, με δύο κατηγορίες όπως ναι-όχι, αρσενικό-θηλυκό, σωστό-λάθος κ.λπ. [5]

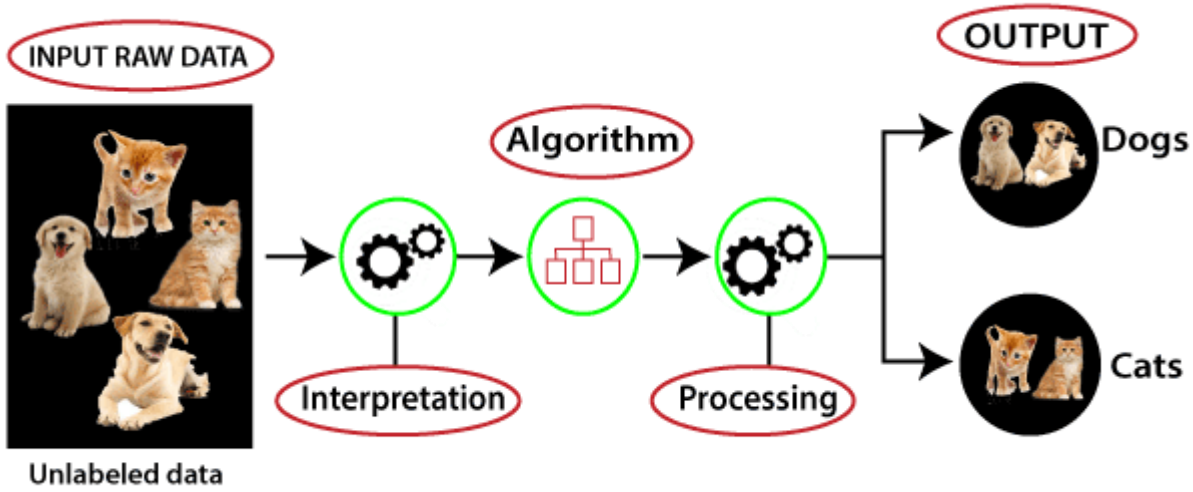
1.1.2.2 Μη επιβλεπόμενη μάθηση

Η μη επιβλεπόμενη μάθηση (unsupervised learning) είναι μία προσέγγιση μηχανικής μάθησης όπου τα μοντέλα δεν καθοδηγούνται από δεδομένα εκπαίδευσης με ετικέτα. Αντίθετα, αυτά τα μοντέλα ανιχνεύουν ανεξάρτητα κρυφά μοτίβα από τα παρεχόμενα δεδομένα, με παρόμοιο τρόπο όπως μαθαίνει ο ανθρώπινος εγκέφαλος όταν αποκτά μια νέα γνώση. Στην ουσία, η μη επιβλεπόμενη μάθηση μπορεί να οριστεί ως εξής:

Η μη επιβλεπόμενη μάθηση περιλαμβάνει μοντέλα εκπαίδευσης που χρησιμοποιούν σύνολα δεδομένων χωρίς ετικέτα και τους επιτρέπει να επεξεργάζονται τα δεδομένα αυτά χωρίς κάποια καθοδήγηση ή επίβλεψη.

Σε αντίθεση με την επιβλεπόμενη μάθηση, η μη επιβλεπόμενη μάθηση δεν μπορεί να εφαρμοστεί άμεσα σε προβλήματα παλινδρόμησης ή ταξινόμησης, επειδή δεν υπάρχουν διαθέσιμα αντίστοιχα δεδομένα εξόδου. Ο πρωταρχικός στόχος της μη επιβλεπόμενης μάθησης είναι να αποκαλύψει την υποκείμενη δομή ενός συνόλου δεδομένων, να ομαδοποιήσει παρόμοια κομμάτια δεδομένων μαζί και να αναπαραστήσει το σύνολο δεδομένων σε συμπυκνωμένη μορφή.

Για παράδειγμα, ας εξετάσουμε έναν αλγόριθμο μη επιβλεπόμενης μάθησης πάνω σε ένα σύνολο δεδομένων εισόδου που περιέχει εικόνες από διάφορες γάτες και σκύλους. Αυτός ο αλγόριθμος δεν έχει εκπαιδευτεί ποτέ σε αυτό το συγκεκριμένο σύνολο δεδομένων και δεν έχει προηγούμενη γνώση των χαρακτηριστικών (features) του. Το καθήκον του αλγορίθμου μάθησης χωρίς επίβλεψη είναι να αναγνωρίζει αυτόνομα τα διακριτικά χαρακτηριστικά των εικόνων. Αυτό το επιτυγχάνει ομαδοποιώντας παρόμοιες εικόνες μαζί.



Εικόνα 1.3 Παράδειγμα Μη Επιβλεπόμενης μάθησης [6]

Στο σενάριο που παρουσιάζεται στην εικόνα 1.3, τα δεδομένα εισόδου είναι χωρίς ετικέτα, που σημαίνει ότι δεν έχουν προκαθορισμένες κατηγορίες και δεν παρέχονται αντίστοιχες έξοδοι. Αυτά τα δεδομένα εισόδου χωρίς ετικέτα χρησιμοποιούνται για την εκπαίδευση του μοντέλου μηχανικής μάθησης. Αρχικά, το μοντέλο αναλύει τα ακατέργαστα δεδομένα για να αποκαλύψει κρυφά μοτίβα και στη συνέχεια εφαρμόζει κατάλληλους αλγορίθμους όπως η ομαδοποίηση k-means ή τα δέντρα

αποφάσεων. Μόλις εφαρμοστεί ο κατάλληλος αλγόριθμος, τα αντικείμενα δεδομένων χωρίζονται σε ομάδες με βάση τις ομοιότητες και τις διαφορές τους.

Οι αλγόριθμοι μη επιβλεπόμενης μάθησης ταξινομούνται σε δύο κύριους τύπους προβλημάτων:

1. **Ομαδοποίηση (Clustering):** Η ομαδοποίηση περιλαμβάνει την ομαδοποίηση παρόμοιων αντικειμένων για να διασφαλιστεί ότι τα αντικείμενα μέσα σε μία ομάδα παρουσιάζουν υψηλές ομοιότητες και έχουν ελάχιστες ή καθόλου ομοιότητες με αντικείμενα άλλων ομάδων. Αυτή η διαδικασία προσδιορίζει κοινά σημεία μεταξύ των αντικειμένων και τα κατηγοριοποιεί με βάση την παρουσία ή την απουσία αυτών των κοινών χαρακτηριστικών.

2. **Συσχέτιση (Association):** Οι κανόνες συσχέτισης είναι μέθοδοι μη επιβλεπόμενης μάθησης που χρησιμοποιούνται για την ανακάλυψη σχέσεων μεταξύ μεταβλητών σε μεγάλες βάσεις δεδομένων. Ανακαλύπτουν σύνολα αντικειμένων που συχνά συνυπάρχουν στο σύνολο δεδομένων. Οι κανόνες συσχέτισης χρησιμοποιούνται για τη βελτίωση των στρατηγικών μάρκετινγκ, όπως η αναγνώριση ότι οι πελάτες που αγοράζουν το προϊόν X (π.χ. ψωμί) είναι επίσης πιθανό να αγοράσουν και το προϊόν Y (π.χ. βούτυρο/μαρμελάδα). Η ανάλυση καλαθιού αγοράς χρησιμεύει ως τυπικό παράδειγμα κανόνων συσχέτισης [6].

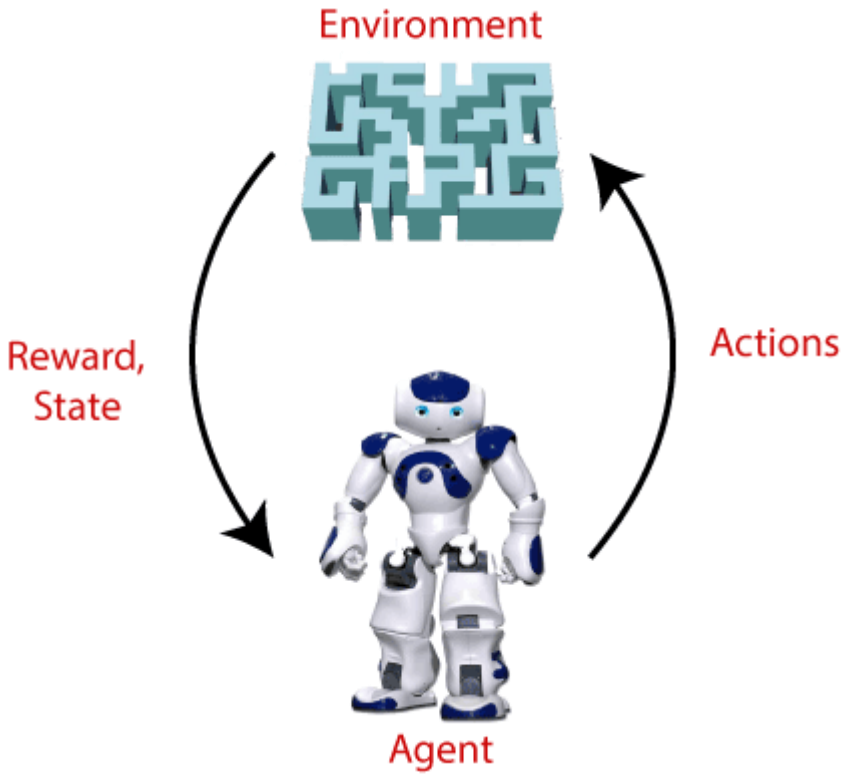
1.1.2.3 Ενισχυτική μάθηση

Η ενισχυτική μάθηση (reinforcement learning) είναι μία τεχνική μηχανικής μάθησης που χρησιμοποιεί ανατροφοδότηση για να εκπαιδεύσει έναν πράκτορα να μπορέσει να λειτουργήσει μέσα σε ένα συγκεκριμένο περιβάλλον. Με τον όρο πράκτορα νοείται μια οντότητα που μπορεί να εξερευνήσει ένα περιβάλλον και να αλληλεπιδράσει με αυτό. Ο πράκτορας μαθαίνει εκτελώντας ενέργειες και παρατηρώντας τις συνέπειες αυτών, λαμβάνοντας θετική ανατροφοδότηση για επιθυμητές ενέργειες και αρνητική ανατροφοδότηση ή κυρώσεις για μη επιθυμητές ενέργειες.

Σε αντίθεση με την επιβλεπόμενη μάθηση, η ενισχυτική δεν βασίζεται σε δεδομένα με ετικέτα. Ο πράκτορας αποκτά γνώσεις αποκλειστικά μέσα από τις δικές του εμπειρίες, καθώς δεν παρέχονται προϋπάρχουσες πληροφορίες. Αυτός ο τύπος μάθησης είναι ιδιαίτερα πλεονεκτικός για προβλήματα που περιλαμβάνουν διαδοχική λήψη αποφάσεων και μακροπρόθεσμους στόχους, όπως είναι τα παιχνίδια και η ρομποτική. Ο πράκτορας ασχολείται ενεργά με το περιβάλλον, εξερευνώντας και μαθαίνοντας από τις ενέργειές του. Ο πρωταρχικός στόχος του πράκτορα στην ενισχυτική μάθηση είναι να μεγιστοποιήσει τις θετικές ανταμοιβές και να ενισχύσει την απόδοσή του. Μέσα από μία διαδικασία δοκιμής και λάθους, ο πράκτορας μαθαίνει να εκτελεί εργασίες πιο αποτελεσματικά με βάση τις συσσωρευμένες εμπειρίες του. Στην ουσία, η ενισχυτική μάθηση μπορεί να οριστεί ως μια προσέγγιση μηχανικής μάθησης όπου ένας ευφυής πράκτορας αλληλεπιδρά με το περιβάλλον και μαθαίνει να ενεργεί ανάλογα.

Ένα αξιοσημείωτο παράδειγμα ενισχυτικής μάθησης είναι ένας ρομποτικός σκύλος που προσπαθεί να τελειοποιήσει την κίνηση των χεριών του. Αποτελεί θεμελιώδες στοιχείο της τεχνητής νοημοσύνης και όλοι οι πράκτορες αυτής λειτουργούν με βάση τις αρχές της ενισχυτικής μάθησης. Ένας πράκτορας δεν απαιτεί προ-προγραμματισμό αλλά μαθαίνει αυτόνομα από τις δικές του εμπειρίες, χωρίς ανθρώπινη παρέμβαση.

Για παράδειγμα, οραματιστείτε έναν πράκτορα τεχνητής νοημοσύνης τοποθετημένο σε περιβάλλον που μοιάζει με λαβύρινθο με στόχο την εύρεση ενός διαμαντιού. Ο πράκτορας αλληλεπιδρά με το περιβάλλον αναλαμβάνοντας ενέργειες, οι οποίες οδηγούν σε αλλαγές στην κατάστασή του και λαμβάνει ανατροφοδότηση με τη μορφή ανταμοιβών ή κυρώσεων. Ο πράκτορας εμπλέκεται συνεχώς σε αυτόν τον κύκλο δράσης, αλλαγής κατάστασης (ή παραμονής στην ίδια κατάσταση) και λήψης ανατροφοδότησης. Μέσω αυτής της επαναληπτικής διαδικασίας, μαθαίνει και εξερευνά το περιβάλλον, ανακαλύπτοντας ποιες ενέργειες αποφέρουν θετικές ανταμοιβές και ποιες οδηγούν σε αρνητικές τιμωρίες. Οι θετικές ανταμοιβές υποδηλώνονται με θετικούς βαθμούς, ενώ οι τιμωρίες αντιπροσωπεύονται με αρνητικούς βαθμούς [7].

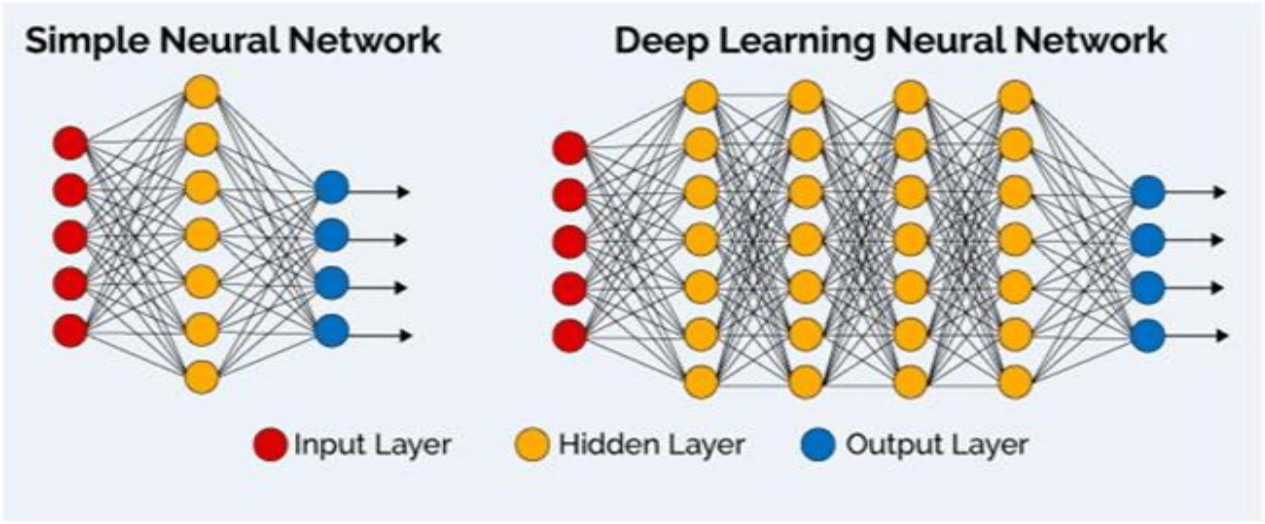


Εικόνα 1.4 Ο πράκτορας και η αλληλεπίδρασή του με ένα περιβάλλον [7]

1.2 Βασικές αρχές βαθιάς μάθησης

1.2.1 Ορισμός της βαθιάς μάθησης και της σχέσης της με τα νευρωνικά δίκτυα

Η βαθιά μάθηση (deep learning) είναι ένα υποσύνολο της τεχνητής νοημοσύνης άρα και της μηχανικής μάθησης, το οποίο ασχολείται με πολύπλοκα νευρωνικά δίκτυα (neural networks) που περιέχουν κρυμμένα επίπεδα (hidden layers). Αυτά τα νευρωνικά δίκτυα προσπαθούν να λειτουργήσουν όπως ένας ανθρώπινος εγκέφαλος με σκοπό να μάθουν καινούργιες πληροφορίες μέσω τεράστιων συνόλων από δεδομένα. Ένα νευρωνικό δίκτυο με μόνο ένα επίπεδο μπορεί να κάνει προβλέψεις που να βγαίνουν σωστές, παρόλα αυτά ένα αντίστοιχο με επιπλέον κρυφά επίπεδα κάνει σίγουρα ακόμα καλύτερες προβλέψεις αφού έχει μεγαλύτερη ακρίβεια και απόδοση.
Η λέξη «βαθιά» έχει να κάνει με τη ποσότητα των κρυμμένων επιπέδων που περιέχει ένα νευρωνικό δίκτυο τα οποία ξεκινάνε από τρία και μπορούν να φτάσουν όσα επιλέξει κάποιος π.χ. 140. Τα απλά νευρωνικά δίκτυα που δεν ανήκουν στην βαθιά μάθηση συνήθως έχουν μέχρι 3 κρυμμένα επίπεδα. Παρόλα αυτά εάν ένα νευρωνικό δίκτυο έχει ένα μόνο κρυφό επίπεδο αλλά με πάρα πολλούς νευρώνες, μπορεί να θεωρηθεί βαθύ.

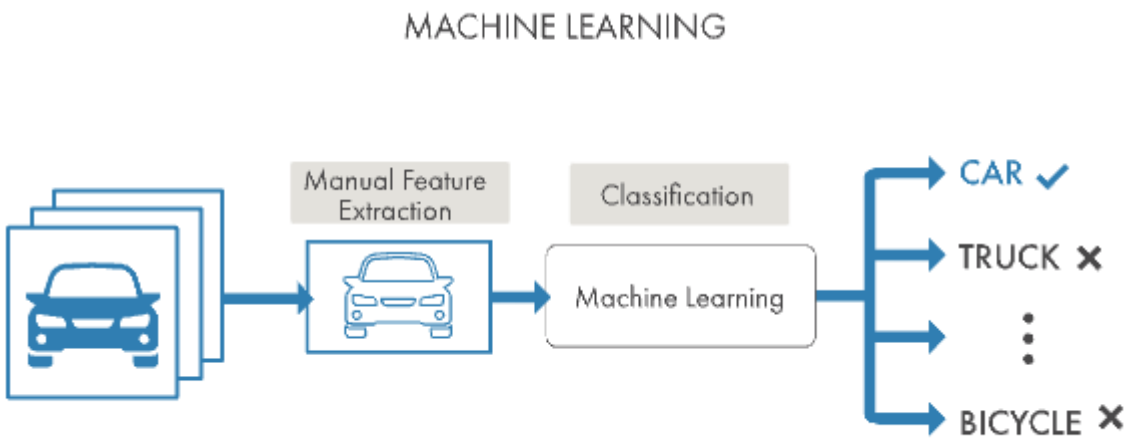


Εικόνα 1.5 Διαφορά απλού νευρωνικού δικτύου με βαθύ νευρωνικού δικτύου [8]

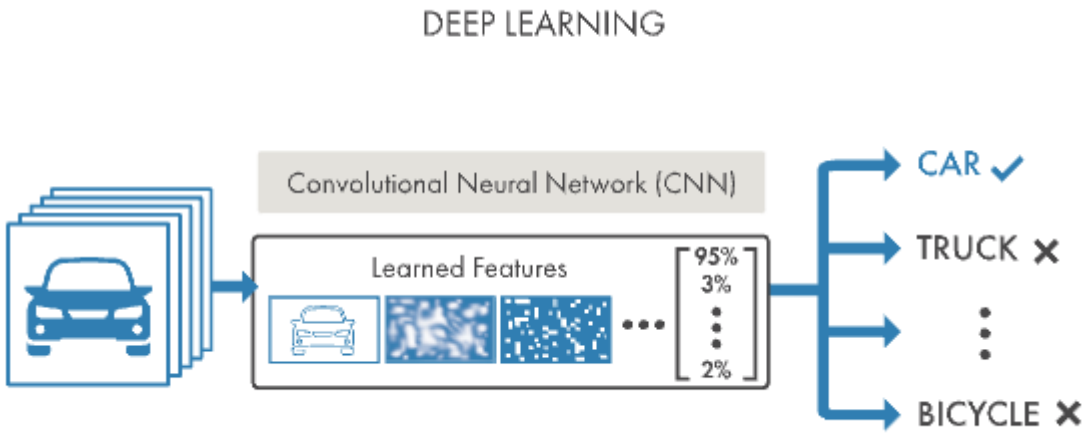
Η βαθιά μάθηση είναι ένα εξειδικευμένο πεδίο της μηχανικής μάθησης που διαθέτει διαφορετικά χαρακτηριστικά σε σύγκριση με τις συμβατικές μεθόδους. Σε μια τυπική διαδικασία μηχανικής μάθησης, τα χαρακτηριστικά από τις εικόνες εξάγονται χειροκίνητα, τα οποία στη συνέχεια χρησιμοποιούνται για την κατασκευή ενός μοντέλου ικανού να κατηγοριοποιήσει αντικείμενα που περιέχονται σε αυτές. Αντίθετα, η βαθιά μάθηση λειτουργεί με διαφορετικό τρόπο αφού εξάγει αυτόματα τα σημαντικά χαρακτηριστικά απευθείας από τις εικόνες. Επιπλέον, η βαθιά μάθηση χαρακτηρίζεται από τη λογική της «από άκρο σε άκρο μάθησης», κατά την οποία ένα δίκτυο το οποίο τροφοδοτείτε με ακατέργαστα δεδομένα και έχει να εκτελέσει έναν συγκεκριμένο σκοπό, όπως η ταξινόμηση, μαθαίνει αυτόνομα να εκτελεί αυτόν τον σκοπό.

Μία αξιοσημείωτη διάκριση μεταξύ της βαθιάς μάθησης και άλλων προσεγγίσεων της είναι η αντιμετώπισή της ως προς τις μεγάλες ποσότητες δεδομένων. Ενώ οι απλές μέθοδοι μάθησης φτάνουν μέχρι ένα επίπεδο απόδοσης, όσα περισσότερα δεδομένα εκπαίδευσης και αν εισαχθούν στο δίκτυο, οι αλγόριθμοι βαθιάς μάθησης έχουν την ικανότητα να βελτιώνουν συνεχώς την απόδοσή τους καθώς αυξάνεται το μέγεθος του συνόλου δεδομένων. Αυτή η ικανότητα αποτελεί

σημαντικό πλεονέκτημα των δικτύων βαθιάς μάθησης [9]. Στις εικόνες 1.6 και 1.7 φαίνεται η διαφορά μιας διαδικασίας ταξινόμησης οχημάτων με μηχανική μάθηση και με βαθιά μάθηση.



Εικόνα 1.6 Ταξινόμηση οχημάτων με μηχανική μάθηση [9]



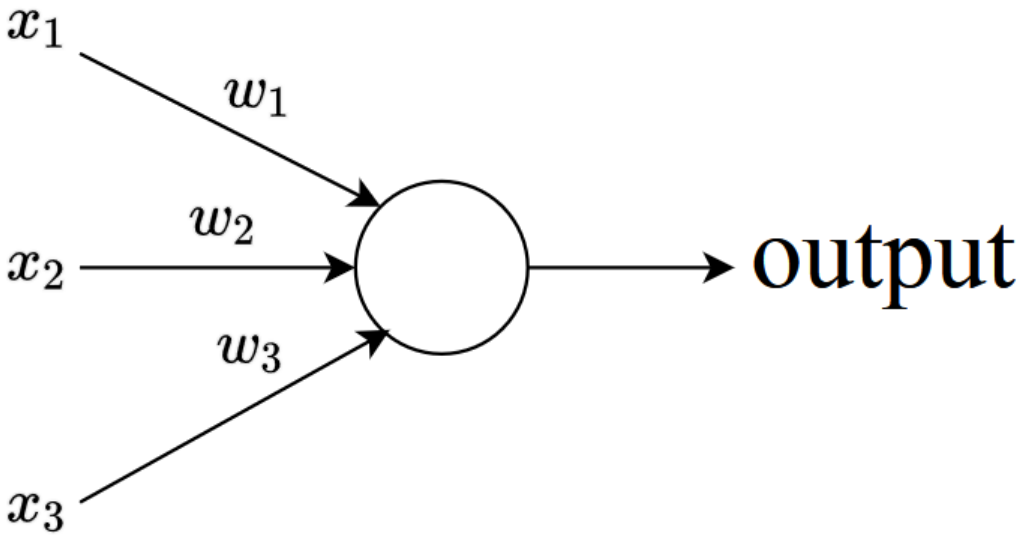
Εικόνα 1.7 Ταξινόμηση οχημάτων με βαθιά μάθηση [9]

1.2.2 Επισκόπηση της βασικής αρχιτεκτονικής και λειτουργίας νευρωνικών δικτύων

Για την κατανόηση της λειτουργίας ενός νευρωνικού δικτύου πρέπει να γίνει πρώτα ανάλυση και επεξήγηση των επιμέρους κομματιών που το αποτελούν και συγκεκριμένα των νευρώνων του. Για την κατανόηση ενός νευρώνα θα αναφερθούμε στον πιο παλιό τεχνητό νευρώνα που σχεδιάστηκε από τον επιστήμονα Frank Rosenblatt, με όνομα perceptron (αντίληπτρο).

1.2.2.1 Perceptron

Ο Perceptron (Αντίληπτρο) είναι η πιο παλιά και απλή μορφή τεχνητού νευρώνα και η μορφή του είναι αυτή της εικόνας 1.8. Θα μπορούσαμε να τον φανταστούμε δηλαδή σαν μία μικρή μπάλα η οποία δέχεται στην είσοδό του από 1 έως πολλές δυαδικές εισόδους και παράγει μία μόνο δυαδική έξοδο.



Εικόνα 1.8 Ο τεχνητός νευρώνας Perceptron με τρείς εισόδους και μία έξοδο [10]

Έστω ότι ο νευρώνας μας έχει τρείς δυαδικές εισόδους που συμβολίζονται ως x_1, x_2, x_3. Με βάση τη θεωρία του Rosenblatt για τον υπολογισμό της εξόδου, για κάθε είσοδο υπάρχει αντίστοιχα μία μεταβλητή που την επηρεάζει με ονομασία βάρος (weight) και συμβολίζεται ως w_1, w_2, w_3. Τα βάρη πρακτικά είναι αληθινά νούμερα που το καθένα έχει μία συγκεκριμένη βαρύτητα ως προς την έξοδο. Η έξοδος του νευρώνα θα έχει τιμή 0 ή 1 και υπολογίζεται ως το άθροισμα των γινομένων $w_i * w_i$ το οποίο συγκρίνεται με ένα κατώφλι (threshold) και μας δίνει το αποτέλεσμα. Το κατώφλι αυτό ουσιαστικά ονομάζεται πόλωση (bias) b και πρακτικά δεν συγκρίνεται αλλά προστίθεται στο τελικό άθροισμα του γινομένου έτσι ώστε να μεταβάλλει ανάλογα το αποτέλεσμα εξόδου [11]. Οι δύο παραλλαγές των συναρτήσεων εξόδου φαίνονται παρακάτω:

Συνάρτηση εξόδου Perceptron χωρίς πόλωση

$$\text{output} = \begin{cases} 0 & \text{if} \quad \sum_j w_j x_j \leq \text{threshold} \\ 1 & \text{if} \quad \sum_j w_j x_j > \text{threshold} \end{cases} \tag{1.1}$$

Συνάρτηση εξόδου Perceptron με πόλωση

$$\text{output} = \begin{cases} 0 & \text{if} \quad \sum_j w_j x_j + b \leq 0 \\ 1 & \text{if} \quad \sum_j w_j x_j + b > 0 \end{cases} \tag{1.2}$$

Ένα παράδειγμα για την καλύτερη κατανόηση της λειτουργίας του Perceptron είναι το παρακάτω. Έστω ότι διοργανώνεται μία εκδρομή για τα Μετέωρα και ο κ. Γιάννης θέλει πολύ να πάει. Το αν θα πάει στην εκδρομή εξαρτάται από τις τρείς εξής παραμέτρους:

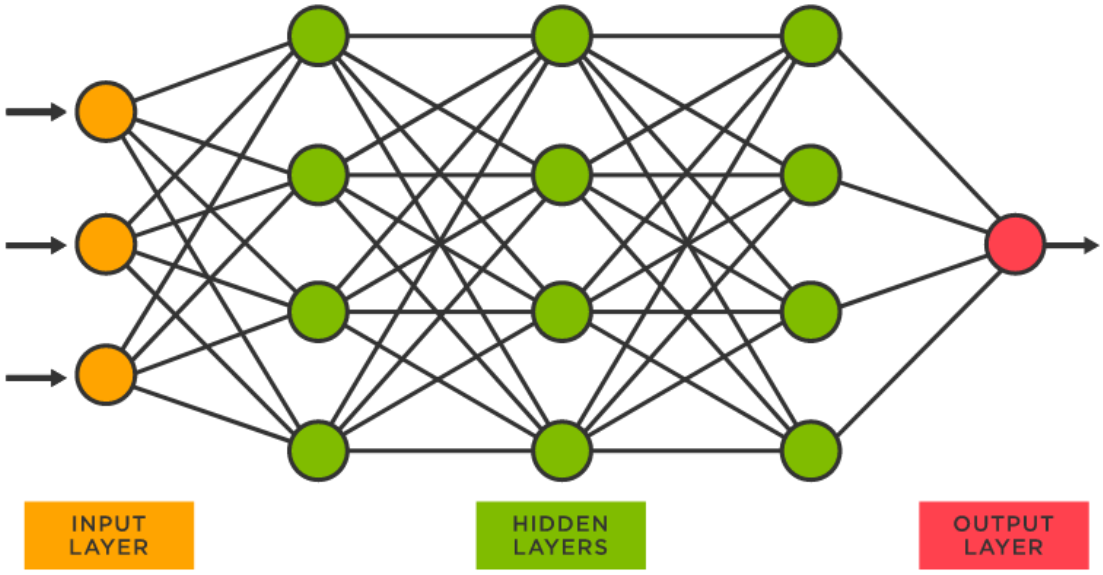
1. Θα έχει ήλιο εκείνη την ημέρα;
2. Θα μπορέσει να πάει μαζί του ο φίλος του ο κ. Γιώργος για παρέα;
3. Θα είναι κάποιος διαθέσιμος εκείνη τη μέρα να τον μεταφέρει μέχρι τα ΚΤΕΛ για να πάρει το πούλμαν;

Κάθε μία από αυτές τις παραμέτρους αντιστοιχεί σε μια είσοδο του νευρώνα και ονομάζονται x_1, x_2, x_3. Έτσι για παράδειγμα αν σε μία κατάσταση η απάντηση μπορεί να δοθεί θετικά τότε η αντίστοιχη είσοδος θα παίρνει τιμή 1 ενώ αν είναι αρνητική 0. Δηλαδή αν εκείνη τη μέρα δεν έχει ήλιο αλλά βρέχει τότε η παράμετρος x_1 θα είναι ίση με 0 ή αν μπορέσει να πάει ο κ. Γιώργος στην εκδρομή τότε η x_2 θα είναι 1.
Έτσι αν υποθέσουμε ότι ο κ. Γιάννης θέλει οπωσδήποτε να πάει στην εκδρομή ανεξαρτήτως των τριών παραμέτρων που ορίσαμε αλλά πρακτικά δεν γίνεται να πάει λόγω βροχής εκείνη τη μέρα, μπορούμε να χρησιμοποιήσουμε τον νευρώνα Perceptron να πάρει μια απόφαση για αυτόν αναλόγως την κατάσταση. Αρχικά πρέπει να οριστούν τα βάρη για κάθε παράμετρο ανάλογα με το πόση σημασία έχει η κάθε παράμετρος για τον κ. Γιάννη. Δηλαδή αν ορίσουμε ως $x_1=6$, $x_2=1$, $x_3=1$ αυτό σημαίνει πως εάν έχει καλό καιρό έξω εκείνη τη μέρα με ήλιο είναι πολύ σημαντικό για το αν θα πάει ο κ. Γιάννης στην εκδρομή ενώ το εάν έρθει ο κ. Γιώργος στην εκδρομή δεν έχει τόση μεγάλη σημασία όπως και το αν θα βρεθεί κάποιος διαθέσιμος να μεταφέρει τον κ. Γιάννη στα ΚΤΕΛ. Οπότε όσο μεγαλύτερο το βάρος τόσο περισσότερη σημασία έχει για τον κ. Γιάννη η συγκεκριμένη παράμετρος. Έτσι έστω ότι επιλέγεται ένα κατώφλι με τιμή 2, επειδή τα γινόμενα από τα βάρη και τις αντίστοιχες παραμέτρους τους x_2 και x_3 είναι μικρότερα από το κατώφλι που επιλέχθηκε, οι παράμετροι αυτές δεν θα έχουν καμία σημασία για την τελική απόφαση του perceptron. Αντιθέτως η παράμετρος x_1 θα είναι η μόνη που θα ληφθεί υπόψιν σε αυτό το συγκεκριμένο μοντέλο αποφάσεων που δημιουργήθηκε και το αποτέλεσμα θα είναι 1 εάν έχει καλό καιρό και 0 εάν δεν έχει.

1.2.2.2 Απλό νευρωνικό δίκτυο

Αφού αναλύσαμε το perceptron που είναι το βασικό κομμάτι ενός νευρωνικού δικτύου, μπορούμε να προχωρήσουμε στην επεξήγηση ενός απλού νευρωνικού δικτύου που αποτελείται από αρκετούς Perceptron.

Ένα νευρωνικό δίκτυο μπορεί να παίρνει πιο έξυπνες αποφάσεις από έναν μόνο perceptron λόγω της πολυπλοκότητάς και του μεγέθους του. Αρχικά περιέχει επίπεδα (layers) από νευρώνες τα οποία όσα περισσότερα είναι τόσο πιο στοχευμένη θα είναι η πρόβλεψη του. Το πρώτο επίπεδο , δηλαδή η πρώτη στήλη από νευρώνες όπως βλέπουμε και στην εικόνα 1.9 δέχονται στις εισόδους τους όλες τις παραμέτρους που ορίστηκαν. Έπειτα κάθε νευρώνας του πρώτου επιπέδου συνδέει την έξοδό του στην είσοδο κάθε νευρώνα του δευτέρου επιπέδου. Έτσι κάθε νευρώνας του δευτέρου επιπέδου υπολογίζει την έξοδό του με βάση τα δεδομένα που δέχτηκε στην είσοδό του και η έξοδός του τροφοδοτείται σε κάθε νευρώνα του επόμενου (τρίτου) επιπέδου.



Εικόνα 1.9 Απλό νευρωνικό δίκτυο [12]

Οι νευρώνες του τρίτου επιπέδου παίρνουν και αυτοί αποφάσεις ανάλογα με τις αποφάσεις που δέχτηκαν από το δεύτερο επίπεδο. Άρα όσα περισσότερα ενδιάμεσα επίπεδα υπάρχουν τόσο πιο μεγάλη ανάλυση των προηγούμενων αποφάσεων γίνεται, άρα και η τελική απόφαση θα είναι πιο συγκεκριμένη. Η τελική πρόβλεψη γίνεται στο τελευταίο επίπεδο ενώ όλα τα ενδιάμεσα επίπεδα μεταξύ του πρώτου επιπέδου και του τελευταίου ονομάζονται «κρυφά» (hidden).

1.2.3 Συνελικτικά νευρωνικά δίκτυα

Τα συνελικτικά νευρωνικά δίκτυα (convolutional neural networks) ή αλλιώς CNN, είναι πολύ σημαντικά για την ταξινόμηση εικόνων και την εξαγωγή δεδομένων μέσα από αυτές. Έχουν ενισχύσει πολλούς κλάδους όπως είναι ο εκπαιδευτικός και η ιατρική διευκολύνοντας τη ζωή των ανθρώπων. Αποτελούν την πιο πρακτική και αποτελεσματική κατηγορία νευρωνικών δικτύων της βαθιάς μάθησης.

Ένα βασικό πλεονέκτημα των CNN σε αντίθεση με τους παραδοσιακούς αλγόριθμους μηχανικής μάθησης, είναι ότι αυτά μπορούν να ανιχνεύσουν μοτίβα σε μία εικόνα ανεξάρτητα από τη θέση τους. Αυτό επιτυγχάνεται μέσω συνελικτικών επιπέδων που εφαρμόζουν φίλτρα, επιτρέποντας την ανίχνευση χαρακτηριστικών ανεξάρτητα από το που βρίσκονται. Έτσι, τα CNN μπορούν να ταξινομούν με ακρίβεια τις εικόνες, χωρίς να επηρεάζονται από παραλλαγές στον προσανατολισμό ή την τοποθέτηση των εικόνων αυτών. Αυτή η ιδιότητά τους τα καθιστά εξαιρετικά αξιόπιστα σε πραγματικά σενάρια της καθημερινότητας.

Τα CNN υπερέχουν επίσης στην αποτελεσματικότητα των δεδομένων, απαιτώντας λιγότερα δεδομένα εκπαίδευσης σε σύγκριση με τις παραδοσιακές μεθόδους. Καταγράφοντας κοινά χαρακτηριστικά από τις εικόνες και βελτιώνοντας την ικανότητα γενίκευσής τους πάνω σε δεδομένα που δεν έχουν ξαναδεί, τα CNN μπορούν να μάθουν αποτελεσματικά από έναν περιορισμένο αριθμό δεδομένων εκπαίδευσης. Αυτή η ιδιότητα τα καθιστά ιδιαίτερα κατάλληλα για καταστάσεις όπου μεγάλα σύνολα δεδομένων με ετικέτα δεν είναι άμεσα διαθέσιμα. Τα CNN

έχουν τη δυνατότητα να εξάγουν σημαντικές πληροφορίες από περιορισμένα δεδομένα, κάνοντας την εκπαίδευσή τους πιο αποδοτική και αποτελεσματική.
Ένα άλλο σημαντικό πλεονέκτημα των CNN είναι η ικανότητά τους στη μεταφορά μάθησης (transfer learning). Μπορούν δηλαδή, να αξιοποιήσουν τη γνώση που αποκτήθηκε από μια εργασία και να την εφαρμόσουν σε σχετικές εργασίες. Αυτό επιτυγχάνεται με τη χρήση προεκπαιδευμένων μοντέλων, τα οποία εκπαιδεύονται σε εκτεταμένα σύνολα δεδομένων και μπορούν να βελτιστοποιηθούν για συγκεκριμένες εργασίες ταξινόμησης εικόνων. Η μεταφορά μάθησης μειώνει σημαντικά την ανάγκη για περεταίρω εκπαιδευτικά δεδομένα, με αποτέλεσμα την βελτίωση στην απόδοση μιας ταξινόμησης. Αυτή η ικανότητα επιτρέπει στα CNN να εκμεταλλεύονται τη δύναμη της γνώσης που αποκτήθηκε προηγουμένως σε άλλες εργασίες, επιτρέποντας την ταχύτερη και πιο ακριβή μάθηση.

Τα CNN χαρακτηρίζονται από επεκτασιμότητα, με αποτέλεσμα να προσαρμόζονται σε εργασίες ταξινόμησης εικόνων διαφορετικής πολυπλοκότητας. Η αρχιτεκτονική τους μπορεί να τροποποιηθεί προσθέτοντας ή αφαιρώντας επίπεδα, προσαρμόζοντας τον αριθμό των φίλτρων σε κάθε επίπεδο και αλλάζοντας το μέγεθος των φίλτρων που χρησιμοποιούνται στα συνελικτικά επίπεδα. Αυτή η ευελιξία επιτρέπει στα CNN να αντιμετωπίζουν ένα ευρύ φάσμα εφαρμογών, από απλή ταξινόμηση εικόνων έως πιο προηγμένες εργασίες όπως η ανίχνευση αντικειμένων (object detection) και η τμηματοποίηση (segmentation). Με την επεκτασιμότητα τους, τα CNN μπορούν να ανταποκριθούν στις απαιτήσεις διαφορετικών και εξελισσόμενων προβληματικών τομέων [13].

Τα CNN αποτελούνται είτε από όλα είτε από το συνδυασμό κάποιων συγκεκριμένων επιπέδων που κάθε ένα από αυτά παίζει συγκεκριμένο ρόλο στην επεξεργασία των δεδομένων που δέχεται. Αυτά τα επίπεδα είναι [13] [14] [15]:

1. **Επίπεδο εισόδου (Input layer):** Το επίπεδο εισόδου χρησιμεύει ως το αρχικό στάδιο, λαμβάνοντας εικόνες και προσαρμόζοντας το μέγεθός τους πριν αυτές περάσουν στα επόμενα στρώματα για εξαγωγή χαρακτηριστικών.

2. **Επίπεδο συνέλιξης (Convolution layer)**: Το επίπεδο συνέλιξης λειτουργεί ως φίλτρο, που εξάγει χαρακτηριστικά από τις εικόνες και προσδιορίζει τα κοινά σημεία των χαρακτηριστικών κατά τη διάρκεια μιας δοκιμής σε ένα σύνολο δοκιμαστικών δεδομένων.

3. **Επίπεδο συγκέντρωσης (Pooling layer):** Τα εξαγόμενα σύνολα χαρακτηριστικών κατευθύνονται στη συνέχεια στο επίπεδο συγκέντρωσης, το οποίο μειώνει το μέγεθος της εικόνας διατηρώντας παράλληλα σημαντικές πληροφορίες. Επιλέγοντας τη μέγιστη τιμή σε κάθε παράθυρο, το επίπεδο συγκέντρωσης διατηρεί τα καλύτερα χαρακτηριστικά προσαρμογής. Ως παράθυρο νοείται ένα φίλτρο για παράδειγμα διαστάσεων 3x3 pixel το οποίο σαρώνει όλη την εικόνα από την αρχή μέχρι το τέλος της και πράττοντας τα προαναφερόμενα.

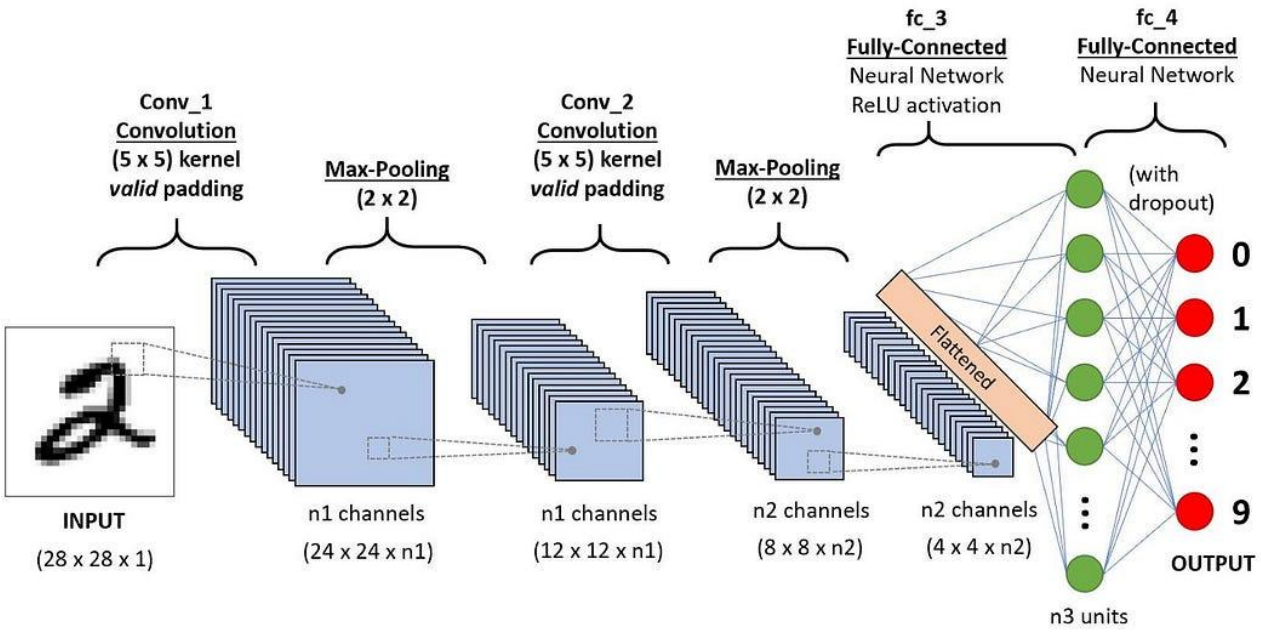
4. **Επίπεδο ισοπέδωσης (Flatten layer):** Η ισοπέδωση είναι μία τεχνική που χρησιμοποιείται για τη μετατροπή των συγκεντρωτικών χαρτών των χαρακτηριστικών (pooled feature maps), οι οποίοι αναπαρίστανται ως δισδιάστατοι πίνακες, σε ένα ενιαίο επιμήκη γραμμικό διάνυσμα. Αυτός ο μετασχηματισμός επιτρέπει στον ισοπεδωμένο πίνακα να χρησιμοποιηθεί ως είσοδος για το πλήρως συνδεδεμένο επίπεδο, επιτρέποντας εργασίες ταξινόμησης εικόνων.

5. **Επίπεδο ReLU (Rectified Linear Units layer):** Το επίπεδο ReLU αντικαθιστά τους αρνητικούς αριθμούς του επιπέδου συγκέντρωσης ή του επιπέδου ισοπέδωσης με μηδενικά, συμβάλλοντας στη μαθηματική σταθερότητα του CNN. Αυτό αποτρέπει τις εκπαιδευμένες τιμές του δικτύου από το να πάρουν τιμή κοντά στο μηδέν ή να μεγαλώσουν υπερβολικά.

6. **Πλήρως συνδεδεμένο επίπεδο (Fully Connected layer):** Το πλήρως συνδεδεμένο επίπεδο λαμβάνει τις εξαιρετικά φιλτραρισμένες εικόνες και τις αντιστοιχίζει σε συγκεκριμένες κατηγορίες δίνοντάς τους αντίστοιχες ετικέτες.

7. **Επίπεδο εγκατάλειψης (Dropout layer):** Το επίπεδο εγκατάλειψης είναι ένα πολύτιμο εργαλείο, καθώς συμβάλλει στη βελτίωση της απόδοσης των μοντέλων μηχανικής εκμάθησης, αποτρέποντας την υπερπροσαρμογή και απλοποιώντας το δίκτυο. Με την απενεργοποίηση ενός ποσοστού τυχαίων νευρώνων κατά τη διάρκεια της εκπαίδευσης, η εγκατάλειψη επιτρέπει τη δημιουργία ενός πιο ισχυρού και γενικευμένου μοντέλου μειώνοντας αποτελεσματικά την πολυπλοκότητα του. Η εγκατάλειψη ουσιαστικά δεν τοποθετείται μετά από κάποιο επίπεδο αλλά εφαρμόζεται σαν μάσκα πάνω σε κάποια από τα πλήρως συνδεδεμένα επίπεδα που θα επιλεχθούν.

Τα θεμελιώδη μέρη ενός CNN περιλαμβάνουν το συνελικτικό επίπεδο, το επίπεδο συγκέντρωσης και το πλήρως συνδεδεμένο επίπεδο. Αυτά τα επίπεδα αποτελούν τα βασικά στοιχεία ενός συνελικτικού νευρωνικού δικτύου.



Εικόνα 1.10 Επίπεδα Συνελικτικού νευρωνικού δικτύου [14]

1.2.4 Συναρτήσεις ενεργοποίησης

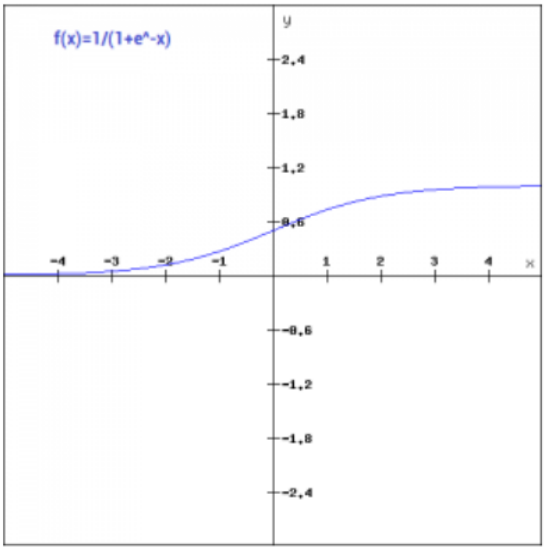
Οι συναρτήσεις ενεργοποίησης (activation functions) χρησιμοποιούνται και εφαρμόζονται κυρίως στα κρυφά επίπεδα των CNN και έχουν σκοπό τη μετατροπή των εξόδων αυτών των επιπέδων από γραμμική σε μη γραμμική. Για την δημιουργία καλύτερων και πιο αποδοτικών μοντέλων νευρωνικών δικτύων οι συναρτήσεις ενεργοποίησης είναι απαραίτητες. Από την άλλη εάν ένα νευρωνικό δίκτυο έχει γραμμικές εξόδους τότε μέγιστη απόδοσή του θα είναι περιορισμένη και η εκπαίδευσή του δεν θα επιφέρει τα πιο επιθυμητά αποτελέσματα ειδικά εάν πρόκειται για εκπαίδευση σε μεγάλο όγκο δεδομένων. Μη γραμμικές θεωρούνται οι συναρτήσεις που κατά την απεικόνισή τους σε κάποιο γράφημα εμφανίζουν καμπυλότητα.
Η εφαρμογή μιας συνάρτησης ενεργοποίησης σε ένα νευρωνικό δίκτυο έχει μεγάλη σημασία για την εισαγωγή δυναμισμού και τη διευκόλυνση της εξαγωγής περίπλοκων πληροφοριών από τα δεδομένα. Μία συνάρτηση ενεργοποίησης παίζει σημαντικό ρόλο στην αναπαράσταση των μη γραμμικών και πολύπλοκων συνδέσεων μεταξύ εισόδων και εξόδων. Με την ενσωμάτωση μη γραμμικών συναρτήσεων ενεργοποίησης, το δίκτυο καθίσταται ικανό να δημιουργεί μη γραμμικές αντιστοιχίσεις μεταξύ των εισόδων και εξόδων.

Η ιδιότητα παραγώγισής της συνάρτησης ενεργοποίησης είναι απαραίτητη καθώς επιτρέπει την βελτιστοποίηση (optimization) μέσω της μεθόδου οπισθοδρόμησης (backpropagation), υπολογίζοντας τα σφάλματα ή τις απώλειες σχετικά με τα βάρη του δικτύου. Αυτή η διαδικασία με τη σειρά της, διευκολύνει τη βελτιστοποίηση των βαρών χρησιμοποιώντας τεχνικές όπως η κάθοδος βασισμένη στην κλίση (Gradient Descent) ή άλλες μεθόδους βελτιστοποίησης για την ελαχιστοποίηση των σφαλμάτων [16].

Μερικές από τις πιο συχνά χρησιμοποιούμενες συναρτήσεις ενεργοποίησης αναφέρονται στις παρακάτω ενότητες.

1.2.4.1 Σιγμοειδής

Η σιγμοειδής (Sigmoid) συνάρτηση ενεργοποίησης χρησιμοποιείται ευρέως λόγω της ικανότητάς της να εισάγει μη γραμμικότητα. Με την εφαρμογή αυτής της συνάρτησης, οι τιμές μετασχηματίζονται ώστε να βρίσκονται στην περιοχή από 0 έως 1. Μαθηματικά, μπορεί να αναπαρασταθεί ως *f(x) = 1/(1 + e^(-x))*. Η σιγμοειδής συνάρτηση παρουσιάζει ομαλή καμπύλη σχήματος S και εξασφαλίζει συνεχή παραγώγιση. Η παράγωγός του δίνεται από το *f'(x) = 1 - σιγμοειδής(x)*. Ωστόσο, είναι σημαντικό να σημειωθεί ότι η σιγμοειδής συνάρτηση δεν είναι συμμετρική γύρω από το μηδέν, με αποτέλεσμα όλες οι τιμές εξόδου των νευρώνων να μοιράζονται το ίδιο πρόσημο. Αυτό το ζήτημα μπορεί να μετριαστεί με την κλιμάκωση της σιγμοειδούς συνάρτησης.

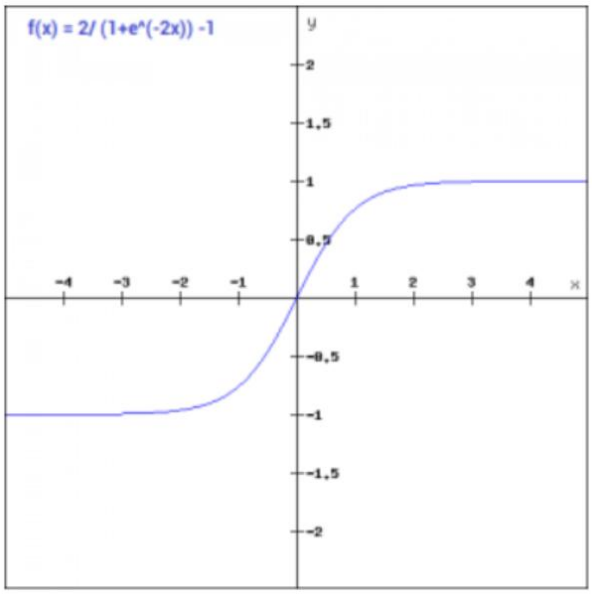


Εικόνα 1.11 Σιγμοειδής συνάρτηση ενεργοποίησης [16]

1.2.4.2 Υπερβολική συνάρτηση εφαπτομένης - Tanh

Η συνάρτηση tanh, γνωστή και ως υπερβολική συνάρτηση εφαπτομένης, είναι παρόμοια με τη συνάρτηση σιγμοειδούς, αλλά έχει συμμετρικό σχήμα γύρω από την αρχή των αξόνων. Αυτή η συμμετρία έχει ως αποτέλεσμα οι έξοδοι από τα προηγούμενα επίπεδα να έχουν διαφορετικά πρόσημα μεταξύ τους. Μαθηματικά, η συνάρτηση tanh μπορεί να εκφραστεί ως *f(x) = 2sigmoid(2x) - 1*. Είναι μία συνεχής και παραγωγίσιμη συνάρτηση που παράγει τιμές οι οποίες κυμαίνονται από -1 έως 1. Εάν συγκρίνουμε την κλίση της σιγμοειδούς συνάρτησης με την κλίση της συνάρτησης tanh, η δεύτερη είναι πιο απότομη. Επίσης η συνάρτηση tanh προτιμάται συχνά έναντι της σιγμοειδούς συνάρτησης επειδή οι κλίσεις της δεν περιορίζονται σε μια συγκεκριμένη κατεύθυνση και είναι κεντραρισμένη γύρω από το μηδέν.

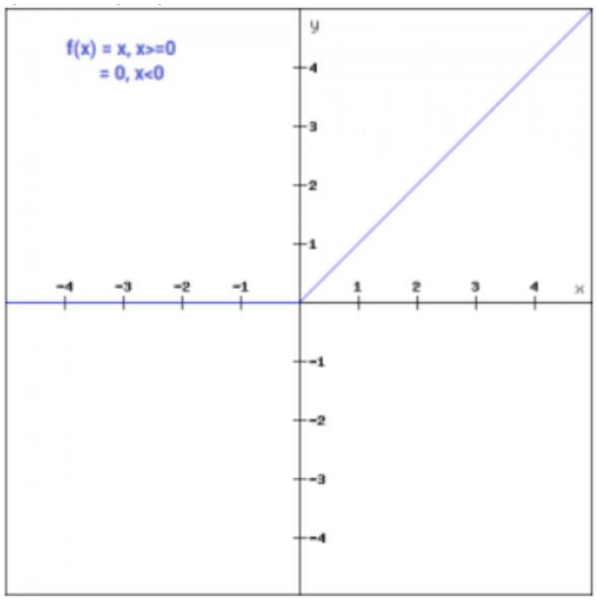
•



Εικόνα 1.12 Συνάρτηση ενεργοποίησης Tanh [16]

1.2.4.3 Διορθωμένη γραμμική μονάδα

Η διορθωμένη γραμμική μονάδα (rectified linear unit) ή αλλιώς ReLU, είναι μια ευρέως χρησιμοποιούμενη συνάρτηση μη γραμμικής ενεργοποίησης στα νευρωνικά δίκτυα. Προσφέρει το πλεονέκτημα της επιλεκτικής ενεργοποίησης νευρώνων, που σημαίνει ότι δεν ενεργοποιούνται όλοι οι νευρώνες ταυτόχρονα. Αντίθετα, ένας νευρώνας απενεργοποιείται μόνο όταν η έξοδος του γραμμικού μετασχηματισμού είναι μηδέν. Μαθηματικά, η ReLU μπορεί να οριστεί ως f(x) = max(0, x). Αυτή η συνάρτηση ενεργοποίησης είναι πιο αποτελεσματική από άλλες επειδή ενεργοποιεί έναν συγκεκριμένο αριθμό νευρώνων κάθε φορά, παρά όλους τους νευρώνες μαζί. Ωστόσο, είναι σημαντικό να σημειωθεί ότι σε ορισμένες περιπτώσεις, η κλίση μπορεί να είναι μηδέν, γεγονός που μπορεί να έχει ως αποτέλεσμα να μην ενημερώνονται τα βάρη και οι πολώσεις κατά τη διάρκεια της διαδικασίας οπισθοδιάδοσης στην εκπαίδευση νευρωνικών δικτύων.



Εικόνα 1.13 Συνάρτηση ενεργοποίησης ReLU [16]

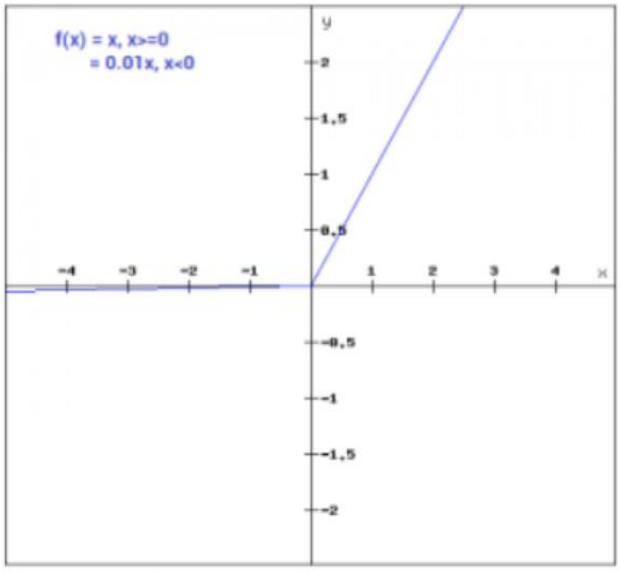
1.2.4.4 Διορθωμένη γραμμική μονάδα Leaky

Η συνάρτηση Διορθωμένη γραμμική μονάδα Leaky (rectified linear unit Leaky) ή αλλιώς Leaky ReLU, είναι μια βελτιωμένη έκδοση της συνάρτησης ReLU που ξεπερνά έναν από τους περιορισμούς της. Αντί να εκχωρεί μια τιμή μηδέν για αρνητικές εισόδους, η συνάρτηση Leaky ReLU εκχωρεί μία μικρή γραμμική συνιστώσα του x. Αυτό μπορεί να εκφραστεί μαθηματικά ως εξής:

Για $x < 0$: $f(x) = 0{,}01x$ (1.3)

Για $x \geq 0$: $f(x) = x$ (1.4)

Με την εισαγωγή αυτής της μικρής κλίσης για αρνητικές εισόδους, η συνάρτηση Leaky ReLU διασφαλίζει ότι οι νευρώνες μπορούν να ενεργοποιηθούν ακόμα και όταν η είσοδος είναι αρνητική. Αυτό αντιμετωπίζει το ζήτημα των "νεκρών νευρώνων" ("dead neurons") που μπορεί να προκύψουν με την τυπική συνάρτηση ReLU.



Εικόνα 1.14 Συνάρτηση ενεργοποίησης Leaky ReLU [16]

1.2.4.5 Softmax

Η συνάρτηση softmax συνδυάζει πολλαπλές σιγμοειδείς συναρτήσεις και έχει σχεδιαστεί ειδικά για εργασίες ταξινόμησης εικόνων πολλαπλών κλάσεων. Ενώ οι σιγμοειδείς συναρτήσεις χρησιμοποιούνται συνήθως για δυαδική ταξινόμηση εικόνων, η συνάρτηση softmax επεκτείνει την εφαρμογή της σε σενάρια που περιλαμβάνουν πολλαπλές κλάσεις. Σε αντίθεση με τις σιγμοειδείς συναρτήσεις, οι οποίες παρέχουν πιθανότητες για μία κλάση, η συνάρτηση softmax εκχωρεί πιθανότητες σε κάθε σημείο δεδομένων σε όλες τις μεμονωμένες κλάσεις. Μπορεί να αναπαρασταθεί μαθηματικά ως $f(x) = exp(zj) / \Sigma k(exp(zk))$ όπου $j = 1,...,K$ και K αναπαριστά το σύνολο των κλάσεων.

Κατά την δημιουργία ενός νευρωνικού δικτύου ή ενός μοντέλου για ταξινόμηση πολλαπλών κλάσεων, το επίπεδο εξόδου του δικτύου θα περιέχει έναν αριθμό νευρώνων ίσο με τον συνολικό αριθμό κλάσεων που θέλουμε να υπάρχουν.

1.2.5 Κάθοδος βασισμένη στην κλίση

Ένα πολύ σημαντικό κομμάτι της βαθιάς μάθησης και των νευρωνικών δικτύων είναι ο αλγόριθμος εν ονόματι κάθοδος βασισμένη στην κλίση (gradient descent) και ανήκει στην κατηγορία των αλγορίθμων βελτιστοποίησης. Ο αλγόριθμος αυτός αναζητά το τοπικό ελάχιστο (local minima) μιας συνάρτησης απωλειών (loss function) και κατ' επέκταση το ολικό ελάχιστο μιας συνάρτησης κόστους (cost function). Αυτό έχει ως στόχο τον μηδενισμό των απωλειών ενός νευρωνικού δικτύου και τη βελτιστοποίησή του όσο το δυνατόν περισσότερο.

Η εφαρμογή αυτού του αλγορίθμου ξεκινά πάνω σε ένα σημείο της συνάρτησης απωλειών του οποίου υπολογίζεται η κλίση παραγωγίζοντας τη συνάρτηση σε αυτό το σημείο. Έτσι γνωρίζοντας την κλίση στο συγκεκριμένο σημείο, ο αλγόριθμος προσπαθεί να την μειώσει σε κάθε επανάληψη εκπαίδευσης του νευρωνικού δικτύου με στόχο κάποια στιγμή να την μηδενίσει.

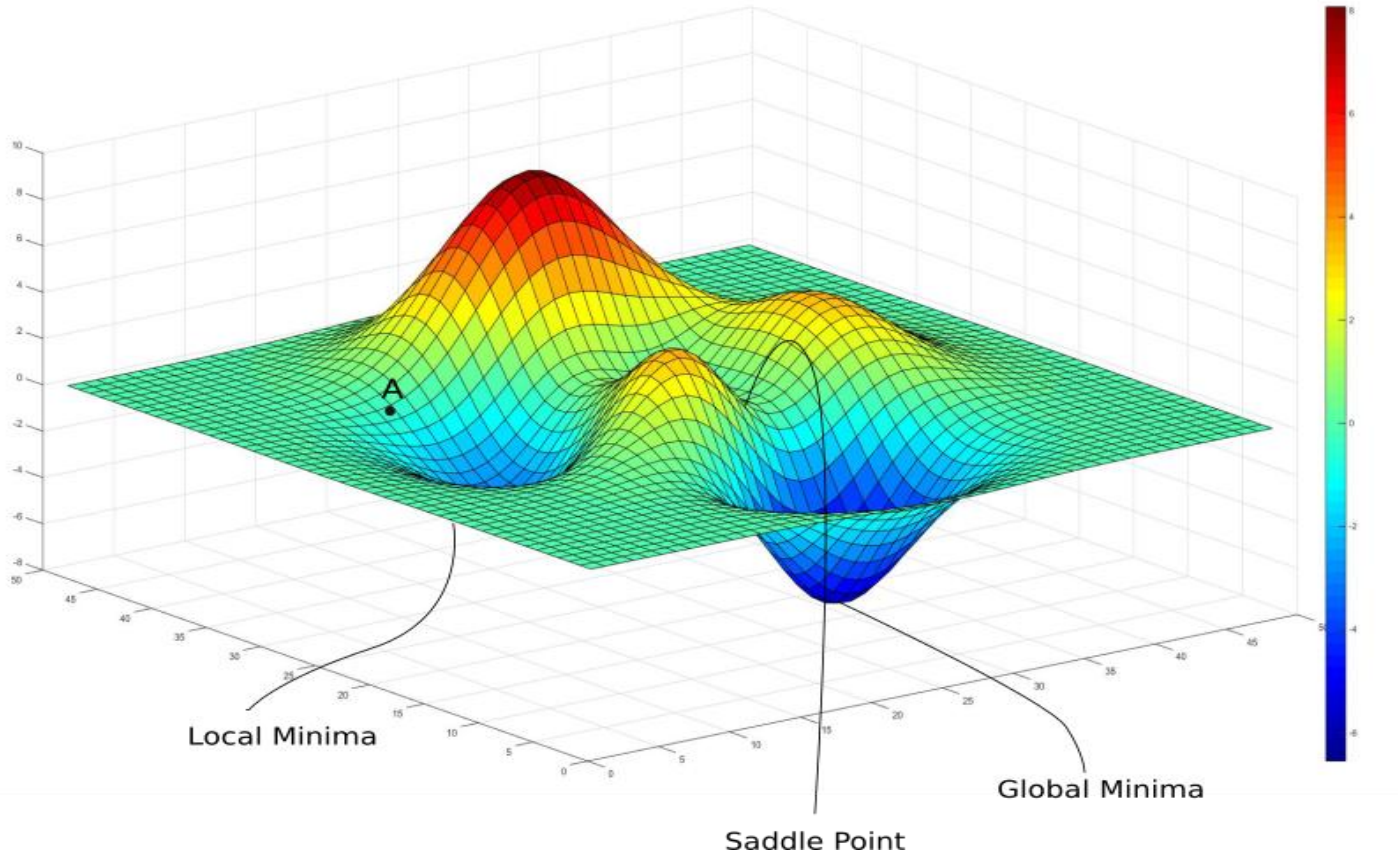
Στο τέλος κάθε επανάληψης που το νευρωνικό δίκτυο εκπαιδεύεται με κάποια δεδομένα υπολογίζεται το νέο σημείο πάνω στην συνάρτηση απωλειών το οποίο μετατοπίστηκε κατά μία απόσταση που εξαρτάται από την υπερπαράμετρο "ρυθμός εκπαίδευσης" (learning rate) γνωστή και ως βήμα της καθόδου. Η υπερπαράμετρος επηρεάζει σημαντικά την επιτυχή εύρεση του τοπικού ελαχίστου και του μηδενισμού της κλίσης και θα την αναλύσουμε περισσότερο στην ενότητα 1.3.1.

Πρέπει να σημειωθεί ότι για την εύρεση του τοπικού ελαχίστου και τον μηδενισμό της κλίσης της συνάρτησης απωλειών, η σωστή κατεύθυνση που πρέπει να μετακινούμε το σημείο είναι αντίθετη από αυτήν της κλίσης. Εάν μετακινούμε το σημείο ως προς την κατεύθυνση της κλίσης τότε θα πλησιάζουμε το τοπικό μέγιστο (local maxima), πράγμα το οποίο δεν επιθυμούμε σε καμία περίπτωση.

Επίσης όσες περισσότερες είναι οι επαναλήψεις ή αλλιώς εποχές (epochs), όπως ονομάζεται σωστά η υπερπαράμετρος, που εκπαιδεύεται το νευρωνικό δίκτυο με τα δεδομένα, τόσο πιο επιτυχής θα είναι η προσέγγιση του τοπικού ελαχίστου άρα και η βελτιστοποίηση του δικτύου.

Ακόμα, είναι σημαντικό να διευκρινιστεί η διαφορά της συνάρτησης κόστους από τη συνάρτηση απωλειών (loss function) που παραπλανούν πολλές φορές τους ανθρώπους και τις θεωρούν ίδιες. Μία συνάρτηση απωλειών αναπαρίσταται ως μία παραβολή που αντιπροσωπεύει ένα μόνο δεδομένο εκπαίδευσης. Από την άλλη μια συνάρτηση κόστους εκφράζει όλα τα δεδομένα εκπαίδευσης άρα αποτυπώνει τον μέσο όρο πολλών συναρτήσεων απωλειών μαζί.

Έτσι, εφόσον στην εκπαίδευση ενός νευρωνικού δικτύου χρησιμοποιούνται πολλά δεδομένα, αναφερόμαστε συνήθως στη συνάρτηση κόστους όταν το εκπαιδεύουμε αφού μας ενδιαφέρει περισσότερο το γενικό κόστος του μοντέλου. Μιλώντας βέβαια για συναρτήσεις κόστους θα πρέπει να αναφερθεί πως δεν θα υπάρχει ένα τοπικό ελάχιστο αλλά πολλά μέσα στη συνάρτηση. Οπότε σε αυτήν την περίπτωση ο αλγόριθμος προσπαθεί να βρει το ολικό ελάχιστο (global minima) που είναι το τοπικό ελάχιστο με την μικρότερη τιμή.



Εικόνα 1.15 Τρισδιάστατη αναπαράσταση μιας συνάρτησης κόστους που απεικονίζει ένα τυχαίο σημείο Α, ένα τοπικό ελάχιστο (local minima), ένα σημείο που η κλίση της συνάρτησης μηδενίζει αλλά δεν είναι πραγματικό ελάχιστο (saddle point) και το ολικό μέγιστο (global minima) [17]

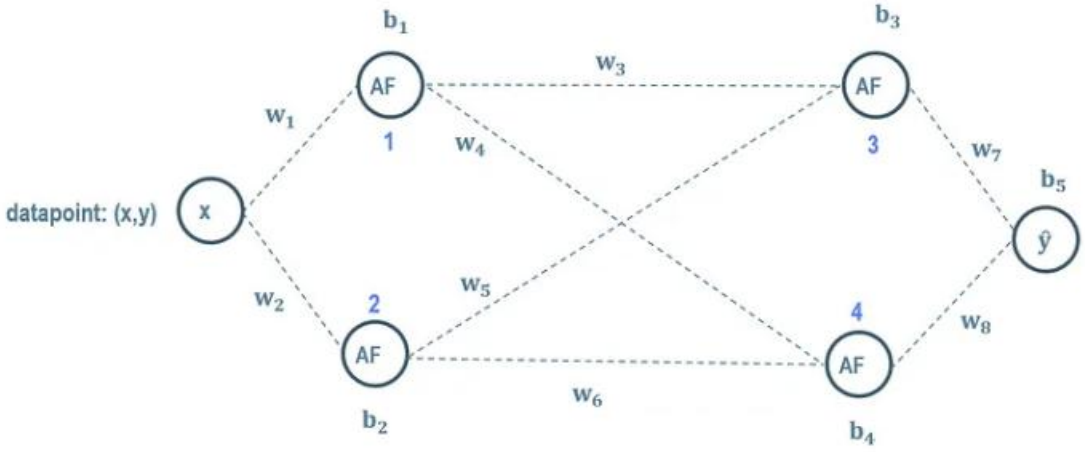
1.2.6 Οπισθοδιάδοση και εμπρόσθια διάδοση

Η μέθοδος της οπισθοδιάδοσης (backpropagation) είναι άλλο ένα σημαντικό κομμάτι των νευρωνικών δικτύων και ειδικότερα των συνελικτικών που περιέχουν πολλά κρυφά επίπεδα. Αυτό που επιτυγχάνει είναι να μεταβάλλει τις τιμές των βαρών και των πολώσεων του δικτύου μετά από μια επανάληψη/εποχή εκπαίδευσης έτσι ώστε να τις βελτιστοποιήσει. Είναι μία τεχνική απαραίτητη η οποία κάνει καλύτερο συνεχώς το μοντέλο και διορθώνει βήμα-βήμα την ακρίβεια των προβλέψεων ενός νευρωνικού δικτύου.

Η οπισθοδιάδοση χρησιμοποιεί τον αλγόριθμο της καθόδου βασισμένη στην κλίση καθώς και την τεχνική του μέσου τετραγωνικού σφάλματος (Root Mean Squared Error). Το μέσο τετραγωνικό σφάλμα μετράει τον μέσο όρο της διαφοράς των τιμών που προέβλεψε ένα νευρωνικό δίκτυο από τις πραγματικές τιμές των δεδομένων εκπαίδευσης. Όσο πιο χαμηλό είναι το αποτέλεσμα τόσο καλύτερα προβλέπει το δίκτυο άρα έχει καλύτερη απόδοση αφού ο μέσος όρος της διαφοράς είναι μικρός.
Πριν αναλύσουμε την οπισθοδιάδοση πρέπει να γίνει αναφορά στην εμπρόσθια διάδοση (forward

propagation) που αποτελεί την διαδικασία ακριβώς πριν την εκτέλεση της μεθόδου της οπισθοδιάδοσης. Οι διαδικασίες της εμπρόσθιας διάδοσης και οπισθοδιάδοσης εκτελούνται και αλλάζουν δεδομένα μόνο μέσα στα κρυφά επίπεδα ενός νευρωνικού δικτύου αφού εκτελούνται μαθηματικές πράξεις σε σχέση με τις τιμές που αναφέρονται στους νευρώνες.

Έστω ότι έχουμε το απλό νευρωνικό δίκτυο τεσσάρων επιπέδων της εικόνας 1.16,



Εικόνα 1.16 Παράδειγμα απλού νευρωνικού δικτύου [18]

Πιο αναλυτικά:

- **Εμπρόσθια διάδοση:** Αρχικά τα βάρη w_1, w_2, ..., w_8 και οι πολώσεις b_1, b_2, …, b_5 του νευρωνικού δικτύου παίρνουν τυχαίες τιμές πριν την εκκίνηση της εμπρόσθιας διάδοσης. Η διαδικασία της εμπρόσθιας διάδοσης ξεκινά από το επίπεδο εισόδου που περιέχει έναν μόνο νευρώνα τον x. Ο νευρώνας x περιέχει την τιμή που δέχτηκε από ένα δεδομένο εκπαίδευσης και υποθέτουμε ότι η τιμή εισόδου αυτή είναι η a_0. Στη συνέχεια ο νευρώνας x στέλνει την έξοδό του δηλαδή στο επόμενο κρυφό επίπεδο άρα στην είσοδο του νευρώνα 1 και 2. Επειδή κάθε σύναψη (synapse), δηλαδή κάθε σύνδεση μεταξύ δυο νευρώνων, έχει τη δικιά της τιμή βάρους, το a_0 πολλαπλασιάζεται αντίστοιχα με το κάθε βάρος. Οι νευρώνες 1 και 2 υπολογίζουν ο καθένας ξεχωριστά την έξοδό τους με βάση τη συνάρτηση (1.2) του κεφαλαίου 1.2.2,. Δηλαδή για παράδειγμα ο νευρώνας 1 υπολογίζει την εξής έξοδο $a_0 * w_1 + b_1$ ενώ ο νευρώνας 2 την $a_0 * w_2 + b_2$.

  Για να κάνουμε πιο εύκολη τη συνέχεια υποθέτουμε ότι όλες οι τιμές των νευρώνων του πρώτου επιπέδου, που στην προκειμένη περίπτωση είναι μόνο ένας αλλά δεν έχει διαφορά, αντιπροσωπεύονται από τον πίνακα A_0 όπου 0 το πρώτο επίπεδο. Αντίστοιχα όλα τα βάρη του δευτέρου επιπέδου τοποθετούνται σε έναν πίνακα W_1 και οι πολώσεις στο B_1, όπου 1 είναι ο δείκτης για το δεύτερο επίπεδο. Έτσι όλες οι έξοδοι των νευρώνων του δευτέρου επιπέδου που υπολογίστηκαν μπορούν να τοποθετηθούν στον πίνακα Z_1 όπου $Z_1 = W_1 * A_0 + B_1$. Έπειτα όλες οι έξοδοι Z_1 προσαρμόζονται κατάλληλα από μια συνάρτηση ενεργοποίησης και αποθηκεύονται στον πίνακα A_1 ως εξής $A_1 = f(Z_1)$. Με αυτόν τον τρόπο όλες οι έξοδοι που περιέχονται στο A_1 στέλνονται στο επόμενο κρυφό επίπεδο νευρώνων, το τρίτο και πολλαπλασιάζονται με τα αντίστοιχα βάρη των νέων συνάψεων.

Έτσι ομοίως με το δεύτερο επίπεδο οι έξοδοι του επιπέδου 3 θα είναι $Z_2 = W_2 * A_1 + B_2$. Με τη σειρά τους οι νευρώνες του επιπέδου 3 θα διαμορφώσουν κατάλληλα τις εξόδους τους με την επιλεγμένη συνάρτηση ενεργοποίησης ως εξής $A_2 = f(Z_2)$. Όλες οι τιμές του A_2 στέλνονται στους τελικούς νευρώνες ή συγκεκριμένα εδώ στον τελικό νευρώνα, όπου γίνεται η ίδια διαδικασία υπολογισμού με αποτέλεσμα αυτός να υπολογίσει την τελική έξοδό του ως $A_3 = f(Z_3)$.

Σε αυτό το σημείο οι τελικές τιμές ή προβλέψεις A_3 λαμβάνονται υπόψιν από την διαδικασία εκπαίδευσης του μοντέλου και υπολογίζεται η συνάρτηση κόστους που απεικονίζει το πόσο απέχει ακόμα το μοντέλο από το να προβλέπει καλά τα δεδομένα που επεξεργάστηκε. Ο υπολογισμός της συνάρτησης κόστους μέσω της τεχνικής του μέσου τετραγωνικού σφάλματος μπορεί να αναπαρασταθεί μαθηματικά ως εξής:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N} (Predicted_i - Actual_i)^2}{N}} \quad (1.5)$$

Όπου RMSE το αγγλικό ακρωνύμιο της συνάρτηση κόστους, Predicted οι τελικές τιμές που προβλέφθηκαν από το νευρωνικό δίκτυο, Actual οι αληθινές τιμές των δεδομένων που έπρεπε να προβλεφθούν και N όλες οι προβλέψεις που έγιναν [19].

- **Οπισθοδιάδοση:** Η οπισθοδιάδοση ξεκινά με το πέρας της εμπρόσθιας διάδοσης η οποία υπολόγισε την συνάρτηση κόστους. Η οπισθοδιάδοση έχει ένα σκοπό, ο οποίος είναι να μειώσει το λάθος (error) της συνάρτησης κόστους όσο το δυνατόν είναι εφικτό. Αυτό πρακτικά πραγματοποιείται πηγαίνοντας προς τα πίσω από το σημείο που σταμάτησε η εμπρόσθια διάδοση και σταματώντας σε κάθε επίπεδο με νευρώνες μεταβάλλοντας τις τιμές των βαρών και των πολώσεων τους.

  Σε αυτή τη διαδικασία βοηθάει ο αλγόριθμος της καθόδου βασισμένη στην κλίση όπου είναι υπεύθυνη για τον υπολογισμό των νέων τιμών σε κάθε επίπεδο. Έτσι για παράδειγμα για το τέταρτο επίπεδο, που στη συγκεκριμένη περίπτωση είναι ο νευρώνας y, θα υπολογιστούν όλες οι νέες τιμές πόλωσης για κάθε νευρώνα που ανήκει σε αυτό το επίπεδο καθώς και όλα τα νέα βάρη των συνάψεων που τους συνδέουν με το προηγούμενο επίπεδο. Η μαθηματική αναπαράσταση του υπολογισμού για τα βάρη είναι $W_{3νέο} = W_3 – ρυθ\_εκπ*(dcost/dW_3)$ ενώ για τις πολώσεις $B_{3νέο} = B_3 – ρυθ\_εκπ*(dcost/dB_3)$ όπου $W_{3νέο}$, $B_{3νέο}$ οι νέες τιμές βαρών και πολώσεων αντίστοιχα, ρυθ_εκπ η υπερπαράμετρος ρυθμός εκπαίδευσης (το βήμα που θα αλλάξουν οι τιμές), $dcost/dW_3$ η παράγωγος του κόστους που υπολογίστηκε ως προς τα βάρη και $dcost/dB_3$ η παράγωγος του κόστους ως προς τις πολώσεις. Οι παραπάνω παράγωγοι είναι ουσιαστικά οι κλίσεις της συναρτήσεως κόστους.

Έτσι η διαδικασία της οπισθοδιάδοσης ολοκληρώνεται και η εκπαίδευση του νευρωνικού δικτύου συνεχίζεται προχωρώντας στην επόμενη επανάληψη/ εποχή. Όσες περισσότερες εποχές εκπαιδευτεί το δίκτυο τόσο πιο μικρή θα είναι η τιμή της συνάρτησης κόστους στο τέλος [20].

1.2.7 Υπερπροσαρμογή

Η υπερπροσαρμογή (overfitting) ενός μοντέλου νευρωνικού δικτύου είναι η κατάσταση στην οποία το μοντέλο έχει μάθει πάρα πολύ καλά τα δεδομένα που του δόθηκαν κατά την εκπαίδευσή του με αποτέλεσμα να μην ανταποκρίνεται καλά σε δεδομένα που δεν έχει ξαναδεχτεί. Αυτό σημαίνει πως το μοντέλο αυτό δεν έχει τόσο καλή ικανότητα γενίκευσης (generalization abillity) ώστε να μπορεί να ανιχνεύσει δεδομένα με παρόμοια χαρακτηριστικά αλλά αντιθέτως θυμάται τέλεια τα δεδομένα με τα οποία εκπαιδεύτηκε. Προφανώς η κατάσταση στην οποία η ικανότητα μνήμης ενός μοντέλου είναι πολύ καλή αλλά η ικανότητα γενίκευσης κακή δεν ευνοεί καθόλου αφού συνήθως εκπαιδεύουμε μοντέλα για να αναγνωρίζουν καινούργια στοιχεία.

Η ποσότητα των δεδομένων που προορίζονται για την εκπαίδευση ενός μοντέλου δεν πρέπει να επιλέγεται αυθαίρετα. Ένα μοντέλο χρειάζεται έναν συγκεκριμένο αριθμό δεδομένων για να μπορέσει να επιτύχει το στόχο του και να προβλέπει με ακρίβεια και ελάχιστα σφάλματα, νέα δεδομένα. Εάν όμως εκπαιδευτεί με τον λάθος αριθμό δεδομένων τότε κινδυνεύει από υπερπροσαρμογή.

Επίσης πρέπει να λαμβάνεται υπόψιν το πόσο βαθύ, δηλαδή μεγάλο, είναι το νευρωνικό δίκτυο και συγκεκριμένα τα κρυφά επίπεδά του διότι συνήθως όσο μεγαλύτερο είναι τόσο περισσότερα δεδομένα εκπαίδευσης χρειάζονται για να μπορεί να φτάσει σε ένα βέλτιστο σημείο εκπαίδευσης. Παρόλα αυτά δεν υπάρχει για όλα τα νευρωνικά δίκτυα μία σίγουρη απάντηση για την ποσότητα των δεδομένων και για αυτό γίνονται διάφορες και πολλές δοκιμές εκπαίδευσης πριν εντοπιστεί το ιδανικότερο μοντέλο. Ένα παράδειγμα μοντέλου υπερπροσαρμογής είναι να εμφανίζει μετά την εκπαίδευσή του 99% ακρίβεια στα δεδομένα εκπαίδευσης και 60% ακρίβεια στα δεδομένα επικύρωσης [21].

1.2.8 Υποπροσαρμογή

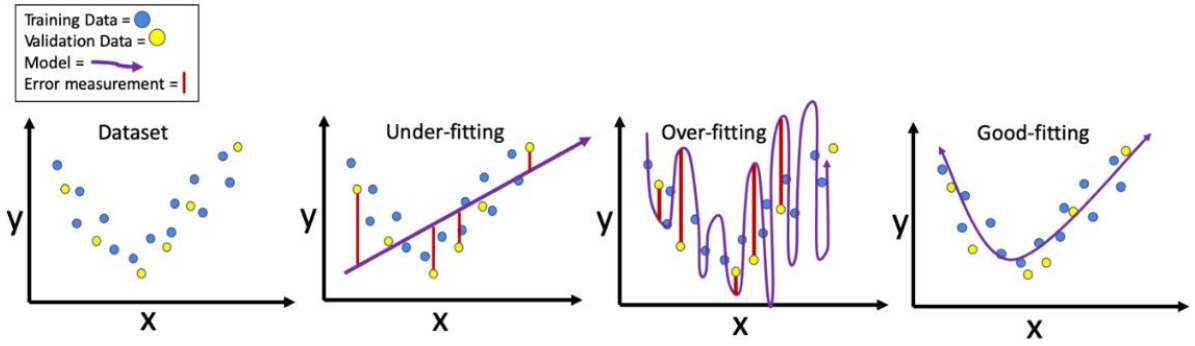
Η κατάσταση της υποπροσαρμογής (underfitting) είναι η ακριβώς αντίθετη κατάσταση από την υπερπροσαρμογή. Η υποπροσαρμογή έχει τη συνέπεια ένα μοντέλο να μην μπορεί να αναγνωρίσει σωστά ούτε νέες εικόνες αλλά πολύ πιθανόν ούτε και τις εικόνες που εκπαιδεύτηκε. Αυτό συμβαίνει διότι το μοντέλο εκπαιδεύτηκε με λιγότερα δεδομένα από ότι έπρεπε ή δεν εκπαιδεύτηκε για πολλές επαναλήψεις/εποχές.

Επίσης είναι σημαντικό το πόσο μεγάλο είναι το νευρωνικό δίκτυο, διότι εάν είναι απλό με ελάχιστα επίπεδα τότε πολύ πιθανόν να υπάρξει υποπροσαρμογή αφού δεν θα μπορούν να αναλυθούν αρκετά χαρακτηριστικά των δεδομένων. Οπότε σε αυτήν την περίπτωση η λύση είναι η προσθήκη μερικών επιπέδων παραπάνω μήπως και διορθωθεί το πρόβλημα.

Ένα παράδειγμα μοντέλου υποπροσαρμογής είναι να εμφανίζει μετά την εκπαίδευσή του 50% ακρίβεια στα δεδομένα εκπαίδευσης και 80% ακρίβεια στα δεδομένα επικύρωσης [21]. Αυτό σημαίνει πως σε νέα δεδομένα το μοντέλο κάνει ακριβής προβλέψεις αλλά λανθασμένες.

Στην εικόνα 1.17 παρουσιάζονται από αριστερά προς τα δεξιά τέσσερα γραφήματα. Το πρώτο γράφημα περιλαμβάνει κάποια δεδομένα εκπαίδευσης μπλε χρώματος και δεδομένα επικύρωσης

κίτρινου χρώματος. Στο δεύτερο γράφημα απεικονίζεται ένα παράδειγμα υποπροσαρμογής με βάση τα δεδομένα του πρώτου γραφήματος. Αντίστοιχα στο τρίτο γράφημα παρουσιάζεται ένα παράδειγμα υπερπροσαρμογής και τέλος στο τέταρτο ένα ιδανικό μοντέλο σωστής προσαρμογής. Τα μωβ βελάκια αναπαριστούν το μοντέλο κάθε περίπτωσης ενώ οι κόκκινες κάθετοι από τα μοντέλα μέχρι κάποια δεδομένα επικύρωσης αποτυπώνουν το σφάλμα.



Εικόνα 1.17 Δεδομένα εκπαίδευσης και επικύρωσης, Υποπροσαρμογή, Υπερπροσαρμογή, σωστή προσαρμογή μοντέλου [22]

1.2.9 Μέθοδοι Τακτοποίησης

Οι μέθοδοι τακτοποίησης (regularization methods) αποσκοπούν στην εξάλειψη του προβλήματος της υπερπροσαρμογής των μοντέλων, προσπαθώντας να βελτιώσουν την ικανότητα γενίκευσής τους. Μερικές από τις πιο γνωστές μεθόδους αυτές, αναλύονται στις παρακάτω ενότητες [21], [22].

1.2.9.1 Λάσο

Η μέθοδος Λάσο (Lasso/L1) επεξεργάζεται το μαθηματικό κομμάτι της συνάρτησης κόστους που σχετίζεται άμεσα με τα απόλυτα μεγέθη των βαρών του μοντέλου. Αυτό έχει ως συνέπεια την αραίωση, καταφέρνοντας έτσι ορισμένα βάρη να μετατρέψουν την τιμή τους σε μηδενική. Το αποτέλεσμα είναι το μοντέλο να επιλέγει και να διατηρεί, κατά την εκπαίδευση του, μόνο τα πιο σημαντικά χαρακτηριστικά από τα δεδομένα εκπαίδευσης.

1.2.9.2 Κορυφή

Η μέθοδος Κορυφή (Ridge/L2) επεξεργάζεται και αυτή, όπως η μέθοδος Λάσο, ένα μαθηματικό κομμάτι της συνάρτησης κόστους που σχετίζεται με τις τετραγωνικές τιμές των βαρών του μοντέλου. Αυτό οδηγεί τα βάρη να έχουν μικρές τιμές και να είναι ομοιόμορφα κατανεμημένα, μειώνοντας με αυτόν τον τρόπο το αντίκτυπο που έχουν τα μεμονωμένα βάρη και αποφεύγοντας την υπερβολική ενασχόλησή τους με κάποιο μεμονωμένο χαρακτηριστικό των δεδομένων εκπαίδευσης.

1.2.9.3 Εγκατάλειψη

Η μέθοδος της Εγκατάλειψης (Dropout) όπως αναφέρθηκε και στην ενότητα 1.2.3 είναι ειδικά σχεδιασμένη για τα νευρωνικά δίκτυα. Αυτό που κάνει είναι να εγκαταλείπει/απενεργοποιεί τυχαία ένα ποσοστό των νευρώνων κατά τη διάρκεια κάθε επανάληψης εκπαίδευσης, δημιουργώντας ουσιαστικά μία συλλογή από μικρότερα υποδίκτυα. Αυτή η τεχνική βοηθά στην αποφυγή της υπερπροσαρμογής μειώνοντας την αλληλεξάρτηση μεταξύ των νευρώνων και οδηγώντας έτσι σε ένα πιο ισχυρό και γενικευμένο μοντέλο.

1.2.9.4 Πρόωρο σταμάτημα

Το πρόωρο σταμάτημα (Early stopping) **ε**ίναι μία μέθοδος κατά την οποία η εκπαίδευση του μοντέλου σταματάει αυτόματα όταν αυτό έχει καταλήξει πολύ κοντά στο ολικό ελάχιστο της συνάρτησης κόστους. Αυτός ο έλεγχος γίνεται με βάση τα δεδομένα επικύρωσης όπου το σφάλμα στη συνάρτηση κόστους ξεκινά να αυξάνεται αντί να μειώνεται ή να παραμένει χαμηλό, τότε η εκπαίδευση σταματά.

1.2.9.5 Αύξηση δεδομένων

Η αύξηση δεδομένων (Data augmentation) είναι μία μέθοδος που επεκτείνει τα δεδομένα εκπαίδευσης επεξεργάζοντας και παραμορφώνοντας με ποικίλους τρόπους το ήδη υπάρχον σύνολο δεδομένων. Με τη δημιουργία πρόσθετων παραλλαγών των δεδομένων, το σετ με τα δεδομένα εκπαίδευσης διευρύνεται, γεγονός που βοηθά το μοντέλο να βελτιώσει την ικανότητα γενίκευσής του. Περισσότερη ανάλυση για αυτήν την μέθοδο γίνεται στην ενότητα 1.3.2.

1.2.9.6 Κανονικοποίηση υποσυνόλου δεδομένων

Η μέθοδος της κανονικοποίησης υποσυνόλου δεδομένων (Batch normalization) χρησιμοποιείται στα νευρωνικά δίκτυα για την κανονικοποίηση των εισόδων κάθε επιπέδου τους σε ένα μικρό υποσύνολο κατά τη διάρκεια της εκπαιδευτικής διαδικασίας. Αυτή η κανονικοποίηση ενισχύει τη σταθερότητα της εκπαίδευσης και βελτιώνει την ικανότητα γενίκευσης του μοντέλου επιταχύνοντας την ελαχιστοποίηση της συνάρτησης κόστους.

1.2.9.7 Μεταφορά μάθησης

Όπως αναφέρθηκε και στην ενότητα 1.2.3, η μεταφορά μάθησης (Transfer learning) είναι μια μέθοδος τακτοποίησης που χρησιμοποιείται στη μηχανική μάθηση για τη βελτίωση της απόδοσης ενός μοντέλου. Αυτό επιτυγχάνεται αξιοποιώντας τις γνώσεις που αποκτήθηκαν κατά την εκπαίδευση ενός μοντέλου και μεταφέροντάς αυτές για την εκπαίδευση ενός άλλου μοντέλου. Αντί δηλαδή να εκπαιδευτεί ένα νέο μοντέλο από την αρχή, η μεταφορά μάθησης χρησιμοποιεί προεκπαιδευμένα μοντέλα που έχουν εκπαιδευτεί σε μεγάλα και διαφορετικά σύνολα δεδομένων. Έτσι μεταφέροντας τις γνώσεις από το προεκπαιδευμένο μοντέλο, η μεταφορά μάθησης επιτρέπει στο νέο μοντέλο να χρησιμοποιεί την ικανότητα γενίκευσής του πιο αποτελεσματικά και να ξεπερνά τους περιορισμούς που μπορεί να συναντήσει στα δεδομένα εκπαίδευσης.

1.3 Εκπαίδευση ενός μοντέλου μηχανικής μάθησης

Τα μοντέλα μηχανικής μάθησης (machine learning models) είναι προγράμματα υπολογιστών που χρησιμοποιούνται για την αναγνώριση προτύπων σε δεδομένα ή την πρόβλεψη αποτελεσμάτων.
Αυτά τα μοντέλα δημιουργούνται μέσω της εφαρμογής αλγορίθμων μηχανικής μάθησης, οι οποίοι εκπαιδεύονται χρησιμοποιώντας δεδομένα με ετικέτες, χωρίς ετικέτες ή μικτά δεδομένα. Διάφοροι αλγόριθμοι μηχανικής μάθησης είναι κατάλληλοι για διαφορετικούς σκοπούς, όπως η ταξινόμηση ή η πρόβλεψη. Έτσι, οι επιστήμονες δεδομένων (data scientists) επιλέγουν διάφορους αλγορίθμους ως βάση για τα μοντέλα τους και καθώς τα δεδομένα εισέρχονται στους αλγορίθμους αυτούς, τροποποιούνται και προσαρμόζονται για να αντιμετωπίσουν αποτελεσματικά συγκεκριμένες εργασίες, μετατρέποντας αυτά τελικά σε πλήρη μοντέλα μηχανικής μάθησης [23].

1.3.1 Υπερπαράμετροι ενός μοντέλου και ο αντίκτυπός τους στην απόδοσή του

Πριν πραγματοποιηθεί η εκπαίδευση οποιουδήποτε μοντέλου, πρέπει να ρυθμιστούν κάποιες ειδικές παράμετροι που ονομάζονται υπερπαράμετροι (hyperparameters). Αυτές είναι υπεύθυνες για το πως θα εκπαιδευτεί το μοντέλο και ποιες θα είναι οι τελικές τιμές των παραμέτρων που θα εκπαιδευτεί όπως αυτές των βαρών (weights) και των πολώσεων (biases). Οι υπερπαράμετροι λειτουργούν ως οδηγοί για την τελική διαμόρφωση των παραμέτρων οι οποίες μετά το πέρας της εκπαίδευσης ονομάζονται παράμετροι μοντέλου. Με τη ρύθμιση διαφόρων υπερπαραμέτρων μπορούμε να βελτιώσουμε σημαντικά την απόδοση ενός CNN μοντέλου.

Στις ενότητες που ακολουθούν αναλύονται οι τρεις βασικότερες υπερπαράμετροι που χρησιμοποιήθηκαν για την εκπαίδευση του μοντέλου αυτής της διπλωματικής εργασίας.

1.3.1.1 Εποχές

Στον τομέα της μηχανικής μάθησης, μια εποχή (epoch) αντιπροσωπεύει μια πλήρη επανάληψη σε ένα σύνολο δεδομένων κατά τη διάρκεια της διαδικασίας εκπαίδευσης ενός μοντέλου. Σε κάθε εποχή, το μοντέλο εκτίθεται σε ολόκληρο το σύνολο δεδομένων εκπαίδευσης και τα βάρη και οι πολώσεις του προσαρμόζονται για να ελαχιστοποιηθεί το σφάλμα των δεδομένων εκπαίδευσης. Συνήθως, η εκπαίδευση ενός μοντέλου περιλαμβάνει πολλαπλές εποχές, με κάθε εποχή να συμβάλλει στην ενίσχυση της ακρίβειας του μοντέλου. Στη βαθιά μάθηση, τα μοντέλα μπορούν να εκπαιδευτούν για αριθμό πολλών εποχών, οι οποίες μπορεί να απαιτούν σημαντικό χρόνο, ιδιαίτερα για μοντέλα με μεγάλο αριθμό παραμέτρων.

Η επιλογή του κατάλληλου αριθμού εποχών είναι μία κρίσιμη υπερπαράμετρος που απαιτεί προσεκτική εξέταση. Πολύ λίγες εποχές μπορεί να οδηγήσουν σε ένα υποεκπαιδευμένο μοντέλο που δεν αποδίδει τη βέλτιστη απόδοση, ενώ από την άλλη η επιλογή πάρα πολλών εποχών μπορεί να οδηγήσει σε υπερπροσαρμογή (overfitting). Η υπερπροσαρμογή είναι μια κατάσταση στην οποία το μοντέλο εξειδικεύεται πολύ στα δεδομένα εκπαίδευσης και αποτυγχάνει να αναγνωρίσει με επιτυχία νέα δεδομένα τα οποία δεν έχει ξαναδεί ποτέ.

Η έννοια των εποχών είναι θεμελιώδης στην εκπαίδευση μοντέλων μηχανικής μάθησης και επηρεάζει σημαντικά τη συνολική απόδοση του μοντέλου. Η σωστή επιλογή του αριθμού των

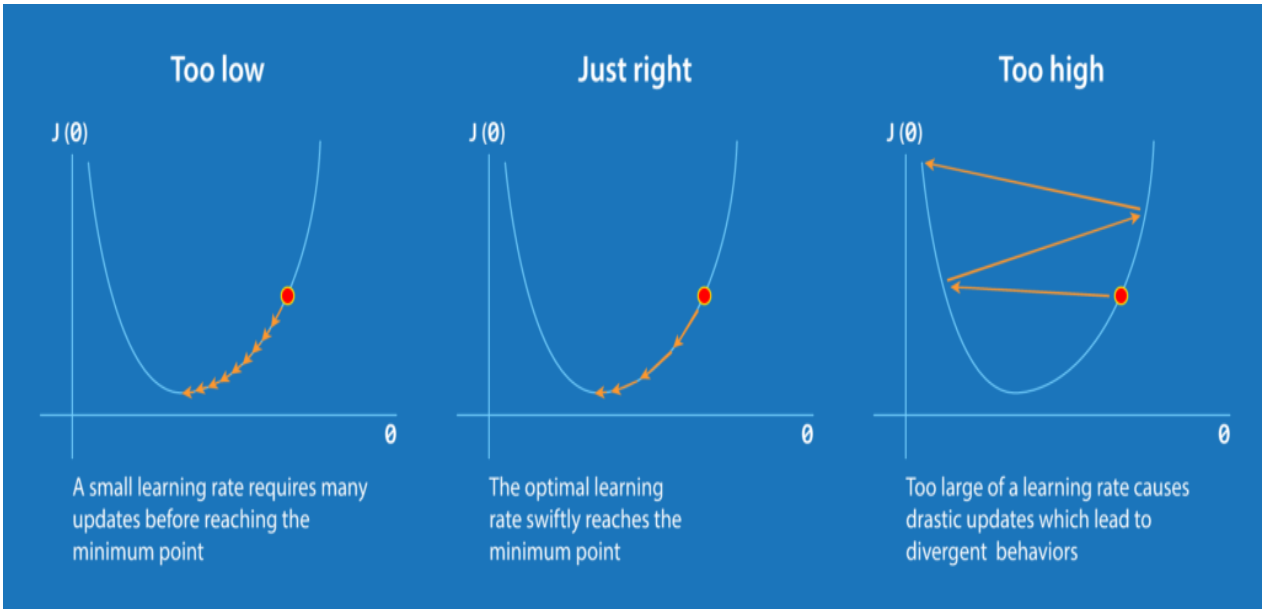
εποχών, σε συνδυασμό με άλλες υπερπαραμέτρους, επηρεάζει σε μεγάλο βαθμό την επιτυχία ενός έργου μηχανικής μάθησης [24].

1.3.1.2 Ρυθμός εκπαίδευσης

Η έννοια του ρυθμού εκπαίδευσης (learning rate) χρησιμοποιείται συνήθως στους τομείς της μηχανικής μάθησης και της στατιστικής. Αναφέρεται στην ταχύτητα με την οποία ένας αλγόριθμος προχωρά προς την εύρεση μιας λύσης. Στο πλαίσιο της εκπαίδευσης των νευρωνικών δικτύων, ο ρυθμός μάθησης έχει σημαντική σημασία ως υπερπαράμετρος.

Όταν εκπαιδεύουμε ένα μοντέλο νευρωνικών δικτύων, συχνά χρησιμοποιούνται τεχνικές βελτιστοποίησης που βασίζονται στην κάθοδο βασισμένη στην κλίση. Αυτό περιλαμβάνει τον υπολογισμό της κλίσης της συνάρτησης κόστους σε σχέση με τα βάρη του μοντέλου, η οποία παρέχει την κατεύθυνση προς το ολικό ελάχιστο. Για να καθοδηγήσουμε τις προσαρμογές βάρους του μοντέλου προς αυτή την κατεύθυνση και να επιτύχουμε βελτιστοποίηση, χρησιμοποιούμε την υπερπαράμετρο του ρυθμού εκπαίδευσης. Ο ρυθμός εκπαίδευσης καθορίζει το μέγεθος των βημάτων που γίνονται από την κλίση της καθόδου προς το τοπικό ελάχιστο. Εάν αυτός ρυθμιστεί πολύ χαμηλός, η διαδικασία όδευσης προς το τοπικό ελάχιστο γίνεται πιο αργή, παρατείνοντας το χρόνο που χρειάζεται για να επιτευχθεί η βέλτιστη λύση. Αντίθετα, εάν ο ρυθμός μάθησης είναι πολύ υψηλός, η κλίση της καθόδου μπορεί να πάρει λάθος κατεύθυνση, εμποδίζοντας την επίτευξη της βέλτιστης λύσης [25].

Οι πιο συχνές τιμές που παίρνει ο ρυθμός εκπαίδευσης κατά την εκπαίδευση μοντέλων στην βαθιά μάθηση είναι 0.1, 0.01 ή 0.001.



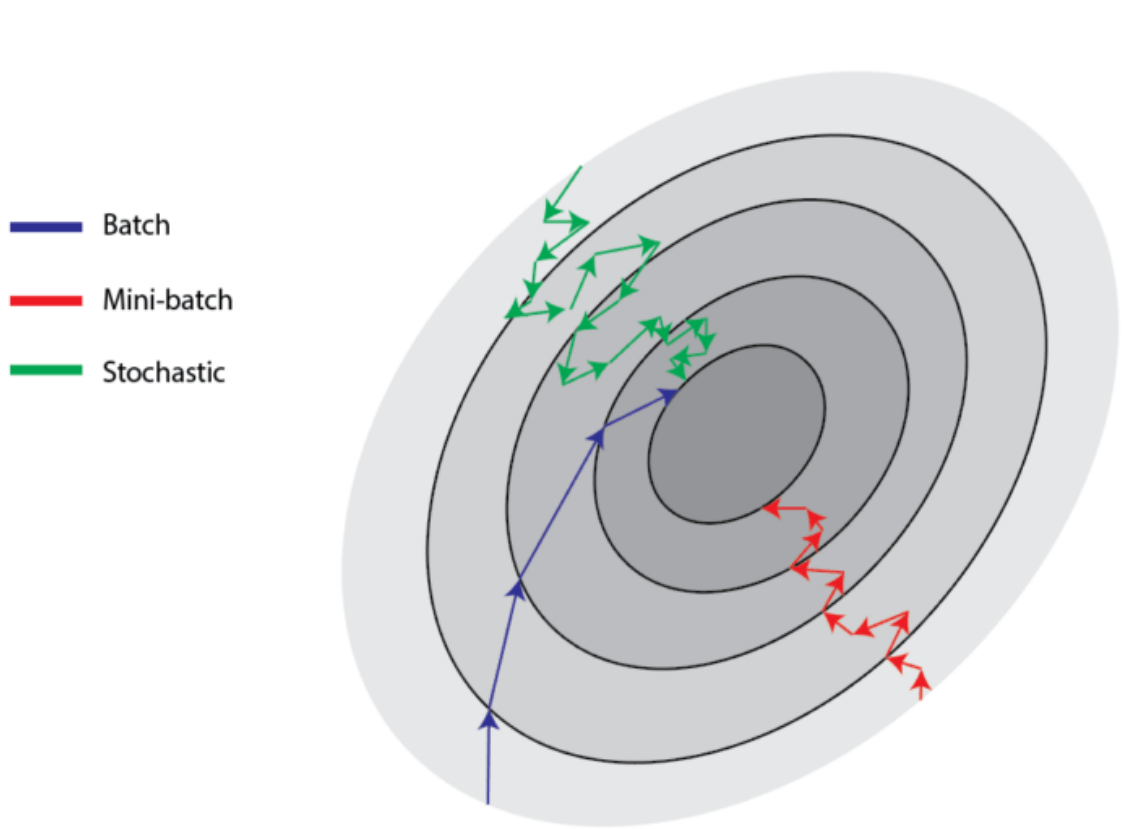
Εικόνα 1.18 Διαφορές κατά την κάθοδο της συνάρτησης κόστους, ανάλογα την τιμή του ρυθμού εκπαίδευσης. Αριστερά παρουσιάζεται η περίπτωση για μικρό ρυθμό εκπαίδευσης, στη μέση για τον ιδανικό και δεξιά για τον μεγάλο [26]

1.3.1.3 Μέγεθος υποσυνόλου δεδομένων

Για να επιταχυνθεί η διαδικασία εκπαίδευσης, συνήθως χρησιμοποιούνται μικρές ομάδες δεδομένων εκπαίδευσης. Αντί να χρησιμοποιείται ολόκληρο το σύνολο δεδομένων σε κάθε επανάληψη, χρησιμοποιούνται μικρά υποσύνολα δεδομένων (batches). Αυτή η προσέγγιση επιτρέπει πιο συχνές ενημερώσεις στο μοντέλο, επιτυγχάνοντας δυνητικά βελτιωμένη απόδοση.

Υπάρχουν τρείς διαφορετικοί τρόποι βελτιστοποίησης του μοντέλου με τη μέθοδο κάθοδος βασισμένη στη κλίση και έχουν να κάνουν με τον τρόπο που θα επιλέξουμε το μέγεθος του υποσυνόλου δεδομένων (Batch size). Ο ένας τρόπος είναι το “Batch gradient descent” που χρησιμοποιεί το συνολικό πλήθος των δεδομένων εκπαίδευσης σε κάθε εποχή για την εκπαίδευση του μοντέλου. Ο δεύτερος τρόπος ονομάζεται “Stochastic gradient descent” και χρησιμοποιεί μόνο ένα τυχαίο δεδομένο εκπαίδευσης σε κάθε εποχή κατά την εκπαίδευση. Ο τρίτος που είναι ο πιο συχνά χρησιμοποιούμενος και είναι και αυτός που χρησιμοποιήθηκε για την εκπαίδευση του μοντέλου αυτής της διπλωματικής εργασίας είναι ο “Mini-batch gradient descent”.

Το “Mini-batch gradient descent” επιφέρει τα καλύτερα αποτελέσματα σε σχέση με τους άλλους τρόπους και η διαφορά του είναι πως χρησιμοποιεί έναν συγκεκριμένο αριθμό δεδομένων εκπαίδευσης σε κάθε εποχή τον οποίο καθορίζουμε εμείς. Το μέγεθος που θα καθορίσουμε για αυτήν την υπερπαράμετρο θα επηρεάσει την εκπαίδευση του μοντέλου είτε θετικά είτε αρνητικά σε σχέση με την απόδοση του μοντέλου και τον συνολικό χρόνο εκπαίδευσης αυτού. Επίσης η τιμή που χρησιμοποιείται συνήθως είναι το 32 αλλά γενικότερα επιλέγεται ένας αριθμός της δύναμης του 2 και μεταξύ των τιμών 16 και 512 [25].



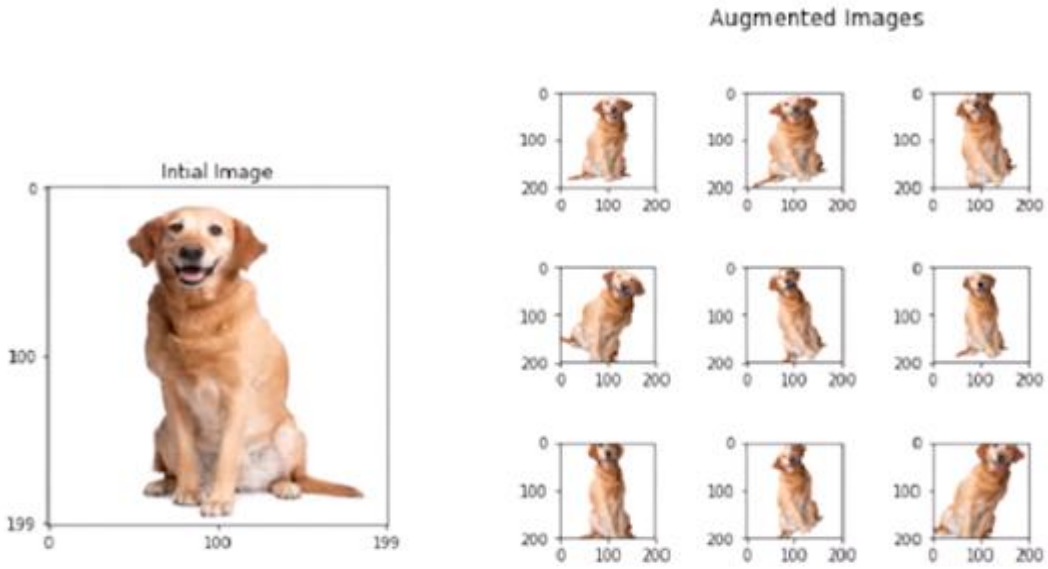
Εικόνα 1.19 Η επιρροή των διαφορετικών τρόπων επιλογής του μεγέθους υποσυνόλου δεδομένων, κατά την κάθοδο της συνάρτησης κόστους [25]

1.3.2 Αύξηση δεδομένων

Πολύ συχνά κατά την εκπαίδευση μοντέλου πραγματοποιείται αύξηση των ήδη υπαρχόντων δεδομένων (data augmentation) εκπαίδευσης σε ακόμα περισσότερα, Πιο συγκεκριμένα σε ένα CNN μοντέλο γίνεται αύξηση των εικόνων που αυτό θα εκπαιδευτεί έτσι ώστε να έχει περισσότερο υλικό να εξασκηθεί και να επιφέρει καλύτερα αποτελέσματα. Συνήθως στα μοντέλα βαθιάς μάθησης όσο περισσότερα είναι τα δεδομένα που εκπαιδεύεται ένα μοντέλο τόσο καλύτερη θα είναι η ακρίβειά του.
Στην πράξη, η αύξηση δεδομένων μπορεί να γίνει είτε πριν την εκπαίδευση ενός μοντέλου είτε κατά την εκπαίδευσή του μέσω κάποιας συνάρτησης που πληθαίνει τις εικόνες αυτόματα κάτω υπό συγκεκριμένες προϋποθέσεις. Οι προϋποθέσεις ή παράμετροι αυτές καθορίζουν τον τρόπο που θα γίνει επεξεργασία πάνω στις υπάρχουσες εικόνες για να δημιουργηθούν οι νέες. Άρα με βάση τα δεδομένα που ήδη υπάρχουν δημιουργούνται νέα δεδομένα τα οποία έχουν διαφορετικά χαρακτηριστικά.

Μερικές παράμετροι αλλαγής των νέων εικόνων είναι η μεγέθυνση ή σμίκρυνση, η περιστροφή, η αναστροφή, η μετατόπιση μέχρι ένα ποσοστό του πλάτους τις πρωτότυπης εικόνας ή αντίστοιχα του ύψους και η τυχαία διαστρέβλωση τους ως προς τον άξονα x ή y. Συνήθως υπάρχει και κάποια παράμετρος που ενεργοποιεί τον εντοπισμό των νέων ή των διαγραμμένων pixel, αυτών των νέων εικόνων, η οποία τα διορθώνει και τα χρωματίζει ανάλογα με τα γειτονικά τους. Κατά την αύξηση των εικόνων συνήθως δημιουργούνται 3-4 νέες εικόνες από κάθε πρωτότυπη [21].

Είναι σημαντικό να αναφερθεί πως κάνοντας αλλαγές στις ήδη υπάρχουσες εικόνες και δημιουργώντας νέες οι οποίες μοιάζουν σε αυτές αλλά έχουν άλλη γωνία θέασης ή είναι γυρισμένες υπό άλλη γωνία, πετυχαίνουμε καλύτερη ακρίβεια κατά την εκπαίδευση του μοντέλου. Αυτό συμβαίνει διότι το μοντέλο αναγνωρίζει τα όμοια χαρακτηριστικά των δύο εικόνων οπότε στο μέλλον θα μπορεί πιο εύκολα να καταλάβει σε νέες εικόνες αν υπάρχει κάποιο αντικείμενο με τα ίδια χαρακτηριστικά.



Εικόνα 1.20 Αριστερά διακρίνεται μια πρωτότυπη εικόνα ενός σκύλου και δεξιά οι παραλλαγές τις εικόνας αυτής που δημιουργήθηκαν από την αύξηση δεδομένων [21]

1.3.3 Δυαδική διασταυρούμενη εντροπία

Η απόδοση του μοντέλου ταξινόμησης εικόνων που δημιουργήθηκε υπολογίστηκε από την συνάρτηση δυαδικής διασταυρούμενης εντροπίας (binary cross entropy) που ανήκει στην κατηγορία των συναρτήσεων απωλειών. Αυτή η συνάρτηση αρχικά υπολογίζει όλες τις επιμέρους συναρτήσεις απωλειών και στο τέλος δημιουργεί τη συνάρτηση κόστους για όλα τα δεδομένα εκπαίδευσης. Αυτή η συνάρτηση χρησιμοποιείται συνήθως για δυαδικά μοντέλα ταξινόμησης εικόνων, όπου δηλαδή η πρόβλεψη είναι για παράδειγμα ναι-όχι, μαύρο-άσπρο, κλπ.

Αρχικά η δυαδική διασταυρούμενη εντροπία υπολογίζει για κάθε δεδομένο εκπαίδευσης την συνάρτηση απωλειών του, βρίσκοντας τη διαφορά μεταξύ της πρόβλεψης που έκανε το μοντέλο και της πραγματικής δυαδικής ετικέτας που έπρεπε να προβλέψει σωστά. Αυτή η διαφορά αντιπροσωπεύει την απώλεια (loss) η το σφάλμα (error) ως προς το συγκεκριμένο δεδομένο εκπαίδευσης, όπου όσο μεγαλύτερη είναι αυτή η τιμή τόσο μεγαλύτερη είναι η απώλεια άρα και η διαφορά της πιθανότητας πρόβλεψης από την αληθινή ετικέτα.

Αφού λοιπόν υπολογιστούν όλες αυτές οι επιμέρους διαφορές για όλα τα δεδομένα εκπαίδευσης, αθροίζονται και έπειτα υπολογίζεται ο μέσος όρος τους. Το τελικό αποτέλεσμα είναι το κόστος (cost) του μοντέλου τη συγκεκριμένη επανάληψη εκπαίδευσης και δείχνει τον μέσο όρο της διαφοράς των προβλέψεων από τις πραγματικές ετικέτες όλων των δεδομένων εκπαίδευσης που εξετάστηκαν [27].

Ο μαθηματικός τύπος για τον υπολογισμό της συνάρτησης απώλειας ενός μεμονωμένου δυαδικού δεδομένου εκπαίδευσης είναι ο εξής:

$$\text{Απώλεια} = y * \log(p) - (1 - y) * \log(1 - p) \qquad (1.5)$$

Όπου y είναι η πραγματική τιμή της ετικέτας δηλαδή 0 ή 1 και p η πιθανότητα της πρόβλεψης που έγινε με τιμή από 0 έως 1.

Όσο πιο κοντά είναι η πρόβλεψη στην πραγματική ετικέτα, η τιμή της απώλειας πλησιάζει το μηδέν, δηλαδή το τοπικό ελάχιστο της συνάρτησης άρα και τον στόχο για την βελτιστοποίηση του μοντέλου.

Αφού γίνει ο υπολογισμός όλων των επιμέρους τιμών των απωλειών , μένει το τελικό βήμα που είναι η άθροισή τους και ο υπολογισμός του μέσου όρου τους που αντιστοιχεί στο κόστος και έχει την εξής μαθηματική μορφή:

$$\text{Κόστος} = -(1/N) * \sum[y * \log(p) + (1 - y) * \log(1 - p)] \qquad (1.6)$$

Όπου N το σύνολο των δεδομένων εκπαίδευσης και Σ ο αθροιστής όλων των απωλειών που υπολογίζονται από τον τύπο της δυαδικής διασταυρούμενης εντροπίας (1.5). Το αρνητικό πρόσημο στην αρχή της μαθηματικής έκφρασης είναι πολύ σημαντικό διότι συμβολίζει πως θέλουμε να μειώσουμε το κόστος του μοντέλου από την προηγούμενη τιμή κόστους που υπολογίστηκε και όχι να το αυξήσουμε.

1.3.4 Βελτιστοποιητής προσαρμοστικής εκτίμησης ροπής Adam

Για την βελτιστοποίηση του μοντέλου μας χρησιμοποιήθηκε ο αλγόριθμος προσαρμοστικής εκτίμησης ροπών ή αλλιώς Adam (adaptive moment estimation). Ο συγκεκριμένος αλγόριθμος λειτουργεί με τη λογική της καθόδου βασισμένη στην κλίση και συνδυάζει πλεονεκτήματα από τους αλγορίθμους της προσαρμοστικής κλίσης (Adaptive Gradient Algorithm ή AdaGrad) και της μέσης τετραγωνικής διάδοσης ρίζας (Root Mean Square Propagation ή RMSProp).

Ο αλγόριθμος AdaGrad έχει σχεδιαστεί για να βελτιώνει την απόδοση σε μοντέλα με αραιές κλίσεις, όπως τα προβλήματα όρασης υπολογιστή (computer vision). Αυτό το επιτυγχάνει διατηρώντας μια συγκεκριμένη τιμή ρυθμού εκπαίδευσης για κάθε παράμετρο του μοντέλου, επιτρέποντάς του να χειρίζεται αποτελεσματικά περιπτώσεις στις οποίες οι κλίσεις είναι αραιές.

Από την άλλη πλευρά, ο αλγόριθμος RMSProp προσαρμόζει επίσης τους ρυθμούς εκπαίδευσης ανά παράμετρο, αλλά σε αυτή την περίπτωση, η προσαρμογή βασίζεται στα μέσα μεγέθη των πρόσφατων κλίσεων για κάθε βάρος. Αυτή η προσέγγιση καθιστά αυτόν τον αλγόριθμο κατάλληλο για διαδικτυακά και μη σταθερά προβλήματα, όπως αυτά που περιλαμβάνουν θόρυβο και μεταβαλλόμενα περιβάλλοντα.

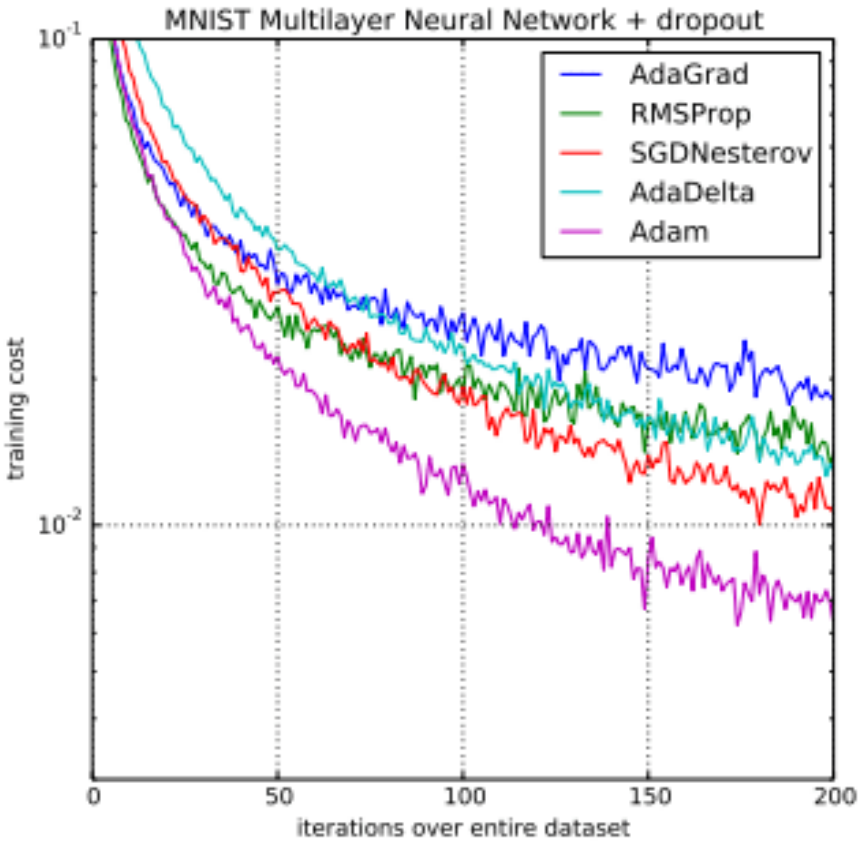
Όσον αφορά τον βελτιστοποιητή Adam, αυτός είναι υπεύθυνος για τη διατήρηση ενός συγκεκριμένου ρυθμού εκπαίδευσης για κάθε παράμετρο σε ένα νευρωνικό δίκτυο και την ενημέρωση των παραμέτρων αυτών με βάση τον μέσο όρο των προηγούμενων κλίσεων και των τετραγωνικών κλίσεων. Αυτή η προσαρμοστική προσέγγιση επιτρέπει την πιο αποτελεσματική εκπαίδευση ενός μοντέλου.

Για να εκτελεστεί ο αλγόριθμος Adam πραγματοποιούνται κάποια βήματα πριν έρθει η σειρά του κατά την εκπαίδευση ενός μοντέλου. Αρχικά τα διανύσματα της πρώτης (first moment) και δεύτερης στιγμής (second moment) αρχικοποιούνται με τιμή μηδέν και έχουν το ίδιο μέγεθος με το σύνολο των παραμέτρων του μοντέλου. Ως πρώτη στιγμή θεωρείται ένα διάνυσμα που αποθηκεύει τον μέσο όρο όλων των κλίσεων που υπολογίστηκαν για κάθε παράμετρο του δικτύου μέχρι την κάθε επανάληψη εκπαίδευσης. Ως δεύτερη στιγμή ορίζεται το διάνυσμα που περιέχει τα τετράγωνα αυτών των μέσων όρων των κλίσεων.

Στη συνέχεια γίνεται υπολογισμός των κλίσεων των παραμέτρων του νευρωνικού δικτύου, οι οποίες υπολογίζονται χρησιμοποιώντας τη μέθοδο της οπισθοδιάδοσης. Έτσι γίνεται ενημέρωση του διανύσματος της πρώτης στιγμής λαμβάνοντας τον μέσο όρο της τρέχουσας κλίσης των παραμέτρων και της τιμής της προηγούμενης πρώτης στιγμής, χρησιμοποιώντας μια παράμετρο που ονομάζεται beta1 και συνήθως παίρνει την τιμή 0.9. Σε αυτό το σημείο το μοντέλο γνωρίζει την γενική κατεύθυνση των κλίσεων.

Αμέσως μετά την ενημέρωση της πρώτης στιγμής γίνεται ενημέρωση της δεύτερης στιγμής. Η δεύτερη στιγμή ενημερώνεται υπολογίζοντας τον μέσο όρο από τις τετραγωνικές κλίσεις και την τιμή της προηγούμενης δεύτερης στιγμής, χρησιμοποιώντας την παράμετρο beta2 που συχνότερα ορίζεται με την τιμή 0.999. Η δεύτερη στιγμή παρακολουθεί τη διακύμανση ή την ομαλότητα των κλίσεων.

Έπειτα γίνεται διόρθωση των τιμών των πολώσεων του δικτύου και των παραμέτρων του. Οι παράμετροι ενημερώνονται χρησιμοποιώντας την υπολογισμένη πρώτη και δεύτερη στιγμή, μαζί με έναν παράγοντα του ρυθμού εκπαίδευσης. Ο ρυθμός εκπαίδευσης για κάθε παράμετρο αλλάζει ανάλογα με την εκτιμώμενη διακύμανση των κλίσεων. Λαμβάνοντας υπόψη την πρώτη και τη δεύτερη στιγμή των κλίσεων, ο Adam προσαρμόζει τον ρυθμό εκπαίδευσης για κάθε παράμετρο ξεχωριστά, οδηγώντας έτσι στην ταχύτερη μείωση του κόστους και στον καλύτερο χειρισμό των αραιών κλίσεων [28].



Εικόνα 1.21 Σύγκριση του αλγορίθμου βελτιστοποίησης Adam με άλλους αλγορίθμους κατά την εκπαίδευση ενός νευρωνικού δικτύου [28]

1.3.5 Η αρχιτεκτονική του μοντέλου: MobileNetV2

Το μοντέλο αυτής της διπλωματικής εργασίας σχεδιάστηκε χρησιμοποιώντας τη μέθοδο της μεταφοράς μάθησης η οποία επιτρέπει την εισαγωγή γνώσεων από ένα προεκπαιδευμένο νευρωνικό δίκτυο ως βάση για το νέο μοντέλο. Αυτές οι γνώσεις είναι ουσιαστικά κάποια από τα επίπεδα που περιέχει το προεκπαιδευμένο μοντέλο μαζί με τις ήδη εκπαιδευμένες παραμέτρους του, δηλαδή τα βάρη και τις πολώσεις.

Στο μοντέλο αυτής της διπλωματικής εργασίας χρησιμοποιήθηκε ως βάση η αρχιτεκτονική του μοντέλου MobileNetV2 της Google, μια από τις πιο γνωστές αρχιτεκτονικές δικτύων βαθιάς μάθησης. Το MobileNetV2 επιλέχθηκε συγκεκριμένα διότι είναι πιο ελαφρύ σε σχέση με άλλα μοντέλα και είναι ειδικά σχεδιασμένο για εφαρμογή σε συσκευές αιχμής (edge devices) όπως είναι το raspberry pi 4 (rpi4). Λόγω της μικρότερης υπολογιστικής του δύναμης σε σχέση με έναν υπολογιστή, το rpi4 δεν θα μπορούσε να αντέξει ένα πιο βαρύ μοντέλο ειδικά για την ταξινόμηση εικόνων μέσω ζωντανής ροής βίντεο. Γενικότερα το MobileNetV2 στοχεύει να βρει μια ισορροπία μεταξύ του μεγέθους, της αποτελεσματικότητας και της ακρίβειας του μοντέλου.

Στις επόμενες δύο ενότητες περιγράφονται τα δυο μέρη του μοντέλου που δημιουργήθηκε, όπου το πρώτο έχει την αρχιτεκτονική του MobileNetV2 ενώ το δεύτερο προγραμματίστηκε από την αρχή με σκοπό την καλύτερη και πιο εξειδικευμένη ταξινόμηση των εικόνων.

1.3.5.1 BaseModel

Το baseModel είναι η μεταβλητή/ το αντικείμενο που χρησιμοποιήθηκε και στον κώδικα για την εκπαίδευση του συνελικτικού νευρωνικού δικτύου και αντιπροσωπεύει/περιέχει το κομμάτι που αποτελεί την βάση του μοντέλου. Το baseModel έχει υιοθετήσει την αρχιτεκτονική του MobileNetV2 άρα και τα επίπεδα που περιέχει αυτό μέχρι το σημείο που ξεκινάνε τα κρυμμένα επίπεδα. Επίσης μαζί με τα επίπεδα έχουν κλωνοποιηθεί και οι παράμετροί τους οι οποίες έχουν τις κατάλληλες τιμές για την σωστή επεξεργασία των δεδομένων εκπαίδευσης. Το MobileNetV2 έχει εκπαιδευτεί πάνω σε ένα τεράστιο σύνολο δεδομένων που ονομάζεται ImageNet και περιέχει πολλές διαφορετικές κλάσεις/κατηγορίες δεδομένων. Αυτό σημαίνει πως οι παράμετροί του είναι πολύ καλά προσαρμοσμένες για την επεξεργασία των δεδομένων με ακρίβεια και αποτελεσματικότητα. Ειδικότερα τα επίπεδα που χρησιμοποιούνται στο baseModel αφορούν την εισαγωγή των δεδομένων εκπαίδευσης στο δίκτυο, δηλαδή των εικόνων και την κατάλληλη επεξεργασία τους έτσι ώστε να εξαχθούν τα σημαντικά χαρακτηριστικά από αυτές [29]. Τα συνελικτικά επίπεδα που περιέχει το MobileNetV2 είναι αυτά του πίνακα 1.1.

Πίνακας 1.1 Δομή του δικτύου MobileNetV2 [30]

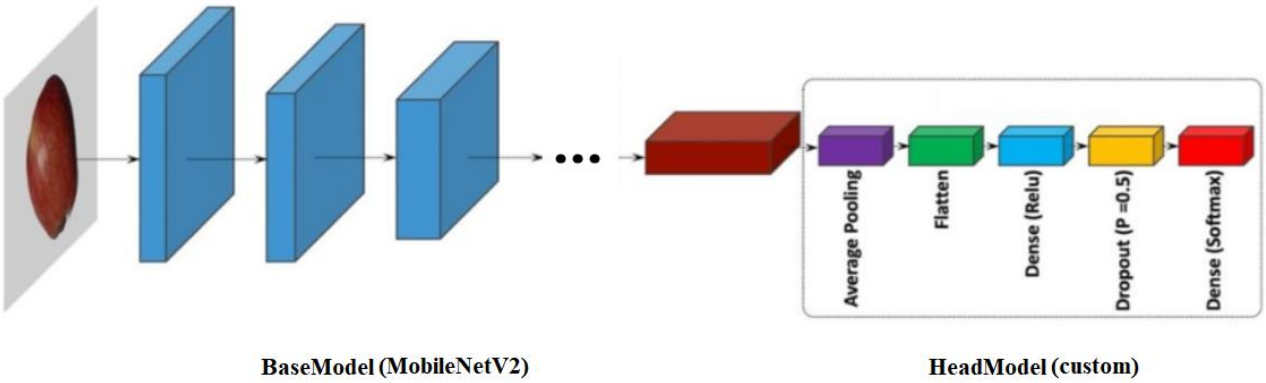
Number of Layers	Input Size	Operator and Convolution Kernel	N	S
1	224 × 224 × 3	Conv2d 3 × 3	1	2
2	112 × 112 × 32	Bottleneck 3 × 3 1 × 1	1	1
3–4	112 × 112 × 16	Bottleneck 3 × 3 1 × 1	2	2
5–7	56 × 56 × 24	Bottleneck 3 × 3 1 × 1	3	2
8–11	28 × 28 × 32	Bottleneck 3 × 3 1 × 1	4	2
12–14	14 × 14 × 64	Bottleneck 3 × 3 1 × 1	3	1
15–17	14 × 14 × 96	Bottleneck 3 × 3 1 × 1	3	2
18	7 × 7 × 160	Bottleneck 3 × 3 1 × 1	1	1
19	7 × 7 × 320	Conv2d 1 × 1	1	1
20	7 × 7 × 1280	Avgpool 7 × 7	1	-
21	1 × 1 × 1280	Conv2d 1 × 1	1	-

Στον πίνακα 1.1 κάθε γραμμή αποτελείται από ένα επίπεδο η μια ομάδα επιπέδων. Το Number of Layers δείχνει με τη σειρά τον αριθμό των επιπέδων. Το input size είναι οι διαστάσεις των δεδομένων που δέχεται κάθε επίπεδο. Η στήλη Operator and Convolution Kernel περιέχει τα ονόματα των επιπέδων ή των ομάδων των επιπέδων που χρησιμοποιούνται. Το N αντιπροσωπεύει τον αριθμό των επαναλήψεων κάθε σειράς. Το S είναι το βήμα (stride) της συνέλιξης του πρώτου επιπέδου κάθε σειράς. Όσον αφορά τα ίδια τα επίπεδα, το Conv2d είναι ένα επίπεδο συνέλιξης, το Bottleneck είναι μία ομάδα επιπέδων που αρχικά αυξάνουν τις διαστάσεις των χαρακτηριστικών και έπειτα τις μειώνουν και το Avgpool είναι το επίπεδο συγκέντρωσης [30].

1.3.5.2 HeadModel

Το headModel αποτελεί την συνέχεια του baseModel και αντικαθιστά το κομμάτι εκείνο του MobileNetV2 που είναι υπεύθυνο για την ταξινόμηση δηλαδή τα κρυφά του επίπεδα. Η αντικατάσταση του συγκεκριμένου μέρους γίνεται για να προσαρμόσουμε το μοντέλο ώστε αυτό να ταξινομεί με μεγαλύτερη ακρίβεια τα δεδομένα που εμείς θέλουμε. Το headModel είναι το μόνο κομμάτι του μοντέλου που θα υποστεί εκπαίδευση και οι παράμετροί του θα αλλάξουν μέχρι να πάρουν τις βέλτιστες τιμές σχετικά με την ταξινόμηση. Αντίθετα οι παράμετροι του baseModel θα παγώσουν κατά την εκπαίδευση και δεν θα αλλάξουν τιμές μιας και στο κομμάτι της επεξεργασίας και εξαγωγής χαρακτηριστικών αυτές οι παράμετροι είναι πολύ καλές. Πιο συγκεκριμένα το headModel θα αποκτήσει μία εξειδίκευση στο να ταξινομεί τις κατηγορίες εικόνων που θέλουμε εμείς. Το πρώτο από τα επίπεδα που δημιουργήθηκαν για την ταξινόμηση είναι το επίπεδο συγκέντρωσης με μέγεθος παραθύρου 7x7. Ακολουθεί ένα επίπεδο ισοπέδωσης το οποίο συνδέεται στη συνέχεια με ένα πλήρως συνδεδεμένο κρυφό επίπεδο 128 νευρώνων. Αυτό το κρυφό επίπεδο οδηγεί τις εξόδους του στη συνάρτηση ενεργοποίησης **“**διορθωμένη γραμμική μονάδα" και έπειτα του εφαρμόζεται η τεχνική τακτοποίησης “εγκατάλειψη” με ποσοστό τυχαίας απενεργοποίησης νευρώνων 50%. Τέλος δημιουργείται το τελευταίο επίπεδο που αποτελείται από δυο νευρώνες πλήρως συνδεδεμένους με το προηγούμενο επίπεδο και οι οποίοι είναι υπεύθυνοι για την τελική ταξινόμηση των δεδομένων. Οι έξοδοι αυτών των δυο νευρώνων περνάνε από τη συνάρτηση ενεργοποίησης Softmax και παίρνουν την τελική τους κατάλληλη μορφή.

Στην εικόνα 1.22 απεικονίζεται όλο το μοντέλο που εκπαιδεύτηκε και αποτελείται από τα δύο μέρη baseModel και headModel.



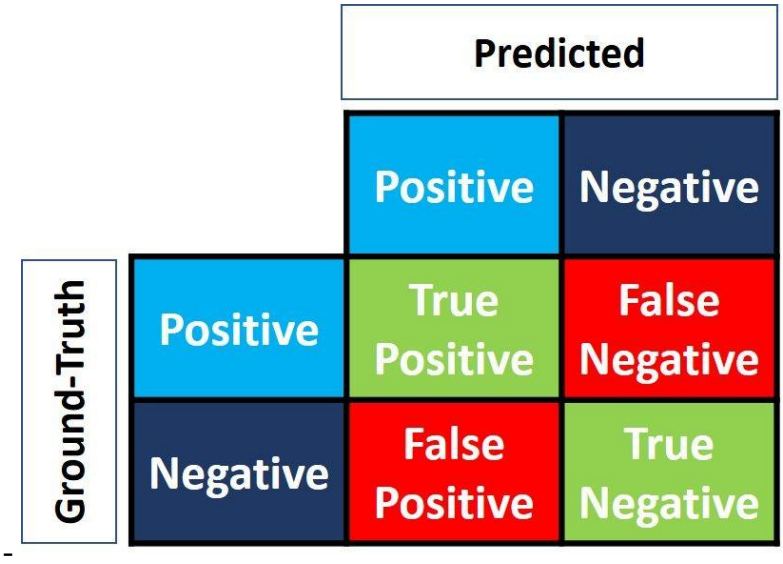
Εικόνα 1.22 Αρχιτεκτονική του μοντέλου αυτής της διπλωματικής εργασίας [31]

1.4 Αξιολόγηση ενός μοντέλου μηχανικής μάθησης

Μετά την εκπαίδευση ενός μοντέλου ακολουθεί η αξιολόγησή του από ένα διαφορετικό σύνολο δεδομένων σε σχέση με αυτό της εκπαίδευσης. Κατά την αξιολόγηση το μοντέλο δοκιμάζεται με βάση αυτό το δοκιμαστικό σύνολο δεδομένων και προκύπτουν κάποιες μετρικές (metrics) σχετικά με την απόδοση και την ακρίβεια του [32].

1.4.1 Πίνακας σύγχυσης

Για την αξιολόγηση ενός μοντέλου και τον υπολογισμό των διαφόρων μετρικών του, πρώτα πρέπει να υπολογιστούν κάποιες άλλες επιμέρους τιμές ενός πίνακα που ονομάζεται πίνακας σύγχυσης (confusion matrix). Αυτός ο πίνακας περιέχει σημαντικές πληροφορίες σχετικά με τα αποτελέσματα εκπαίδευσης ενός μοντέλου αφού δείχνει το πόσο αποτελεσματικά ταξινομεί το μοντέλο τα δεδομένα. Στον πίνακα παρουσιάζονται οι σχέσεις που έχουν οι προβλέψεις που έγιναν κατά την εκπαίδευση με τις αληθινές ετικέτες που έπρεπε να προβλεφθούν. Ένα πίνακας σύγχυσης έχει τη μορφή της εικόνας 1.23.



Εικόνα 1.23 Πίνακας σύγχυσης [33]

Οι σειρές που βρίσκονται δεξιά από τον τίτλο “Ground truth” αντιπροσωπεύουν τις αληθινές ετικέτες των δεδομένων ενώ οι στήλες που βρίσκονται κάτω από τον τίτλο “Predicted” έχουν να κάνουν με τις ίδιες τις προβλέψεις. Πρέπει να διευκρινιστεί πως μιλώντας για μια δυαδική ταξινόμηση δηλαδή για ταξινόμηση με ετικέτες π.χ. αν φοράει κάποιος μάσκα ή όχι δεν φοράει, ορίζουμε ως θετική την κατάσταση του να φοράει κάποιος μάσκα. Ανάλογα λοιπόν με τα αποτελέσματα των προβλέψεων σε σχέση με τις αληθινές ετικέτες δημιουργούνται τα παρακάτω:

- **Σωστά θετικά (True Positives/TP):** Αυτή η μέτρηση δείχνει ότι το μοντέλο έκανε μια *θετική πρόβλεψη* και η αληθινή ετικέτα που έπρεπε να προβλεφθεί ήταν και αυτή *θετική*. Άρα π.χ. προβλέφθηκε ότι κάποιος φοράει μάσκα και επιβεβαιώθηκε πως όντως φοράει μάσκα.
- **Λάθος θετικά (False Positives/FP):** Αυτή η μέτρηση δείχνει ότι το μοντέλο έκανε μια *θετική πρόβλεψη* αλλά η αληθινή ετικέτα που έπρεπε να προβλεφθεί ήταν *αρνητική*. Άρα π.χ. προβλέφθηκε ότι κάποιος φοράει μάσκα αλλά στην πραγματικότητα έγινε λάθος πρόβλεψη διότι δεν φοράει.

- **Λάθος αρνητικά (False Negatives/FN):** Αυτή η μέτρηση δείχνει ότι το μοντέλο έκανε μία *αρνητική πρόβλεψη* αλλά η αληθινή ετικέτα που έπρεπε να προβλεφθεί ήταν *θετική*. Άρα π.χ. προβλέφθηκε ότι κάποιος δεν φοράει μάσκα αλλά στην πραγματικότητα έγινε λάθος πρόβλεψη διότι φοράει.
- **Σωστά αρνητικά (True Negatives/TN):** Αυτή η μέτρηση δείχνει ότι το μοντέλο έκανε μία *αρνητική πρόβλεψη* και η αληθινή ετικέτα που έπρεπε να προβλεφθεί ήταν και αυτή *αρνητική*. Άρα π.χ. προβλέφθηκε ότι κάποιος δεν φοράει μάσκα και επιβεβαιώθηκε πως όντως δεν φοράει μάσκα.

Από όλες αυτές τις πληροφορίες που περιέχει ο πίνακας σύγχυσης, προκύπτουν οι υπόλοιπες μετρικές αξιολόγησης (evaluation metrics) που θα αναλύσουμε στην επόμενη ενότητα (1.4.2).

1.4.2 Μετρικές αξιολόγησης

1.4.2.1 Ορθότητα

Η ορθότητα (accuracy) είναι μία μετρική που μετρά τη γενική απόδοση του μοντέλου αλλά συνήθως δεν προτιμάτε, ειδικά για μη ισορροπημένα δεδομένα. Αυτό συμβαίνει διότι υπάρχει περίπτωση κατά την εκπαίδευση ενός μοντέλου τα δεδομένα μιας κλάσης (class)/ κατηγορίας, είτε αυτά είναι θετικά είτε αρνητικά, να είναι περισσότερα από την άλλη κλάση. Έτσι το μοντέλο οδηγείται σε ανισόρροπη ταξινόμηση των προβλέψεων και η ορθότητα να μπορεί να δείχνει υψηλή αλλά στην πραγματικότητα να οφείλεται για αυτό μόνο η μία εκ των δύο κλάσεων και η άλλη να έχει πολύ χαμηλότερη ορθότητα. Εάν βέβαια γνωρίζουμε καλά τα δεδομένα εκπαίδευσης ενός μοντέλου και τα χωρίσουμε ομοιόμορφα τότε η ορθότητα δεν θα είναι άστοχη. Ο υπολογισμός της υπολογίζεται με τον εξής τύπο [34]:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{1.7}$$

1.4.2.2 Ακρίβεια

Η ακρίβεια (precision) μας δείχνει την ικανότητα ενός μοντέλου να προβλέπει σωστά τα θετικά δεδομένα μεταξύ όλων των θετικών δεδομένων είτε δηλαδή σωστών είτε λάθος. Η τιμή του ποσοστού είναι δεκαδική από το 0 έως το 1 και μας δείχνει κατά πόσο οι θετικές προβλέψεις είναι πραγματικά θετικές. Υπολογίζεται από [34]:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{1.8}$$

1.4.2.3 Ανάκληση

Η ανάκληση (recall) υπολογίζει την ικανότητα του μοντέλου να εντοπίζει τα θετικά δεδομένα σε σχέση με όλα τα σωστά θετικά δεδομένα. Χρησιμοποιείται συνήθως για να μειώσουμε τη ποσότητα των λάθος αρνητικών προβλέψεων (FN). Υπολογίζεται από [34]:

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{1.9}$$

1.4.2.4 Βαθμολογία F1

Η βαθμολογία F1 (F1-score) προκύπτει από τον συνδυασμό της ακρίβειας και της ανάκλησης και είναι ο αρμονικός τους μέσος παρέχοντας μια τιμή που αντιπροσωπεύει την αντιστάθμιση μεταξύ των δύο. Χρησιμοποιείται σε περιπτώσεις που θέλουμε να λάβουμε υπόψιν την ακρίβεια και την ανάκληση ταυτόχρονα. Όσο πιο ψηλή είναι η βαθμολογία F1 τόσο καλύτερη ισορροπία υπάρχει μεταξύ αυτών των δύο μετρικών. Υπολογίζεται ως [34]:

$$f_1\text{-score} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \tag{1.10}$$

1.4.2.5 Μέσοι όροι

Στις μετρικές αξιολόγησης μπορούν να υπολογιστούν δύο είδη μέσων όρων, πιο συγκεκριμένα ο μακρύς μέσος όρος (macro average) και ο σταθμισμένος μέσος όρος (weighted average) [35].

- **Μακρύς μέσος όρος:** Ο μακρύς μέσος όρος υπολογίζει τον μέσο όρο της ακρίβειας, της ανάκλησης και της βαθμολογίας F1 μεταξύ όλων των κλάσεων.

- **Σταθμισμένος μέσος όρος:** Ο σταθμισμένος μέσος όρος υπολογίζει και αυτός τον μέσο όρο των μετρικών που αναφέρθηκαν στον μακρύ μέσο όρο, αλλά κατά τον υπολογισμό δίνει περισσότερη βαρύτητα στις κλάσεις με τα περισσότερα δεδομένα για να πραγματοποιήσει έναν πιο δίκαιο υπολογισμό.

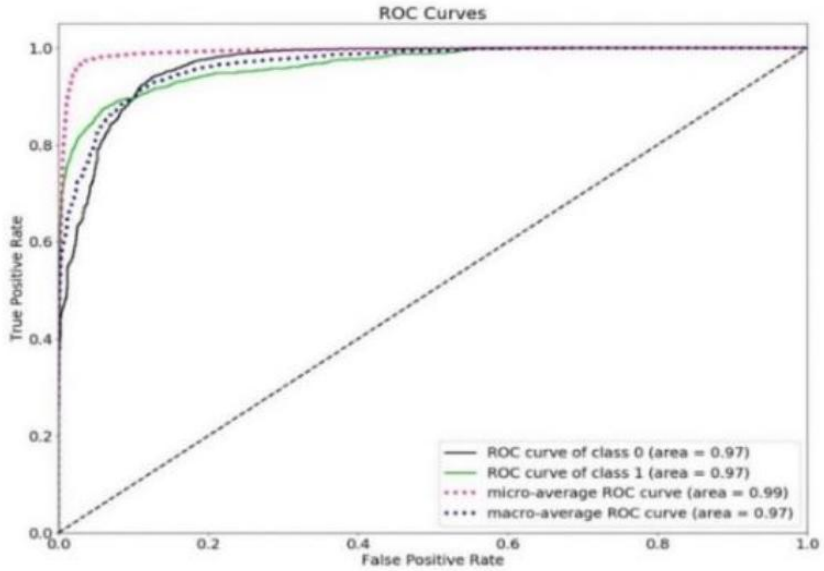
1.4.2.6 Καμπύλη λειτουργικού χαρακτηριστικού δέκτη

Η καμπύλη λειτουργικού χαρακτηριστικού δέκτη (receiver operating characteristic curve) ή αλλιώς ROC μετράει την απόδοση της ταξινόμησης ενός μοντέλου με βάση τον σωστό θετικό ρυθμό (true positive rate/TPR) και τον λάθος θετικό ρυθμό (false positive rate/FPR). Ο σωστός θετικός ρυθμός είναι ουσιαστικά η ανάκληση όπου είδαμε πως υπολογίζεται (1.9) ενώ ο λάθος θετικός ρυθμός υπολογίζεται από την εξής μαθηματική έκφραση:

$$\text{FP rate} = \frac{\text{FP}}{\text{FP} + \text{TN}}. \tag{1.11}$$

Η καμπύλη λειτουργικού χαρακτηριστικού δέκτη είναι μια γραφική αναπαράσταση αυτής της αντιστάθμισης μεταξύ του σωστού θετικού ρυθμού και του λάθος θετικού ρυθμού όπου υπολογίζονται για διαφορετικά κατώφλια (threshold) και τοποθετούνται στο ίδιο γράφημα.

Εάν το μοντέλο ταξινομεί σωστά τα δεδομένα του τότε η τιμή του σωστού θετικού ρυθμού θα είναι υψηλή ενώ του λάθος θετικού ρυθμού χαμηλή για κάθε κατώφλι. Αυτή η κατάσταση αποτυπώνεται γραφικά με τις καμπύλες να είναι τοποθετημένες προς την αριστερή μεριά του γραφήματος όπως φαίνεται στην εικόνα 1.24.



Εικόνα 1.24 Καμπύλη λειτουργικού χαρακτηριστικού δέκτη [32]

1.4.2.7 Περιοχή κάτω από την καμπύλη λειτουργικού χαρακτηριστικού δέκτη

Η περιοχή κάτω από την καμπύλη λειτουργικού χαρακτηριστικού δέκτη (area under the ROC curve) ή αλλιώς βαθμολογία AUC, δείχνει την ποιότητα της καμπύλης αυτής. Πιο συγκεκριμένα παρουσιάζει το πόσο καλά ένα μοντέλο ταξινομεί τις προβλέψεις του και αντιπροσωπεύει την πιθανότητα ότι μια τυχαία επιλεγμένη θετική πρόβλεψη κατατάσσεται υψηλότερα από μια τυχαία επιλεγμένη αρνητική πρόβλεψη.

2 ΚΕΦΑΛΑΙΟ 2ο : Τεχνικό υπόβαθρο

Στο κεφάλαιο αυτό παρουσιάζονται κυρίως τα εργαλεία και τα προγράμματα που χρησιμοποιήθηκαν για την ανάπτυξη του κώδικα εκπαίδευσης μοντέλου ανίχνευσης μάσκας καθώς και την ανάπτυξη του κώδικα για την ανίχνευση προσώπου και μάσκας. Τα εργαλεία αυτά ήταν απαραίτητα για την ολοκλήρωση του πρακτικού μέρους της διπλωματικής εργασίας και βοήθησαν το καθένα με τον δικό του τρόπο. Επίσης περιγράφονται οι βιβλιοθήκες που χρησιμοποιήθηκαν για τους κώδικες, τα εξαρτήματα που χρησιμοποιήθηκαν για την εφαρμογή τους καθώς και κάποια άλλα λογισμικά.

2.1 Χαρακτηριστικά του υπολογιστή και των περιφερειακών

Για την δημιουργία των κωδίκων, την εκπαίδευση των μοντέλων, την εφαρμογή των κωδίκων στη πράξη, την διαχείριση του Raspberry pi 4 (rpi4) καθώς και τη συγγραφή αυτής της διπλωματικής χρησιμοποιήθηκε ένας σταθερός υπολογιστής με 2 οθόνες. Τα χαρακτηριστικά του υπολογιστή φαίνονται παρακάτω:

- **Λειτουργικό σύστημα:** Windows 10 Home 64-bit, έκδοση 21H2,
- **Επεξεργαστής:** Intel(R) Core(TM) i5-6600 CPU @ 3.30GHz 3.31 GHz
- **Μνήμη RAM:** 16 GB, με τύπο DDR4
- **Μητρική κάρτα:** B150 GAMING M3 (MS-7978)
- **Κάρτα γραφικών:** NVIDIA GeForce GTX 960

Οι οθόνες που χρησιμοποιήθηκαν είναι το μοντέλο C27F390 της Samsung, ανάλυσης 1920x1080 pixels, 60 Hz, 27 ιντσών και το μοντέλο M2362D της LG, ανάλυσης 1920x1080 pixels, 60 Hz, 24 ιντσών.

Επίσης για τις δοκιμές της εφαρμογής των μοντέλων μηχανικής μάθησης χρησιμοποιήθηκε η κάμερα διαδικτύου (webcam) της Microsoft και μοντέλο LifeCam VX-1000.

2.2 Python

Η Python, μια ευέλικτη και υψηλού επιπέδου γλώσσα προγραμματισμού, προσφέρει ένα πλούσιο σύνολο δυνατοτήτων για διάφορες εφαρμογές. Ξεχωρίζει με τη σχεδιαστική της φιλοσοφία που δίνει έμφαση στην αναγνωσιμότητα κώδικα μέσω της χρήσης σημαντικής εσοχής (identation), ακολουθώντας τον κανόνα off-side. Ως μία δυναμικά πληκτρολογούμενη γλώσσα με ενσωματωμένη συλλογή «σκουπιδιών», η Python υποστηρίζει πολλαπλά πρότυπα προγραμματισμού, όπως ο διαδικαστικός και ο αντικειμενοστραφής προγραμματισμός.

Εικόνα 2.1 Το λογότυπο της Python [36]

Είναι γνωστή ως μία γλώσσα που συνοδεύεται από μία εκτεταμένη βασική βιβλιοθήκη, η οποία συχνά αναφέρεται ως "οι μπαταρίες περιλαμβάνονται" και παρέχει ένα ευρύ φάσμα προκατασκευασμένων βιβλιοθηκών και εργαλείων. Η Python κερδίζει συνεχώς δημοτικότητα και διατηρεί τη θέση της μεταξύ των πιο ευρέως χρησιμοποιούμενων γλωσσών προγραμματισμού.

Ο Guido van Rossum ξεκίνησε την ανάπτυξη της Python στα τέλη της δεκαετίας του 1980 ως τον διάδοχο της γλώσσας προγραμματισμού ABC. Η πρώτη επίσημη κυκλοφορία ήταν η Python 0.9.0 και παρουσιάστηκε το 1991. Ένα σημαντικό επίτευγμα πραγματοποιήθηκε με την κυκλοφορία της Python 2.0 το 2000. Ωστόσο, η κυκλοφορία της Python 3.0 το 2008 έφερε σημαντικές αλλαγές που δεν ήταν πλήρως συμβατές με τις προηγούμενες εκδόσεις. Η Python 2.7.18, που κυκλοφόρησε το 2020, σηματοδότησε την τελική κυκλοφορία της Python 2 [37].

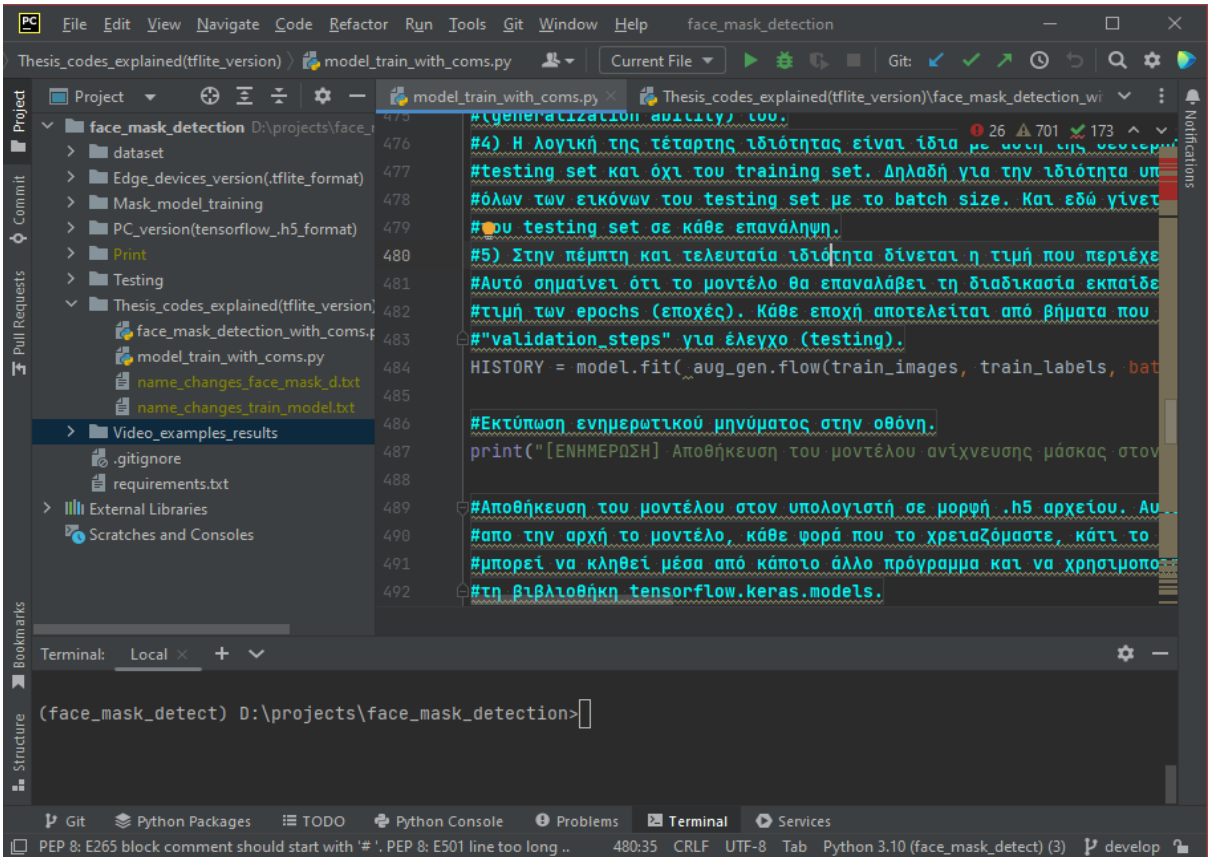
Για τη διπλωματική εργασία αυτή, χρησιμοποιήθηκε η Python 3.10 που είναι η νεότερη έκδοση αυτή τη στιγμή (2022-2023) έτσι ώστε η εφαρμογή να χρησιμοποιεί πιο πρόσφατες και ενημερωμένες εκδόσεις βιβλιοθηκών.

2.3 PyCharm Community Edition

Το PyCharm Community Edition είναι ένα ισχυρό ολοκληρωμένο εργαλείο ανάπτυξης (integrated development tool) κώδικα, προσαρμοσμένο ειδικά για προγραμματιστές Python. Δημιουργήθηκε και κυκλοφόρησε από την εταιρία JetBrains και είναι ένα δωρεάν ανοικτού κώδικα λογισμικό. Η αντίστοιχη επί πληρωμή έκδοση ονομάζεται PyCharm Professional και παρέχει κάποια επιπλέον εργαλεία, τα οποία ήταν περιττά για την διπλωματική αυτή.

Τόσο η Community Edition όσο και η επαγγελματική έκδοση του PyCharm είναι συμβατές με λειτουργικά συστήματα Apple Mac, Microsoft Windows και Linux. Η εισαγωγή της PyCharm Community Edition αντανακλά την αυξανόμενη ζήτηση για δεξιότητες προγραμματισμού Python σε διάφορους τομείς που σχετίζονται με την τεχνολογία.

Παρέχει προηγμένες δυνατότητες όπως η συμπλήρωση και η επιθεώρηση κώδικα και δίνει στους χρήστες τη δυνατότητα να αναπτύσσουν, να διορθώνουν, να εκτελούν και να δοκιμάζουν τα προγράμματα Python τους με ευκολία. Η διαισθητική διεπαφή χρήστη (user interface) της κονσόλας Python και η εύκολη σύνδεση με συστήματα ελέγχου έκδοσης (version control systems) ενισχύουν την εμπειρία ανάπτυξης κώδικα, επιτρέποντας την καλύτερη διαχείριση των εκδόσεων κώδικα [38].



Εικόνα 2.2 Στιγμιότυπο του προγράμματος PyCharm Community edition

Στην εικόνα 2.2 παρουσιάζεται το PyCharm διαμορφωμένο με βάση την διπλωματική εργασία αυτή. Στην αριστερή μεριά φαίνεται το όνομα του project δηλαδή face_mask_detection που χρησιμοποιείται ως φάκελος και περιέχει μέσα όλους τους υποφακέλους που φαίνονται από κάτω του. Εκεί μέσα βρίσκονται όλα τα απαραίτητα αρχεία για την εφαρμογή της ανίχνευσης μάσκας στα πρόσωπα ανθρώπων που δημιουργήθηκε.

Στο κεντρικό, δεξιά παράθυρο αναπτύσσεται ο κώδικας όπου στην συγκεκριμένη περίπτωση φαίνεται ένα κομμάτι του κώδικα εκπαίδευσης μοντέλου για την ανίχνευση μάσκας. Χαμηλά κάτω κεντρικά βρίσκεται το τερματικό που εμφανίζονται τα μηνύματα και τα αποτελέσματα κατά την εκτέλεση του κώδικα. Χαμηλά κάτω δεξιά αναγράφεται η έκδοση της Python (3.10) που έχει επιλεχθεί για την ανάπτυξη της εφαρμογής καθώς και η διακλάδωση (branch) του Github αποθετηρίου (repository) που χρησιμοποιείται εκείνη τη στιγμή π.χ. develop.

2.4 Anaconda

Το Anaconda είναι μία δημοφιλής διανομή (distribution) ανοιχτού κώδικα γλωσσών προγραμματισμού Python και R, ειδικά προσαρμοσμένη για την επιστήμη δεδομένων (data science) και τη μηχανική μάθηση. Προσφέρει μία ολοκληρωμένη σειρά εργαλείων και βιβλιοθηκών που διευκολύνουν την εργασία με δεδομένα και την εκτέλεση σύνθετων αναλυτικών εργασιών.

Το Anaconda συνοδεύεται από μία ευρεία γκάμα προεγκατεστημένων βιβλιοθηκών και εργαλείων που χρησιμοποιούνται συνήθως στους τομείς της επιστήμης δεδομένων και της μηχανικής μάθησης. Κάποιες από αυτές είναι η NumPy, η Pandas, η Matplotlib, ηscikit-learn και το Jupyter Notebook. Αυτές οι βιβλιοθήκες είναι σημαντικές για τον χειρισμό των δεδομένων, την ανάλυσή τους, την οπτικοποίηση και την μοντελοποίηση.

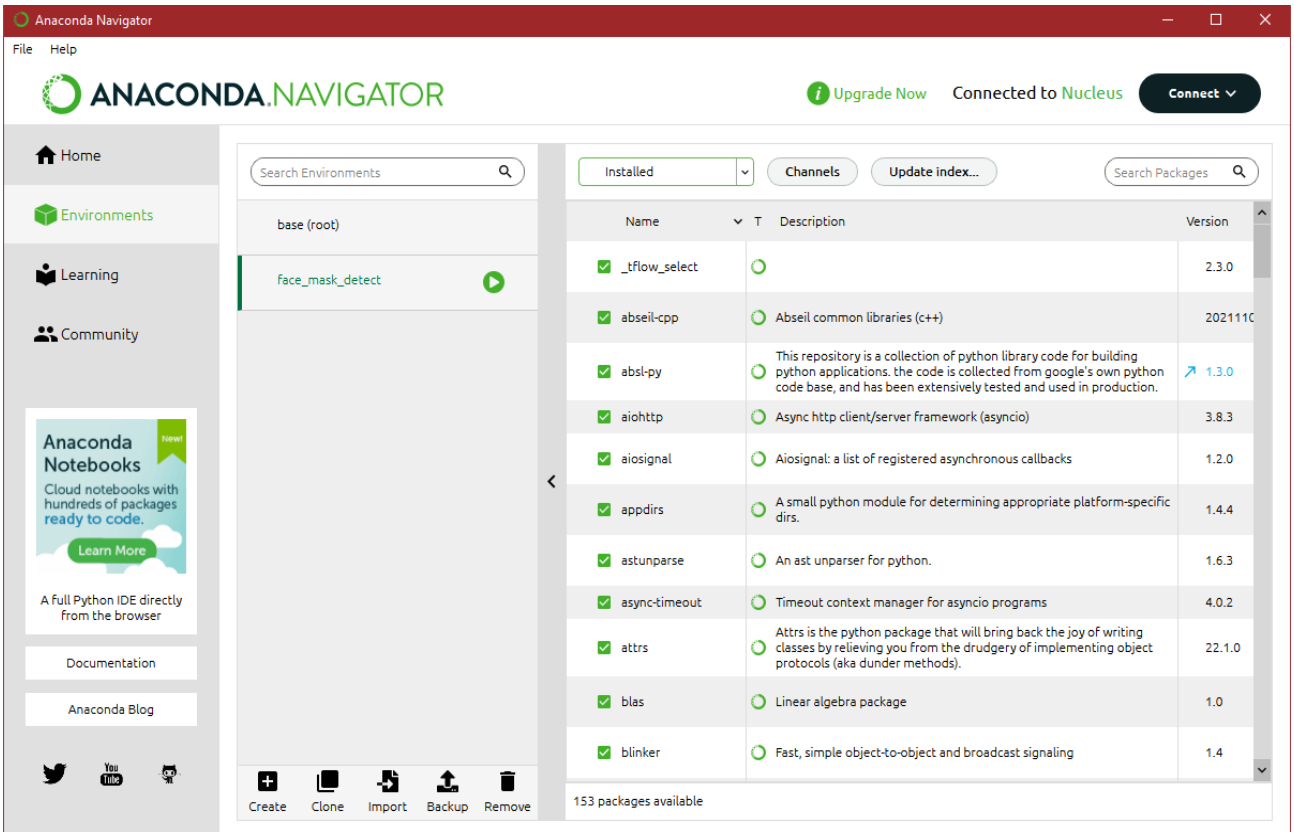
Ένα από τα βασικά χαρακτηριστικά του Anaconda είναι το σύστημα διαχείρισης πακέτων που ονομάζεται conda. Απλοποιεί την εγκατάσταση, τη διαχείριση και την ενημέρωση των πακέτων Python και R, καθώς και των εξαρτημένων πακέτων τους. Αυτό διασφαλίζει την ομαλή ρύθμιση των περιβαλλόντων ανάπτυξης κώδικα με το χειρισμό των εγκαταστάσεων πακέτων και την επίλυση τυχόν αντικρουόμενων εκδόσεων πακέτων.

Ένα άλλο πλεονέκτημα του Anaconda είναι η συμβατότητά του σε πολλαπλές πλατφόρμες. Είναι διαθέσιμο για Windows, macOS και Linux, επιτρέποντας στους χρήστες να έχουν ένα συνεπές περιβάλλον ανάπτυξης σε διαφορετικά λειτουργικά συστήματα.

Το Anaconda επιτρέπει τη δημιουργία απομονωμένων εικονικών περιβαλλόντων, τα οποία βοηθούν στη διαχείριση διαφορετικών εκδόσεων με τις συγκεκριμένες βιβλιοθήκες και τα πακέτα τους. Αυτή η δυνατότητα επιτρέπει στους προγραμματιστές να εργάζονται σε πολλές διαφορετικές εργασίες (project) ταυτόχρονα χωρίς να υπάρχουν διενέξεις μεταξύ των διαφορετικών εκδόσεων πακέτων κάθε εργασίας [39].

Για να βελτιώσει την εμπειρία χρήστη, η Anaconda προσφέρει το Anaconda Navigator, ένα ολοκληρωμένο περιβάλλον ανάπτυξης (integrated development environment/IDE) με γραφικό φιλικό περιβάλλον εργασίας προς τον χρήστη. Παρέχει μία φιλική προς το χρήστη διεπαφή για τη διαχείριση περιβαλλόντων, την εγκατάσταση πακέτων, την εκκίνηση του Jupyter Notebook και την

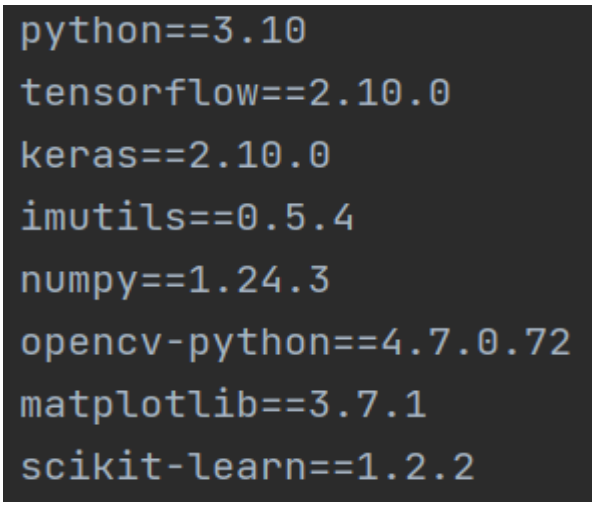
πρόσβαση σε άλλα εργαλεία ανάπτυξης. Η έκδοση του Anaconda Navigator που χρησιμοποιήθηκε είναι η 2.4.0 και στην εικόνα 2.3 απεικονίζεται ένα στιγμιότυπο από το ανοιχτό παράθυρο του Anaconda Navigator. Το παράθυρο περιέχει το περιβάλλον που δημιουργήθηκε για την συλλογή όλων των απαραίτητων βιβλιοθηκών και εκδόσεων τους σχετικά με τον προγραμματισμό της εκπαίδευσης του μοντέλου και της εφαρμογής ανίχνευσης μάσκας. Το περιβάλλον ονομάζεται face_mask_detect.



Εικόνα 2.3 Στιγμιότυπο του παραθύρου Anaconda Navigator

2.5 Βιβλιοθήκες

Για την ανάπτυξη του κώδικα εκπαίδευσης μοντέλου ανίχνευσης μάσκας καθώς και του κώδικα για την εφαρμογή ανίχνευσης μάσκας χρησιμοποιήθηκαν κάποιες συγκεκριμένες βιβλιοθήκες που περιείχαν τις κατάλληλες εντολές για την επίτευξη του σκοπό που θέλαμε. Μέσω του προγράμματος PyCharm δημιουργήθηκε το έγγραφο με όνομα "requirements.txt" που περιέχει όλες τις βιβλιοθήκες που χρειάζεται κάποιος και τις αντίστοιχες εκδόσεις τους για να εκτελέσει τους κώδικες αυτής της διπλωματικής εργασίας. Το περιεχόμενο αυτού του εγγράφου φαίνεται στην εικόνα 2.4



Εικόνα 2.4 Στιγμιότυπο του εγγράφου requirements.txt

2.5.1 TensorFlow

Ίσως η πιο σημαντική από όλες τις βιβλιοθήκες είναι η TensorFlow, μία ισχυρή και δημοφιλής βιβλιοθήκη λογισμικού ανοιχτού κώδικα που χρησιμοποιείται ευρέως για το πεδίο της μηχανικής μάθησης και της τεχνητής νοημοσύνης. Αναπτύχθηκε αρχικά από την ομάδα Google Brain με σκοπό την εσωτερική έρευνα και παραγωγή.

Από την αρχική του κυκλοφορία υπό την άδεια Apache 2.0 το 2015, το TensorFlow έχει κερδίσει σημαντική έλξη στην κοινότητα της μηχανικής μάθησης. Τον Σεπτέμβριο του 2019, η Google παρουσίασε το TensorFlow 2.0, το οποίο έφερε αρκετές βελτιώσεις και αναβαθμίσεις.

Το TensorFlow προσφέρει μια ολοκληρωμένη γκάμα χαρακτηριστικών και εργαλείων, με ιδιαίτερη έμφαση στην εκπαίδευση σε βαθιά νευρωνικά δίκτυα και στην εξαγωγή συμπερασμάτων. Ένα αξιοσημείωτο πλεονέκτημά του είναι η γλωσσική του ευελιξία, καθώς υποστηρίζει πολλές γλώσσες προγραμματισμού όπως η Python, η JavaScript, η C++ και η Java. Αυτή η προσαρμοστικότητα επιτρέπει στους προγραμματιστές να χρησιμοποιήσουν το TensorFlow σε διαφορετικές εφαρμογές και βιομηχανίες, καθιστώντας το έναν ανεκτίμητο πόρο για τη δημιουργία και την ανάπτυξη μοντέλων τεχνητής νοημοσύνης [40].

2.5.1.1 Keras

Η Keras είναι μία δημοφιλής βιβλιοθήκη ανοιχτού κώδικα που χρησιμεύει για τη δημιουργία τεχνητών νευρωνικών δικτύων. Έχει σχεδιαστεί ειδικά για να παρέχει ένα φιλικό προς τον χρήστη και ευέλικτο πλαίσιο (framework) για την ανάπτυξη μοντέλων βαθιάς μάθησης. Έχει αναγνωριστεί ευρέως για τη συμβατότητά της με διάφορα backend, συμπεριλαμβανομένων των TensorFlow, Microsoft Cognitive Toolkit, Theano και PlaidML. Ωστόσο, ξεκινώντας από την έκδοση 2.4, η Keras εστιάζει αποκλειστικά στην υποστήριξη του backend κομματιού του TensorFlow. Αυτό σημαίνει πως είναι μέρος του Tensorflow και λειτουργεί πάνω σε αυτό.

Η βιβλιοθήκη αυτή αναπτύχθηκε ως μέρος του έργου ONEIROS, το οποίο είχε ως στόχο τη δημιουργία ενός ανοιχτού τύπου νευροηλεκτρονικού, ευφυούς και ρομποτικού λειτουργικού συστήματος. Ο François Chollet, μηχανικός της Google, είναι ο κύριος συγγραφέας και συντηρητής του Keras. Η Keras έχει γίνει μία δημοφιλής επιλογή τόσο για επαγγελματίες βαθιάς μάθησης όσο και για ερευνητές αφού δίνει έμφαση στην απλότητα, την προσαρμοστικότητα και την επεκτασιμότητα [41].

2.5.1.2 TensorFlow lite

Το TensorFlow Lite, που συχνά αναφέρεται ως TFLite, είναι μία βιβλιοθήκη ανοιχτού κώδικα που δημιουργήθηκε από την Google ειδικά σχεδιασμένη για την ανάπτυξη μοντέλων μηχανικής μάθησης σε συσκευές αιχμής (edge devices) όπως οι κινητές συσκευές (iOS/Android), τα ενσωματωμένα συστήματα (embedded devices) και συσκευές όπως το Raspberry Pi. Παρέχει υποστήριξη για διάφορα πεδία όπως η επεξεργασία της φυσικής γλώσσας και η όραση υπολογιστή. Ο πρωταρχικός στόχος του TensorFlow Lite είναι να επιτρέψει την αποτελεσματική ανάπτυξη μοντέλων σε συσκευές με περιορισμένους πόρους. Αυτό το επιτυγχάνει προσφέροντας δυνατότητες

μετατροπής μοντέλων οι οποίες βελτιστοποιούν το μέγεθος, την καθυστέρηση και την απόδοση των μοντέλων.

Με το TensorFlow Lite, δίνετε η δυνατότητα μετατροπής του μοντέλου TensorFlow σε αρχείο .tflite χρησιμοποιώντας τον μετατροπέα TensorFlow Lite. Αυτό το αρχείο μπορεί στη συνέχεια να φορτωθεί και να εκτελεστεί στη συσκευή-στόχο, στη συγκεκριμένη περίπτωση αυτής της διπλωματικής εργασίας στο rpi4. Αυτή η διαδικασία μετατροπής διασφαλίζει ότι το μοντέλο είναι προσαρμοσμένο ώστε να λειτουργεί αποτελεσματικά σε συσκευές αιχμής, με μειωμένη χρήση μνήμης και χαμηλότερη καθυστέρηση αποτελεσμάτων.

Ένα από τα βασικά πλεονεκτήματα του TensorFlow Lite είναι η υποστήριξή του σε εφαρμογές σχετικά με την όραση υπολογιστή. Αυτό επιτρέπει να εκτελούνται εργασίες όπως η ταξινόμηση εικόνων και η ανίχνευση αντικειμένων απευθείας από τη συσκευή, χωρίς να χρειάζεται να βασιστεί η συσκευή στο σύννεφο (cloud) [42].

2.5.2 Sklearn

Η Sklearn, γνωστή και ως scikit-learn, είναι μία ευρέως χρησιμοποιούμενη και ισχυρή βιβλιοθήκη μηχανικής μάθησης ανοιχτού κώδικα για την Python. Προσφέρει μια εκτεταμένη γκάμα εργαλείων και αλγορίθμων που καλύπτουν διάφορες εργασίες μηχανικής μάθησης, όπως η ταξινόμηση, η παλινδρόμηση, η ομαδοποίηση, η μείωση διαστάσεων και η αξιολόγηση μοντέλων [43].

Η βιβλιοθήκη περιλαμβάνει ενότητες αφιερωμένες σε βασικές εργασίες όπως η προεπεξεργασία δεδομένων, η εξαγωγή χαρακτηριστικών, η επιλογή μοντέλου και η αξιολόγηση απόδοσης. Επίσης περιέχει πολλούς αλγορίθμους, όπως τα δέντρα αποφάσεων ή η k-means που επιτρέπουν στον χρήστη να εφαρμόζει διάφορες τεχνικές στα σύνολα δεδομένων του.

Επιπλέον, η Sklearn παρέχει βοηθητικά εργαλεία για ειδικές διαδικασίες όπως ο διαχωρισμός των δεδομένων, η διασταυρούμενη επικύρωση και ο συντονισμός υπερπαραμέτρων. Συνεργάζεται χωρίς επιπλοκές με άλλες δημοφιλείς βιβλιοθήκες όπως η NumPy και η panda, επιτρέποντας στους χρήστες να αξιοποιήσουν τις δυνατότητές τους και να δημιουργήσουν ολοκληρωμένα συστήματα μηχανικής μάθησης.

2.5.3 NumPy

Η NumPy είναι μία ισχυρή βιβλιοθήκη της Python για αριθμητικές πράξεις και επιστημονικούς υπολογισμούς. Παρέχει χρήσιμους πολυδιάστατους πίνακες και μαθηματικές συναρτήσεις, που είναι απαραίτητα για εργασίες όπως η ανάλυση δεδομένων και η μηχανική μάθηση. Με τους γρήγορους υπολογισμούς και την εύκολη συνεργασία της με άλλες βιβλιοθήκες, η NumPy χρησιμοποιείται ευρέως σε διάφορους τομείς, χρησιμεύοντας ως βάση για την προηγμένη επεξεργασία δεδομένων και επιστημονικές εφαρμογές [44].

2.5.4 Matplotlib

Η Matplotlib είναι μία ευέλικτη βιβλιοθήκη για τη δημιουργία οπτικοποιήσεων και γραφικών παραστάσεων. Παρέχει ένα ευρύ φάσμα σχεδίασης συναρτήσεων και επιλογών επεξεργασίας, καθιστώντας τη κατάλληλη για τη δημιουργία βασικών αλλά και προηγμένων γραφημάτων. Υποστηρίζει διάφορους τύπους γραφικών όπως τα διαγράμματα διασποράς ή τα ιστογράμματα.

Με τη διαισθητική διεπαφή της και τη συμβατότητά της με τα διανύσματα της βιβλιοθήκης NumPy, η Matplotlib είναι μία δημοφιλής επιλογή για περιπτώσεις που αφορούν την οπτικοποίηση δεδομένων σε τομείς όπως η ανάλυση δεδομένων, η επιστημονική έρευνα και η μηχανική μάθηση [45].

2.5.5 Imutils

Η Imutils είναι μία βιβλιοθήκη που παρέχει ένα σύνολο βοηθητικών συναρτήσεων για εργασίες επεξεργασίας εικόνας και όρασης υπολογιστή. Προσφέρει βολικές συναρτήσεις για διαδικασίες όπως η αλλαγή μεγέθους, η περιστροφή και η περικοπή εικόνων. Απλοποιεί τις κοινές τεχνικές επεξεργασίας εικόνας, επιτρέποντας στους προγραμματιστές να χειρίζονται και να προεπεξεργάζονται εικόνες γρήγορα και εύκολα. Με εστίαση στην απλότητα και την αποτελεσματικότητα, η βιβλιοθήκη Imutils αποτελεί ένα πολύτιμο εργαλείο για εργασίες που σχετίζονται με την ανάλυση εικόνας, την ανίχνευση αντικειμένων και τις εφαρμογές όρασης υπολογιστή [46].

2.5.6 OpenCV

Η βιβλιοθήκη OpenCV (Open Source Computer Vision) είναι επίσης μία από τις πιο σημαντικές βιβλιοθήκες που χρησιμοποιήθηκαν. Είναι μία βιβλιοθήκη ανοιχτού κώδικα που χρησιμοποιείται κυρίως για περιπτώσεις του πεδίου όρασης υπολογιστή και της επεξεργασίας εικόνας.

Παρέχει ένα ολοκληρωμένο σύνολο συναρτήσεων και αλγορίθμων για διαδικασίες όπως ο χειρισμός της εικόνας και βίντεο ή η ανίχνευση και παρακολούθηση αντικειμένων. Υποστηρίζει διάφορες γλώσσες προγραμματισμού, συμπεριλαμβανομένων των Python, C++ και Java, καθιστώντας τη προσβάσιμη και ευέλικτη για τους προγραμματιστές [47].

2.6 Αρχεία μοντέλου ανίχνευσης προσώπων

Για την υλοποίηση της εφαρμογής ανίχνευσης μάσκας σε πρόσωπα, χρησιμοποιήθηκε ένα προεκπαιδευμένο μοντέλο που δημιουργήθηκε με βάση τον αλγόριθμο βαθιάς μάθησης Single Shot Multibox Detector ή αλλιώς SSD. για τον αποτελεσματικό εντοπισμό των ανθρώπινων προσώπων. Το μοντέλο αποτελείται από δύο αρχεία τα οποία φορτώνονται στον κώδικα και έπειτα εκτελείται η συναρμολόγηση του μοντέλου και η χρήση του από την εφαρμογή.

Το πρώτο αρχείο έχει τη μορφή απλού κειμένου με ονομασία "deploy.prototxt" και περιέχει πληροφορίες σχετικά με την αρχιτεκτονική του δικτύου, τα επίπεδά του, τον τρόπο που αυτά συνδέονται μεταξύ τους και τις ρυθμίσεις του δικτύου.

Το δεύτερο αρχείο έχει τη μορφή αρχείου ".caffemodel" και όνομα "res10_300x300_ssd_iter_140000.caffemodel". Αυτό περιέχει τις παραμέτρους του δικτύου όπως τα βάρη και τις πολώσεις του με τιμές διαμορφωμένες μετά από την εκπαίδευση του μοντέλου ανίχνευσης προσώπου πάνω σε συγκεκριμένο σύνολο δεδομένων. Το όνομα αυτού του αρχείου δείχνει κάποια από τα χαρακτηριστικά του όπως οι διαστάσεις των εικόνων που δέχεται (300x300 pixel), και ότι έχει εκπαιδευτεί για 140000 επαναλήψεις.

Τα αρχεία του μοντέλου κλωνοποιήθηκαν από το αποθετήριο face-detection του χρήστη keyurr2 στη σελίδα Github [48].

2.7 Github

Το GitHub είναι μία πλατφόρμα στο ίντερνετ στην οποία ένας χρήστης μπορεί να δημιουργήσει με τον δικό του λογαριασμό ένα αποθετήριο για την αποθήκευση των κωδίκων ή και άλλων αρχείων που δημιουργεί για μία συγκεκριμένη εργασία (project). Για τη συγκεκριμένη διπλωματική εργασία δημιουργήθηκε στο GitHub ένα ιδιωτικό μέχρι στιγμής αποθετήριο με όνομα "face_mask_detection".

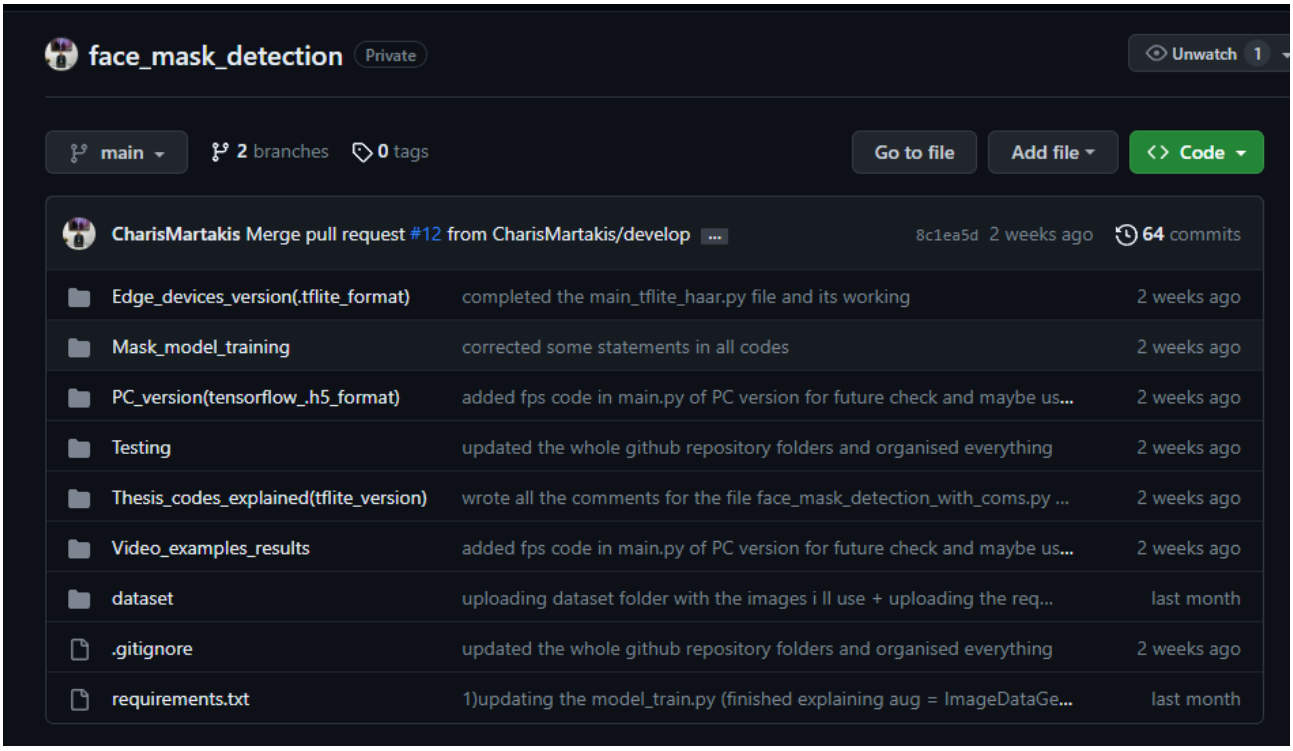
Το GitHub είναι χτισμένο πάνω στο σύστημα ελέγχου έκδοσης με όνομα Git, το οποίο βοηθά τους προγραμματιστές να παρακολουθούν και να διαχειρίζονται τις αλλαγές που γίνονται στον κώδικά τους με την πάροδο του χρόνου. Αυτό σημαίνει πως τους επιτρέπεται να διατηρούν ένα ιστορικό των αλλαγών που έγιναν στους κώδικές τους, δίνοντάς τους τη δυνατότητα να επιστρέψουν σε προηγούμενες εκδόσεις αυτών εάν χρειάζεται. Αυτό διασφαλίζει ότι ο πολύτιμος κώδικας δεν θα χαθεί και παρέχει ένα δίχτυ ασφαλείας για πειραματισμό και ανάπτυξη.

Το Git υποστηρίζει επίσης τη συνεργασία μεταξύ χρηστών είτε αυτοί είναι προγραμματιστές είτε για παράδειγμα ο υπεύθυνος καθηγητής αυτής της διπλωματικής εργασίας, επιτρέποντας έτσι σε πολλά άτομα ταυτόχρονα να παρακολουθούν ή να επεξεργάζονται τους κώδικες και τα αρχεία του αποθετηρίου.

Με το GitHub που υποστηρίζεται από το σύστημα ελέγχου εκδόσεων Git επιτυγχάνεται η καλύτερη διαχείριση και οργάνωση του κώδικα, η ασφάλεια του κώδικα, αφού βρίσκεται στο σύννεφο και

υπάρχουν αποθηκευμένες πιο παλιές εκδόσεις του καθώς και η παρακολούθηση των αλλαγών στον κώδικα με σχόλια κατά την ενημέρωση μιας έκδοσης κώδικα [49].

Η σύνδεση του GitHub αποθετηρίου και του προγράμματος Pycharm για την συγγραφή των κωδίκων γίνεται μέσω του Git και της επιλογής που παρέχει το ίδιο το Pycharm έτσι ώστε να γίνει σύνδεση του χρήστη με τα στοιχεία του (email και κωδικός). Μετά τη σύνδεσή του, ο χρήστης μπορεί να κλωνοποιήσει το αποθετήριό του στο Pycharm και να το επεξεργάζεται όπως θέλει μέσω αυτού πλέον. Στην εικόνα 2.5 απεικονίζεται ένα στιγμιότυπο του αποθετηρίου που δημιουργήθηκε στο GitHub και περιέχει όλους τους φακέλους με τους κώδικες και άλλες πληροφορίες που φτιάχτηκαν για αυτή τη διπλωματική εργασία.



Εικόνα 2.5 Στιγμιότυπο του αποθετηρίου αυτής της διπλωματικής εργασίας στο Github

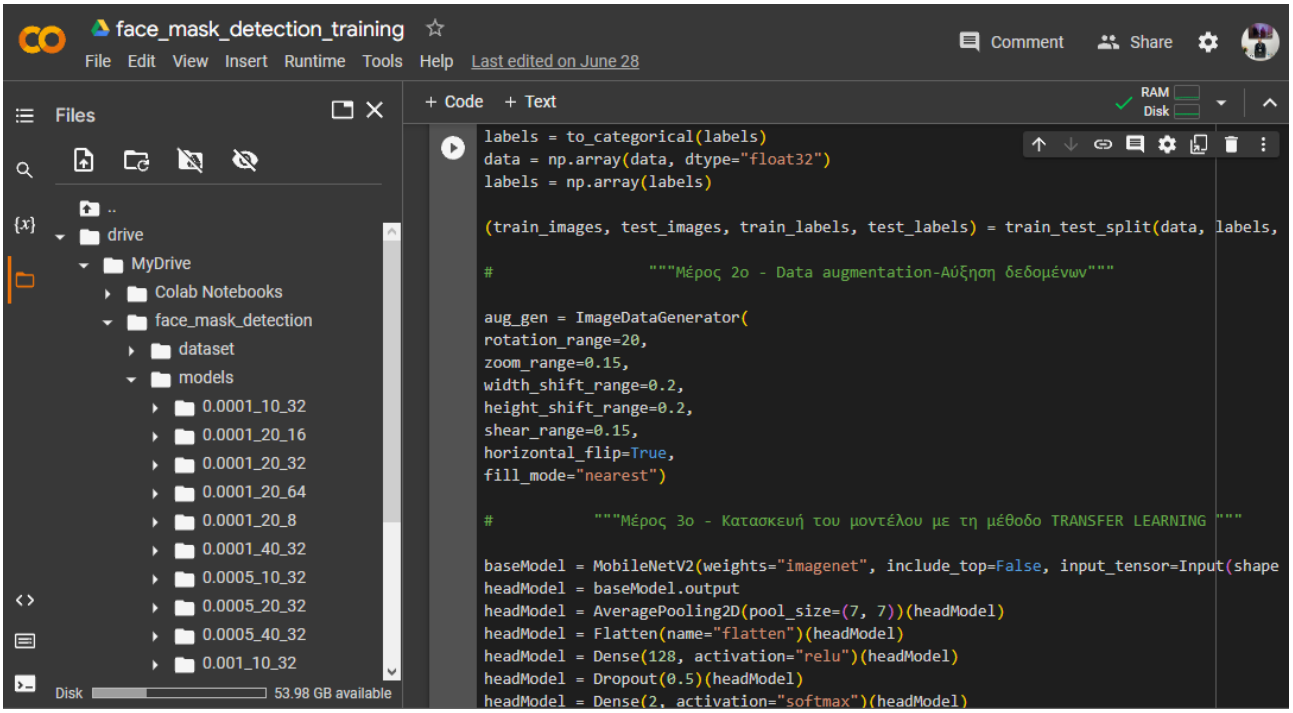
2.8 Google Colab

Ένα πολύ σημαντικό εργαλείο που χρησιμοποιήθηκε για την εκτέλεση του κώδικα εκπαίδευσης μοντέλων ανίχνευσης μάσκας, είναι το Google Colab. Το google Colab είναι μια πλατφόρμα στο σύννεφο σχεδιασμένη για την εκτέλεση κώδικα γραμμένου σε γλώσσα Python και βασισμένη στο Jupyter Notebook. Ο χρήστης έχει τη δυνατότητα να δημιουργήσει ένα αρχείο που ονομάζεται "σημειωματάριο" μέσα στο οποίο μπορεί να γράψει κώδικα και να τον εκτελέσει άμεσα. Μέσα σε αυτό το σημειωματάριο δεν χρειάζεται κάποια συγκεκριμένη ρύθμιση ή εγκατάσταση αφού όλα εκτελούνται στο σύννεφο και είναι έτοιμα.

Το πιο σημαντικό κομμάτι του Google Colab είναι η επιτάχυνση υλικού (hardware acceleration) μέσω της παροχής δωρεάν πόρων, δηλαδή επεξεργαστών αλλά και καρτών γραφικών. Αυτό βοηθάει πολύ σε εργασίες σχετικά με τη μηχανική και βαθιά μάθηση αφού επιταχύνονται οι υπολογισμοί και η εκπαίδευση των μοντέλων μπορεί να γίνει με μεγαλύτερη ταχύτητα και ευκολία.

Πρέπει βέβαια να αναφερθεί πως οι δωρεάν κάρτες γραφικών εάν χρησιμοποιηθούν για κάποιες ώρες συνεχόμενα, τότε το Google Colab τις αποκλείει από τον χρήστη για κάποιο συγκεκριμένο χρονικό διάστημα. Αυτό συμβαίνει διότι υπάρχει και η επί πληρωμή έκδοσή του που παρέχει απεριόριστους πόρους.

Άλλο ένα θετικό είναι η σύνδεση του με την πλατφόρμα Google Drive στην οποία για παράδειγμα έχει αποθηκευτεί το σύνολο δεδομένων για την εκπαίδευση των μοντέλων ανίχνευσης μάσκας, που είναι αρκετά μεγάλο για να φορτωθεί απευθείας στο Google Colab. Έτσι με τη σύνδεσή τους το Google Colab δέχεται αυτά τα δεδομένα απευθείας από το Google Drive και εκτελεί την εκπαίδευση των μοντέλων με επιτυχία [50]. Στην εικόνα 2.6 φαίνεται ένα στιγμιότυπο από το σημειωματάριο που δημιουργήθηκε στο Google Colab για την παραγωγή των μοντέλων.



Εικόνα 2.6 Στιγμιότυπο από το σημειωματάριο του Google Colab

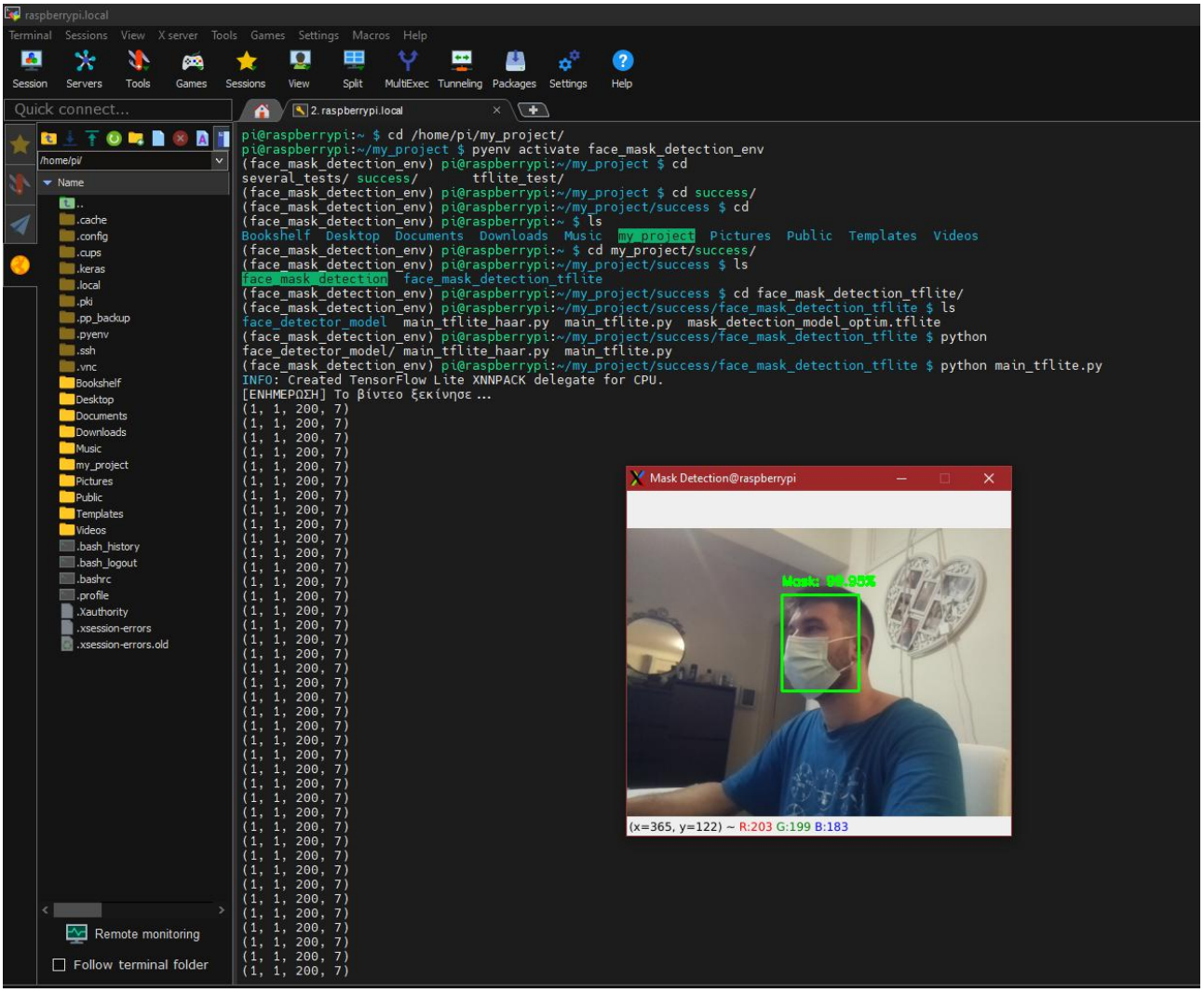
2.9 MobaXterm

Το MobaXterm είναι ένα πρόγραμμα που αποτελείται από πολλά εργαλεία και χρησίμευσε συγκεκριμένα για την σύνδεση με το rpi4 και την διαχείρισή του μέσω του υπολογιστή και χωρίς να χρειάζεται επιπλέον σύνδεση περιφερειακών στο rpi4. Το μόνο που χρειάστηκε για τη σύνδεση του MobaXterm με το rpi4 είναι να συνδεθεί το rpi4 αρχικά με ethernet στο router του σπιτιού έτσι ώστε να πραγματοποιηθεί σύνδεση μέσω του πρωτοκόλλου SSH.

Το MobaXterm συνδέεται στο rpi4 μέσω τερματικού και δίνεται επίσης η δυνατότητα να μεταφερθούν με ευκολία από τον υπολογιστή στο rpi4 αρχεία μέσω του ειδικού συστήματος διαχείρισης φακέλων του rpi4 από το παράθυρο του MobaXterm (πρωτόκολλα SFTP και SCP). Επίσης το πρόγραμμα υποστηρίζει την προώθηση X11, που επιτρέπει σε εφαρμογές με γραφικά να εκτελούνται στο rpi4 και να εμφανίζονται στον υπολογιστή [51]. Μία τέτοια εφαρμογή είναι το

παράθυρο που θα εμφανίζεται με το ζωντανό βίντεο για την ανίχνευση μάσκας σε ανθρώπινα πρόσωπα.

Για τη σύνδεση στο rpi4 αρκεί να δημιουργήσουμε μια σύνδεση με το πρωτόκολλο SSH και να πληκτρολογήσουμε τα στοιχεία που έχουν ορισθεί στο rpi4, κατά την εγκατάσταση του λειτουργικού συστήματός του, σχετικά με την σύνδεση αυτού του πρωτοκόλλου. Αυτά τα στοιχεία είναι ένα συγκεκριμένο όνομα συνοδευόμενο από έναν κωδικό. Ένα στιγμιότυπο της σύνδεσης του rpi4 με τον υπολογιστή μέσω του MobaxTerm και της εκτέλεσης του κώδικα ανίχνευσης μάσκας, απεικονίζεται στην εικόνα 2.7.

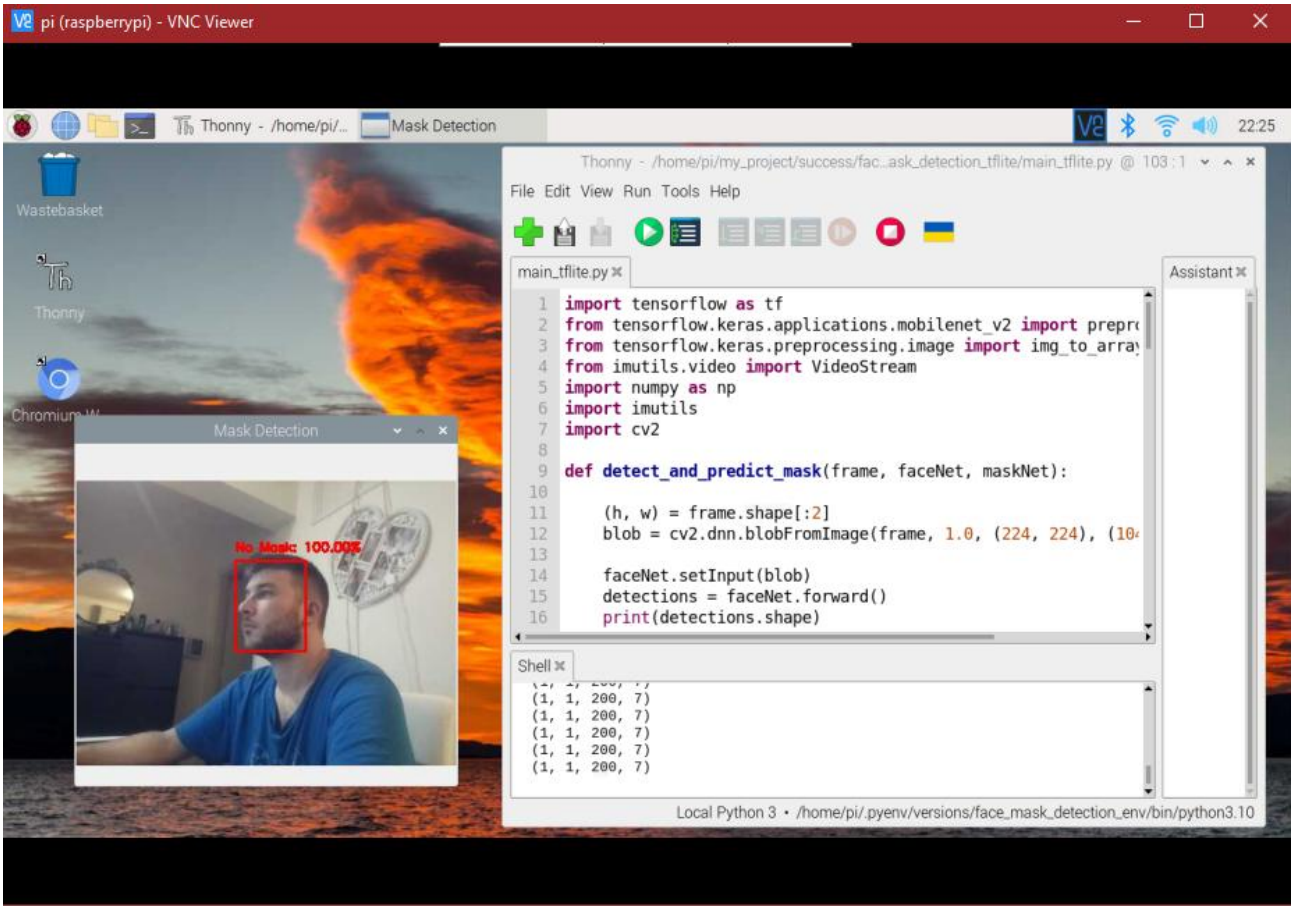


Εικόνα 2.7 Στιγμιότυπο μέσα από το περιβάλλον του MobaxTerm κατά τη σύνδεσή του με το Raspberry Pi 4

2.10 VNC Viewer

Το VNC Viewer είναι ένα πρόγραμμα που έχει παρόμοιο σκοπό με το MobaxTerm, δηλαδή χρησιμοποιείται και αυτό για τη σύνδεση στο rpi4. Η διαφορά του είναι πως αυτό δεν συνδέεται σε περιβάλλον τερματικού αλλά κατευθείαν στο γραφικό περιβάλλον του λειτουργικού συστήματος του rpi4 ξεκινώντας από την επιφάνεια εργασίας του. Αυτό χρησιμεύει σε περιπτώσεις που θέλει ο χρήστης να χειριστεί το rpi4 μέσω γραφικών είτε γιατί του είναι πιο εύκολο είτε επειδή μπορεί να θέλει να προγραμματίσει το rpi4 μέσω κάποιου λογισμικού του όπως το Thonny.

Δυστυχώς το γραφικό περιβάλλον χρησιμοποιεί περισσότερους πόρους οπότε χρησιμοποιήθηκε μόνο για τον προγραμματισμό και τις δοκιμές του κώδικα στο rpi4. Στην εικόνα 2.8 προβάλλεται ένα στιγμιότυπο από το παράθυρο του VNC Viewer που έχει πραγματοποιηθεί σύνδεση με το rpi4 και εκτελείται μία δοκιμή της εφαρμογής ανίχνευσης μάσκας μέσω του Thonny.

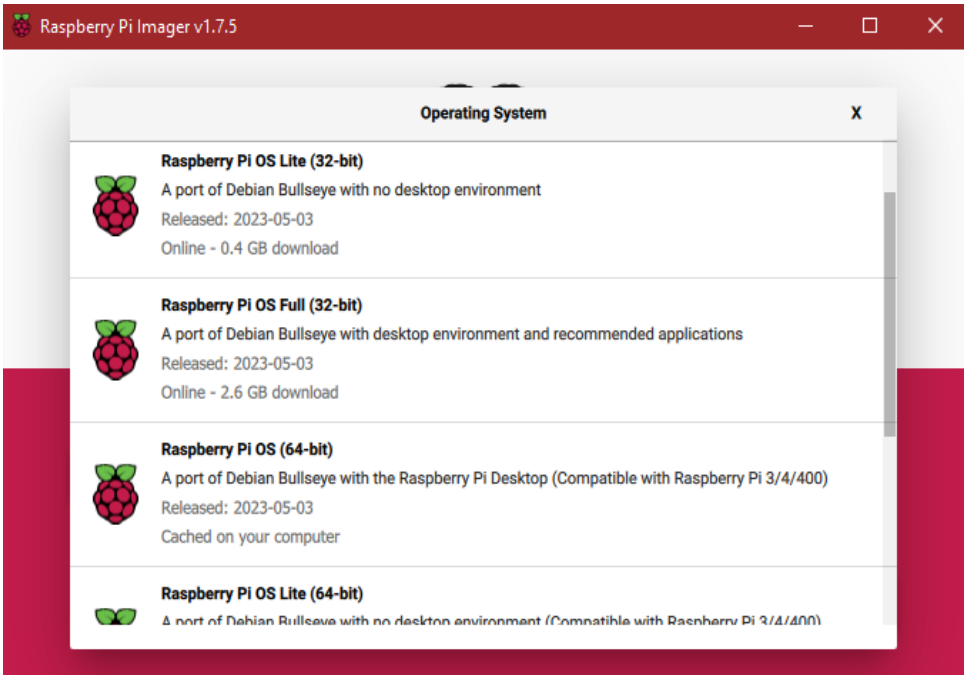


Εικόνα 2.8 Στιγμιότυπο του προγράμματος VNC Viewer κατά τη σύνδεσή του με το Raspberry Pi 4

2.11 Imager 1.7.5

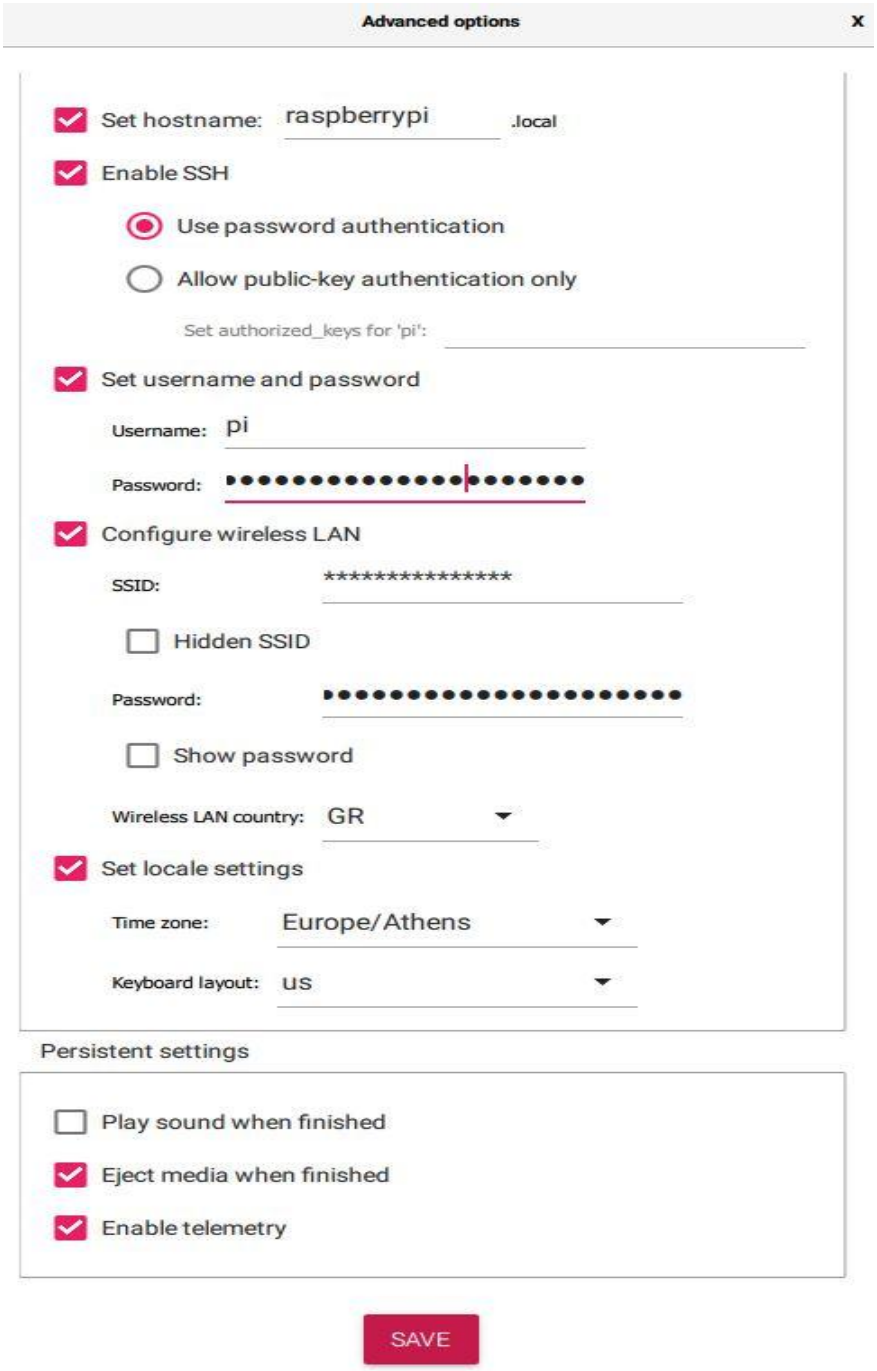
Το Imager, με έκδοση 1.7.5, είναι ένα πρόγραμμα εγκατάστασης λειτουργικού συστήματος σε ένα Raspberry Pi και συγκεκριμένα η εγκατάσταση γίνεται στην κάρτα μνήμης SD του rpi4 που χρησιμοποιήθηκε. Το Imager έχει δημιουργηθεί από την ίδια την εταιρία του Raspberry pi και μπορεί να κατεβεί από την σελίδα της [52].

Για την εγκατάσταση του λειτουργικού συστήματος αρχικά πρέπει να γίνει σύνδεση του υπολογιστή με την κάρτα SD. Στην περίπτωσή μας τοποθετήθηκε η κάρτα SD σε έναν αντάπτορα που συνδέεται στον υπολογιστή μέσω θύρας USB. Έπειτα εκτελείται η εφαρμογή του Imager η οποία εμφανίζει το κεντρικό μενού της και μας οδηγεί να επιλέξουμε πρώτα το λειτουργικό που επιθυμούμε και έπειτα την συσκευή που θα εγκατασταθεί αυτό. Στην εικόνα 2.9 φαίνονται μερικές από τις επιλογές λειτουργικού συστήματος που μας δίνει.



Εικόνα 2.9 Μενού επιλογών λειτουργικού συστήματος για το Raspberry Pi 4

Αφού επιλεχθεί το κατάλληλο λειτουργικό και ο προορισμός του, πραγματοποιούνται κάποιες τελικές ρυθμίσεις για να είναι το rpi4 έτοιμο κατά την πρώτη εκκίνησή του. Αυτές οι ρυθμίσεις απεικονίζονται στην εικόνα 2.10. Τέλος μένει η επιλογή του κουμπιού “WRITE” για να ξεκινήσει η εγκατάσταση, η οποία μόλις ολοκληρωθεί εμφανίζει κατάλληλο μήνυμα.



Εικόνα 2.10 Ρυθμίσεις του Raspberry Pi 4 πριν από την εγκατάσταση του λειτουργικού του συστήματος

2.12 Win32DiskImager

Η εφαρμογή Win32DiskImager χρησιμοποιείται για τη δημιουργία αντίγραφου ασφαλείας όλου του λογισμικού και των αρχείων που περιέχονται στην sd κάρτα του rpi4 τα οποία αποθηκεύονται σε ένα αρχείο μορφής .img. Το αντίγραφο ασφαλείας χρειάζεται έτσι ώστε εάν με κάποιες επόμενες ρυθμίσεις ή αλλαγές στο rpi4 που θα γίνουν, δεν ξαναλειτουργεί το λογισμικό του ή κάτι αποτύχει να εκτελεστεί, θα μπορούμε να επαναφέρουμε ξανά το rpi4 στην κατάσταση εκείνη που λειτουργούσε κανονικά.

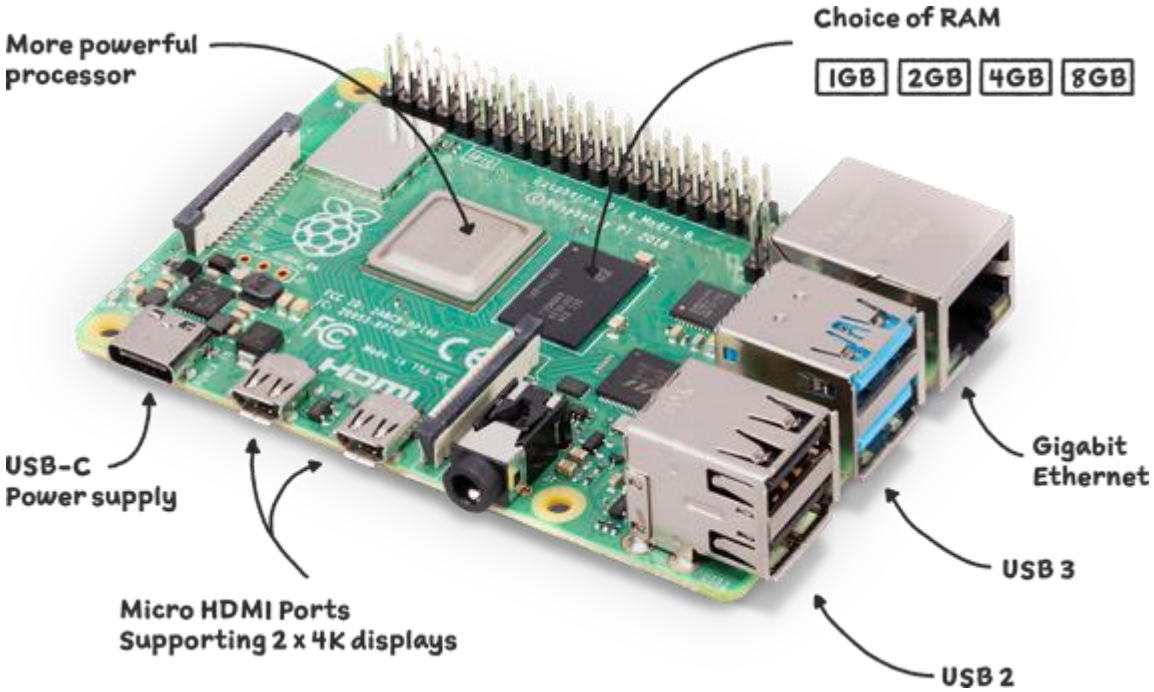
2.13 Raspberry Pi 4 model B

Η εφαρμογή ανίχνευσης μάσκας σε πρόσωπα ανθρώπων αποθηκεύτηκε και προσαρμόστηκε καταλλήλως για να εκτελεστεί μέσα σε μια συσκευή Raspberry pi 4 ή αλλιώς rpi4.

Το rpi4 είναι ένας υπολογιστής με μικρό μέγεθος και χαμηλό κόστος που χωράει στην παλάμη του χεριού μας. Αποτελείται από μία πλακέτα που έχει πάνω της διάφορα εξαρτήματα όπως ένας επεξεργαστής και είναι βολικό για την εκτέλεση εφαρμογών σε οποιοδήποτε μέρος εάν βρίσκεται κάποιος αρκεί να έχει ρεύμα, να είναι συνδεδεμένο στο διαδίκτυο και να υπάρχει διαθέσιμος στο ίδιο δίκτυο ένας φορητός ή σταθερός υπολογιστής για τον χειρισμό του.

Το rpi4 χρησιμοποιείται για την δοκιμή διαφόρων ειδών εφαρμογών όπως για παράδειγμα είναι οι αυτοματισμοί σπιτιού, η φιλοξενία μιας ιστοσελίδας, η χρήση του σαν αποθηκευτικό χώρο στο σύννεφο με πρόσβαση από οποιοδήποτε σημείο στον κόσμο, Στη συγκεκριμένη περίπτωση χρησιμοποιήθηκε με σκοπό την ανίχνευση αντικειμένων με τη βοήθεια μιας κάμερας που συνδέεται με αυτό.

Υπάρχουν διάφορα μοντέλα Raspberry Pi αλλά επιλέχθηκε το Raspberry Pi 4 model B διότι είναι το πιο καινούργιο μοντέλο μέχρι στιγμής άρα το ταχύτερο και ευκολότερο να το διαχειριστεί κανείς.



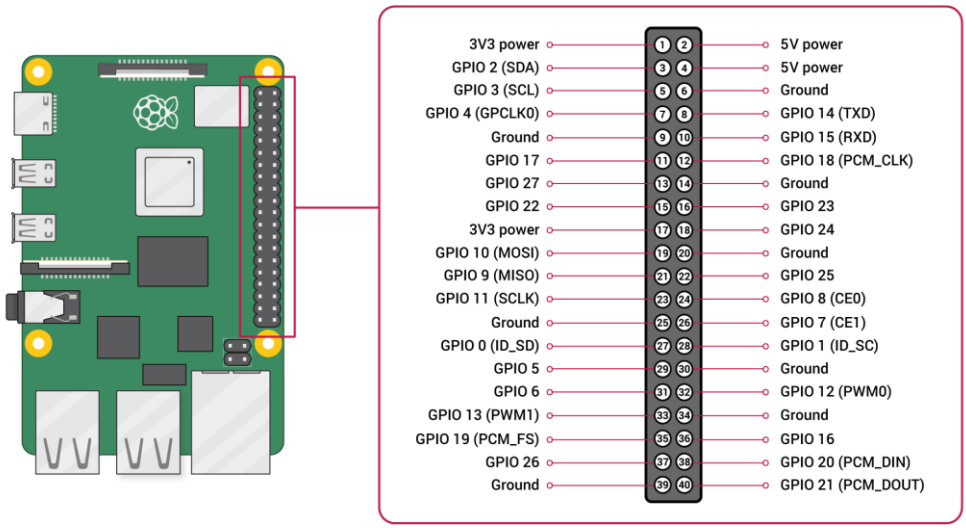
Εικόνα 2.11 Το Raspberry Pi 4 model B [53]

2.13.1 Υλικό μέρος

2.13.1.1 Πλακέτα

Η πλακέτα του rpi4 αποτελείται από ένα σύνολο ηλεκτρονικών εξαρτημάτων τα οποία μαζί αποτελούν το υλικό του κομμάτι. Η ανάλυση αυτών των επιμέρους στοιχείων γίνεται παρακάτω [54]:

- **Επεξεργαστής:** Broadcom BCM2711, quad-core Cortex-A72 (ARM v8) 64-bit SoC @ 1.8GHz
- **Μνήμη RAM:** 4GB LPDDR4-3200 SDRAM
- **Wi-Fi:** 2.4 GHz και 5.0 GHz IEEE 802.11ac ασύρματο
- **Bluetooth:** Bluetooth 5.0, BLE
- **Θύρα Ethernet:** Gigabit Ethernet για την σύνδεση με το διαδίκτυο. Επίσης μπορεί να χρησιμοποιηθεί ως θύρα για την τροφοδοσία του rpi4 τοποθετώντας τη συσκευή PoE HAT που αγοράζεται ξεχωριστά
- **Θύρες USB:** 2 θύρες USB 3.0 και 2 θύρες USB 2.0
- **Ακροδέκτες GPIO:** 40 ακροδέκτες GPIO header του Raspberry Pi, πλήρως συμβατοί με προηγούμενες εκδόσεις του Raspberry Pi οι οποίοι φαίνονται αναλυτικά στην εικόνα 2.12



Εικόνα 2.12 Οι ακροδέκτες GPIO του Raspberry Pi 4 [55]

- **Θύρες HDMI:** 2 θύρες micro-HDMI για σύνδεση μέχρι και δύο οθονών (με υποστήριξη ανάλυσης μέχρι και 4kp60)
- **Θύρα MIPI DSI:** 1 θύρα 2 σειρών MIPI DSI για τη σύνδεση μιας οθόνης κινητού για παράδειγμα
- **Θύρα MIPI CSI:** 1 θύρα 2 σειρών MIPI CSI για τη σύνδεση της κάμερας
- **Θύρα ήχου και βίντεο:** 1 θύρα τεσσάρων-πόλων στερεοφωνικού ήχου και σύνθετου βίντεο
- **Αποκωδικοποίηση/κωδικοποίηση και απόδοση γραφικών:** H.265 (4kp60 decode), H264 (1080p60 decode, 1080p30 encode), OpenGL ES 3.1, Vulkan 1.0

- **Θύρα κάρτας SD:** 1 θύρα για την κάρτα SD που περιέχει το λειτουργικό σύστημα και λειτουργεί σαν σκληρός
- **Θύρα USB-C:** 1 θύρα USB-C για την τροφοδοσία του rpi4 των 5V DC και με ελάχιστη τιμή 3A, εάν βέβαια το rpi4 δεν καταναλώνει συνολικά πάνω από 500mA τότε μπορεί να τροφοδοτηθεί και με 2.5A
- **Ακροδέκτης τροφοδοσίας:** 1 ακροδέκτης που παρέχει 5V DC και 3A/2.5A τροφοδοσία
- **Θερμοκρασίας λειτουργίας:** Σε περιβάλλον που έχει 0 έως 50 βαθμούς Κελσίου

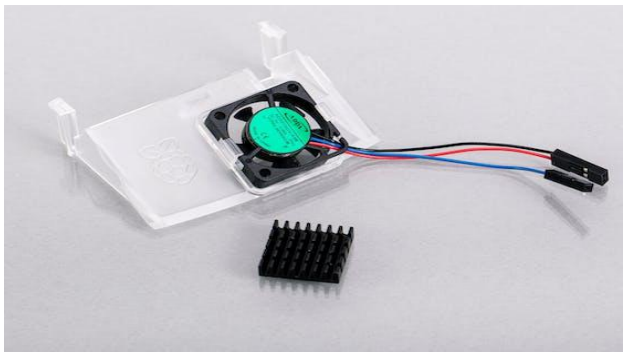
2.13.1.2 Περιφερειακά

Το rpi4 εκτός από την ίδια την πλακέτα του, χρειάζεται κάποια επιπλέον περιφερειακά εξαρτήματα για να λειτουργήσει σωστά. Τα εξαρτήματα αυτά που χρησιμοποιήθηκαν για την επιτυχή ολοκλήρωση της εργασίας αναλύονται παρακάτω μαζί με τα χαρακτηριστικά τους:

- **Θήκη προστασίας:** Για το rpi4 χρησιμοποιήθηκε μία μαύρη θήκη προστασίας από υψηλής ποιότητας υλικά και συγκεκριμένα από συμπολυμερές ακρυλονιτριλίου-βουταδιενίου-στυρενίου (Acrylonitrile Butadiene Styrene/ABS).

Εικόνα 2.13 Η θήκη του Raspberry Pi 4 [56]

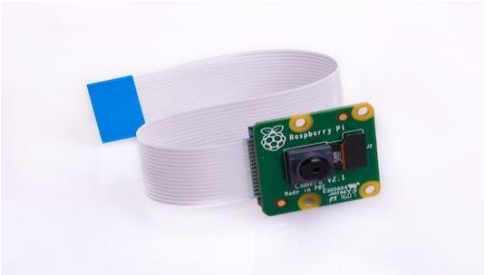
- **Τροφοδοσία:** Για την τροφοδοσία του rpi4 χρησιμοποιήθηκε ένα καλώδιο USB-C σε USB-A της εταιρίας Baseus, μαζί με έναν μετασχηματιστή της εταιρίας Powertech που παράγει 6.5V και 3A.
- **Ψύξη:** Για την ψύξη του επεξεργαστή χρησιμοποιήθηκε μία μικρή ψήκτρα και ένα μικρό ανεμιστηράκι συμβατό με τη θήκη προστασίας, το οποίο συνδέθηκε στους ακροδέκτες 5V και γείωση.



Εικόνα 2.14 Η ψήκτρα και το ανεμιστηράκι του Raspberry Pi 4 [57]

- **Μνήμη SD:** Για την εκτέλεση του λειτουργικού συστήματος και την αποθήκευση των αρχείων της εφαρμογής που δημιουργήθηκε, χρησιμοποιήθηκε η κάρτα microSDHC των 32GB class 10 της εταιρίας Intenso. Θα μπορούσε να χρησιμοποιηθεί οποιαδήποτε SD κάρτα τουλάχιστον 8GB αλλά επιλέχθηκε η συγκεκριμένη διότι η περισσότερη μνήμη επιτυγχάνει περισσότερο το rpi4. Επιπλέον η κλάση 10 επιλέχθηκε και αυτή για τον λόγο ότι προσφέρει μεγαλύτερες ταχύτητες μεταφοράς δεδομένων σε αντίθεση με άλλες κλάσεις.

- **Raspberry Pi Camera V2:** Ένα από τα πιο σημαντικά εξαρτήματα για την υλοποίηση αυτής της διπλωματικής εργασίας είναι η κάμερα V2 του Raspberry Pi. Η συγκεκριμένη κάμερα συνδέεται μέσω καλωδιοταινίας στην υποδοχή MIPI CSI του rpi4 και έχει τον αισθητήρα IMX219 8-megapixel της εταιρίας Sony. Επίσης υποστηρίζει λειτουργίες βίντεο στα 1080p30, 720p60 και VGA90 [58].



Εικόνα 2.15 Η κάμερα V2 του Raspberry Pi 4 [58]

2.13.2 Λογισμικό μέρος

2.13.2.1 Λειτουργικό σύστημα Debian 11

Το λειτουργικό σύστημα που εγκαταστάθηκε στην κάρτα SD του Raspberry Pi 4 ονομάζεται Debian με έκδοση νούμερο 11 και ψευδώνυμο bullseye. Το Debian είναι βασισμένο στα Linux οπότε προσφέρει αξιοπιστία, σταθερότητα και υπόσχεται υποστήριξη για αρκετά χρόνια αφού το κοινό που το χρησιμοποιεί είναι ενεργό και μεγάλο.

Η συγκεκριμένη έκδοση που εγκαταστάθηκε, παρέχει γραφικό περιβάλλον που είναι φιλικό προς τον χρήστη ειδικά αν θέλει πρώτα κάποιος να εξοικειωθεί με τα Linux συστήματα. Αυτό το περιβάλλον κατά την πρώτη εκκίνηση του rpi4 ξεκινά από μία επιφάνεια εργασίας και κάποια μηνύματα που καθοδηγούν το χρήστη για να γνωρίσει καλύτερα το Debian. Επίσης περιλαμβάνονται πολλές εφαρμογές προεγκατεστημένες για την διευκόλυνση του χρήστη, όπως ο περιηγητής chromium ή το περιβάλλον προγραμματισμού Thonny.

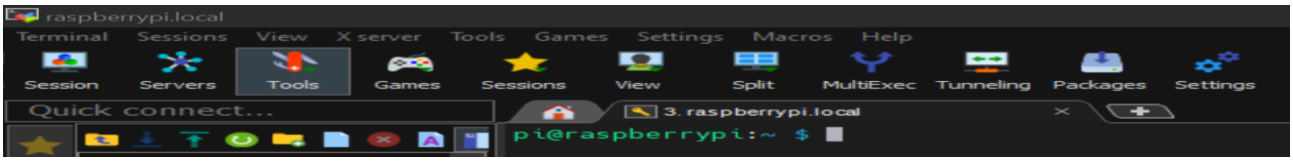
Κάποια πιο συγκεκριμένα χαρακτηριστικά για το λειτουργικό σύστημα του Debian 11 στο rpi4 είναι τα εξής:

- **Ημερομηνία έκδοσης:** 3 Μαΐου 2023
- **Αρχιτεκτονική συστήματος:** 64-bit
- **Έκδοση Kernel (πυρήνα):** 6.1.34-v8+
- **Μέγεθος:** 2.6 GB

2.13.2.2 Ρυθμίσεις και προετοιμασία για την εφαρμογή ανίχνευσης μάσκας

Αφού εγκαταστάθηκε το λειτουργικό σύστημα Debian 11 bullseye στην κάρτα microSD, αυτή τοποθετήθηκε στο rpi4 και αυτό με τη σειρά του συνδέθηκε στην τροφοδοσία για να πραγματοποιηθεί η πρώτη του εκκίνηση. Κατά την πρώτη του εκκίνηση ακολουθήσαμε κάποια βήματα για την σωστή ρύθμιση, ενημέρωση και αναβάθμιση του rpi4.

Αρχικά έγινε σύνδεση στο τερματικό περιβάλλον του rpi4, αφού για τις ενέργειες που ακολούθησαν, το γραφικό περιβάλλον δεν ευνοούσε σε κάποιο κομμάτι. Έτσι ανοίγοντας το λογισμικό MobaXterm στον σταθερό υπολογιστή πραγματοποιήσαμε την πρώτη σύνδεση στο rpi4 διαλέγοντας την επιλογή Session, SSH και πληκτρολογώντας στο πεδίο “Remote host” το “raspberrypi.local”. Αφού ο υπολογιστής και το rpi4 ήταν συνδεδεμένα στο ίδιο δίκτυο, η σύνδεση αποδείχθηκε επιτυχής με το τερματικό να ζητάει πλέον τα στοιχεία του rpi4 που είχαμε δώσει στο πρόγραμμα Imager πριν την εγκατάσταση του λειτουργικού. Εισάγοντας τα στοιχεία μπορούσαμε πλέον να ξεκινήσουμε όλες τις απαραίτητες ενέργειες για τον έλεγχο και την προετοιμασία του rpi4.



Εικόνα 2.16 Στιγμιότυπο από την πρώτη επιτυχή σύνδεση του σταθερού υπολογιστή στο τερματικό περιβάλλον του Raspberry Pi 4

Πρώτα εκτελέσαμε ένα σύνολο από εντολές οι οποίες φαίνονται παρακάτω μαζί με την επεξήγησή τους:

- **sudo apt update:** Η εντολή "sudo" δίνει δικαιώματα διαχειριστή πριν την εκτέλεση μιας εντολής έτσι ώστε να μην υπάρξει άρνηση εκτέλεσης της εντολής από το λειτουργικό σύστημα. Η εντολή "apt update" ενημερώνει το σύστημα μέσω των κατάλληλων αποθετηρίων για τυχόν νέα πακέτα αναβαθμίσεων των βιβλιοθηκών ή εφαρμογών που περιέχει.

- **sudo apt upgrade -y:** Με το "apt upgrade" γίνεται η εγκατάσταση όλων αυτών των πιθανών νέων ενημερώσεων που έλαβε η προηγούμενη εντολή έτσι ώστε τα προγράμματα και οι βιβλιοθήκες να έχουν τις πιο νέες εκδόσεις τους. Το "-y" (από το αγγλικό "yes") ενεργοποιεί την αυτόματη αποδοχή της εγκατάστασης των διαφόρων νέων πακέτων χωρίς να χρειαστεί ο χρήστης να πληκτρολογεί κάθε φορά αν συμφωνεί με την εγκατάσταση ή όχι.

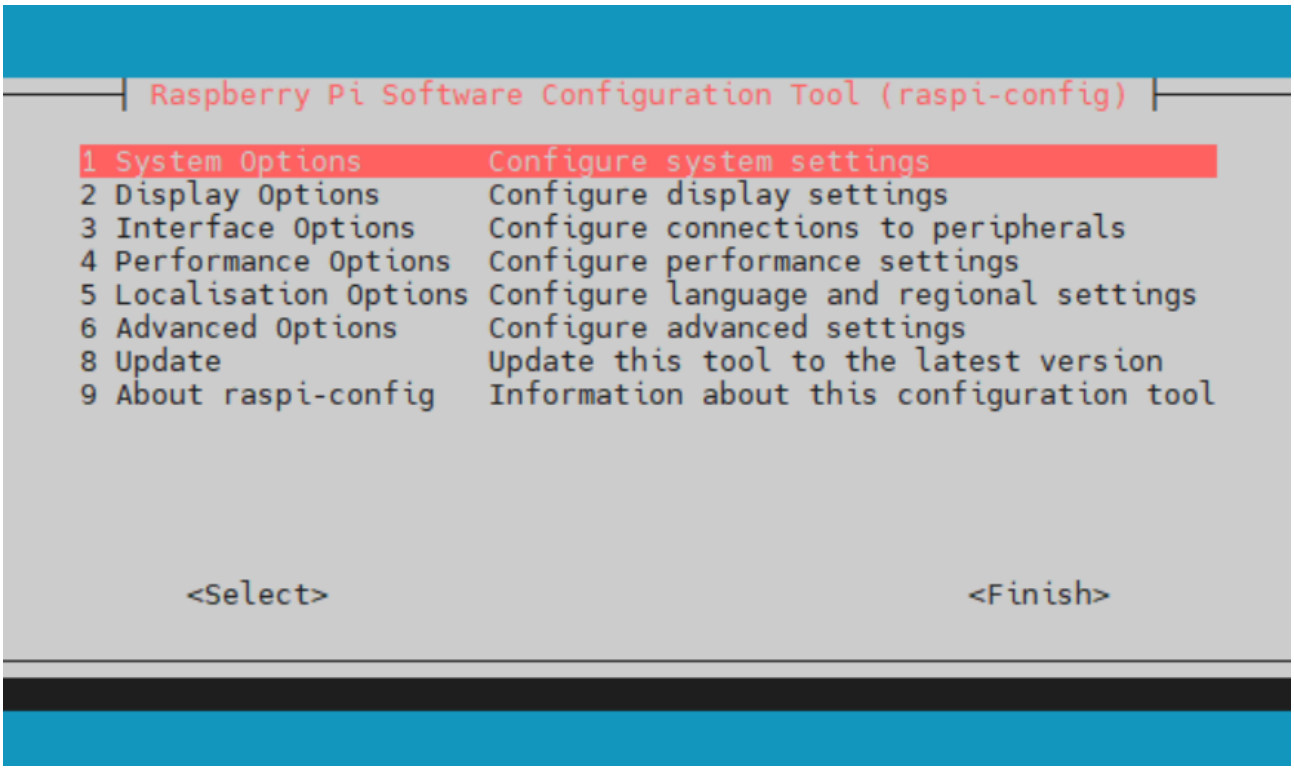
- **sudo rpi-update: Α**υτή η εντολή αναβαθμίζει το ίδιο το rpi4, δηλαδή το υλικολογισμικό (firmware) του. Γενικότερα αυτή η εντολή αποφεύγεται να εκτελείται, γιατί υπάρχει περίπτωση μετά την ενημέρωση να μην συνεργάζεται σωστά το λειτουργικό σύστημα με το υλικό. Στην περίπτωσή μας έγινε δοκιμή αυτής της εντολής, η οποία πέτυχε και όλα λειτούργησαν σωστά με το σύστημα να είναι πλέον πλήρως αναβαθμισμένο.

- **sudo reboot:** Εκτέλεση επανεκκίνησης, έτσι ώστε το σύστημα να ξεκινήσει ανανεωμένο μετά από όλες τις μαζεμένες ενημερώσεις και αλλαγές που δέχτηκε.

- **sudo apt-get install -y python3-pip:** Εγκατάσταση του pip, δηλαδή του επίσημου διαχειριστή πακέτων της Python 3. Με το pip δηλαδή μπορούσαμε αργότερα να εγκαταστήσουμε όλες τις απαραίτητες βιβλιοθήκες για την εφαρμογή που φτιάξαμε. Είναι ένα πολύ χρήσιμο και απαραίτητο εργαλείο, ειδικά για προγραμματιστές της γλώσσας Python.

- **sudo pip3 install --upgrade setuptools:** Αναβάθμιση μέσω της εντολής "pip3", που εγκαταστάθηκε πριν, του πακέτου "setuptools". Το "setuptools" είναι ένα πακέτο που διευκολύνει την εγκατάσταση πακέτων για την Python 3.

- **Sudo raspi-config:** Αυτή η εντολή εμφανίζει ένα παράθυρο ρυθμίσεων που θυμίζει κατά κάποιον τρόπο το μενού των bios ενός σταθερού υπολογιστή με Windows. Δηλαδή περιέχει μέσα ρυθμίσεις όπως η αλλαγή των στοιχείων σύνδεσης του rpi4 με το διαδίκτυο, η αλλαγή της ανάλυσης της οθόνης, η ενεργοποίησης ή απενεργοποίηση πρωτοκόλλων όπως το SSH και άλλα. Εμείς εκτελέσαμε αυτήν την εντολή συγκεκριμένα για τρείς λόγους. Πρώτον για την ενεργοποίηση της εφαρμογής VNC Server, δεύτερον για την ενεργοποίηση της διεπαφής που βρίσκεται συνδεδεμένη η κάμερα V2 πάνω στο rpi4, ώστε αυτή να αναγνωρίζεται από το σύστημα και να λειτουργεί. Τρίτον για την αύξηση του ορίου χρήσης της μνήμης της κάρτας γραφικών (GPU) του επεξεργαστή από τα 128 MB στα 256 MB με στόχο την ταχύτερη εκτέλεση των διαφόρων διεργασιών στο rpi4.

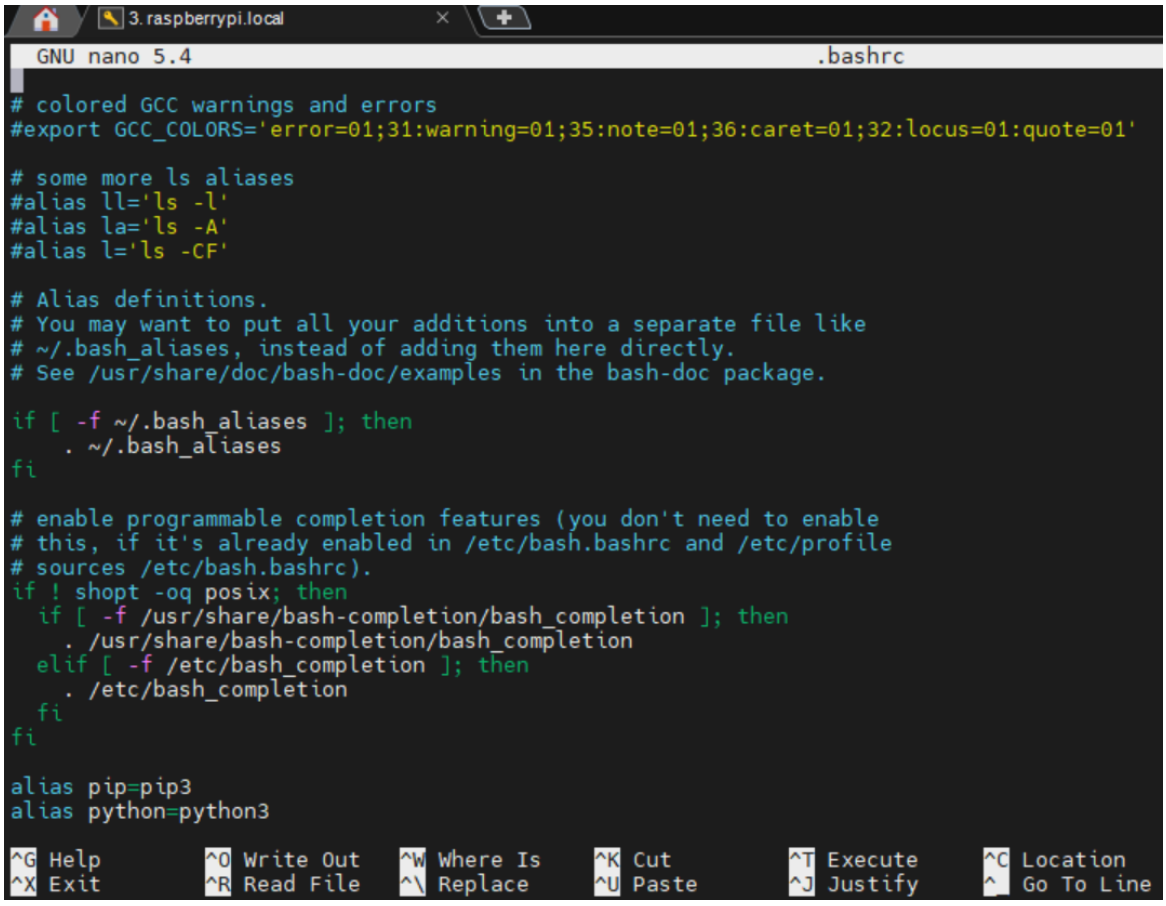


Εικόνα 2.17 Στιγμιότυπο από το μενού, μετά την εκτέλεση της εντολής "sudo raspi-config "

Μετά από τις παραπάνω εντολές εκτελέστηκε πάλι επανεκκίνηση για να ανανεωθεί το rpi4 και να βεβαιωθούμε ότι όλες οι αλλαγές δεν προκάλεσαν κάποια βλάβη στο λειτουργικό. Αφού όλα λειτουργούσαν με επιτυχία συνεχίσαμε ανοίγοντας το αρχείο ".bashrc" ως κείμενο με την βοήθεια της εφαρμογής nano με σκοπό την επεξεργασία των πληροφοριών που περιέχει. Η ακριβής εντολή ήταν "sudo nano .bashrc" και η τοποθεσία του αρχείου ήταν στο "/home/pi/". Αφού το αρχείο άνοιξε, προσθέσαμε δυο σειρές στο τέλος του κειμένου που περιείχε ήδη, οι οποίες φαίνονται ως εξής:

- **alias pip=pip3:** Δίνεται το ψευδώνυμο "pip" για την εντολή "pip3"
- **alias python=python3:** Δίνεται το ψευδώνυμο "python" για την εντολή "python3"

Το αρχείο ".bashrc" γενικότερα είναι ένα αρχείο bash και σκοπός του είναι να ενημερώνει το περιβάλλον του λειτουργικού συστήματος με τις πληροφορίες που περιέχονται σε αυτό. Έτσι με την προσθήκη των δύο παραπάνω σειρών ενημερώνουμε το σύστημα πως πλέον όποτε θα πληκτρολογούμε τη λέξη pip θα είναι σαν να εννοούμε pip3 και όπου python θα είναι σαν να εννοούμε python3. Οπότε με την ενημέρωση του αρχείου μένει η αποθήκευσή του και έπειτα η εκτέλεση της εντολής "source ~/.bashrc" η οποία κάνει μία επανεκκίνηση στην εκτέλεση του αρχείου ώστε το λειτουργικό σύστημα να πληροφορηθεί για τις νέες αλλαγές.



Εικόνα 2.18 Στιγμιότυπο από το εσωτερικό του αρχείου .bashrc κατά την προσθήκη των δύο ψευδωνύμων στο τέλος του

Στη συνέχεια έγινε αλλαγή της δυναμικής ip του rpi4 σε στατική για να το εντοπίζουμε εύκολα στο δίκτυο και να μην αλλάζει διεύθυνση με κάθε νέα εκκίνησή του. Συγκεκριμένα του δόθηκε η διεύθυνση “192.168.1.23” που ήταν σίγουρα διαθέσιμη εκείνη τη στιγμή.

Για την αλλαγή των ρυθμίσεων της ip έπρεπε να ανοιχτεί σε μορφή κειμένου το αρχείο διαχείρισης διεπαφών δικτύου και ρυθμίσεων σχετικά με το πρωτόκολλο DHCP (Dynamic Host Configuration Protocol). Το όνομα του αρχείου αυτού είναι το “dhcpcd.conf” και ανοίχτηκε για επεξεργασία με την εντολή “sudo nano /etc/dhcpcd.conf”.

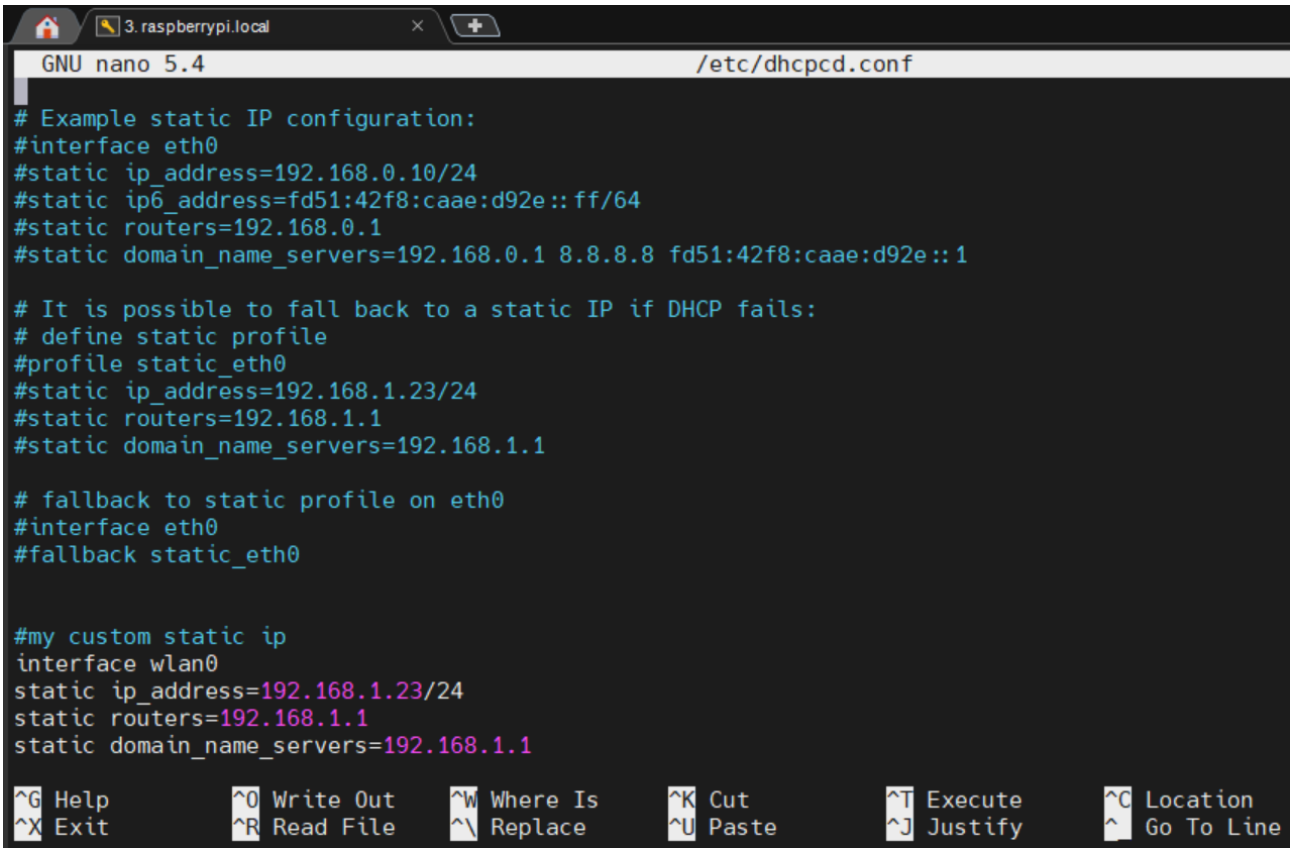
Έτσι στο τέλος του αρχείου προστέθηκαν οι παρακάτω εντολές, όπως φαίνεται και στην εικόνα 2.19, οι οποίες έπειτα αποθηκεύτηκαν και εκτελέστηκε επανεκκίνηση για να τεθούν σε λειτουργία:

interface wlan0

static ip_address=192.168.1.23/24

static routers=192.168.1.1

static domain_name_servers=192.168.1.1



Εικόνα 2.19 Στιγμιότυπο από το εσωτερικό του αρχείου "dhcpcd.conf" κατά την μετατροπή της ip σε στατική

Ακόμα, είναι σημαντικό να αναφερθούν κάποιες εντολές που αποδείχθηκαν γενικότερα χρήσιμες κατά την πορεία αυτής της διπλωματικής εργασίας και επιλύσαν διάφορα προβλήματα:

- **vcgencmd get_camera:** Αυτή είναι η πρώτη εντολή που χρησιμοποιείται για να μας απαντήσει το λειτουργικό εάν αναγνώρισε την κάμερα που συνδέσαμε στο rpi4.
- **libcamera-hello --qt-preview:** Εκτέλεση της εφαρμογής "-hello" της βιβλιοθήκης "libcamera" για να συμπεράνουμε εάν λειτουργεί η κάμερα του rpi4 σωστά. Η εφαρμογή αυτή ουσιαστικά ανοίγει ένα γραφικό παράθυρο για περίπου 5 δευτερόλεπτα και δείχνει σε ζωντανό βίντεο ότι βλέπει η κάμερα. Το "--qt-preview" δίνει εντολή να ανοίξει το γραφικό παράθυρο στο πλαίσιο (framework) του QT.
- **libcamera-jpeg -o test.jpg:** Αυτή η εντολή χρησίμευσε επίσης για την δοκιμή της σωστής λειτουργίας της κάμερας. Η διαφορά με την προηγούμενη εντολή είναι ότι αποτυπώνεται μία στιγμιαία φωτογραφία μέσω της εφαρμογής "-jpeg". Αυτή αποθηκεύεται στο σημείο που βρισκόμαστε εκείνη τη στιγμή μέσα στο rpi4 με όνομα "test.jpg".
- **sudo libcamera-vid -t 10000 -o test.h264:** Αυτή η εντολή πραγματοποιεί εγγραφή βίντεο με την εφαρμογή "-vid", διάρκειας 10000 milliseconds (10 δευτερόλεπτα) και το αποθηκεύει με όνομα "test.h264".
- **sudo chmod -R 755 /τοποθεσία_επιθυμητού_φακέλου/:** Δίνεται το δικαίωμα στον χρήστη του rpi4 από εκείνη τη στιγμή και πέρα να μπορεί να ανοίγει για να διαβάζει το περιεχόμενο

των αρχείων του φακέλου αυτού που επιλέχθηκε. Αυτή η εντολή χρησιμοποιείται σε περίπτωση που το σύστημα αρνήθηκε να μας δώσει πρόσβαση σε κάποιο αρχείο που θέλουμε να επεξεργαστούμε. Το "-R" επισημαίνει στην εντολή να δώσει δικαίωμα ανάγνωσης σε όλα τα αρχεία που περιέχονται στον επιλεγμένο φάκελο.

- **sudo chmod -R 777 /τοποθεσία_επιθυμητού_φακέλου/:** Αυτή η εντολή λειτουργεί όπως ακριβώς η προηγούμενη και συνήθως εκτελούνται μαζί. Η διαφορά τους είναι πως αυτή η εντολή δίνει δικαιώματα και για αλλαγή του περιεχομένου των αρχείων και όχι μόνο για την ανάγνωσή τους.

- **sudo systemctl isolate multi-user.target:** Αυτή η εντολή απενεργοποιεί προσωρινά το γραφικό περιβάλλον του rpi4 και ενεργοποιεί την λειτουργία τερματικού. Αυτό βοηθάει στην εξοικονόμηση πόρων του rpi4 άρα την γρηγορότερη εκτέλεση της εφαρμογής ανίχνευσης μάσκας σε πρόσωπα ή άλλων διεργασιών. Μετά την επανεκκίνηση το σύστημα επανέρχεται στην μορφή που είχε, δηλαδή με γραφικό περιβάλλον.

- **sudo startx:** Αυτή η εντολή ενεργοποιεί άμεσα το γραφικό περιβάλλον του rpi4, σε περίπτωση που αλλάξαμε γνώμη ή δεν θέλουμε να κάνουμε αναγκαστικά επανεκκίνηση.

2.13.3 Απαραίτητα προγράμματα

2.13.3.1 VNC Server

Όπως προαναφέρθηκε στην ενότητα 2.13.2.2, πραγματοποιήθηκε ενεργοποίηση του προεγκατεστημένου προγράμματος VNC Server με σκοπό την επιτυχή σύνδεσή του με το πρόγραμμα VNC Viewer του σταθερού υπολογιστή. Ο VNC Server δίνει την δυνατότητα σε έναν χρήστη να συνδεθεί από τον υπολογιστή του μέσω του προγράμματος VNC viewer στο γραφικό περιβάλλον του rpi4.

Ειδικότερα το VNC Viewer μπορεί να δει κάποια από τα γραφικά που εμφανίζονται στο rpi4, όπως το άνοιγμα ενός παραθύρου που περιέχει μία εικόνα ή ένα βίντεο. Έτσι όταν θα εκτελεστεί ο κώδικας που θα ανοίγει ζωντανό βίντεο μέσω της κάμερας του rpi4, αυτό θα μπορεί να προβληθεί και στον υπολογιστή που χειρίζεται ο χρήστης το rpi4 αυτό.

Επιπλέον ο χρήστης του υπολογιστή θα πρέπει να γνωρίζει το όνομα χρήστη, τον κωδικό, καθώς και την IP διεύθυνση του rpi4 προκειμένου να μπορεί να συνδεθεί σε αυτό με επιτυχία.

2.13.3.2 Pyenv

Όπως και στον σταθερό υπολογιστή έτσι και στο rpi4, για την ανάπτυξη κώδικα χρειάστηκε να χρησιμοποιήσουμε ένα πρόγραμμα δημιουργίας εικονικών περιβαλλόντων. Το Pyenv είναι ένα πρόγραμμα αντίστοιχο του Anaconda που αναφέρθηκε στην ενότητα 2.4. Αυτό σημαίνει πως μας δίνει τη δυνατότητα να δημιουργήσουμε το δικό μας εικονικό περιβάλλον, μέσα στο οποίο μπορούμε να εγκαταστήσουμε όποια έκδοση Python θέλουμε και όποιες άλλες συγκεκριμένες βιβλιοθήκες, συγκεκριμένων εκδόσεων θέλουμε.

Το πλεονέκτημα, όπως και με το Anaconda, είναι ότι μπορούμε να κάνουμε δοκιμές με διαφορετικές εκδόσεις βιβλιοθηκών για να ελέγξουμε εάν η εφαρμογή μας λειτουργεί καλύτερα ή χειρότερα με κάποια συγκεκριμένη έκδοση. Επίσης αποφεύγεται το πρόβλημα της “σύγκρουσης” μεταξύ ήδη εγκατεστημένων βιβλιοθηκών και πακέτων γενικά στο rpi4, αφού με το εικονικό περιβάλλον απομονώνουμε ότι δημιουργούμε εκεί μέσα.

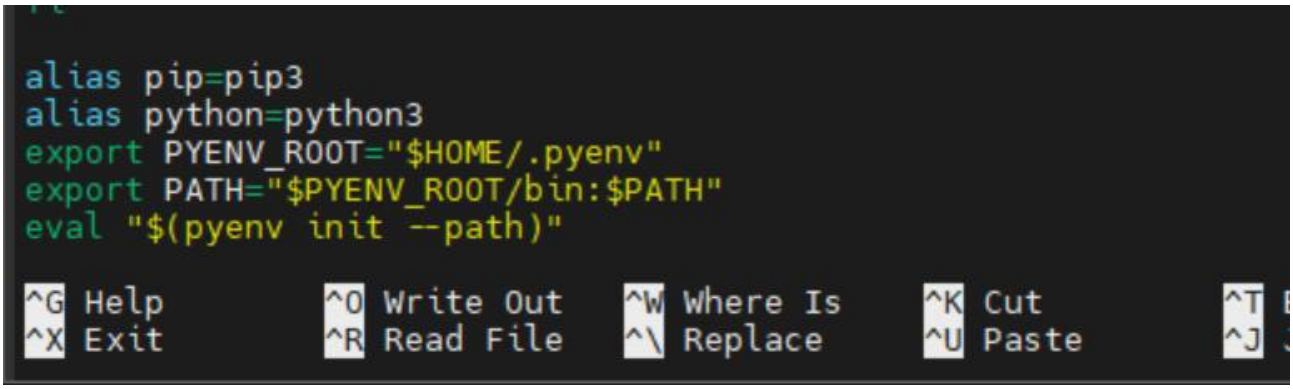
Στη συνέχεια περιγράφεται η διαδικασία εγκατάστασης του Pyenv καθώς και η δημιουργία ενός εικονικού περιβάλλοντος. Αρχικά πραγματοποιήθηκε η εγκατάσταση των πακέτων από τα οποία εξαρτάται το Pyenv και είναι προαπαιτούμενα για τη σωστή λειτουργία του με την εξής εντολή:

- sudo apt install -y make build-essential libssl-dev zlib1g-dev libbz2-dev \ libreadline-dev libsqlite3-dev wget curl llvm libncurses5-dev libncursesw5-dev \ xz-utils tk-dev libffi-dev liblzma-dev python-openssl git

Στη συνέχεια έγινε εγκατάσταση του git με την εντολή “ sudo apt install git” έτσι ώστε να μπορούμε να κλωνοποιήσουμε το Pyenv, από το αποθετήριο του Github. Επίσης το Git θα μας βοηθήσει αργότερα να κλωνοποιήσουμε τους δικούς μας κώδικες μέσα στο rpi4 με ευκολία. Έτσι μέσω της εντολής “git clone https://github.com/pyenv/pyenv.git ~/.pyenv” κλωνοποιήθηκε το Pyenv μέσα στον φάκελο “pyenv” του rpi4.

Έπειτα θέλαμε να ενημερώσουμε το λειτουργικό σύστημα να αναγνωρίζει την εντολή “pyenv” από οποιοδήποτε σημείο του rpi4 και αν πληκτρολογούσαμε μέσω του τερματικού το “pyenv”. Γι’ αυτό χρειάστηκε για ακόμη μια φορά η ενημέρωση του αρχείου “.bashrc” με τις παρακάτω εντολές:

echo 'export PYENV_ROOT="$HOME/.pyenv"' >> ~/.bashrc
echo 'export PATH="$PYENV_ROOT/bin:$PATH"' >> ~/.bashrc
echo 'eval "$(pyenv init --path)"' >> ~/.bashrc



Εικόνα 2.20 Οι τελευταίες γραμμές του αρχείου “.bashrc” όπως έχουν διαμορφωθεί μέχρι τώρα

Αυτή τη φορά δεν ανοίξαμε το αρχείο με την εντολή sudo nano αλλά προσθέσαμε κατευθείαν μέσα στο τέλος του αρχείου τις παραπάνω πληροφορίες με την εντολή echo. Αμέσως μετά εκτελώντας την εντολή “source ~/.bashrc” ανανεώσαμε το σύστημα ώστε να ενημερωθεί με τις νέες πληροφορίες. Επίσης με την εκτέλεση της εντολής “pyenv --version” επαληθεύσαμε πως το Pyenv εγκαταστάθηκε με επιτυχία.

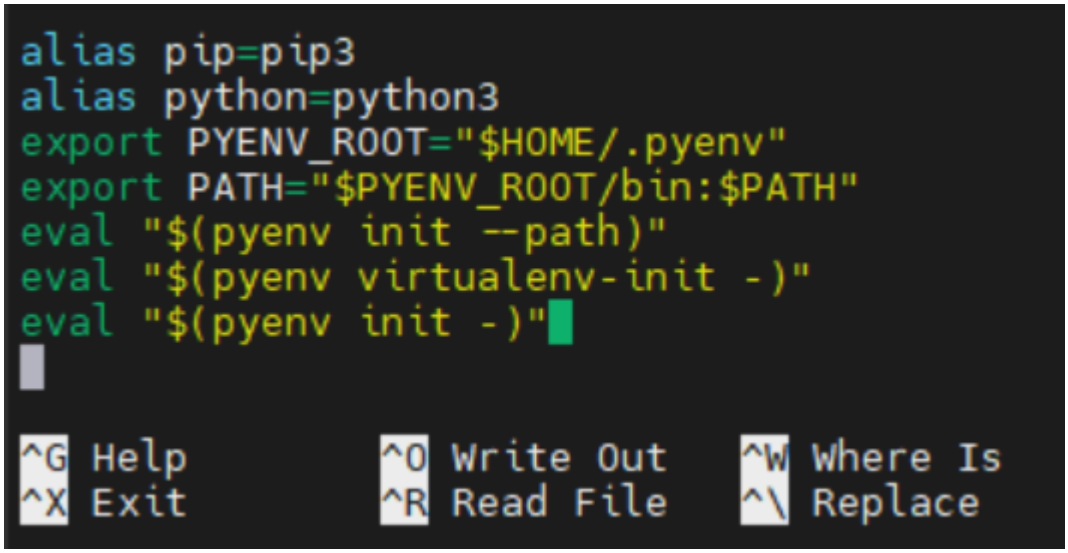
Με την ολοκλήρωση της εγκατάστασης του Pyenv ξεκινά η διαδικασία δημιουργίας του εικονικού περιβάλλοντος που θα δουλέψουμε τους κώδικες. Στην αρχή γίνεται εγκατάσταση των προαπαιτούμενων πακέτων της Python 3.10 πριν την εγκαταστήσουμε στο εικονικό περιβάλλον. Έτσι ξεκινάμε με την εντολή εγκατάστασης:

- sudo apt-get install libbz2-dev libncurses5-dev libffi-dev libreadline-dev libssl-dev libsqlite3-dev python3-tk build-essential tk-dev liblzma-dev

Στη συνέχεια πραγματοποιείται εγκατάσταση της έκδοσης Python που έχει σχεδιαστεί ο κώδικας και άρα θα τρέξει αργότερα. Αυτό γίνεται με την εντολή ”pyenv install 3.10.11”. Ακολουθεί η εγκατάσταση του πρόσθετου εργαλείου “virtualenv” από το Github που θα κάνει τις εντολές πιο εύκολες και κατανοητές στη συνέχεια. Αυτό πραγματοποιείται εκτελώντας πρώτα την εντολή “git clone https://github.com/pyenv/pyenv-virtualenv.git” $(pyenv root)/plugins/pyenv-virtualenv” που αντιγράφει το εργαλείο μέσα στο φάκελο του .Pyenv.

Έπειτα προσθέτουμε στο αρχείο “.bashrc” τις δύο γραμμές:
eval "$(pyenv virtualenv-init -)"
eval "$(pyenv init -)"

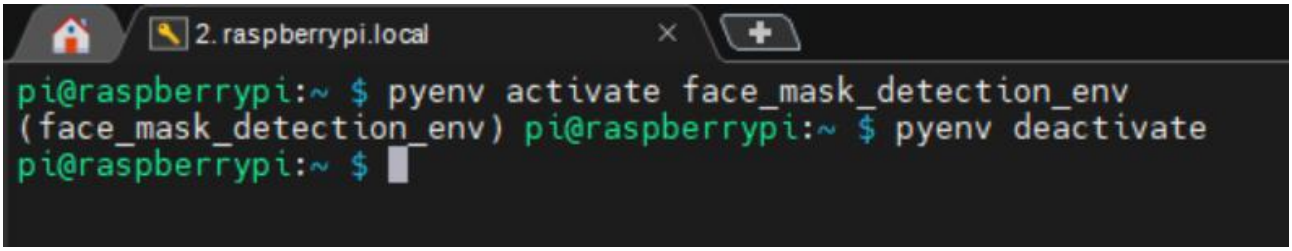
Αυτές οι δύο γραμμές αποσκοπούν στην ενεργοποίηση και την αναγνώριση του νέου εργαλείου από το λειτουργικό σύστημα. Ασφαλώς χρειάζεται πάλι η εντολή “source ~/.bashrc” για την ενημέρωση των παραπάνω στο σύστημα.



Εικόνα 2.21 Τελική διαμόρφωση των τελευταίων γραμμών του αρχείου “.bashrc”

Σε αυτό το σημείο θα δημιουργήσουμε το εικονικό περιβάλλον με όνομα face_mask_detection_env, χρησιμοποιώντας την εντολή “pyenv virtualenv 3.10.11 face_mask_detection_env”.

Για την ενεργοποίηση και τη χρήση αυτού του εικονικού περιβάλλοντος εκτελείται η εντολή “pyenv activate face_mask_detection_env”. Μόλις ενεργοποιηθεί το εικονικό περιβάλλον , θα εμφανιστεί σε παρένθεση το όνομά του, πράγμα που σημαίνει πως η διαδικασία μέχρι τώρα ήταν επιτυχής.



Εικόνα 2.22 Επιτυχής ενεργοποίηση και απενεργοποίηση του εικονικού περιβάλλοντος “face_mask_detection_env”

Σε αυτό το σημείο συνεχίζουμε με την εγκατάσταση όλων των υπόλοιπων απαραίτητων πακέτων και βιβλιοθηκών που θα χρειαστεί ο κώδικας για να τρέξει. Η εγκατάστασή τους γίνεται με την εντολή “pip install package_name” όπου “package_name” το επιθυμητό πακέτο εγκατάστασης.

Στην περίπτωση αυτής της διπλωματικής εργασίας είχε δημιουργηθεί κατά την υλοποίηση του κώδικα το έγγραφο “requirements.txt” το οποίο περιέχει όλες τις πληροφορίες σχετικά με τα απαραίτητα πακέτα και τις ακριβείς εκδόσεις τους. Οπότε χάρη στην βοήθεια της εντολής “pip”

μπορεί να πραγματοποιηθεί η εγκατάστασή τους κατευθείαν από αυτό το αρχείο με την εκτέλεση της εντολής “pip install -r requirements”.

Βέβαια για να εκτελεστεί αυτή η εντολή με επιτυχία πρέπει πρώτα να μεταφερθεί το αρχείο “requirements.txt” στο rpi4 μαζί με όλα τα υπόλοιπα απαραίτητα αρχεία που έχουν συγκεντρωθεί, οργανωθεί και τοποθετηθεί στο ιδιωτικό αποθετήριο που δημιουργήσαμε στο Github. Αυτά τα αρχεία μπορούν να κλωνοποιηθούν με την εκτέλεση της εντολής ”git clone <url_του_αποθετηρίου>”. Αμέσως δημιουργείται ο φάκελος με το όνομα του αποθετηρίου, δηλαδή “face_mask_detection”, στον οποίο μπορεί κάποιος και να προβάλει τα αρχεία που περιέχει, πληκτρολογώντας πρώτα την εντολή “cd όνομα_φακέλου” η οποία μας μεταφέρει στον επιθυμητό φάκελο.

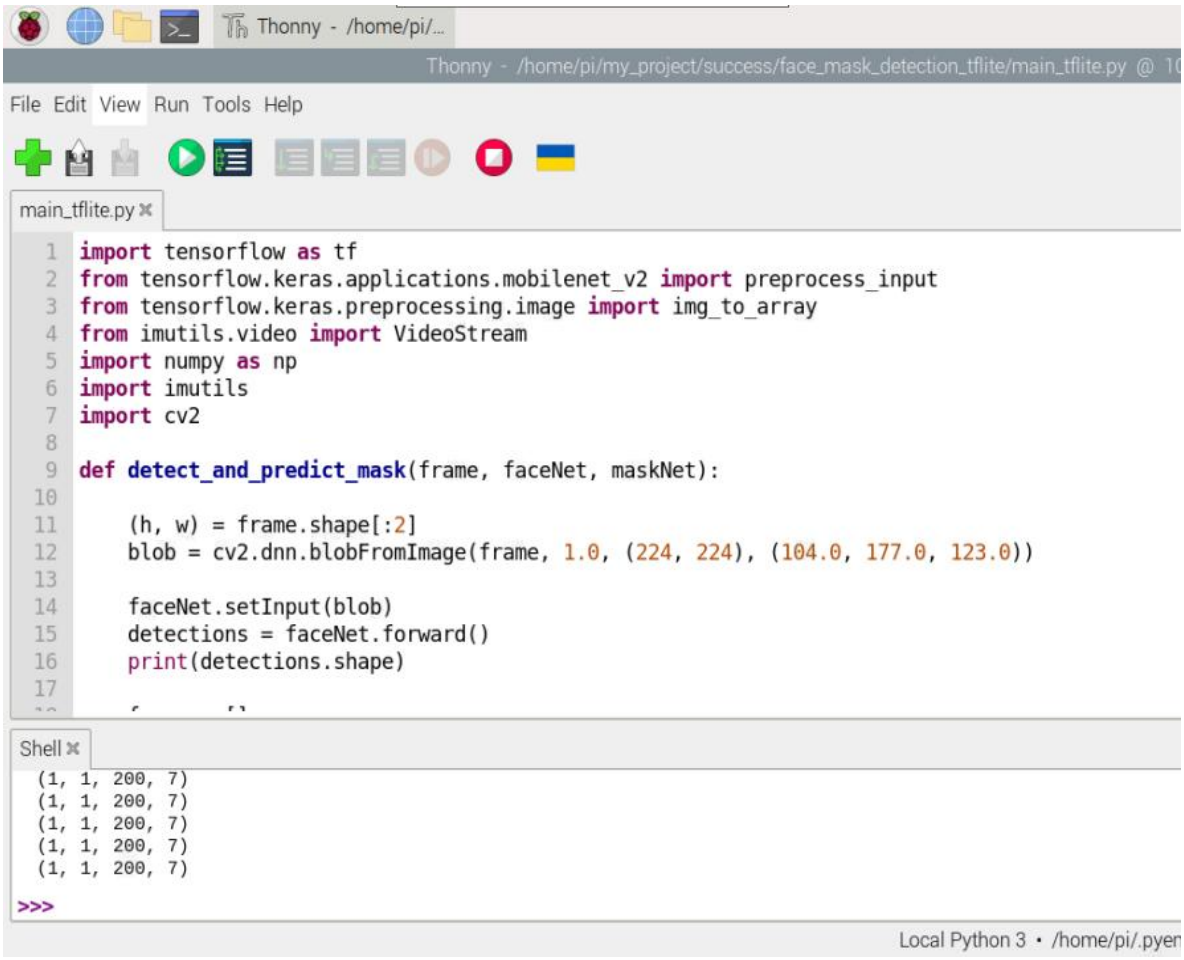
Να επισημανθεί ότι κατά τη δημιουργία του εικονικού περιβάλλοντος “face_mask_detection_env “ φτιάχτηκε ένας φάκελος με το όνομά του και όλα τα σχετικά αρχεία και βιβλιοθήκες με αυτό τοποθετήθηκαν μέσα εκεί. Επιπλέον για την προβολή όλων των πακέτων που έχουν εγκατασταθεί μέχρι στιγμής, μπορεί να εκτελεστεί η εντολή “pip list”.

Μόλις ολοκληρωθούν όλες οι απαραίτητες διαδικασίες και εφόσον δεν χρειάζεται πλέον να είναι ανοιχτό το εικονικό περιβάλλον, αυτό απενεργοποιείται με την εντολή “pyenv deactivate” και έτσι το τερματικό επαναφέρεται στην προηγούμενή του κατάσταση. Σε περίπτωση που θελήσει κάποιος χρήστης να προβάλλει όλα τα εικονικά περιβάλλοντα που έχουν δημιουργηθεί σε μορφή λίστας μπορεί να εκτελέσει την εντολή “pyenv virtualenvs”.

2.13.3.3 Thonny

Το Thonny είναι ένα περιβάλλον ανάπτυξης κώδικα που βρίσκεται προεγκατεστημένο στο rpi4. Με αυτό έγιναν οι δοκιμές και οι περισσότερες αλλαγές στον κώδικα για την προσαρμοσμένη έκδοση της εφαρμογής ανίχνευσης μάσκας σε ανθρώπινα πρόσωπα.

Το περιβάλλον του είναι πολύ φιλικό προς τον χρήστη, εύκολο στον χειρισμό και είναι ειδικά σχεδιασμένο για προγραμματιστές γλώσσας Python. Περιλαμβάνει λειτουργίες όπως η συμπλήρωση κώδικα και η επισήμανση για λάθος εσοχές διευκολύνοντας έτσι τη σύνταξη του κώδικα. Επίσης είναι διαθέσιμο για διάφορα λειτουργικά συστήματα, όπως τα Windows, τα macOS και τα Linux δίνοντας έτσι τη δυνατότητα σε χρήστες με διαφορετικές προτιμήσεις να το χρησιμοποιήσουν.



Εικόνα 2.23 Στιγμιότυπο από το πρόγραμμα ανάπτυξης κώδικα Thonny

3 ΚΕΦΑΛΑΙΟ 3^{ο} : Εκπαίδευση του μοντέλου ανίχνευσης μάσκας

Σε αυτό το κεφάλαιο αναλύεται η διαδικασία εκπαίδευσης του μοντέλου για την ανίχνευση μάσκας στα πρόσωπα των ανθρώπων. Το κεφάλαιο ξεκινά με την πρώτη ενότητα στην οποία περιγράφεται το σύνολο των δεδομένων εκπαίδευσης που επιλέχθηκε και δίνονται κάποια παραδείγματα σχετικά με αυτό.

Έπειτα γίνεται ανάλυση του κώδικα εκπαίδευσης του μοντέλου χωρίζοντάς τον σε μέρη ανάλογα με το αποτέλεσμα που παράγουν οι διάφορες ομάδες εντολών. Να σημειωθεί πως η επεξήγηση του κώδικα στην ενότητα 3.2 γίνεται πιο γενικά έτσι ώστε να μπορεί οποιοσδήποτε να καταλάβει το αποτέλεσμα των γραμμών κώδικα, χωρίς να χρειάζεται να είναι γνώστης της γλώσσας προγραμματισμού Python. Παρόλα αυτά εάν κάποιος ενδιαφέρεται για τις ίδιες τις εντολές, μπορεί να ανατρέξει στο Παράρτημα Α όπου έχει γίνει λεπτομερείς επεξήγηση κάθε γραμμής του κώδικα.

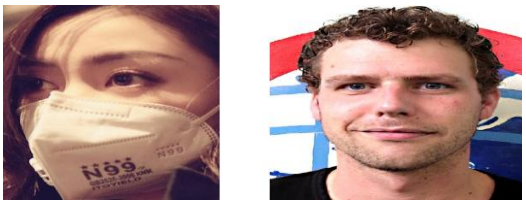
Στο τέλος του κεφαλαίου παρουσιάζονται διάφορα παραδείγματα και αποτελέσματα σχετικά με τα μοντέλα που εκπαιδεύτηκαν.

3.1 Δεδομένα εκπαίδευσης

Για την εκπαίδευση του μοντέλου χρησιμοποιήθηκε ένα σύνολο δεδομένων (dataset) που αποθηκεύτηκε σε έναν φάκελο με ονομασία “dataset” ο οποίος περιέχει δύο υποφακέλους με εικόνες. Οι δύο αυτοί υποφάκελοι αντιπροσωπεύουν τις δύο κλάσεις που θα ταξινομεί τα αποτελέσματά του το μοντέλο, οπότε ο ένας ονομάζεται “with_mask” ενώ ο άλλος “without_mask”.

Ο πρώτος, όπως φαίνεται και από το όνομά του στα αγγλικά, περιέχει εικόνες από ανθρώπους που φοράνε μάσκα, ενώ ο δεύτερος περιέχει εικόνες με ανθρώπους που δεν φοράνε μάσκα. Σχεδόν όλες οι εικόνες και των δύο υποφακέλων απεικονίζουν μόνο τη περιοχή του κεφαλιού των ανθρώπων και όχι το σώμα τους ή γενικά τους ίδιους με κάποιο παρασκήνιο. Αυτό συμβαίνει διότι κατά την εκπαίδευση θέλουμε το μοντέλο να εκπαιδευτεί να αναγνωρίζει μάσκες οι οποίες βρίσκονται στα πρόσωπα ανθρώπων και όχι κάπου γενικά μέσα σε μια εικόνα. Για παράδειγμα αν κάποιος κρατάει στο χέρι του μία μάσκα, το μοντέλο δεν θέλουμε να την αναγνωρίζει διότι δεν είναι αυτός ο σκοπός του.

Ο αριθμός των εικόνων με ανθρώπους που φοράνε μάσκα είναι 1915 ενώ ο αριθμός για τις εικόνες με ανθρώπους χωρίς μάσκα είναι 1918. Οι διαστάσεις των εικόνων δεν είναι ίδιες αλλά αυτό δεν έχει σημασία διότι στον κώδικα προσαρμόζονται όλες σε μια συγκεκριμένη διάσταση για να είναι συμβατές με το μοντέλο. Παρόλα αυτά όλες οι εικόνες που έχουν επιλεχθεί περιέχουν τρία κανάλια χρώματος, δηλαδή RGB και είναι της μορφής “.jpg”. Οι πηγές από τις οποίες συλλέχθηκαν αυτές οι εικόνες είναι η ιστοσελίδα Kaggle [59] και το Google Images.



Εικόνα 3.1 Παραδείγματα των δεδομένων εκπαίδευσης

3.2 Ανάλυση του κώδικα

3.2.1 Εισαγωγή βιβλιοθηκών και ρύθμιση υπερπαραμέτρων

```
import os
import sys
import tensorflow
from tensorflow.keras.preprocessing.image import load_img
from tensorflow.keras.preprocessing.image import img_to_array
from tensorflow.keras.applications.mobilenet_v2 import preprocess_input
from sklearn.preprocessing import LabelBinarizer
from tensorflow.keras.utils import to_categorical
import numpy as np
from sklearn.model_selection import train_test_split
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications import MobileNetV2
from tensorflow.keras.layers import Input
from tensorflow.keras.layers import AveragePooling2D
from tensorflow.keras.layers import Flatten
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import Dropout
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import Adam
from sklearn.metrics import roc_curve, auc
from sklearn.metrics import classification_report
import matplotlib.pyplot as plt

```

Εικόνα 3.2 Εισαγωγή των απαραίτητων βιβλιοθηκών

Ο κώδικας για την εκπαίδευση του μοντέλου ανίχνευσης μάσκας ξεκινά με την εισαγωγή των απαραίτητων βιβλιοθηκών (libraries), οι οποίες περιέχουν συναρτήσεις (functions), κλάσεις για τη δημιουργία αντικειμένων (objects) και εντολές σχετικά με την δημιουργία, εκπαίδευση και αξιολόγηση ενός μοντέλου μηχανικής ή και βαθιάς μάθησης (Εικόνα 3.2). Επίσης δίνεται πρόσβαση σε εντολές που μας δίνουν τη δυνατότητα να επεξεργαστούμε κατάλληλα τα δεδομένα εκπαίδευσης, δηλαδή τις εικόνες και να τις προσαρμόσουμε με τέτοιο τρόπο ώστε να είναι συμβατές με το μοντέλο.

Πρέπει να διευκρινιστεί ότι από κάποιες βιβλιοθήκες δεν εισάγονται όλες οι εντολές που αυτές περιέχουν έτσι ώστε να μην επιβαρυνθεί και γίνει χρονοβόρος ο κώδικας κατά την εκτέλεσή του. Αντιθέτως εισάγονται μέρη αυτών των βιβλιοθηκών όπως για παράδειγμα συμβαίνει με πολλές συναρτήσεις της βιβλιοθήκης Keras, η οποία βρίσκεται πάνω στην βιβλιοθήκη του Tensorflow. Χαρακτηριστικό παράδειγμα αποτελεί η γραμμή 4 κατά την οποία εισάγεται μόνο η συνάρτηση “load_img” του κομματιού “preprocessing.image” της βιβλιοθήκης Keras.

Συγκεκριμένα οι βασικές βιβλιοθήκες που χρησιμοποιούνται για την εισαγωγή όλων των απαραίτητων μέσων για την ανάπτυξη του κώδικα, είναι οι παρακάτω:

- **os**

- **sys**
- **tensorflow**
- **sklearn**
- **numpy**
- **matplotlib.pyplot**

Η βιβλιοθήκες os, sys ανήκουν στην Python και παρέχουν συναρτήσεις με σκοπό την αλληλεπίδραση με το λειτουργικό σύστημα ή κάποιες συγκεκριμένες παραμέτρους του.

```
24  INIT_LR = 1e-4
25  EPOCHS = 20
26  BS = 32
27  IMAGE_SIZE = 224
```

Εικόνα 3.3 Οι τρεις υπερπαράμετροι του μοντέλου και η μεταβλητή IMAGE_SIZE

Στην εικόνα 3.3 αρχικοποιούνται οι τρεις σημαντικές υπερπαράμετροι του μοντέλου που καθορίζουν τον τρόπο με τον οποίο θα εκπαιδευτεί και οφείλονται για την επιτυχία ή αποτυχία του. Μαζί με αυτές δίνεται και η τιμή “224” στην μεταβλητή με όνομα IMAGE_SIZE που είναι υπεύθυνη για την πληροφοριοδότηση κάποιων εντολών αργότερα σχετικά με τις νέες διαστάσεις που θα έχουν τα δεδομένα εκπαίδευσης (224x224 pixels).

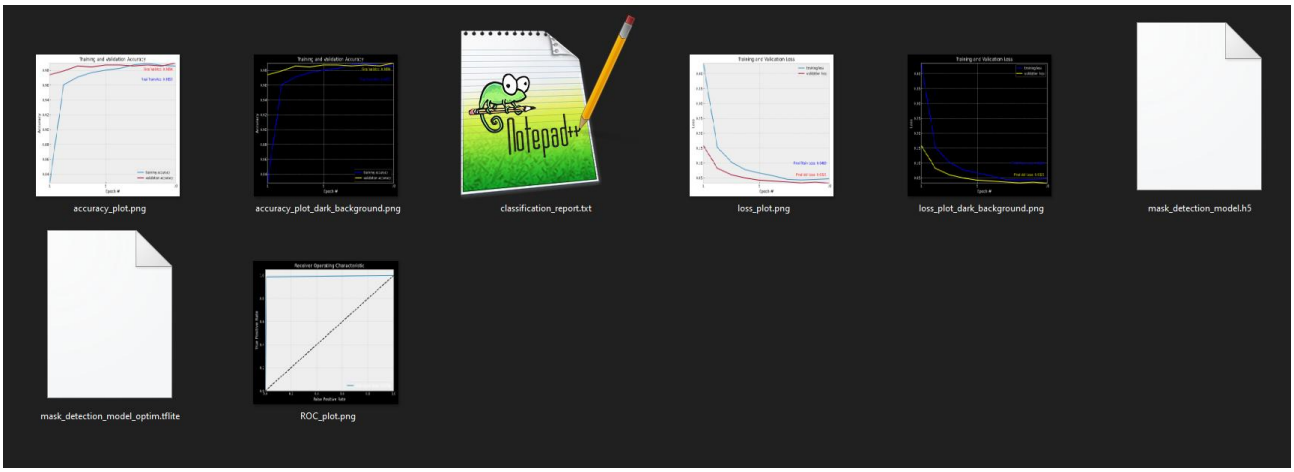
Η μεταβλητή INIT_LR είναι ο ρυθμός εκπαίδευσης ο οποίος δέχεται την τιμή 0.0001, η EPOCHS είναι οι εποχές/επαναλήψεις που θα εκπαιδευτεί το μοντέλο με τα ίδια δεδομένα εκπαίδευσης με τιμή 20 και η BS είναι το μέγεθος υποσυνόλου δεδομένων με τιμή 32. Ο συνδυασμός των συγκεκριμένων τιμών των υπερπαραμέτρων δημιουργούν το μοντέλο με την καλύτερη απόδοση, συμπέρασμα που προέκυψε μετά από τη δοκιμή διαφόρων συνδυασμών και σύγκριση των μοντέλων που παρήχθησαν.

3.2.2 Προεπεξεργασία δεδομένων εκπαίδευσης

```
33  folder_of_model = f"models/{INIT_LR}_{EPOCHS}_{BS}"
34
35  if os.path.exists(folder_of_model):
36      print(f"Αυτή η έκδοση του μοντέλου που προσπαθήσατε να εκπαιδεύσετε ({folder_of_model[7:]}) έχει ήδη δημιουργηθεί."
37      f" Το πρόγραμμα θα τερματιστεί τώρα...")
38      sys.exit()
39
40  print(f"[ΕΝΗΜΕΡΩΣΗ] Η εκπαίδευση του μοντέλου με ονομασία έκδοσης '{folder_of_model[7:]}' ξεκίνησε...")
41
42  os.makedirs(folder_of_model)
```

Εικόνα 3.4 Έλεγχος και δημιουργία μοναδικού φακέλου για το κάθε μοντέλο

Για την αρχειοθέτηση των διαφορετικών μοντέλων που εκπαιδεύτηκαν συντάχθηκε ο κώδικας της εικόνας 3.4, ο οποίος δημιουργεί έναν νέο μοναδικό φάκελο για το κάθε μοντέλο. Το όνομα αυτού του φακέλου καθορίζεται από το συνδυασμό των τιμών των τριών υπερπαραμέτρων που ορίστηκαν στην αρχή του κώδικα. Άρα για παράδειγμα αν το μοντέλο εκπαιδεύεται με τις υπερπαραμέτρους της εικόνας 3.3, τότε ο φάκελος θα ονομάζεται “0.0001_20_32” και μετά το πέρας της εκπαίδευσης του μοντέλου θα περιέχει όλα τα απαραίτητα αρχεία και πληροφορίες σχετικά με το συγκεκριμένο μοντέλο. Το περιεχόμενο ενός τέτοιου φακέλου θα μοιάζει με αυτό της εικόνας 3.5.



Εικόνα 3.5 Παράδειγμα των περιεχομένων του φακέλου ενός μοντέλου

Πριν από τη δημιουργία του φακέλου αυτού, γίνεται πρώτα έλεγχος εάν αυτός υπάρχει και εάν η απάντηση είναι θετική, τότε ο κώδικας τερματίζεται ενημερώνοντας κατάλληλα τον χρήστη με ένα μήνυμα στην οθόνη. Με αυτόν τον τρόπο διασφαλίζουμε ότι δεν θα εκπαιδεύσουμε ξανά ένα ίδιο ακριβώς μοντέλο, οπότε γλιτώνουμε χρόνο μιας και η διαδικασία εκπαίδευσης δεν είναι γρήγορη.

Να αναφερθεί πως όλοι οι νέοι φάκελοι των μοντέλων που εκπαιδεύονται τοποθετούνται μέσα σε έναν φάκελο με ονομασία “models”. Εάν ο κώδικας εντοπίσει πως ο φάκελος δεν υπάρχει, τότε τον δημιουργεί και προχωράει στην εκτέλεση των εντολών της εικόνας 3.6 που αφορά την προεπεξεργασία των δεδομένων εκπαίδευσης.

```
46  dataset_location = r"D:\projects\face_mask_detection\dataset"
47  dataset_classes = ["with_mask", "without_mask"]
48  data = []
49  labels = []
50
51  print("[ΕΝΗΜΕΡΩΣΗ] Η φόρτωση των εικόνων ξεκίνησε...")
52
53  for dataset_class in dataset_classes:
54      path = os.path.join(dataset_location, dataset_class)
55
56      for img_name in os.listdir(path):
57          img_path = os.path.join(path, img_name)
58          image = load_img(img_path, target_size=(IMAGE_SIZE, IMAGE_SIZE))
59          image = img_to_array(image)
60          image = preprocess_input(image)
61          data.append(image)
62          labels.append(dataset_class)
```

Εικόνα 3.6 Μέρος 1^{ο} της προεπεξεργασίας των δεδομένων εκπαίδευσης

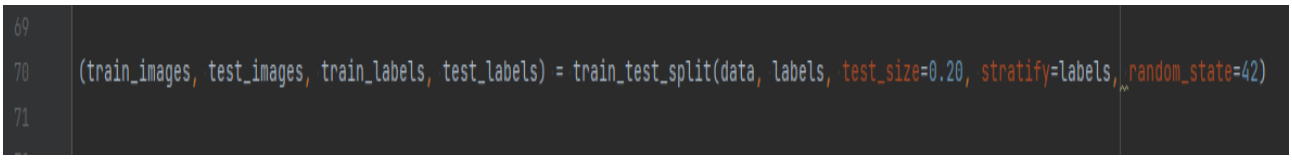
Ο κώδικας συνεχίζει δημιουργώντας δύο λίστες, την “data” και την “labels” στις οποίες θα αποθηκευτούν όλες οι εικόνες για την εκπαίδευση και όλες οι ετικέτες αυτών των εικόνων αντίστοιχα. Ως ετικέτα (label) ορίζεται η κλάση στην οποία ανήκει πραγματικά μια εικόνα, δηλαδή αν μια από τις εικόνες περιέχει έναν άνθρωπο που δεν φοράει μάσκα τότε η αντίστοιχη ετικέτα της θα έχει την ονομασία “without_mask”. Σε αντίθετη περίπτωση η ετικέτα θα ονομάζεται “with_mask”.

Για την τοποθέτηση όλων των εικόνων του θα εκπαιδευτεί το μοντέλο στη λίστα “data”, πρώτα αυτές περνάνε από μία διαδικασία προεπεξεργασίας. Η διαδικασία ξεκινά φορτώνοντας κάθε μία από τις εικόνες μέσα στον κώδικα ως μεταβλητές και μετατρέποντας τις διαστάσεις τους στα 224 x 224 pixels όπως ορίζει η μεταβλητή IMAGE_SIZE της εικόνας 3.3. Έπειτα γίνεται μετατροπή της κάθε εικόνας σε διάνυσμα το οποίο δέχεται συγκεκριμένες αλλαγές για να γίνει συμβατή η εικόνα με το προεκπαιδευμένο μοντέλο MobileNetV2 που θα συναντήσει αργότερα. Έτσι η κάθε εικόνα αποθηκεύεται με τη σειρά στη λίστα “data” και η ετικέτα της στην αντίστοιχη θέση της λίστας “labels”.

```
63  lb = LabelBinarizer()
64  labels = lb.fit_transform(labels)
65  labels = to_categorical(labels)
66  data = np.array(data, dtype="float32")
67  labels = np.array(labels)
```

Εικόνα 3.7 Μέρος 2^{ο} της προεπεξεργασίας των δεδομένων εκπαίδευσης

Ο κώδικας συνεχίζει (εικόνα 3.7) αλλάζοντας τις λίστες “data” και “labels” σε διανύσματα με περιεχόμενο τύπου “float32” και μετατρέποντας τις αλφαριθμητικές ετικέτες του “labels” σε αριθμούς. Πρώτα μετατρέπεται το “with_mask” σε “0”, το “without_mask” σε “1” και έπειτα μέσω της κωδικοποίησης one hot encoding μετατρέπεται το “0” σε “[1. 0.]” και το “1” σε “[0. 1.]”. Αυτό συμβαίνει για λόγους συμβατότητας και τη σωστή κατανόηση των δεδομένων από τον κώδικα εκπαίδευσης.



Εικόνα 3.8 Μέρος 3ο της προεπεξεργασίας των δεδομένων εκπαίδευσης

Το τελευταίο κομμάτι της προεπεξεργασίας των δεδομένων αποτελείται από την εντολή της εικόνας 3.8, η οποία διαιρεί το σύνολο των εικόνων σε δεδομένα που θα εκπαιδευτεί το μοντέλο και σε δεδομένα που θα δοκιμαστεί το μοντέλο αν εκπαιδεύτηκε σωστά. Οι δύο αυτές ομάδες δεδομένων θα συνοδεύονται από άλλες δυο ομάδες που θα περιέχουν τις ετικέτες τους. Συγκεκριμένα για την εκπαίδευση θα δημιουργηθεί το διάνυσμα “train_images” με τις εικόνες και το διάνυσμα “train_labels” με τις ετικέτες τους. Αντίστοιχα για την δοκιμή θα δημιουργηθεί το “test_images” και το “test_labels”. Θα αναφερόμαστε πλέον σε αυτές τις δύο κατηγορίες δεδομένων ως δεδομένα εκπαίδευσης (training set) και ως δοκιμαστικά δεδομένα (test set).

Το σύνολο των εικόνων κατανέμεται στις δυο αυτές κατηγορίες σε ποσοστό 80% για τα δεδομένα εκπαίδευσης και 20% για τα δοκιμαστικά δεδομένα. Η επιλογή αυτού του ποσοστού μπορεί να αλλάξει, ωστόσο είναι μια καλή επιλογή γενικά σε σχέση με τη ποσότητα των δεδομένων που διαθέτουμε. Ακόμα, τα δεδομένα χωρίζονται ομοιόμορφα χωρίς να υπάρχουν για παράδειγμα μόνο εικόνες με μάσκα στα δεδομένα εκπαίδευσης και μόνο εικόνες χωρίς μάσκα στα δοκιμαστικά δεδομένα. Θα υπάρξουν εικόνες και των δύο κλάσεων στην κάθε ομάδα.

Επίσης στην εντολή της εικόνας 3.8 δίνεται ο αριθμός 42 στην ιδιότητα "random_state" η οποία ορίζει τη λογική της τυχαιότητας που θα χωριστούν τα δεδομένα στις ομάδες. Αυτό βοηθάει στην πιο αντικειμενική σύγκριση των μοντέλων που θα εκπαιδευτούν.

3.2.3 Αντικείμενο αύξησης δεδομένων εκπαίδευσης

```
311  aug_gen = ImageDataGenerator(
312  rotation_range=20,
313  zoom_range=0.15,
314  width_shift_range=0.2,
315  height_shift_range=0.2,
316  shear_range=0.15,
317  horizontal_flip=True,
318  fill_mode="nearest")
```

Εικόνα 3.9 Δημιουργία αντικειμένου για την αύξηση των δεδομένων εκπαίδευσης

Στο κομμάτι του κώδικα της εικόνας 3.9 δημιουργείται ένα αντικείμενο της κλάσης "ImageDataGenerator" με ονομασία "aug_gen". Αυτό θα είναι υπεύθυνο αργότερα, κατά την εκπαίδευση του μοντέλου, για την αύξηση των δεδομένων εκπαίδευσης με τη δημιουργία νέων εικόνων μέσω της τροποποίησης αυτών που υπάρχουν ήδη.

Οι τροποποιήσεις που θα δεχτούν οι ήδη υπάρχουσες εικόνες για να δημιουργήσουν νέες, είναι οι εξής:

- Τυχαία περιστροφή
- Τυχαία εφαρμογή μεγέθυνσης ή σμίκρυνσης
- Τυχαία μετατόπιση του πλάτους
- Τυχαία μετατόπιση του ύψους
- Τυχαία διαστρέβλωση
- Τυχαία οριζόντια αναστροφή
- Επιδιόρθωση των pixels που υπέστησαν τροποποιήσεις

3.2.4 Κατασκευή της αρχιτεκτονικής του μοντέλου

```
89  baseModel = MobileNetV2(weights="imagenet", include_top=False, input_tensor=Input(shape=(IMAGE_SIZE, IMAGE_SIZE, 3)))
90
91  headModel = baseModel.output
92  headModel = AveragePooling2D(pool_size=(7, 7))(headModel)
93  headModel = Flatten(name="flatten")(headModel)
94  headModel = Dense(128, activation="relu")(headModel)
95  headModel = Dropout(0.5)(headModel)
96  headModel = Dense(2, activation="softmax")(headModel)
97
98  model = Model(inputs=baseModel.input, outputs=headModel)
```

Εικόνα 3.10 Η αρχιτεκτονική του μοντέλου

Σε αυτό το σημείο (εικόνα 3.10) ο κώδικας δημιουργεί πρώτα το αντικείμενο “baseModel” της κλάσης “MobileNetV2” που θα αποτελέσει την βάση του μοντέλου. Η συγκεκριμένη κλάση διαμορφώνει το πρώτο μέρος του μοντέλου με την αρχιτεκτονική του προεκπαιδευμένου μοντέλου MobileNetV2. Έτσι το “baseModel” θα περιέχει τα επίπεδα και τα βάρη του συνελικτικού αυτού νευρωνικού δικτύου μέχρι το σημείο που ξεκινάνε τα κρυφά επίπεδά του. Τα κρυφά επίπεδα του “MobileNetV2” δεν περιλαμβάνονται στο “baseModel” ενώ τα βάρη που προαναφέρθηκαν θα έχουν τις τιμές που διαμόρφωσε το προεκπαιδευμένο μοντέλο όταν εκπαιδεύτηκε πάνω στο τεράστιο σύνολο δεδομένων “imagenet”. Το “baseModel” περιλαμβάνει την είσοδο του τελικού μοντέλου, την επεξεργασία των δεδομένων εκπαίδευσης, την εξαγωγή των χαρακτηριστικών τους και την παράδοση αυτών στα κρυφά επίπεδα για την ταξινόμηση. Επίσης με τη μεταβλητή “IMAGE_SIZE”ορίζονται οι διαστάσεις που πρέπει να έχουν οι εικόνες για να εισαχθούν στο μοντέλο και αυτό να εκπαιδευτεί δηλαδή 224x224 pixels.

Στη συνέχεια δημιουργείται το δεύτερο μέρος του μοντέλου που αφορά την ταξινόμηση των εικόνων και είναι το αντικείμενο με ονομασία “headModel”. Σε αυτό προστίθενται τα παρακάτω επίπεδα:

- Επίπεδο συγκέντρωσης με παράθυρο μεγέθους 7x7
- Επίπεδο ισοπέδωσης
- Πλήρως συνδεδεμένο κρυφό επίπεδο 128 νευρώνων
- Επίπεδο εφαρμογής της μεθόδου τακτοποίησης εγκατάλειψη με ποσοστό 50%
- Πλήρως συνδεδεμένο επίπεδο 2 νευρώνων

Το “headModel” περιλαμβάνει την ταξινόμηση των δεδομένων εκπαίδευσης με βάση τα χαρακτηριστικά τους που δέχεται από το “baseModel” και είναι η έξοδος του τελικού μοντέλου.

Μετά τη δημιουργία του “headModel” για την εξειδικευμένη ταξινόμηση των δεδομένων με βάση την ανίχνευση μάσκας ή όχι στα ανθρώπινα πρόσωπα, πραγματοποιείται ένωση των δύο κομματιών του μοντέλου σε ένα. Δηλαδή συνδέεται το “baseModel” με το “headModel” και κατασκευάζεται αρχιτεκτονικά το μοντέλο με ονομασία “model”.

```
for layer in baseModel.layers:
	layer.trainable = False

adam_optim = Adam(learning_rate=INIT_LR)

print("[ΕΝΗΜΕΡΩΣΗ] Γίνεται μεταγλώττιση του μοντέλου...")

model.compile(loss="binary_crossentropy", optimizer=adam_optim, metrics=["accuracy"])
```

Εικόνα 3.11 Μεταγλώττιση του μοντέλου

Ο κώδικας συνεχίζει (εικόνα 3.11) απενεργοποιώντας τη δυνατότητα εκπαίδευσης των επιπέδων του πρώτου μέρους του μοντέλου (“baseModel”), αφού όπως αναφέρθηκε θέλουμε οι παράμετροι του προεκπαιδευμένου δικτύου να μείνουν αμετάβλητες. Το μόνο μέρος του μοντέλου που θα εκπαιδευτεί αργότερα και θα διαμορφώσει κατάλληλα τις παραμέτρους του είναι αυτό που περιλαμβάνει το “headModel”. Αυτό συμβαίνει διότι έτσι θα μάθει το μοντέλο να επικεντρώνεται σε χαρακτηριστικά εικόνων με μάσκα και όχι σε χαρακτηριστικά από οτιδήποτε άλλο, διαδικασία που θα βελτιώσει την ακρίβειά του ως προς τα αποτελέσματα

Μετά από την απενεργοποίηση της εκπαίδευσης του πρώτου μέρους του μοντέλου, επιλέγεται ο βελτιστοποιητής για το μοντέλο που είναι ο “Adam”, καθώς και το είδος της συνάρτησης απωλειών που είναι η “binary_crossentropy”.

Όλες αυτές οι πληροφορίες σχετικά με την αρχιτεκτονική του μοντέλου, τον βελτιστοποιητή και την συνάρτηση απωλειών μεταγλωττίζονται (compile) ή αλλιώς συναρμολογούνται και αποθηκεύονται στο αντικείμενο “model” που αποτελεί το μοντέλο μας.

3.2.5 Εκπαίδευση και αποθήκευση του μοντέλου

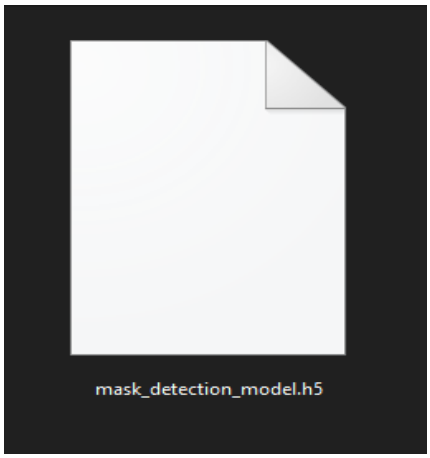
```
113  print("[ΕΝΗΜΕΡΩΣΗ] Η εκπαίδευση του μοντέλου(head μέρος) ξεκίνησε...")
114
115  HISTORY = model.fit(aug_gen.flow(train_images, train_labels, batch_size=BS), steps_per_epoch=len(train_images) // BS,
116                      validation_data=(test_images, test_labels), validation_steps=len(test_images) // BS, epochs=EPOCHS)
117
118  print("[ΕΝΗΜΕΡΩΣΗ] Αποθήκευση του μοντέλου ανίχνευσης μάσκας στον φάκελο...")
119
120  model.save(f"{folder_of_model}/mask_detection_model.h5")
```

Εικόνα 3.12 Εκπαίδευση και αποθήκευση του μοντέλου

Στην εικόνα 3.12 απεικονίζονται οι εντολές που πραγματοποιούνε την εκπαίδευση του μοντέλου και έπειτα την αποθήκευσή του μέσα στον φάκελο που δημιουργήθηκε νωρίτερα για αυτό. Για την εκπαίδευση του μοντέλου δίνονται οι παρακάτω πληροφορίες:

- Το αντικείμενο αύξησης δεδομένων “aug_gen”
- Τα δεδομένα εκπαίδευσης “train_images”
- Οι ετικέτες των δεδομένων εκπαίδευσης “train_labels”
- Η μεταβλητή “BS” της υπερπαραμέτρου του υποσυνόλου δεδομένων, για την κατανομή των δεδομένων εκπαίδευσης σε ομάδες
- Ο αριθμός επαναλήψεων εκπαίδευσης ανά εποχή εκπαίδευσης
- Τα δοκιμαστικά δεδομένα για την επικύρωση της εκπαίδευσης
- Ο αριθμός επαναλήψεων επικύρωσης ανά εποχή εκπαίδευσης
- Οι εποχές εκπαίδευσης του μοντέλου

Όλες οι πληροφορίες σχετικά με την εκπαίδευση του μοντέλου αποθηκεύονται μέσα στο αντικείμενο “HISTORY” από το οποίο θα μπορούμε να τις εξάγουμε και να τις χρησιμοποιήσουμε καταλλήλως αργότερα για την δημιουργία των διαγραμμάτων. Το μοντέλο που αποθηκεύεται έχει την κατάληξη“.h5” και αυτό σημαίνει πως όλες οι πληροφορίες σχετικά με αυτό βρίσκονται μέσα σε ένα μόνο αρχείο και όχι σε κάποιον φάκελο με διαχωρισμένες τις πληροφορίες του. Επίσης το αποθηκευμένο μοντέλο θα έχει όνομα “mask_detection_model.h5”.



Εικόνα 3.13 Το αποθηκευμένο αρχείο του μοντέλου με μορφή “.h5”

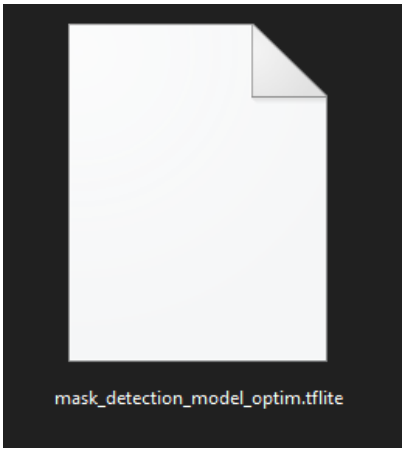
3.2.6 Μετατροπή και αποθήκευση του μοντέλου σε έκδοση TensorFlow Lite

```
126  print("[ΕΝΗΜΕΡΩΣΗ] Μετατροπή του μοντέλου απο .h5 σε μορφή .tflite...")
127
128  loaded_model = tensorflow.keras.models.load_model(f"{folder_of_model}/mask_detection_model.h5")
129  converter = tensorflow.lite.TFLiteConverter.from_keras_model(loaded_model)
130  converter.optimizations = [tensorflow.lite.Optimize.DEFAULT]
131  tflite_model = converter.convert()
132  open(f"{folder_of_model}/mask_detection_model_optim.tflite", "wb").write(tflite_model)
```

Εικόνα 3.14 Μετατροπή και αποθήκευση του μοντέλου σε έκδοση TensorFlow Lite

Για την επιτυχή χρήση του μοντέλου που εκπαιδεύτηκε στη συσκευή Raspberry Pi 4, δημιουργήθηκε ο κώδικας της εικόνας 3.13 ο οποίος μετατρέπει το μοντέλο μορφής ".h5" σε μορφή ".tflite" της TensorFlow Lite.

Πρώτα φορτώνεται στη μεταβλητή "loaded_model" το αποθηκευμένο μοντέλο που εκπαιδεύτηκε και έπειτα γίνεται η μετατροπή του σε μοντέλο TensorFlow Lite εφαρμόζοντας κάποιες επιπλέον αυτοματοποιημένες βελτιστοποιήσεις. Οι βελτιστοποιήσεις αυτές προετοιμάζουν το αρχείο για χρήση σε συσκευές όπως το rpi4, μειώνοντας το μέγεθός του και κάνοντας πιο γρήγορο το ίδιο το μοντέλο. Μετά την μετατροπή η νέα έκδοση του μοντέλου αποθηκεύεται πρώτα στην μεταβλητή "tflite_model" και έπειτα στον φάκελο της εκπαίδευσης του μοντέλου με ονομασία "mask_detection_model_optim.tflite".



Εικόνα 3.15 Το αποθηκευμένο αρχείο του μοντέλου με μορφή ".tflite"

Εικόνα 3.16 Η διαφορά του μεγέθους των μοντέλων ανάλογα με τη μορφή τους, ψηλά πρώτο το μοντέλο ".h5" στα 11.22 MB, από κάτω του το ".tflite" στα 9.2 MB και τέλος το ".tflite" αλλά με την εφαρμογή των αυτοματοποιημένων βελτιστοποιήσεων στα 2.6 MB

3.2.7 Αξιολόγηση του μοντέλου

```
138  print("[ΕΝΗΜΕΡΩΣΗ] Αξιολόγηση του νευρωνικού δικτύου...")
139  predictions = model.predict(test_images, batch_size=BS)
140  test_labels_binary = np.argmax(test_labels, axis=1)
141  predictions_binary = np.argmax(predictions, axis=1)
142  fpr, tpr, thresholds = roc_curve(test_labels_binary, predictions_binary)
143  auc_score = auc(fpr, tpr)
```

Εικόνα 3.17 Προβλέψεις και υπολογισμός της βαθμολογίας AUC από την καμπύλη λειτουργικού χαρακτηριστικού δέκτη

Πριν την αξιολόγηση του μοντέλου προηγούνται κάποιες άλλες εντολές που παρουσιάζονται στην εικόνα 3.17. Αρχικά γίνεται δοκιμή του μοντέλου πάνω στα δοκιμαστικά δεδομένα “test_images” και αποθήκευση των παραγόμενων προβλέψεων που κάνει ως πιθανότητες από το 0 μέχρι το 1 στην μεταβλητή “predictions”. Για την πιο γρήγορη ολοκλήρωση των προβλέψεων, τα δοκιμαστικά δεδομένα χωρίζονται σε ομάδες σύμφωνα με την υπερπαράμετρο μέγεθος υποσυνόλου δεδομένων που έχει οριστεί (“BS”).

Έπειτα δημιουργούνται δύο νέα διανύσματα, το “test_labels_binary” και το “predictions_binary”. Αυτά προκύπτουν από την μετατροπή των περιεχομένων των δοκιμαστικών δεδομένων και των προβλέψεων αντίστοιχα σε δυαδική μορφή. Αυτή η διαδικασία μετατροπής χρειάζεται για τον υπολογισμό των επιμέρους στοιχείων της καμπύλης λειτουργικού χαρακτηριστικού δέκτη τα οποία προκύπτουν από τη συνάρτηση “roc_curve”. Η συνάρτηση αυτή απαιτεί τα δεδομένα εισαγωγής σε αυτήν να έχουν δυαδική μορφή.

Μετά την εύρεση των επιμέρους στοιχείων της καμπύλης, δηλαδή του σωστού θετικού ρυθμού (“tpr”), του λάθος θετικού ρυθμού (“fpr”) και των κατωφλιών τους (“thresholds”), υπολογίζεται η περιοχή κάτω από την καμπύλη λειτουργικού χαρακτηριστικού δέκτη, δηλαδή η βαθμολογία AUC.

Όλοι αυτοί οι παραπάνω υπολογισμοί θα χρειαστούν στην επόμενη ενότητα όπου θα αναπαρασταθεί γραφικά η καμπύλη λειτουργικού χαρακτηριστικού δέκτη.

```
145  predictions = np.argmax(predictions, axis=1)
146
147  report = classification_report(test_labels.argmax(axis=1), predictions, target_names=lb.classes_)
148  print(report)
149
150  with open(f'{folder_of_model}/classification_report.txt', 'w') as file:
151      file.write(report)
152      file.close()
```

Εικόνα 3.18 Δημιουργία και αποθήκευση της αναφοράς σχετικά με τις μετρικές αξιολόγησης, πάνω στα δεδομένα των προβλέψεων που πραγματοποιήθηκαν

Το τελευταίο κομμάτι κώδικα αυτής της ενότητας (εικόνα 3.18) αρχικά μετατρέπει τις ίδιες τις προβλέψεις που είχαν γίνει στην αρχή του κώδικα της εικόνας 3.17 σε δυαδική μορφή, όπως δηλαδή έγινε και για την εύρεση των στοιχείων της καμπύλης λειτουργικού χαρακτηριστικού δέκτη. Ο λόγος που εκτελείται ξανά η εντολή είναι για να δοθεί έμφαση στην κατανόηση και να

διαφοροποιηθεί το προηγούμενο κομμάτι κώδικα σε σχέση με αυτό της εικόνας 3.18 που αφορά την αξιολόγηση.

Έτσι λοιπόν έχοντας τις προβλέψεις σε δυαδική μορφή, αυτές δίνονται σαν πληροφορία στην συνάρτηση "classification_report" μαζί με τις ετικέτες των δοκιμαστικών δεδομένων σε δυαδική μορφή και αποθηκεύεται στο αντικείμενο "report" η αναφορά με τις μετρικές αξιολόγησης.

Αυτή η αναφορά στη συνέχεια αποθηκεύεται σαν αρχείο κειμένου μέσα στον φάκελο που δημιουργήθηκε για το μοντέλο, ώστε να μπορεί να ανατρέξει κάποιος σε αυτό όποτε θέλει και όχι μόνο κατά την εκτέλεση του κώδικα εκπαίδευσης του μοντέλου. Το όνομα που θα έχει το αρχείο είναι το "classification_report.txt" και το περιεχόμενό του θα έχει την μορφή της εικόνας 3.19.

classification_report.txt

	precision	recall	f1-score	support
with_mask	0.99	0.99	0.99	383
without_mask	0.99	0.99	0.99	384
accuracy			0.99	767
macro avg	0.99	0.99	0.99	767
weighted avg	0.99	0.99	0.99	767

Εικόνα 3.19 Παράδειγμα αναφοράς "classification_report.txt" των μετρικών αξιολόγησης

3.2.8 Διαγράμματα μετρικών αξιολόγησης

```
156  print("[ΕΝΗΜΕΡΩΣΗ] Η γραφική απεικόνιση των μετρήσεων ξεκίνησε...")
157
158  final_train_loss = HISTORY.history["loss"][-1]
159  final_train_acc = HISTORY.history["accuracy"][-1]
160  final_val_loss = HISTORY.history["val_loss"][-1]
161  final_val_acc = HISTORY.history["val_accuracy"][-1]
```

Εικόνα 3.20 Υπολογισμοί τελικών τιμών απωλειών και ορθότητας

Η τελευταία ενότητα του κώδικα εκπαίδευσης μοντέλου περιέχει εντολές σχετικά με την δημιουργία διαγραμμάτων που απεικονίζουν διάφορες πληροφορίες σχετικά με την απόδοση του μοντέλου.

Πριν όμως από την δημιουργία των διαγραμμάτων, εκτελούνται τέσσερις εντολές υπολογισμού τελικών τιμών. Οι πρώτες δύο τελικές τιμές που υπολογίζονται αφορούν τις απώλειες, πρώτον σχετικά με την εκπαίδευση του μοντέλου ως “loss” και δεύτερον σχετικά με την επικύρωση των δοκιμαστικών του δεδομένων ως “val_loss”. Οι δεύτερες τελικές τιμές αφορούν την ορθότητα του μοντέλου και χαρακτηρίζονται ως “accuracy” και “val_accuracy” όπου η πρώτη αφορά την εκπαίδευση και η δεύτερη την επικύρωση. Οι τελικές αυτές τιμές βρέθηκαν με τη βοήθεια του αντικειμένου “HISTORY” που δημιουργήθηκε κατά την εκπαίδευση του μοντέλου.

```
163  plt.style.use("bmh")
164  plt.figure(figsize=(8, 6))
165  plt.plot(np.arange(1, EPOCHS+1), HISTORY.history["loss"], label="training loss")
166  plt.plot(np.arange(1, EPOCHS+1), HISTORY.history["val_loss"], label="validation loss")
167  plt.title("Training and Validation Loss")
168  plt.xlabel("Epoch #")
169  plt.ylabel("Loss")
170  plt.legend(loc="upper right")
171  plt.annotate(f"Final Train Loss: {final_train_loss:.4f}", (EPOCHS, final_train_loss), textcoords="offset points",
172      xytext=(-10, 40), ha='right', color='blue')
173  plt.annotate(f"Final Val Loss: {final_val_loss:.4f}", (EPOCHS, final_val_loss), textcoords="offset points",
174      xytext=(-10, 20), ha='right', color='red')
175  plt.margins(x=0, y=0)
176  tick_locations = np.arange(0, EPOCHS+1, 5)
177  tick_locations[0] = 1
178  plt.xticks(tick_locations, tick_locations)
179  plt.savefig(f"{folder_of_model}/loss_plot.png", dpi=300, bbox_inches="tight")
```

Εικόνα 3.21 Δημιουργία διαγράμματος απωλειών

Μετά από τους υπολογισμούς ακολουθεί ο κώδικας του πρώτου διαγράμματος που αφορά τις απώλειες εκπαίδευσης και επικύρωσης (εικόνα 3.21).

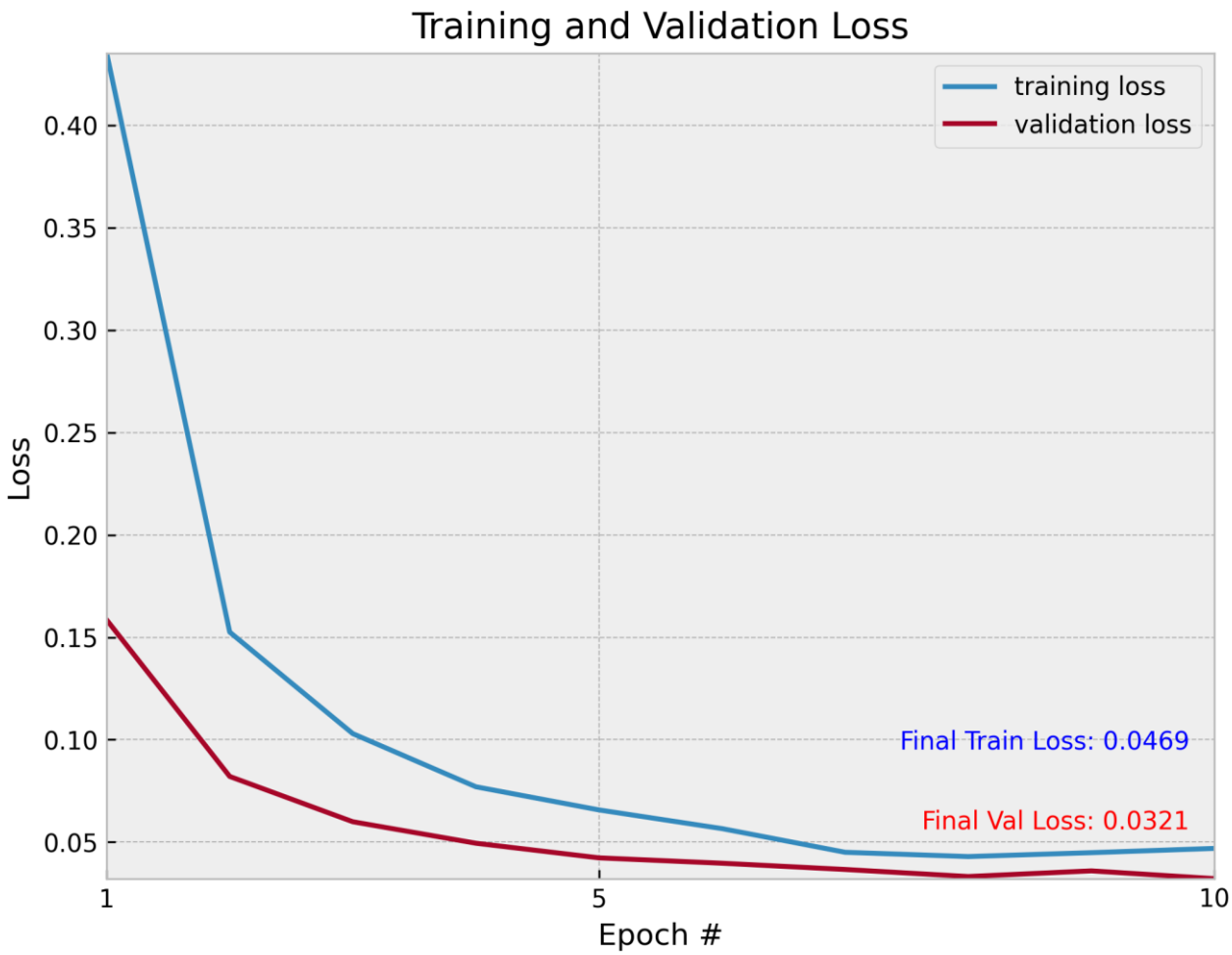
Αρχικά επιλέγεται το στυλ του διαγράμματος ως προς τα χρώματα, το παρασκήνιο και άλλες παρεμφερείς λεπτομέρειες. Συγκεκριμένα επιλέχθηκε για το πρώτο διάγραμμα το στυλ “bmh”. Το μέγεθος του γραφήματος ορίστηκε ως 8 επί 6 ίντσες και οι τιμές των σημείων που θα σχεδιαστούν ως προς τον άξονα x ξεκινάνε από το 1 έως τον συνολικό αριθμό των εποχών που εκτελεί κάθε μοντέλο κατά την εκπαίδευσή του, με βήμα 1. Οι τιμές των σημείων αυτών ως προς τον άξονα y παρέχονται από το “HISTORY” που είναι όλες οι τιμές απωλειών εκπαίδευσης για κάθε εποχή και

όλες οι τιμές απωλειών επικύρωσης για κάθε εποχή. Επίσης για τις γραφικές που δημιουργούνται αναγράφονται σε υπόμνημα οι τίτλοι και το χρώμα τους. Το υπόμνημα βρίσκεται στο πάνω δεξιά σημείο του διαγράμματος.

Για τους άξονες x και y φτιάχνονται τίτλοι με ονομασία “Epoch” και “Loss” αντίστοιχα ενώ στο πάνω μέρος του διαγράμματος εμφανίζεται ο τίτλος “Training and Validation Loss”. Η τελική τιμή απώλειας για την εκπαίδευση αναγράφεται με μπλε χρώμα ενώ για την επικύρωση με κόκκινο. Αυτές οι τιμές τοποθετούνται στο σημείο που τελειώνουν οι γραφικές και είναι υπολογισμένες να βρίσκονται λίγο πιο μέσα από τα όρια του διαγράμματος έτσι ώστε να φαίνονται ευδιάκριτα.

Τα περιθώρια του διαγράμματος έχουν αφαιρεθεί για την καλύτερη εμφάνισή του και επιπλέον γίνεται μία μικρή επιδιόρθωση στο διάγραμμα έτσι ώστε οι τιμές του άξονα x να ξεκινάνε από το 1 και όχι το 0. Στο τέλος γίνεται αποθήκευση του διαγράμματος στον φάκελο του μοντέλου με όνομα “loss_plot.png”.

Στην εικόνα 3.22 απεικονίζεται ένα παράδειγμα ενός τέτοιου διαγράμματος απωλειών.

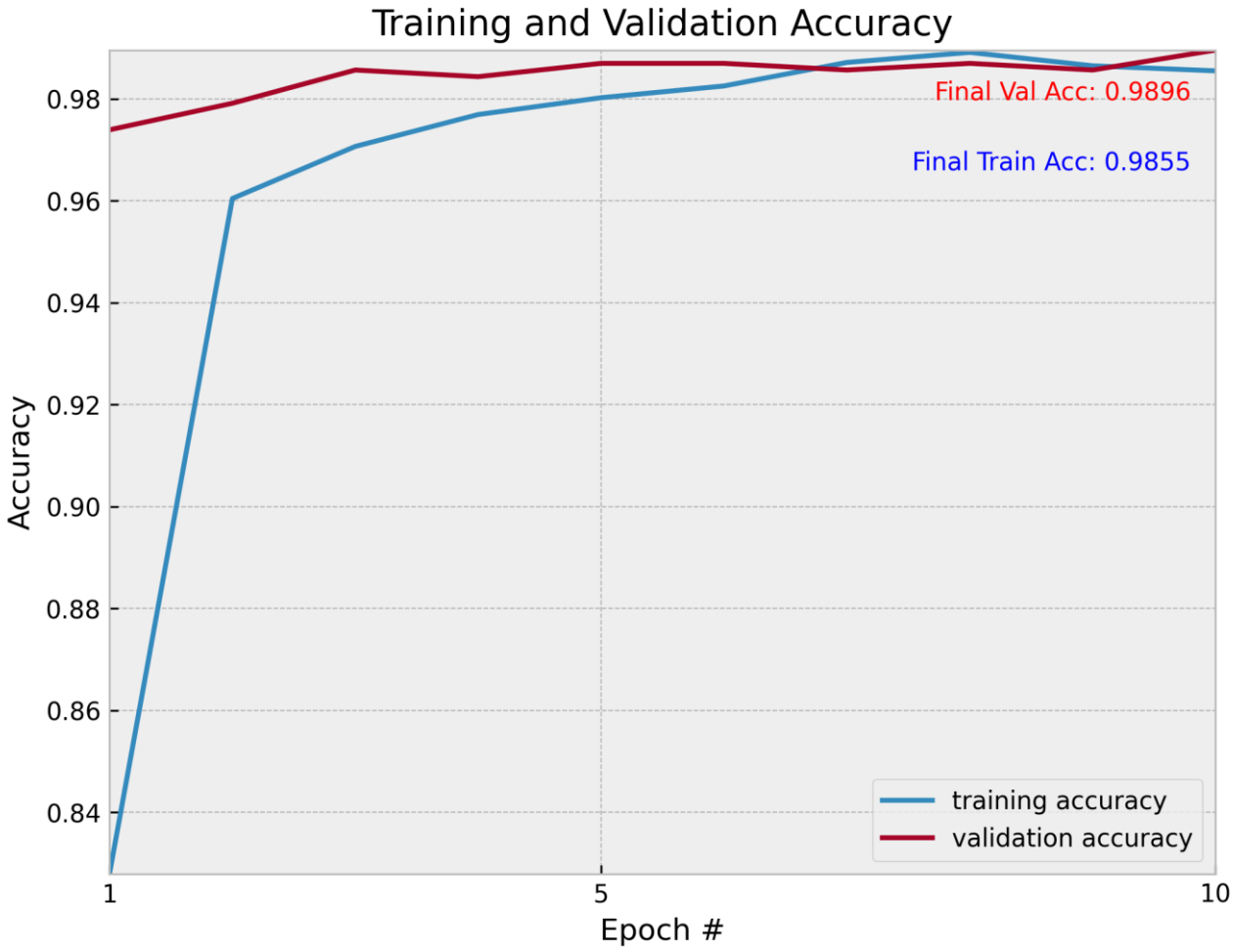


Εικόνα 3.22 Παράδειγμα διαγράμματος απωλειών

```
plt.style.use("bmh")
plt.figure(figsize=(8, 6))
plt.plot(np.arange(1, EPOCHS+1), HISTORY.history["accuracy"], label="training accuracy")
plt.plot(np.arange(1, EPOCHS+1), HISTORY.history["val_accuracy"], label="validation accuracy")
plt.title("Training and Validation Accuracy")
plt.xlabel("Epoch #")
plt.ylabel("Accuracy")
plt.legend(loc="lower right")
plt.annotate(f"Final Train Acc: {final_train_acc:.4f}", (EPOCHS, final_train_acc), textcoords="offset points",
             xytext=(-10, -40), ha='right', color='blue')
plt.annotate(f"Final Val Acc: {final_val_acc:.4f}", (EPOCHS, final_val_acc), textcoords="offset points",
             xytext=(-10, -20), ha='right', color='red')
plt.margins(x=0, y=0)
tick_locations = np.arange(0, EPOCHS+1, 5)
tick_locations[0] = 1
plt.xticks(tick_locations, tick_locations)
plt.savefig(f"{folder_of_model}/accuracy_plot.png", dpi=300, bbox_inches="tight")
```

Εικόνα 3.23 Δημιουργία διαγράμματος ορθοτήτων

Ο κώδικας της εικόνας 3.23 έχει ακριβώς την ίδια λογική με αυτόν της δημιουργίας του διαγράμματος απωλειών, με τη διαφορά ότι δημιουργεί το διάγραμμα των ορθοτήτων. Από θέμα απεικόνισης τα γραφικά είναι ίδια όπως και τα χρώματα που αντιστοιχούν στις τιμές ορθότητας για την εκπαίδευση και την επικύρωση. Το μόνο που αλλάζει πρακτικά είναι τα δεδομένα, οι τίτλοι του γραφήματος καθώς και το όνομα του αρχείου που θα αποθηκευτεί το διάγραμμα, δηλαδή “accuracy_plot.png”. Στην εικόνα 3.24 απεικονίζεται ένα διάγραμμα ορθοτήτων.



Εικόνα 3.24 Παράδειγμα διαγράμματος ορθοτήτων

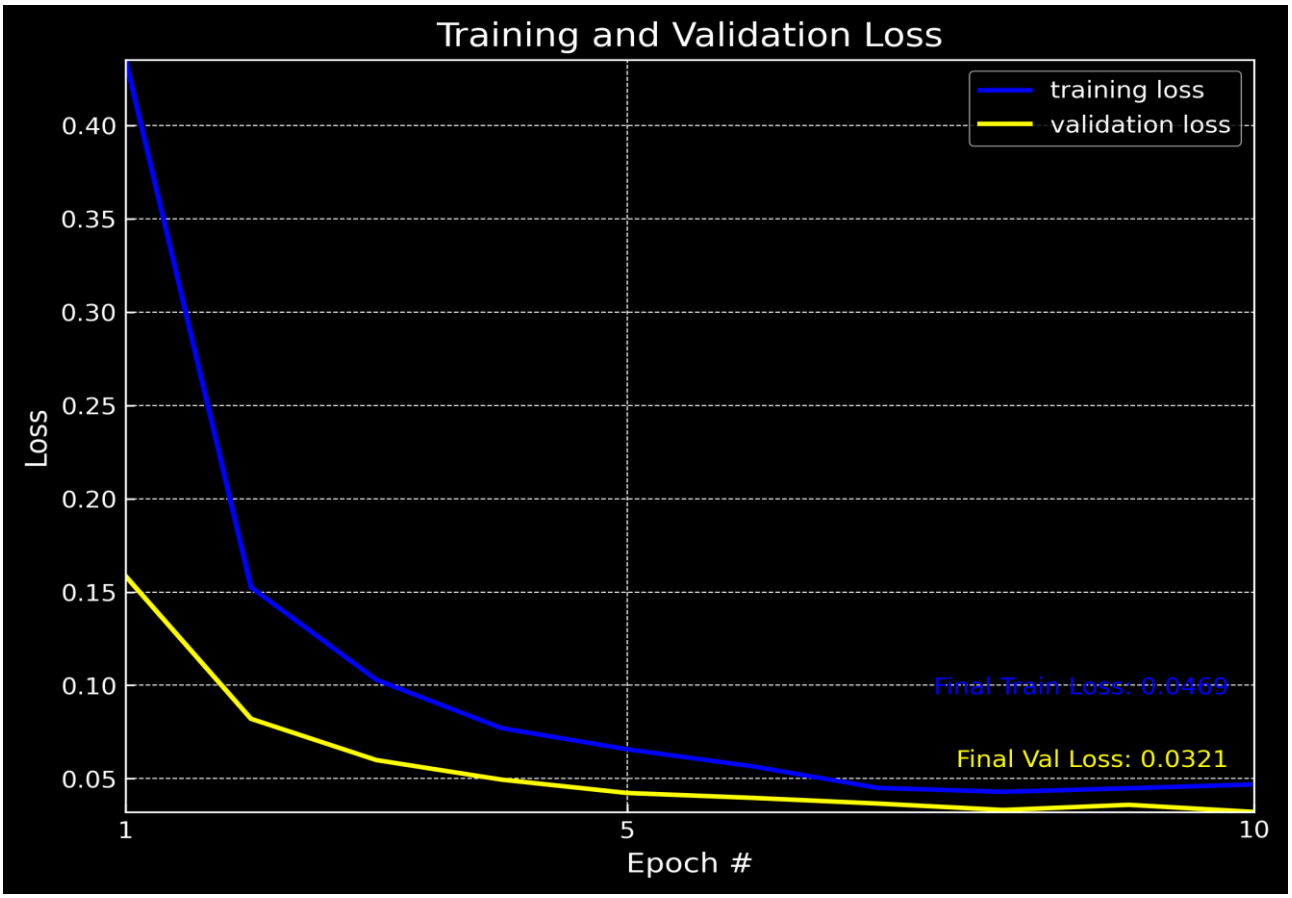
```
plt.style.use("dark_background")
plt.figure(figsize=(8, 6))
plt.plot(np.arange(1, EPOCHS+1), HISTORY.history["loss"], label="training loss", color='blue')
plt.plot(np.arange(1, EPOCHS+1), HISTORY.history["val_loss"], label="validation loss", color='yellow')
plt.title("Training and Validation Loss")
plt.xlabel("Epoch #")
plt.ylabel("Loss")
plt.legend(loc="upper right")
plt.annotate(f"Final Train Loss: {final_train_loss:.4f}", (EPOCHS, final_train_loss), textcoords="offset points",
             xytext=(-10, 40), ha='right', color='blue')
plt.annotate(f"Final Val Loss: {final_val_loss:.4f}", (EPOCHS, final_val_loss), textcoords="offset points",
             xytext=(-10, 20), ha='right', color='yellow')
plt.margins(x=0, y=0)
tick_locations = np.arange(0, EPOCHS+1, 5)
tick_locations[0] = 1
plt.xticks(tick_locations, tick_locations)
plt.savefig(f"{folder_of_model}/loss_plot_dark_background.png", dpi=300, bbox_inches="tight")

plt.style.use("dark_background")
plt.figure(figsize=(8, 6))
plt.plot(np.arange(1, EPOCHS+1), HISTORY.history["accuracy"], label="training accuracy", color='blue')
plt.plot(np.arange(1, EPOCHS+1), HISTORY.history["val_accuracy"], label="validation accuracy", color='yellow')
plt.title("Training and Validation Accuracy")
plt.xlabel("Epoch #")
plt.ylabel("Accuracy")
plt.legend(loc="lower right")
plt.annotate(f"Final Train Acc: {final_train_acc:.4f}", (EPOCHS, final_train_acc), textcoords="offset points",
             xytext=(-10, -40), ha='right', color='blue')
plt.annotate(f"Final Val Acc: {final_val_acc:.4f}", (EPOCHS, final_val_acc), textcoords="offset points",
             xytext=(-10, -20), ha='right', color='yellow')
plt.margins(x=0, y=0)
tick_locations = np.arange(0, EPOCHS+1, 5)
tick_locations[0] = 1
plt.xticks(tick_locations, tick_locations)
plt.savefig(f"{folder_of_model}/accuracy_plot_dark_background.png", dpi=300, bbox_inches="tight")
```

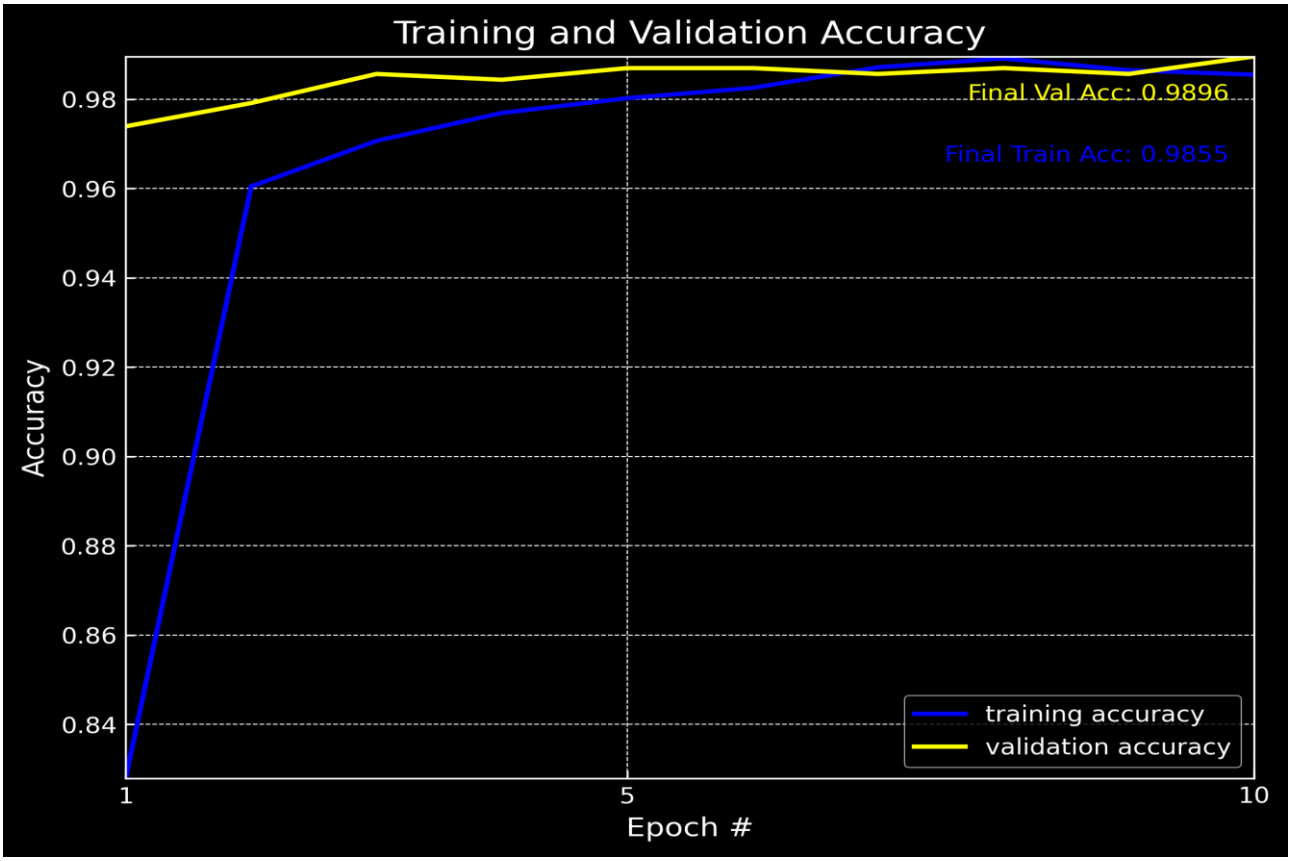
Εικόνα 3.25 Δημιουργία διαγραμμάτων για τις απώλειες και τις ορθότητες με σκούρο παρασκήνιο και διαφορετικό στυλ

Το κομμάτι κώδικα της εικόνας 3.25 είναι προαιρετικό διότι δημιουργεί δυο ίδια διαγράμματα με τους κώδικες των εικόνων 3.21 και 3.23, με τη μόνη διαφορά η αλλαγή του στυλ από “bmh” σε “dark_background”. Επίσης αλλάζουν και τα ονόματα αποθήκευσης των διαγραμμάτων, δηλαδή "loss_plot_dark_background.png" και "accuracy_plot_dark_background.png" πέρα από τα χρώματα και το σκούρο παρασκήνιο.

Αυτή η εναλλακτική απεικόνιση των διαγραμμάτων δημιουργήθηκε έτσι ώστε να καλύψει και ανθρώπους για τους οποίους τα διαγράμματα μπορεί να είναι είτε πιο ευδιάκριτα με το συγκεκριμένο στυλ, είτε αισθητικά να τους αρέσει περισσότερο. Παραδείγματα αυτών των διαγραμμάτων παρουσιάζονται στις εικόνες 3.26 και 3.27.



Εικόνα 3.26 Παράδειγμα διαγράμματος απωλειών με σκούρο παρασκήνιο

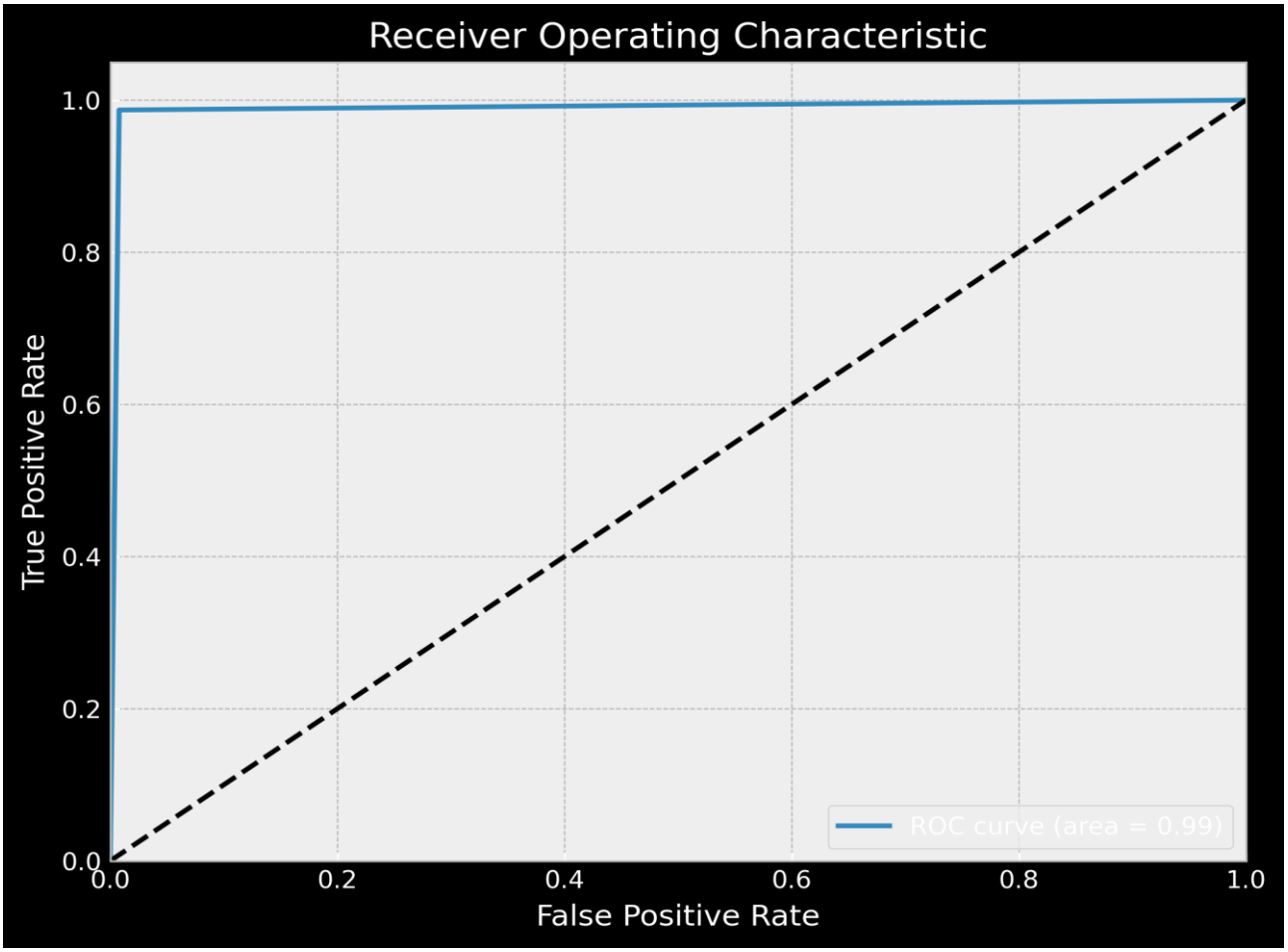


Εικόνα 3.27 Παράδειγμα διαγράμματος ορθοτήτων με σκούρο παρασκήνιο

```
235    plt.style.use("bmh")
236    plt.figure(figsize=(8, 6))
237    plt.plot(fpr, tpr, label='ROC curve (area = %0.2f)' % auc_score)
238    plt.plot([0, 1], [0, 1], 'k--')
239    plt.xlim([0.0, 1.0])
240    plt.ylim([0.0, 1.05])
241    plt.xlabel('False Positive Rate')
242    plt.ylabel('True Positive Rate')
243    plt.title('Receiver Operating Characteristic')
244    plt.legend(loc="lower right")
245    plt.margins(x=0, y=0)
246    plt.savefig(f"{folder_of_model}/ROC_plot.png", dpi=300, bbox_inches="tight")
247
248    print("[ΕΝΗΜΕΡΩΣΗ] Τέλος εκπαίδευσης του μοντέλου. Τερματισμός προγράμματος...")
```

Εικόνα 3.28 Δημιουργία διαγράμματος καμπύλης λειτουργικού χαρακτηριστικού δέκτη

Το τελευταίο κομμάτι εντολών του κώδικα εκπαίδευσης μοντέλου (εικόνα 3.28), δημιουργεί το διάγραμμα καμπύλης λειτουργικού χαρακτηριστικού δέκτη με βάση τα δεδομένα που υπολογίστηκαν νωρίτερα στον κώδικα. Η λογική των εντολών είναι ίδια με αυτήν της δημιουργίας για παράδειγμα του διαγράμματος απωλειών, με διαφορά στα δεδομένα κυρίως της συγκεκριμένης γραφικής και στον τρόπο που σχεδιάζεται. Ένα παράδειγμα τέτοιου διαγράμματος βρίσκεται στην εικόνα 3.29.



Εικόνα 3.29 Παράδειγμα διαγράμματος καμπύλης λειτουργικού χαρακτηριστικού δέκτη

3.3 Αποτελέσματα

Σε αυτήν την ενότητα παρουσιάζονται διάφορα μοντέλα που εκπαιδεύτηκαν με τον κώδικα που αναλύθηκε στην ενότητα 3.2 και πραγματοποιείται ο σχολιασμός των αποδόσεών τους σε σχέση με το βέλτιστο μοντέλο που δημιουργήθηκε. Το βέλτιστο ή ιδανικό μοντέλο αναλύεται πρώτο έτσι ώστε να είναι το πρότυπο παράδειγμα για τις άλλες δοκιμές μοντέλων που έγιναν, παρόλο που πρακτικά επιλέχθηκε ως ιδανικό μετά από τις δοκιμές. Πιο συγκεκριμένα, τα υπόλοιπα μοντέλα που δημιουργήθηκαν ή αλλιώς οι δοκιμές που έγιναν, έχουν να κάνουν με τις διαφορετικές τιμές των υπερπαραμέτρων ή το σύνολο των δεδομένων εκπαίδευσης.

Όλες οι δοκιμές εκτός από την πρώτη, συγκρίνονται με το βέλτιστο μοντέλο με βάση την αλλαγή της τιμής κάποιας από τις τρεις υπερπαραμέτρους του. Οι τρεις υπερπαράμετροι θα αναφέρονται για ευκολία με την ονομασία που έχουν σαν μεταβλητές μέσα στον κώδικα δηλαδή " INIT_LR " για τον ρυθμό εκπαίδευσης, "BS" για το μέγεθος του υποσυνόλου δεδομένων και "EPOCHS" για τις εποχές.

Για κάθε δοκιμή θα υπάρχουν αποτελέσματα, δηλαδή εικόνες διαγραμμάτων σχετικά με την απόδοση του κάθε μοντέλου. Με βάση αυτά τα αποτελέσματα και την αναφορά των μετρικών αξιολόγησης θα γίνεται η σύγκριση με το ιδανικό μοντέλο και η εξαγωγή συμπερασμάτων.

Ένα από τα αποτελέσματα αυτά, είναι το διάγραμμα των απωλειών του με την πορεία προσπάθειας μηδενισμού τους. Ειδικότερα, συγκρίνονται δύο γραφικές απωλειών στο ίδιο διάγραμμα, η μία αφορά τις απώλειες κατά την εκπαίδευση με τα δεδομένα εκπαίδευσης και η άλλη τις απώλειες κατά την επικύρωση με τα δοκιμαστικά δεδομένα. Πιο συγκεκριμένα, στο τέλος κάθε εποχής υπολογίζεται ο μέσος όρος των απωλειών του μοντέλου με βάση τα δεδομένα εκπαίδευσης που δέχτηκε και αντίστοιχα ο μέσος όρος με βάση τα δεδομένα επικύρωσης. Κάθε μέσος όρος φαίνεται σαν ένα σημείο πάνω στο διάγραμμα και όλα τα σημεία της κάθε εποχής δημιουργούν τη γραφική για τα δεδομένα εκπαίδευσης και για τα δεδομένα επικύρωσης. Είναι σημαντικό να παρατηρήσουμε την πορεία και την κατάληξη των τιμών κάθε γραφικής για να καταλάβουμε εάν το μοντέλο έχει κάποιο πρόβλημα σχετικά με την απόδοση ή την ακρίβειά του, όπως για παράδειγμα αυτό της υπερπροσαρμογής.

Το δεύτερο αποτέλεσμα για το μοντέλο είναι το διάγραμμα των ορθοτήτων του όπου και σε αυτήν την περίπτωση περιέχει δυο γραφικές, την πρώτη σχετικά με την εκπαίδευση και την δεύτερη σχετικά με την επικύρωση. Οι ορθότητες λειτουργούν αντίθετα από τις απώλειες, δηλαδή ξεκινάνε από χαμηλές τιμές, πάλι λόγο των τυχαίων παραμέτρων στην αρχή της εκπαίδευσης και έπειτα στοχεύουν να πετύχουν την υψηλότερη τιμή απόδοσης δηλαδή το 1 που αντιστοιχεί στο 100%.

Το τρίτο αποτέλεσμα αφορά την εμφάνιση της καμπύλης λειτουργικού χαρακτηριστικού δέκτη ή αλλιώς καμπύλη ROC μαζί με τη βαθμολογία AUC δηλαδή το ποσοστό της περιοχής κάτω από την καμπύλης. Όσο πιο πολύ τείνει προς τα αριστερά και πάνω όρια του διαγράμματος η καμπύλη, τόσο καλύτερη θεωρείται η απόδοση του μοντέλου. Επίσης όσο μεγαλύτερο είναι το ποσοστό της βαθμολογίας AUC επίσης δείχνει ποσοτικά το πόσο καλό είναι ένα μοντέλο.

Το τέταρτο και τελευταίο αποτέλεσμα είναι το μόνο που δεν απεικονίζεται σαν διάγραμμα, αλλά είναι μία αναφορά των μετρικών αξιολόγησης του μοντέλου. Αυτή η αναφορά περιέχει

πληροφορίες σχετικά με τις τιμές της ακρίβειας, ανάκλησης, βαθμολογίας F1, της ορθότητας του μοντέλου, τον μακρύ μέσο όρο όλων αυτών καθώς και τον σταθμισμένο μέσο όρο τους. Όσο πιο κοντά στο 1 είναι όλες αυτές οι τιμές τόσο καλύτερα λειτουργεί το μοντέλο.

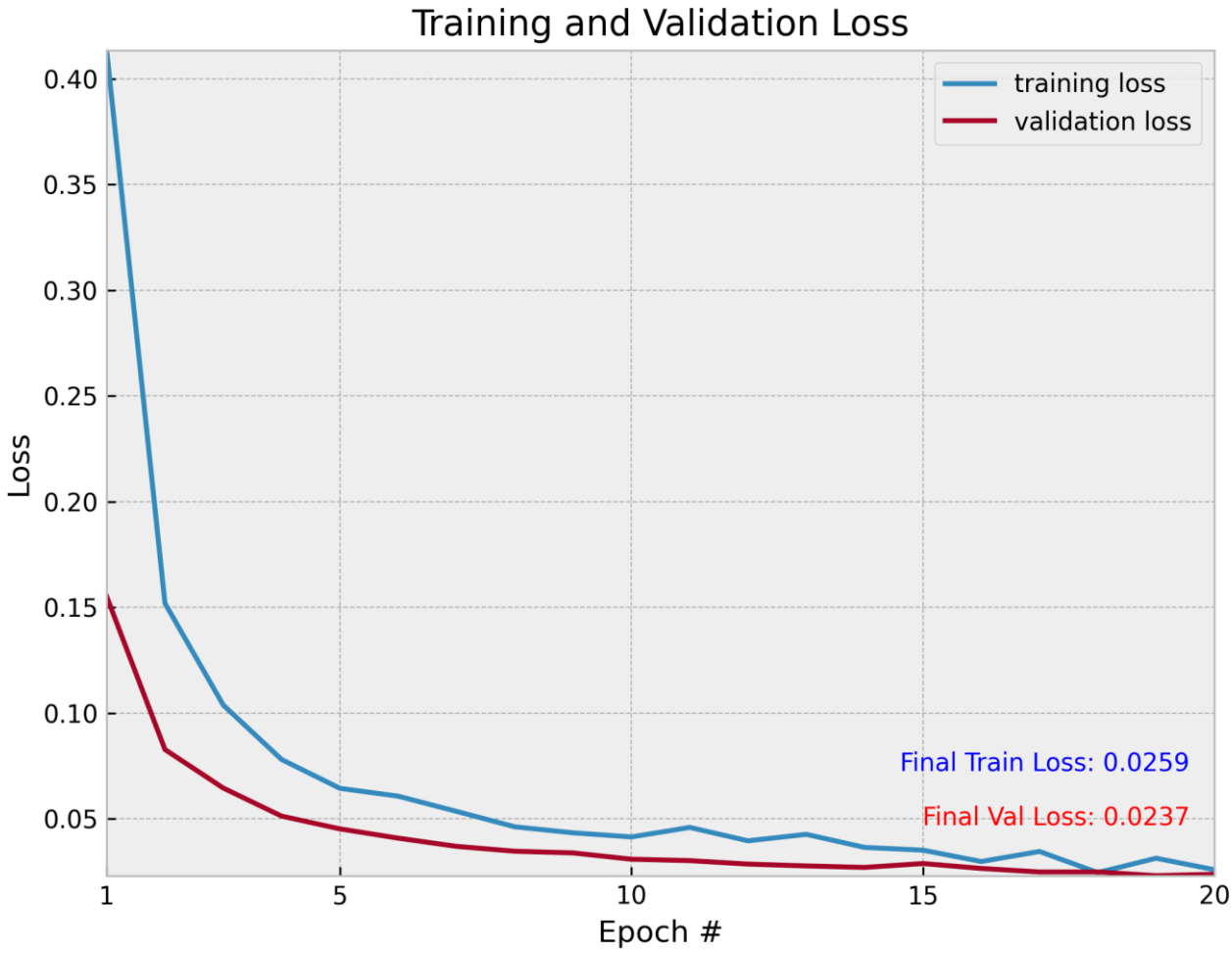
3.3.1 Το Βέλτιστο μοντέλο

Μετά από αρκετές προσπάθειες εκπαίδευσης του μοντέλου ανίχνευσης μάσκας και κάνοντας διάφορους συνδυασμούς με τις τιμές των υπερπαραμέτρων, καταλήξαμε στο βέλτιστο μοντέλο με την καλύτερη ακρίβεια και απόδοση. Το βέλτιστο ή αλλιώς ιδανικό μοντέλο εκπαιδεύτηκε με τις παρακάτω τιμές των υπερπαραμέτρων:

- Αριθμός εποχών/EPOCHS = 20
- Ρυθμός εκπαίδευσης/INIT_LR = 0.0001
- Μέγεθος υποσυνόλου δεδομένων/BS = 32

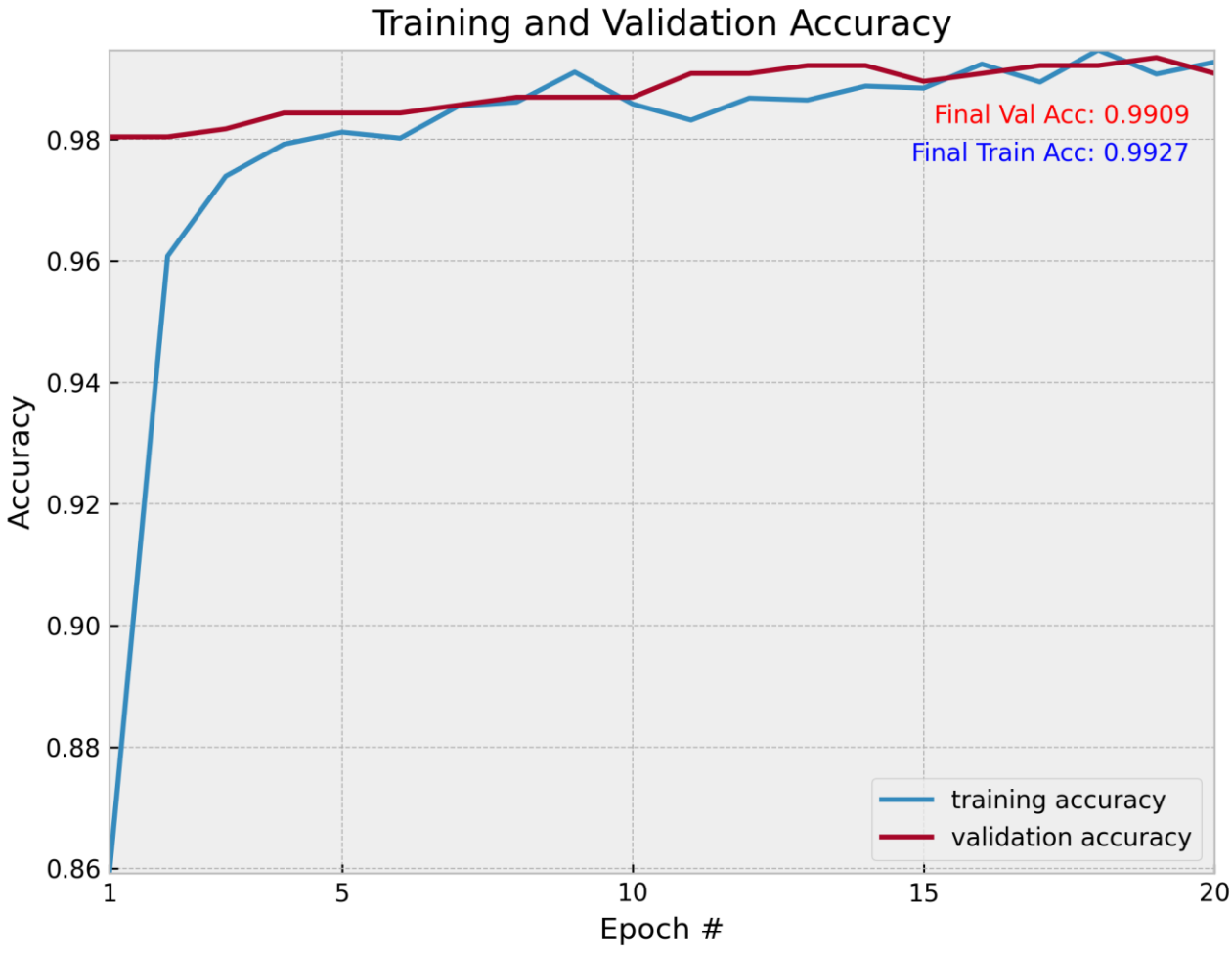
Επίσης εκπαιδεύτηκε πάνω σε ένα σύνολο δεδομένων 3833 εικόνων εκ των οποίων 1915 ήταν εικόνες με ανθρώπους με μάσκα και 1918 εικόνες με ανθρώπους χωρίς μάσκα.

Με βάση όλες αυτές τις ρυθμίσεις το μοντέλο μετά την εκπαίδευσή και την αξιολόγησή του, παρήγαγε τα αποτελέσματα που ακολουθούν.



Εικόνα 3.30 Διάγραμμα απωλειών βέλτιστου μοντέλου

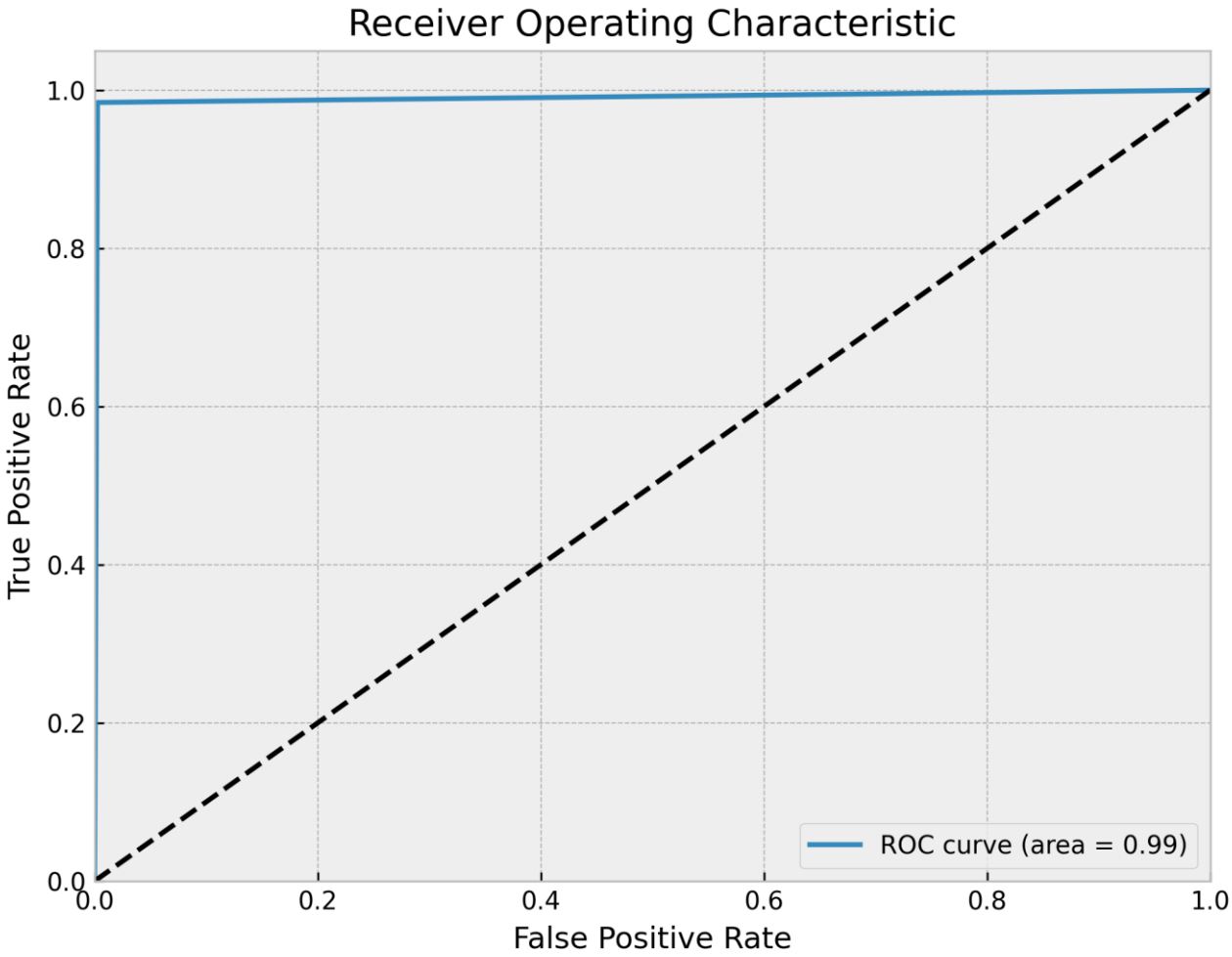
Στην εικόνα 3.30 του διαγράμματος απωλειών, βλέπουμε πως οι τιμές των απωλειών ξεκινάνε από κάπου ψηλά διότι οι παράμετροι του μοντέλου στην αρχή της εκπαίδευσής έχουν τυχαίες τιμές πράγμα που σημαίνει ότι οι προβλέψεις που θα κάνει δεν θα είναι καλές. Στη συνέχεια παρατηρούμε ότι οι τιμές των απωλειών, τόσο για τα δεδομένα εκπαίδευσης όσο και για αυτά της επικύρωσης, μειώνονται σε κάθε επανάληψη σχετικά ομαλά χωρίς απότομες αλλαγές. Αυτό μας δείχνει πως ο συνδυασμός των υπερπαραμέτρων "INIT_LR" και "BS" είναι αρκετά καλός και μειώνει σχετικά σταθερά τις απώλειες σε κάθε εποχή. Επίσης, ο αριθμός των εποχών "EPOCHS" είναι ιδανικός αφού οι τελικές τιμές απωλειών είναι σχεδόν μηδενικές στην 20ή εποχή ενώ αν επιλέγαμε λιγότερες εποχές π.χ. 10 όπως διακρίνεται από το διάγραμμα οι απώλειες θα ήταν υψηλότερες. Από την άλλη αν επιλέγαμε περισσότερες εποχές, τότε θα καθυστερούσαμε απλά την εκπαίδευση του μοντέλου αφού ο στόχος έχει επιτευχθεί ήδη κοντά στην 20ή εποχή. Ακόμα μία σημαντική παρατήρηση είναι πως οι δύο γραφικές απωλειών είναι κοντά συνεχώς η μία με την άλλη και ειδικότερα όσον αφορά τις τελικές τιμές τους. Αυτό σημαίνει πως δεν υπάρχει υπερπροσαρμογή αφού δεν υπάρχει μεγάλη απόκλιση μεταξύ των τιμών τους. Ούτε υποπροσαρμογή υπάρχει αφού οι τελικές τιμές απωλειών δεν είναι πολύ υψηλότερες του μηδενός.



Εικόνα 3.31 Διάγραμμα ορθοτήτων βέλτιστου μοντέλου

Η εικόνα 3.31 παρουσιάζει τις ορθότητες του βέλτιστου μοντέλου οι οποίες μοιάζουν να είναι πολύ καλές. Ενώ αυτές ξεκινούν χαμηλά συνεχίζουν την σχετικά ομαλή τους πορεία προς την τιμή 1 με πολύ μικρές εναλλαγές που δεν επηρεάζουν την απόδοση του μοντέλου σημαντικά. Οι τελικές τιμές τους είναι πολύ κοντά στο ένα, κάτι το οποίο είναι πολύ επιθυμητό και θεωρείται επιτυχία. Η

απόκλιση των δύο ορθοτήτων είναι πολύ μικρή που σημαίνει πως δεν υπάρχει κανένα ιδιαίτερο πρόβλημα σχετικά με το μοντέλο.



Εικόνα 3.32 Διάγραμμα καμπύλης ROC βέλτιστου μοντέλου

Σύμφωνα με το διάγραμμα καμπύλης ROC της εικόνας 3.32, το μοντέλο ανταποκρίνεται πολύ καλά ως προς την σωστή πρόβλεψη των δεδομένων. Η βαθμολογία AUC που αναφέρεται κάτω δεξιά του διαγράμματος ως “area = 0.99” δείχνει πως η απόδοση του μοντέλου είναι κατά 99% άψογη. Αυτό προκύπτει από την γραφική της καμπύλης ROC η οποία τείνει προς τα αριστερά με αποτέλεσμα η περιοχή κάτω από αυτήν να είναι μεγάλη.

classification_report.txt

	precision	recall	f1-score	support
with_mask	0.98	1.00	0.99	383
without_mask	1.00	0.98	0.99	384
accuracy			0.99	767
macro avg	0.99	0.99	0.99	767
weighted avg	0.99	0.99	0.99	767

Εικόνα 3.33 Αναφορά μετρικών αξιολόγησης του βέλτιστου μοντέλου

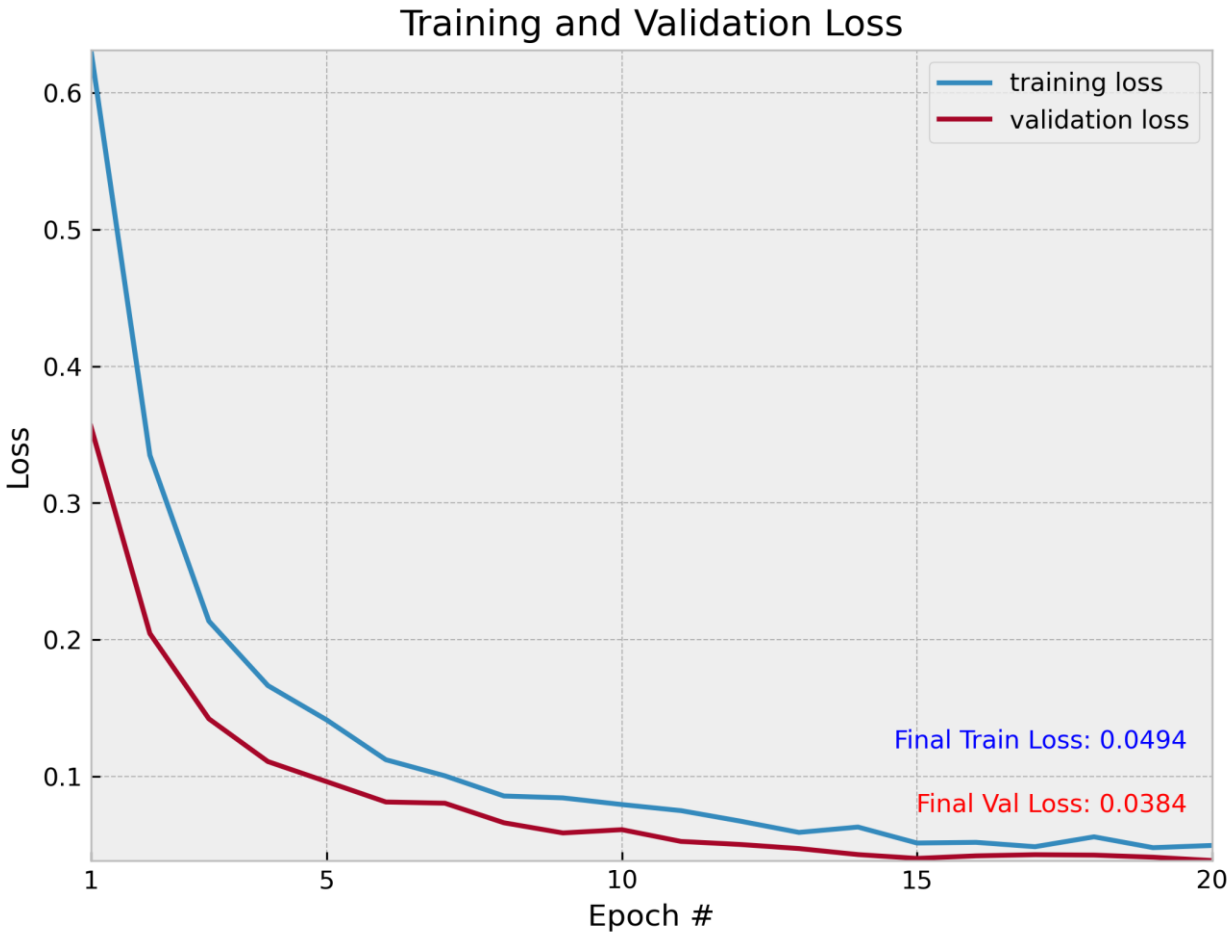
Η αναφορά μετρικών αξιολόγησης της εικόνας 3.33 είναι πολύ καλή αφού οι περισσότερες τιμές είναι κοντά στο ένα αν όχι το ίδιο το ένα. Αρχικά ως support εννοούμαι των αριθμό των εικόνων κάθε κλάσης που εκπαιδεύτηκε το μοντέλο. Για κάθε κλάση έχουν υπολογιστεί οι μετρικές τους

ξεχωριστά. Επίσης έχει μετρηθεί ο μακρύς και ο σταθμισμένος μέσος όρος που προκύπτουν από τα μετρικά και των δύο κλάσεων.

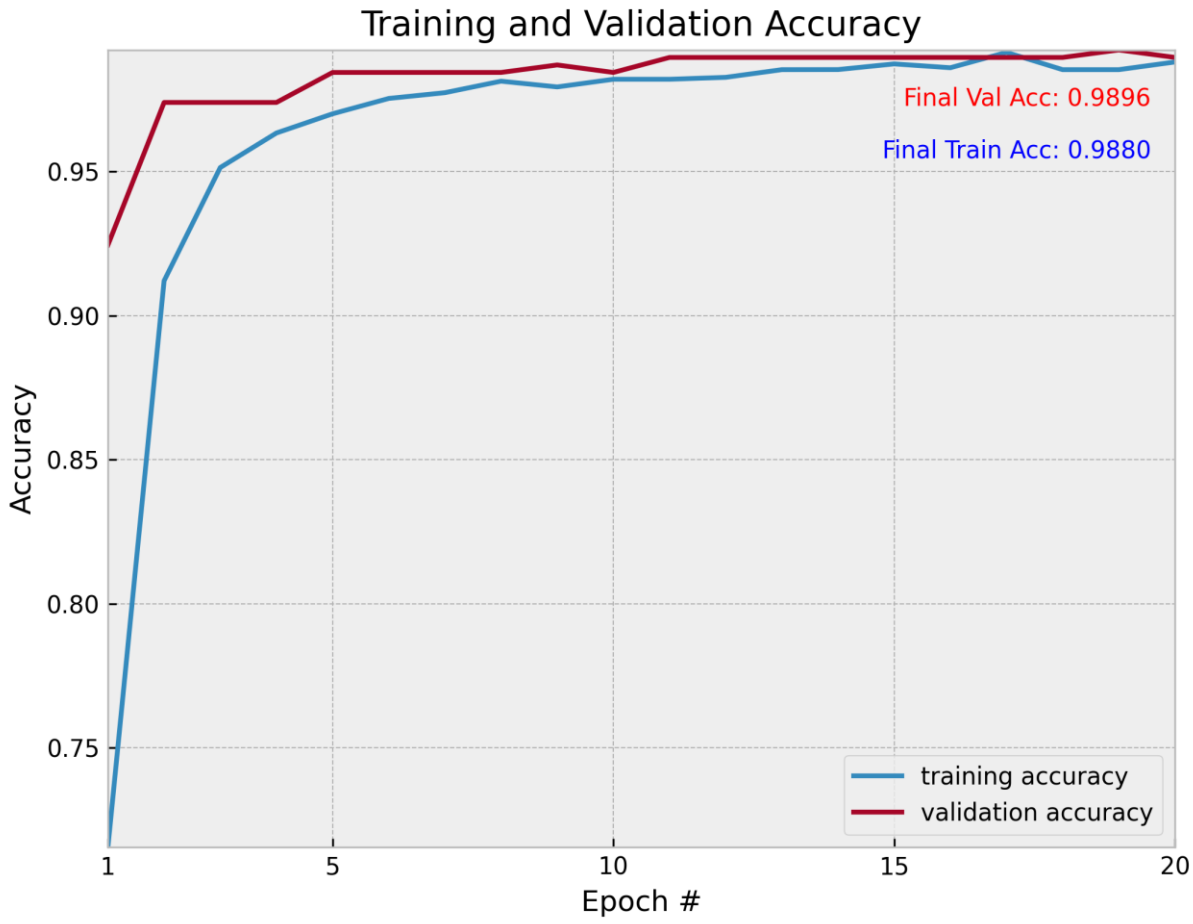
Αρχικά παρατηρούμε ότι η ακρίβεια και των δύο κλάσεων του μοντέλου είναι πολύ υψηλή με την μία κλάση μάλιστα να αγγίζει το 1 δηλαδή το 100%. Έπειτα η ανάκληση παρομοίως είναι υψηλή, όπως η βαθμολογία F1 και η γενική ορθότητα του μοντέλου. Οι τιμές αυτές συνολικά μας δείχνουν πως το μοντέλο δεν έχει κάποιο πρόβλημα που χρειάζεται να διορθώσουμε και πως είναι έτοιμο για χρήση.

3.3.2 Δοκιμή 1: Μειωμένα δεδομένα εκπαίδευσης

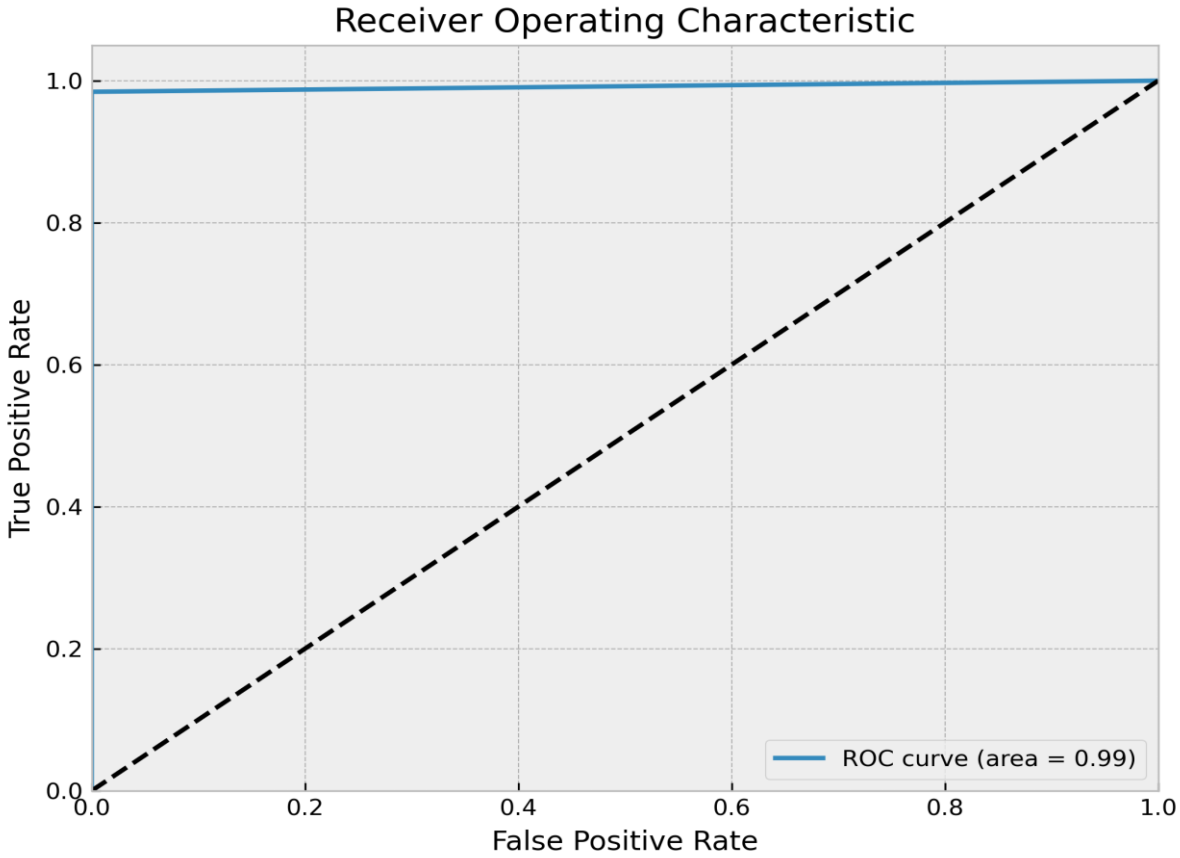
Σε αυτή την δοκιμή εκπαιδεύτηκε ένα μοντέλο ακριβώς με τις ίδιες υπερπαραμέτρους του βέλτιστου αλλά με τα μισά δεδομένα εκπαίδευσης, δηλαδή 1917 εικόνες εκ των οποίων οι 958 περιλάμβαναν ανθρώπους με μάσκα ενώ οι 959 ανθρώπους χωρίς μάσκα. Η γενική παρατήρηση με βάση τα αποτελέσματα ήταν πως το μοντέλο αυτό θα μπορούσε επίσης να θεωρηθεί βέλτιστο διότι έχει τιμές απόδοσης παραπλήσιες με το ιδανικό μοντέλο. Παρόλα αυτά επειδή τα συμπεράσματα πρέπει να βγαίνουν από τη λογική και όχι μόνο την παρατήρηση, δεν θα μπορούσε να θεωρηθεί βέλτιστο αυτό το μοντέλο. Αυτό συμβαίνει διότι όσο καλά και να εκπαιδεύτηκε το μοντέλο στα δεδομένα που του δώσαμε, ακόμα και στα άγνωστα για την επικύρωση, σίγουρα ένα μοντέλο με περισσότερη ποικιλία εικόνων έχει περισσότερες γνώσεις. Όπως θα δείτε παρακάτω οι γραφικές καθώς και οι μετρικές αξιολόγησης είναι σχεδόν ίδιες με του βέλτιστου μοντέλου.



Εικόνα 3.34 Διάγραμμα απωλειών δοκιμής 1



Εικόνα 3.35 Διάγραμμα ορθοτήτων δοκιμής 1



Εικόνα 3.36 Διάγραμμα καμπύλης ROC δοκιμής 1

classification_report.txt

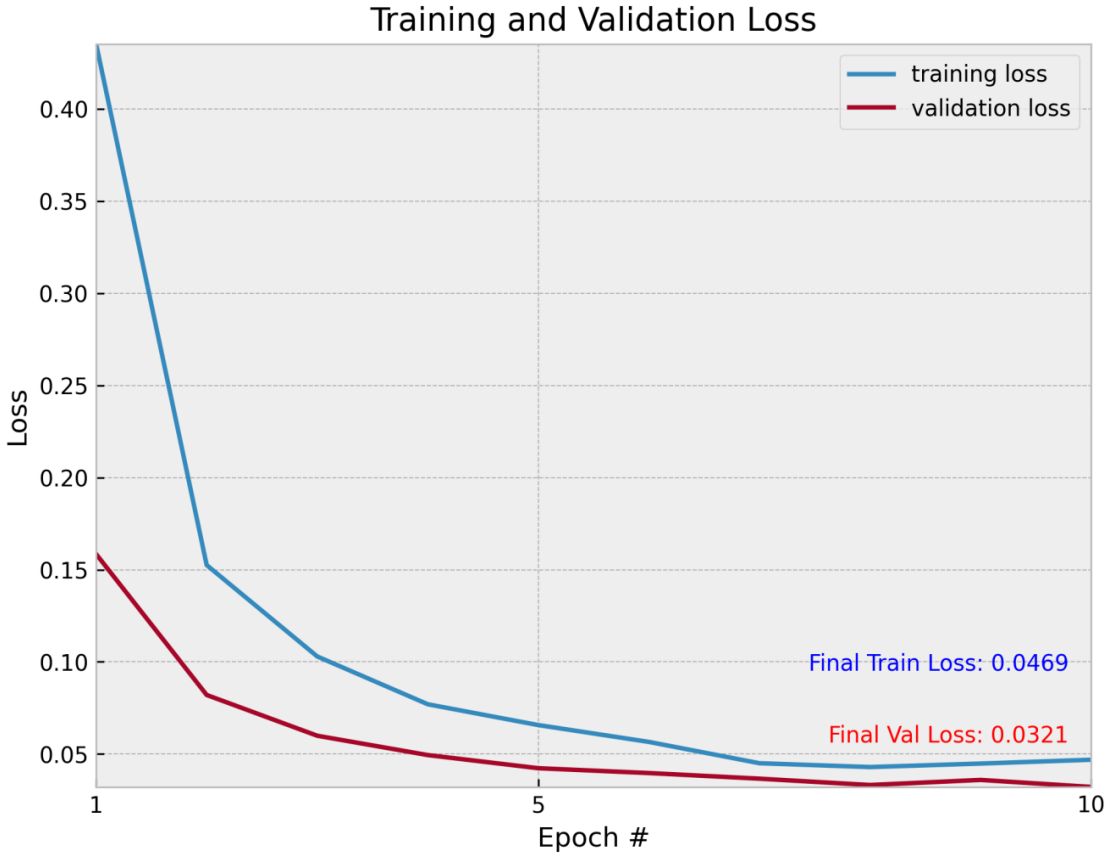
	precision	recall	f1-score	support
with_mask	0.98	0.99	0.99	192
without_mask	0.99	0.98	0.99	192
accuracy			0.99	384
macro avg	0.99	0.99	0.99	384
weighted avg	0.99	0.99	0.99	384

Εικόνα 3.37 Αναφορά μετρικών αξιολόγησης της δοκιμής 1

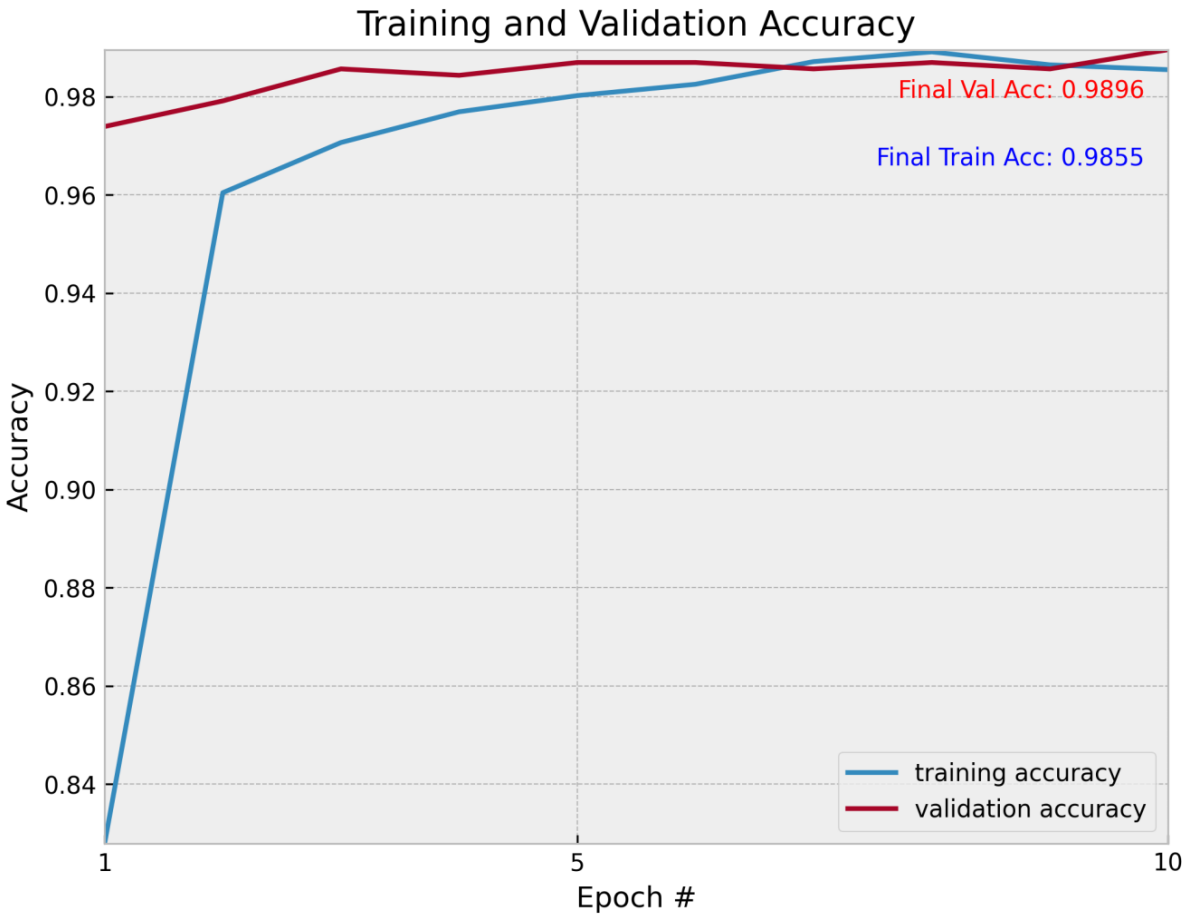
3.3.3 Δοκιμή 2: Σταθερά INIT_LR, BS και αλλαγή EPOCHS

3.3.3.1 Περίπτωση Α: Μειωμένο EPOCHS = 10

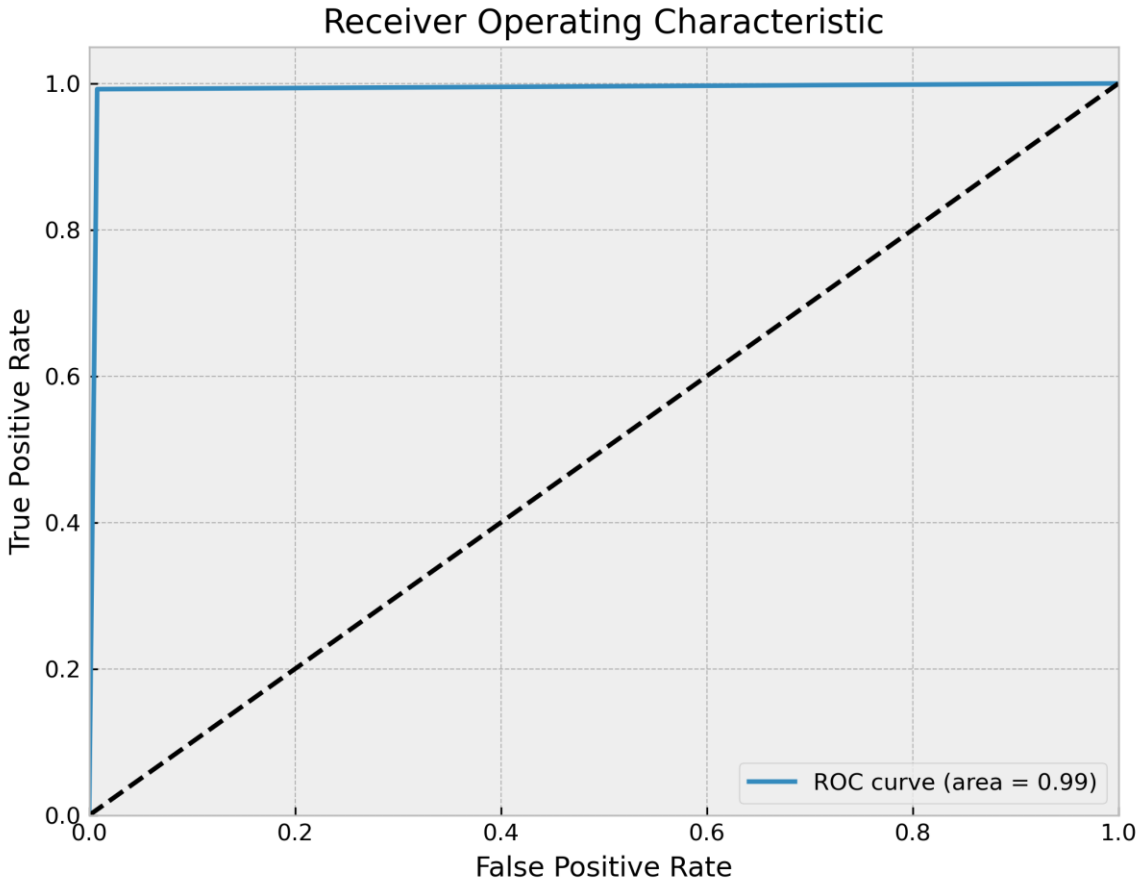
Με την μείωση των εποχών όπως θα παρατηρηθεί και στα διαγράμματα που θα ακολουθήσουν, το μοντέλο δεν προλαβαίνει να εκπαιδευτεί τόσο καλά με αποτέλεσμα οι τιμές απωλειών να είναι λίγο μεγαλύτερες από αυτές του βέλτιστου μοντέλου όπως και οι ορθότητες είναι μικρότερες. Το διάγραμμα ROC δείχνει την γενική απόδοση του μοντέλου ως πολύ καλή, πράγμα που δεν σημαίνει ότι δεν είναι αλλά επειδή η διαφορά κρίνεται στην λεπτομέρεια, θα θέλαμε οι τιμές του μοντέλου να είναι ακόμα καλύτερες εφόσον γνωρίζουμε ότι έχει περιθώριο βελτίωσης. Όσον αφορά τις μετρικές αξιολόγησης ήταν αναμενόμενο πως θα είναι παραπλήσιες με αυτές του ιδανικού μοντέλου αλλά όχι ίδιες.



Εικόνα 3.38 Διάγραμμα απωλειών δοκιμής 2, περίπτωσης Α



Εικόνα 3.39 Διάγραμμα ορθοτήτων δοκιμής 2, περίπτωσης Α



Εικόνα 3.40 Διάγραμμα καμπύλης ROC δοκιμής 2, περίπτωσης Α

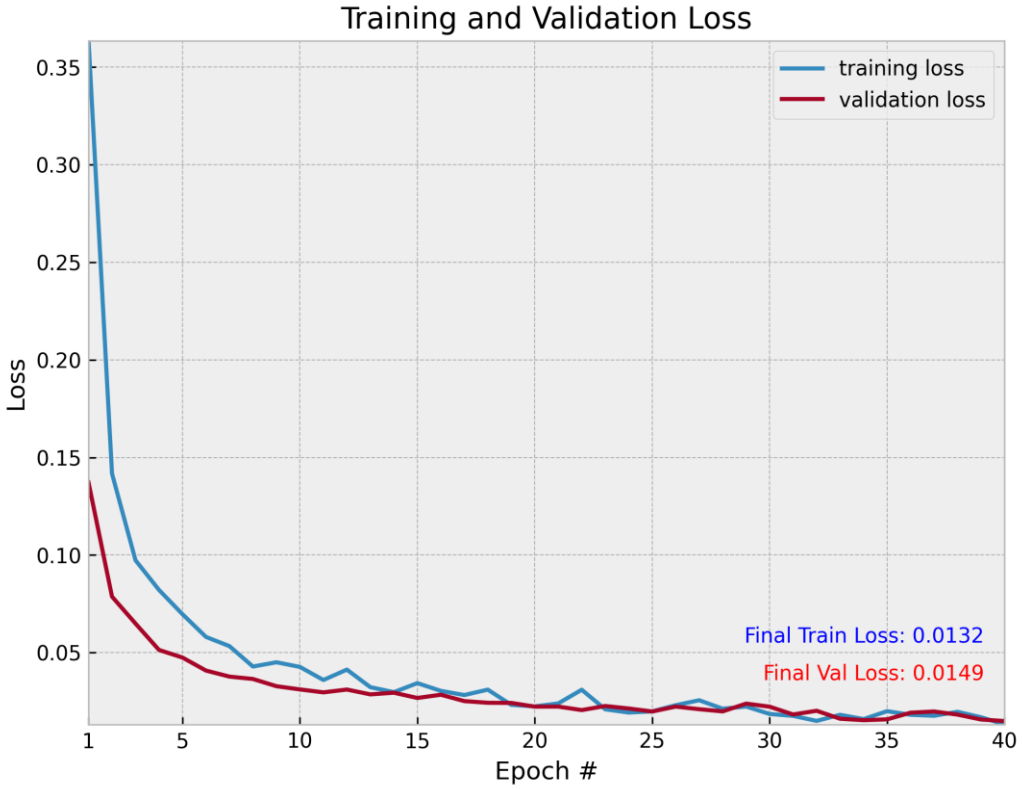
classification_report.txt

	precision	recall	f1-score	support
with_mask	0.99	0.99	0.99	383
without_mask	0.99	0.99	0.99	384
accuracy			0.99	767
macro avg	0.99	0.99	0.99	767
weighted avg	0.99	0.99	0.99	767

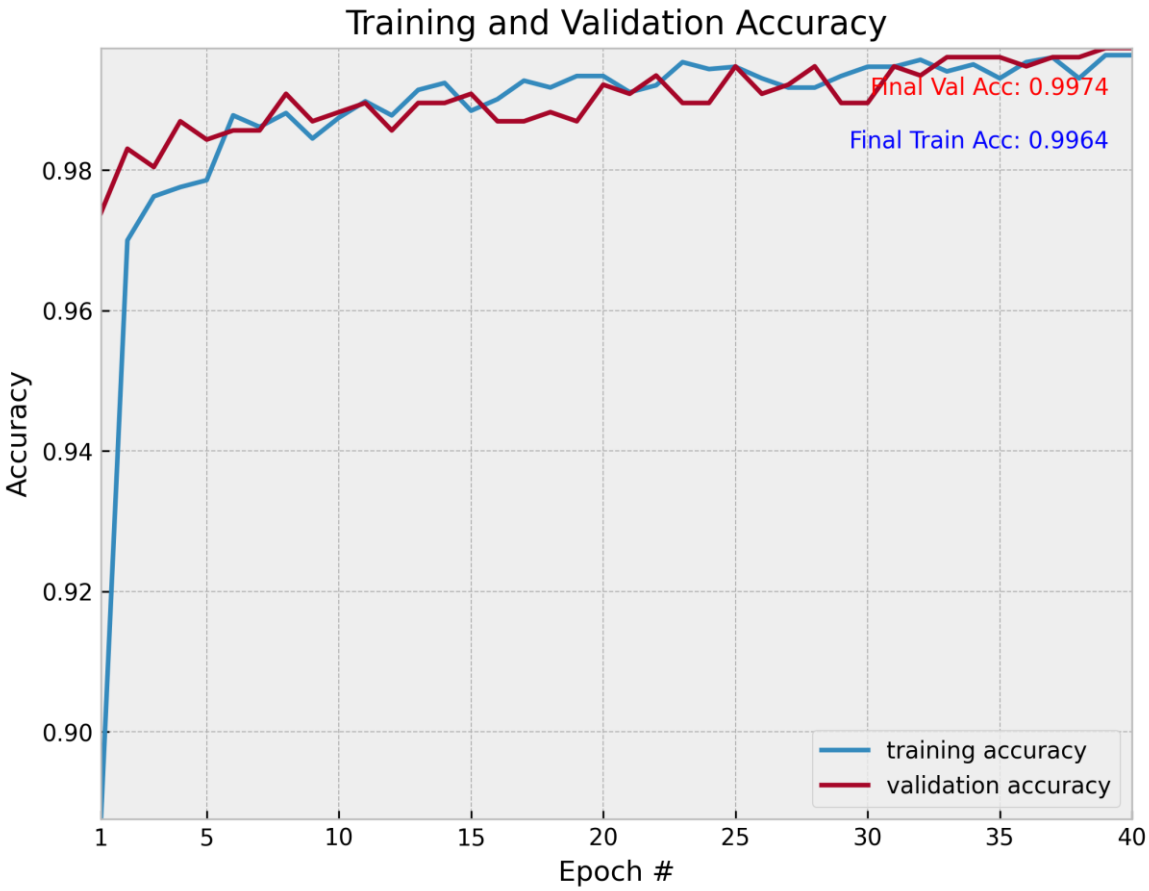
Εικόνα 3.41 Αναφορά μετρικών αξιολόγησης της δοκιμής 2, περίπτωσης Α

3.3.3.2 Περίπτωση Β: Αυξημένο EPOCHS = 40

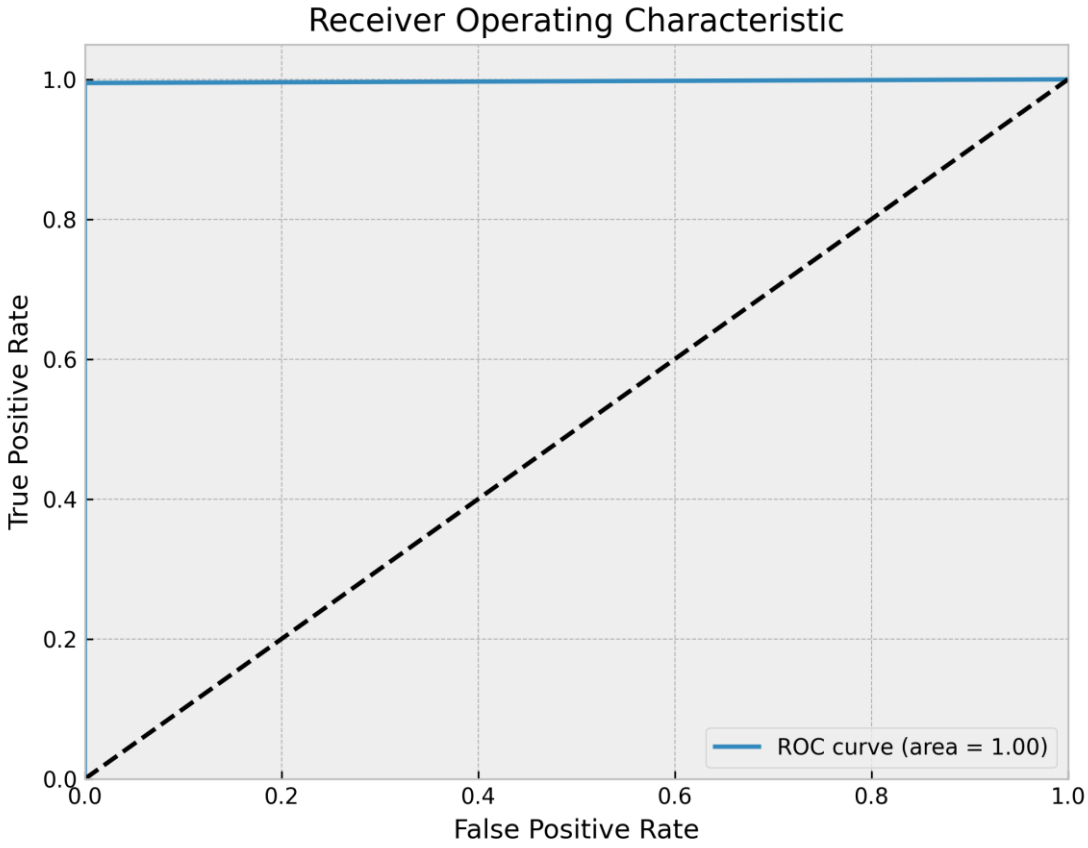
Η αύξηση των εποχών δεν επιφέρει αρνητικά αποτελέσματα σχετικά με την απόδοση του μοντέλου, απλά κάνει πιο χρονοβόρα την διαδικασία εκπαίδευσης χωρίς να το βελτιώνει σημαντικά περισσότερο. Θεωρητικά το μοντέλο περισσότερων εποχών θα μπορούσε να θεωρηθεί βέλτιστο διότι κρίνοντας την λεπτομέρεια στις τιμές των διαγραμμάτων του, έχει καλύτερες τιμές από το ιδανικό μοντέλο. Η διαφορά όμως αυτή των τιμών είναι πολύ μικρή και επειδή μας ενδιαφέρει να κερδίσουμε χρόνο προτιμάμε το μοντέλο με τις 20 εποχές. Αυτό συμβαίνει επειδή στην πράξη το βέλτιστο μοντέλο με το μοντέλο των αυξημένων εποχών επιφέρουν τα ίδια αποτελέσματα κατά την εφαρμογή τους. Οπότε μόνο λόγο αυτής της λογικής προτιμάμε το μοντέλο των 20 εποχών, επειδή θεωρητικά δεν χρειάζεται να εκπαιδευτεί για παραπάνω εποχές το μοντέλο αφού ήδη έχει φτάσει κοντά στο μέγιστο, όσον αφορά την ορθότητα. Αν η απόκλιση για παράδειγμα του μοντέλου 20 εποχών με το μοντέλο 40 εποχών ήταν ακόμα και 3 μονάδες τότε ναι θα προτιμούσαμε το μοντέλο των 40 εποχών.



Εικόνα 3.42 Διάγραμμα απωλειών δοκιμής 2, περίπτωσης Β



Εικόνα 3.43 Διάγραμμα ορθοτήτων δοκιμής 2, περίπτωσης Β



Εικόνα 3.44 Διάγραμμα καμπύλης ROC δοκιμής 2, περίπτωσης Β

classification_report.txt

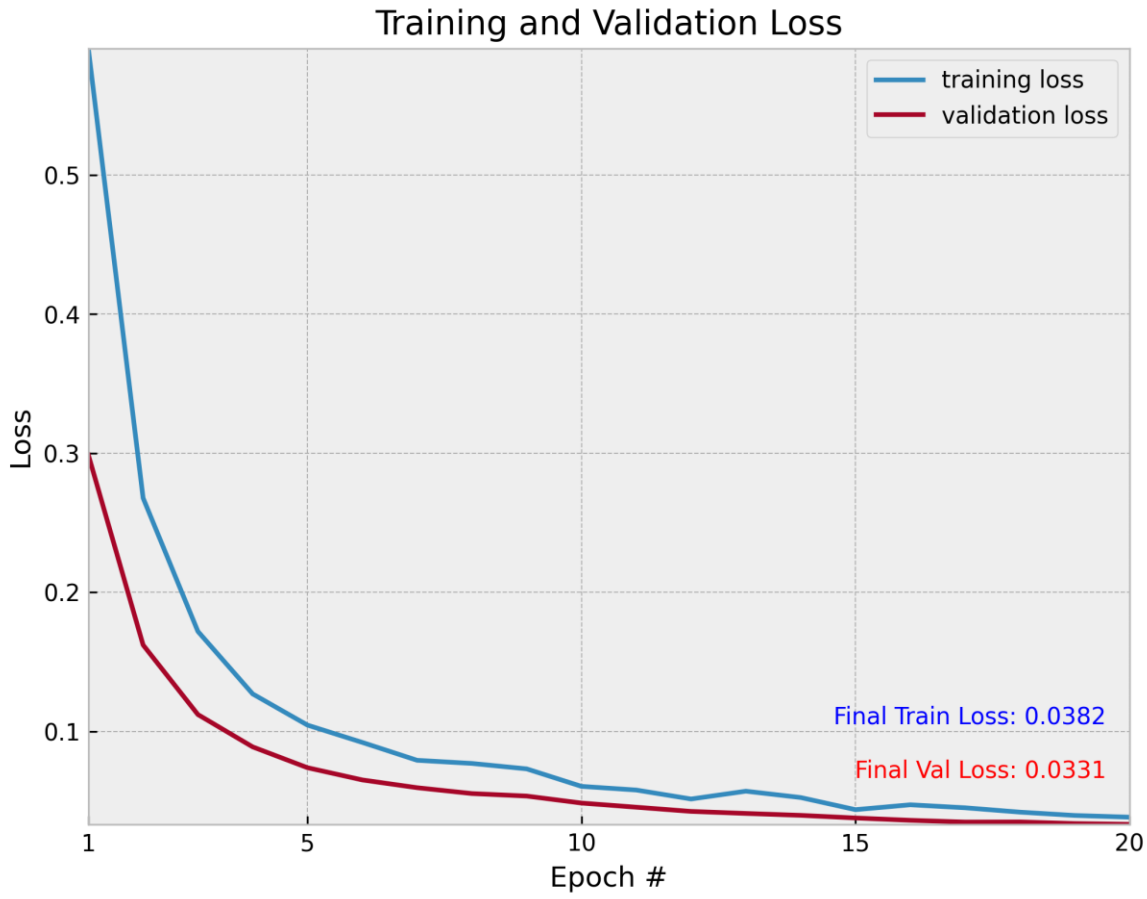
	precision	recall	f1-score	support
with_mask	0.99	1.00	1.00	383
without_mask	1.00	0.99	1.00	384
accuracy			1.00	767
macro avg	1.00	1.00	1.00	767
weighted avg	1.00	1.00	1.00	767

Εικόνα 3.45 Αναφορά μετρικών αξιολόγησης της δοκιμής 2, περίπτωσης Β

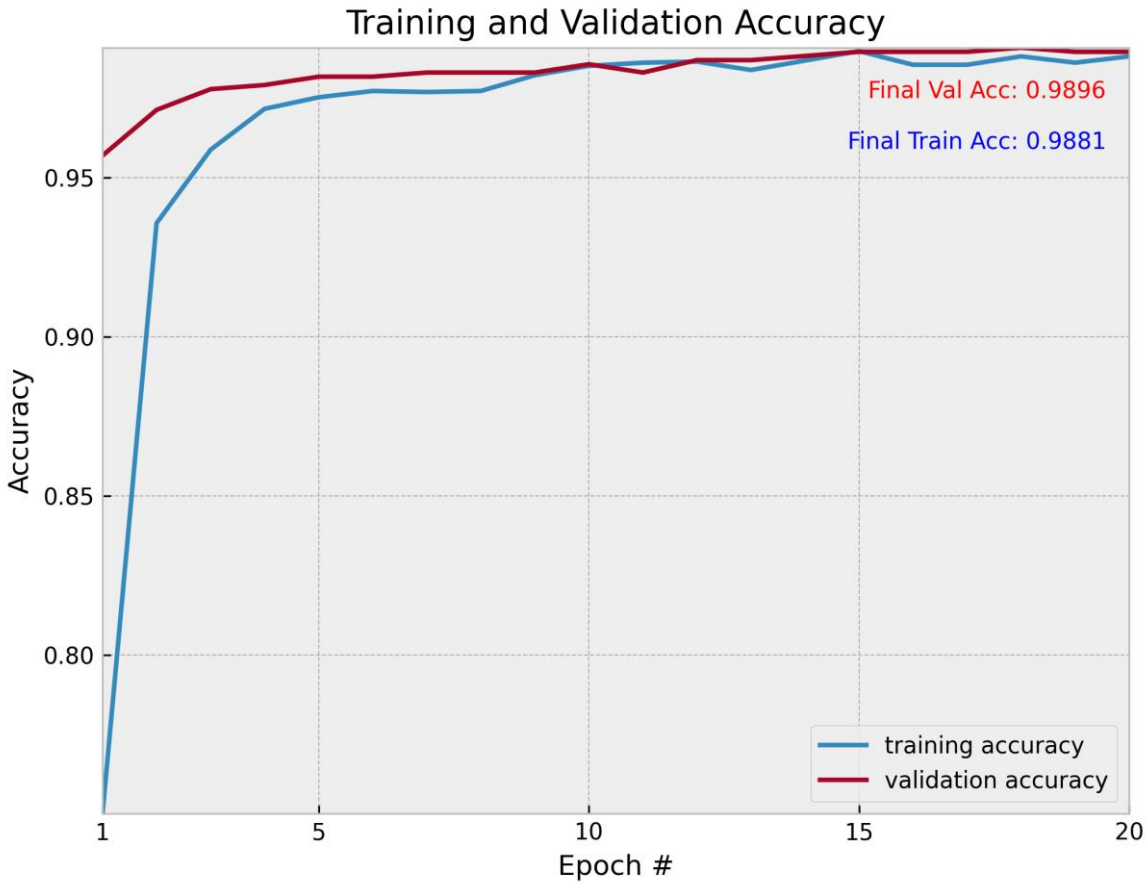
3.3.4 Δοκιμή 3:Σταθερά BS, EPOCHS και αλλαγή INIT_LR

3.3.4.1 Περίπτωση Α: Μειωμένο INIT_LR = 0.00005

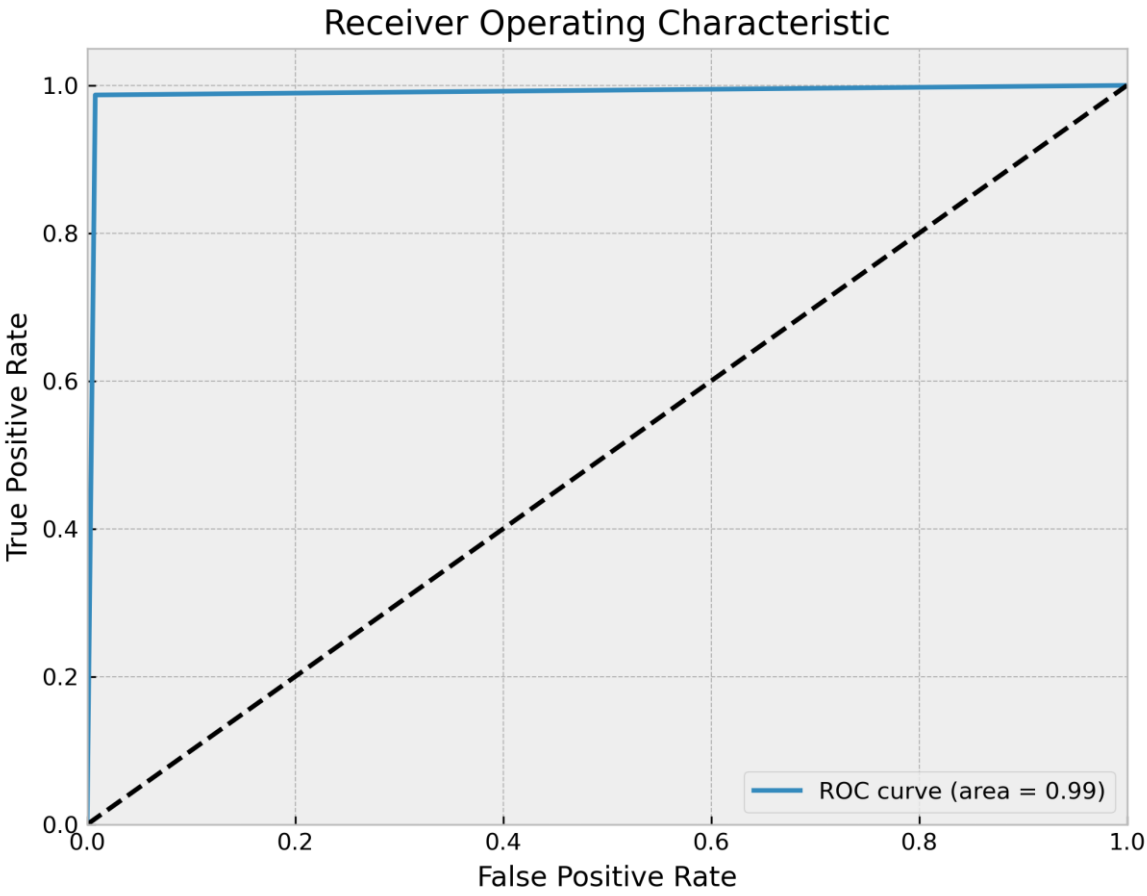
Εκπαιδεύοντας αυτό το μοντέλο με μειωμένο ρυθμό εκπαίδευσης παρατηρούμε, στα διαγράμματα που θα ακολουθήσουν, πως οι τελικές τιμές απωλειών είναι μεγαλύτερες από τις επιθυμητές όπως αντίστοιχα και οι ορθότητες είναι χαμηλότερες από αυτές του βέλτιστου μοντέλου. Αυτό σημαίνει πως μικρότερος ρυθμός εκπαίδευσης από τον ιδανικό που επιλέξαμε, δεν βοηθάει στην βελτιστοποίηση του μοντέλου μας. Ακόμα και οι μετρικές αξιολόγησης, αν και παρόμοιες δεν είναι καλύτερες από του ιδανικού μοντέλου. Η καμπύλη ROC και σε αυτή την περίπτωση είναι σχεδόν πανομοιότυπη με αυτή του βέλτιστου μοντέλου γιατί πρακτικά και αυτό το μοντέλο είναι αρκετά καλό στις προβλέψεις του.



Εικόνα 3.46 Διάγραμμα απωλειών δοκιμής 3, περίπτωσης Α



Εικόνα 3.47 Διάγραμμα ορθοτήτων δοκιμής 3, περίπτωσης Α



Εικόνα 3.48 Διάγραμμα καμπύλης ROC δοκιμής 3, περίπτωσης Α

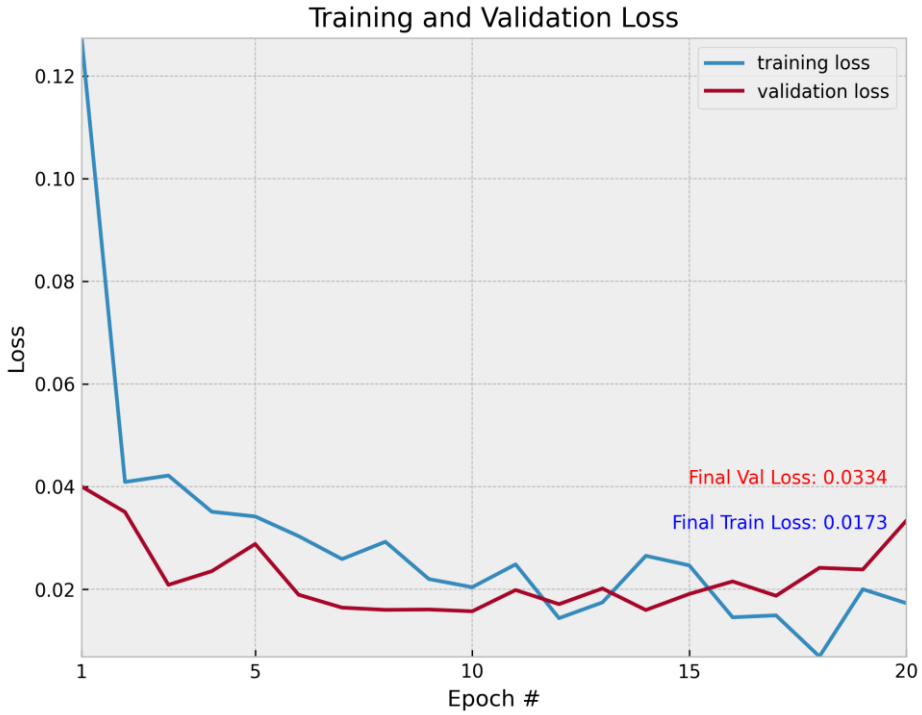
classification_report.txt

	precision	recall	f1-score	support
with_mask	0.98	0.99	0.99	383
without_mask	0.99	0.98	0.99	384
accuracy			0.99	767
macro avg	0.99	0.99	0.99	767
weighted avg	0.99	0.99	0.99	767

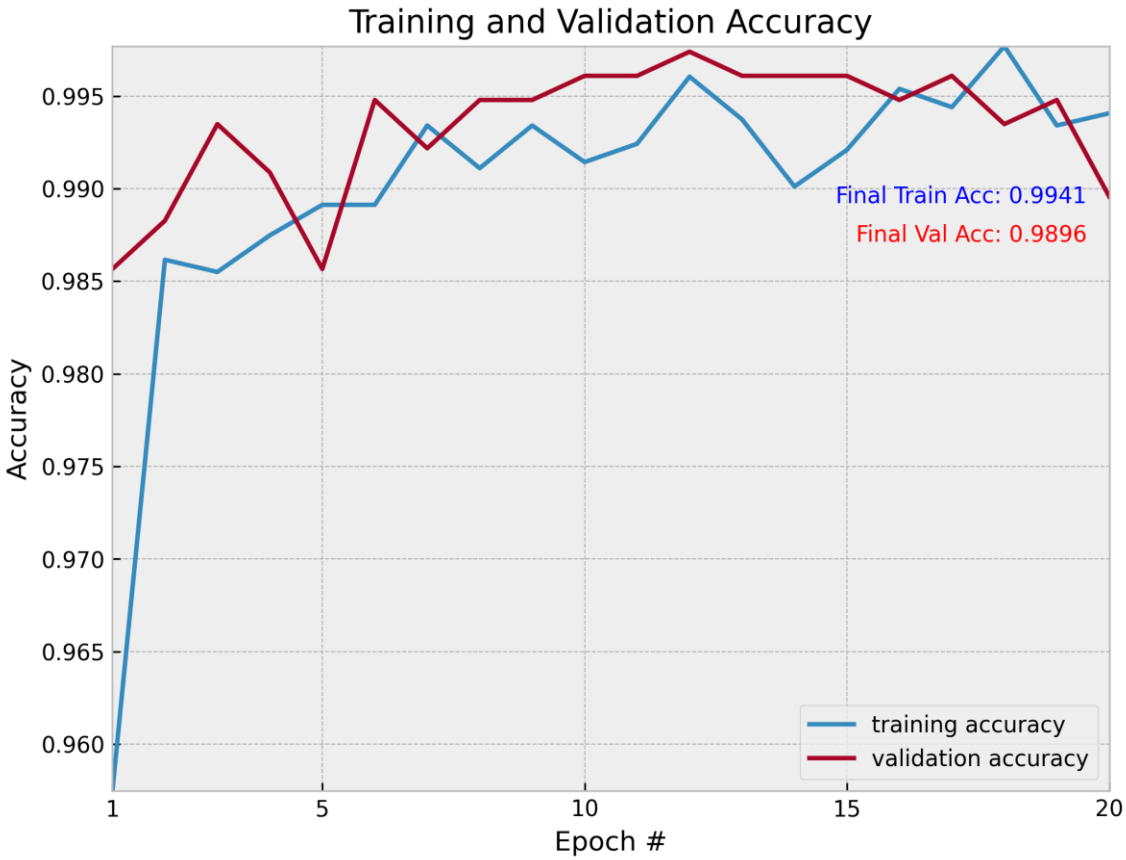
Εικόνα 3.49 Αναφορά μετρικών αξιολόγησης της δοκιμής 3, περίπτωσης Α

3.3.4.2 Περίπτωση Β: Αυξημένο INIT_LR = 0.001

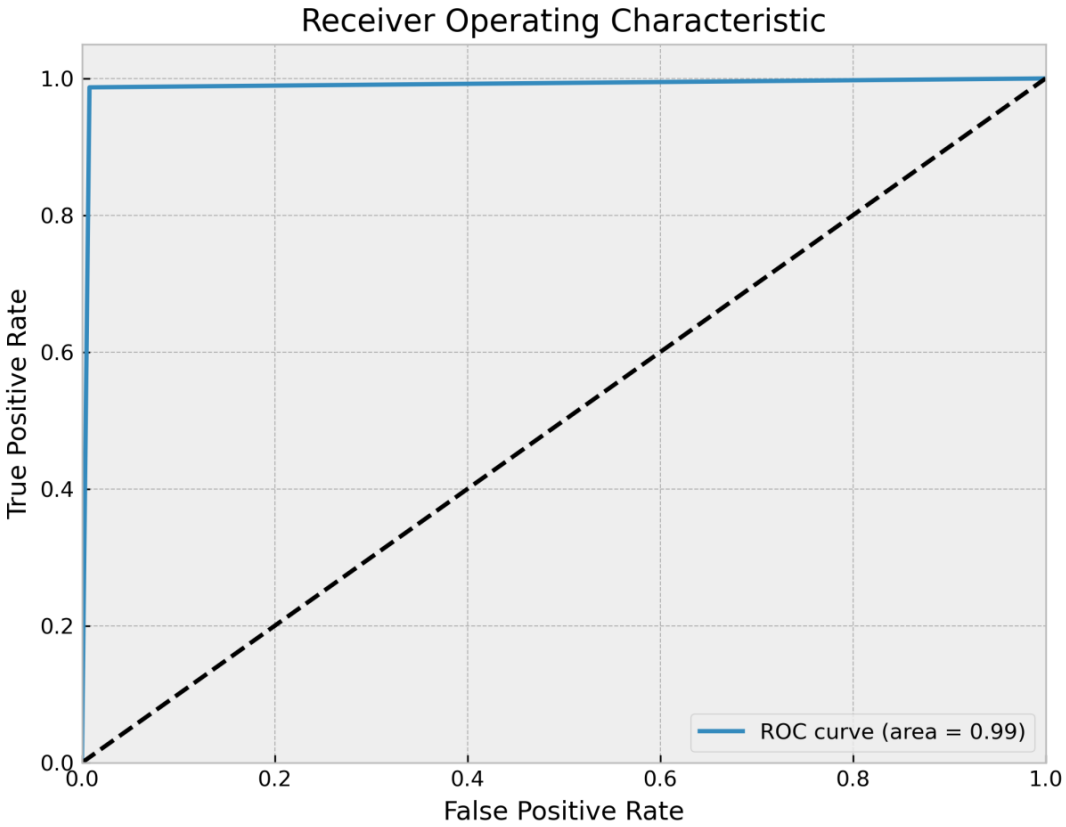
Κατά την αύξηση του ρυθμού εκπαίδευσης του μοντέλου παρατηρούμε πως το μοντέλο δεν είναι καλά εκπαιδευμένο και το αν τα αποτελέσματα δείξουν ότι το μοντέλο έχει καλή απόδοση αυτό κρίνεται στην τύχη του. Αρχικά, στο πρώτο διάγραμμα, οι εναλλαγές κατά την προσπάθεια μείωσης των απωλειών είναι μεγάλες, λόγω του μεγαλύτερου ρυθμού εκπαίδευσης που πρακτικά είναι το μέγεθος αυτών των εναλλαγών ή το βήμα. Αυτό έχει ως αποτέλεσμα να μην επιτυγχάνεται απαραίτητα η μείωση των απωλειών αλλά ίσως και να αυξάνεται όπως φαίνεται. Αυτό κάνει το μοντέλο χειρότερο έχοντας αρκετά μεγαλύτερη τελική τιμή απωλειών σε σχέση με το βέλτιστο μοντέλο. Αντίστοιχα το ίδιο συμβαίνει και με τις ορθότητες οι οποίες τη μία στιγμή βρίσκονται κοντά στην μονάδα και την άλλη μακριά της. Επίσης υπάρχει η πιθανότητα της υπερπροσαρμογής του μοντέλου μιας και η τελική απόκλιση των τιμών απώλειας εκπαίδευσης από την απώλεια επικύρωσης είναι σχετικά μεγάλη. Οι μετρικές αξιολόγησης καθώς και η καμπύλη ROC δείχνουν να έχουν καλές τιμές αλλά αυτό όπως προαναφέρθηκε οφείλεται στην τυχαιότητα. Το μοντέλο δείχνει να εκπαιδεύτηκε σωστά αλλά παρόλα αυτά οι αποκλίσεις του κατά την διαδικασία εκπαίδευσης ήταν μεγάλες σε σχέση πάντα με τις ιδανικές.



Εικόνα 3.50 Διάγραμμα απωλειών δοκιμής 3, περίπτωσης Β



Εικόνα 3.51 Διάγραμμα ορθοτήτων δοκιμής 3, περίπτωσης Β



Εικόνα 3.52 Διάγραμμα καμπύλης ROC δοκιμής 3, περίπτωσης Β

classification_report.txt

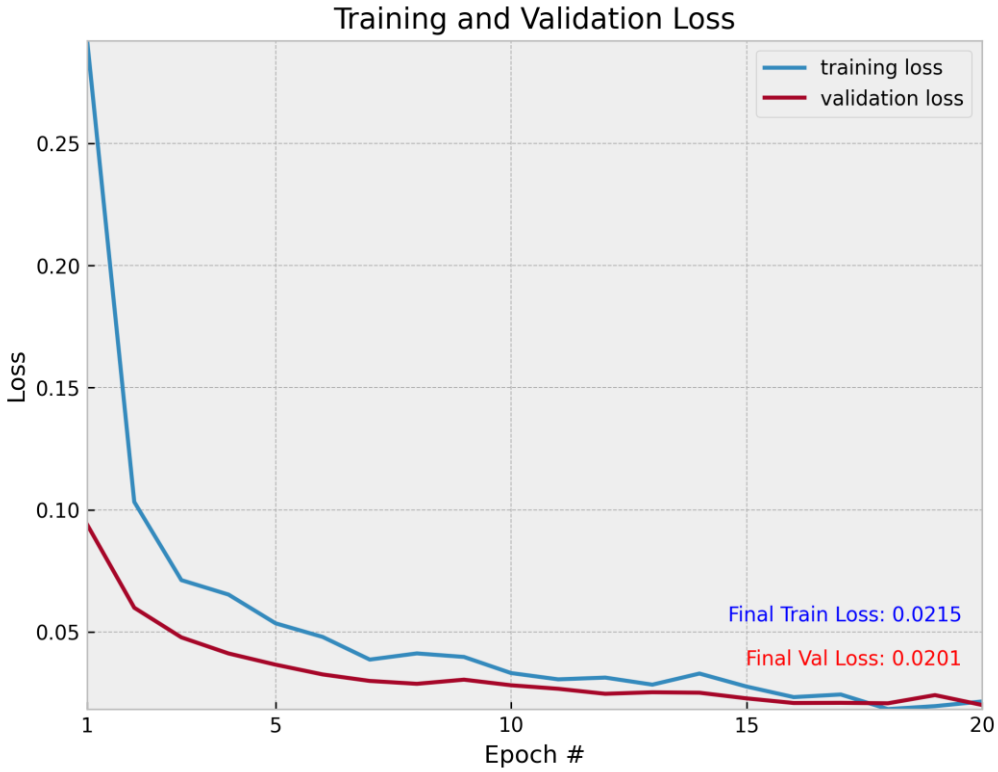
	precision	recall	f1-score	support
with_mask	0.98	0.99	0.99	383
without_mask	0.99	0.98	0.99	384
accuracy			0.99	767
macro avg	0.99	0.99	0.99	767
weighted avg	0.99	0.99	0.99	767

Εικόνα 3.53 Αναφορά μετρικών αξιολόγησης της δοκιμής 3, περίπτωσης Β

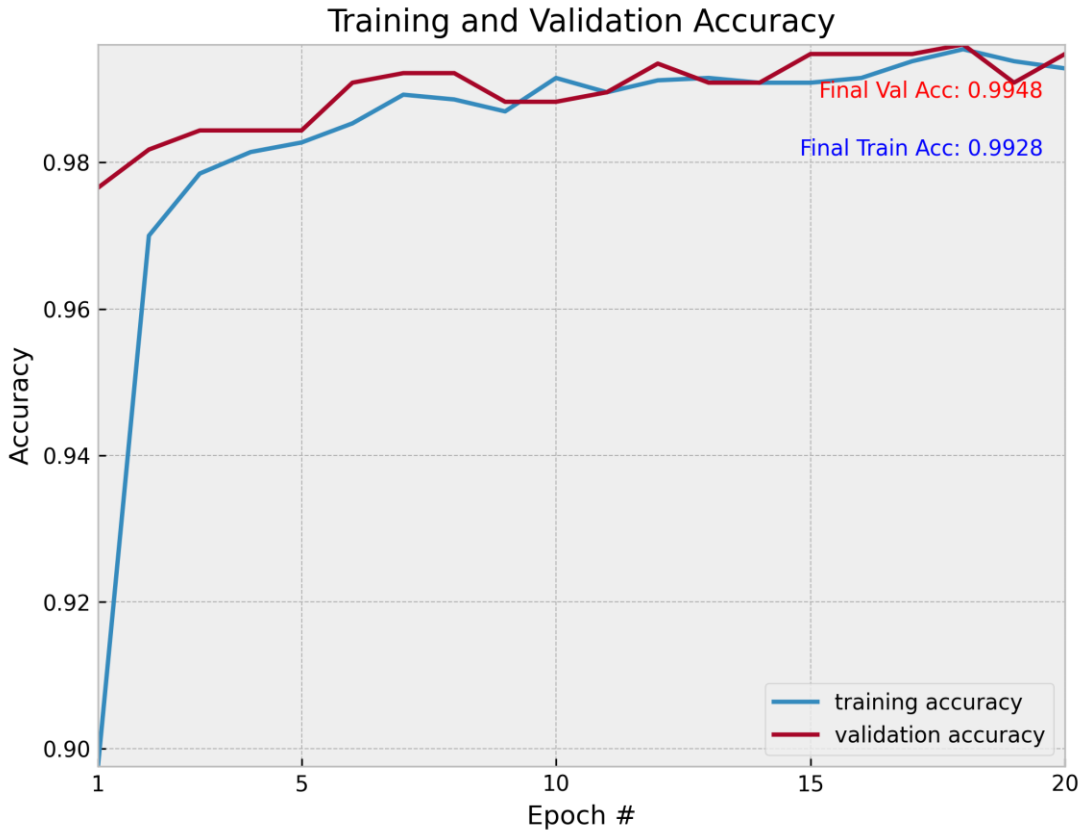
3.3.5 Δοκιμή 4: Σταθερά INIT_LR, EPOCHS και αλλαγή BS

3.3.5.1 Περίπτωση Α: Μειωμένο BS = 16

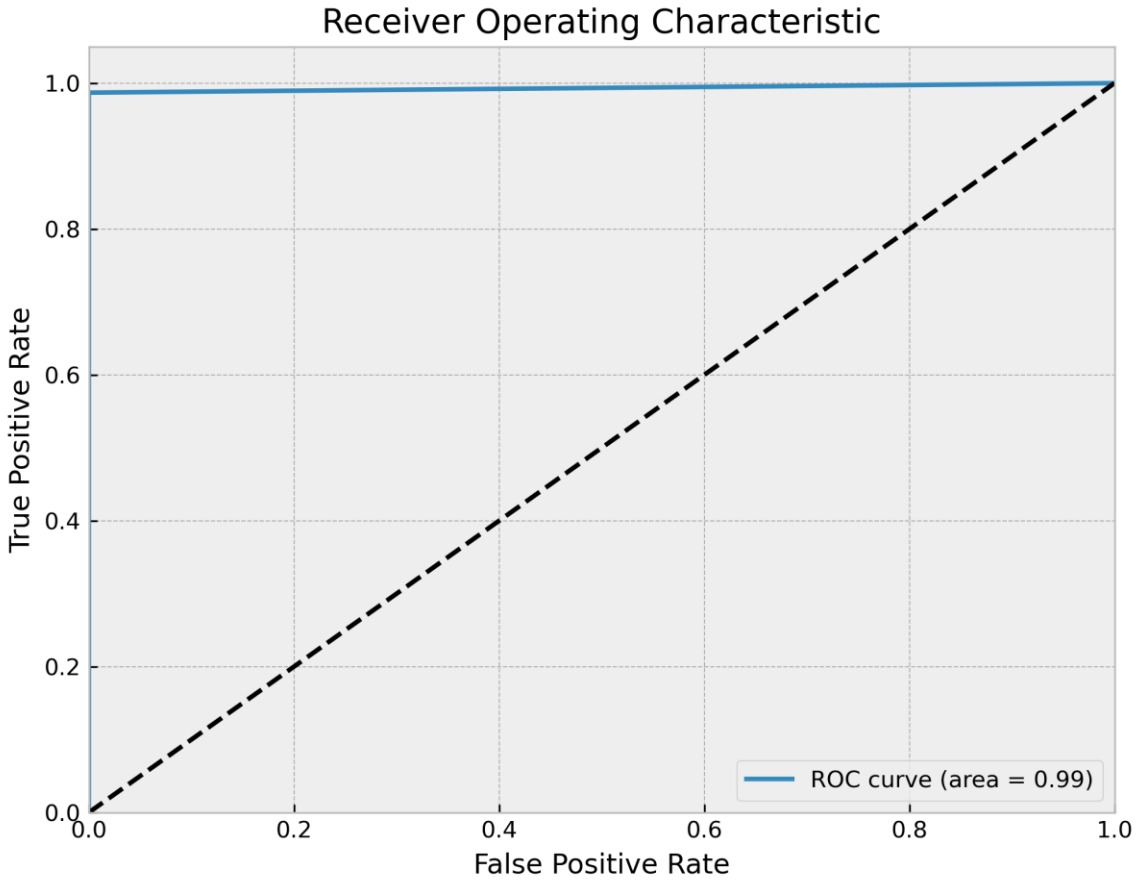
Το μοντέλο μειωμένου μεγέθους υποσυνόλου δεδομένων είναι ένα πάρα πολύ καλό μοντέλο και θα μπορούσε θεωρητικά να θεωρηθεί και αυτό βέλτιστο όπως και το μοντέλο της ενότητας 3.3.3.2. Πρακτικά όμως δεν επιλέχθηκε ως βέλτιστο διότι όπως και το μοντέλο της ενότητας 3.3.3.2 κάνει πιο χρονοβόρα την διαδικασία εκπαίδευσης και δεν προσφέρει μεγάλη διαφορά απόδοσης ή ακρίβειας στην πράξη. Αν και οι τιμές του είναι απειροελάχιστα καλύτερες από του ιδανικού μοντέλου, η απόκλιση μεταξύ των τιμών τους είναι τόσο μικρή που κατά την εκτέλεση του κώδικα ανίχνευσης μάσκας δεν θα έχουν διαφορά στα αποτελέσματα. Αν η απόκλιση των τιμών είχε αρκετή διαφορά τότε αυτό το μοντέλο θα ήταν πολύ πιθανό να θεωρηθεί ως βέλτιστο. Τα διαγράμματά του μοντέλου αυτού απεικονίζουν μια πολύ καλή απόδοση και σχεδόν μηδενικές απώλειες.



Εικόνα 3.54 Διάγραμμα απωλειών δοκιμής 4, περίπτωσης Α



Εικόνα 3.55 Διάγραμμα ορθοτήτων δοκιμής 4, περίπτωσης Α



Εικόνα 3.56 Διάγραμμα καμπύλης ROC δοκιμής 4, περίπτωσης Α

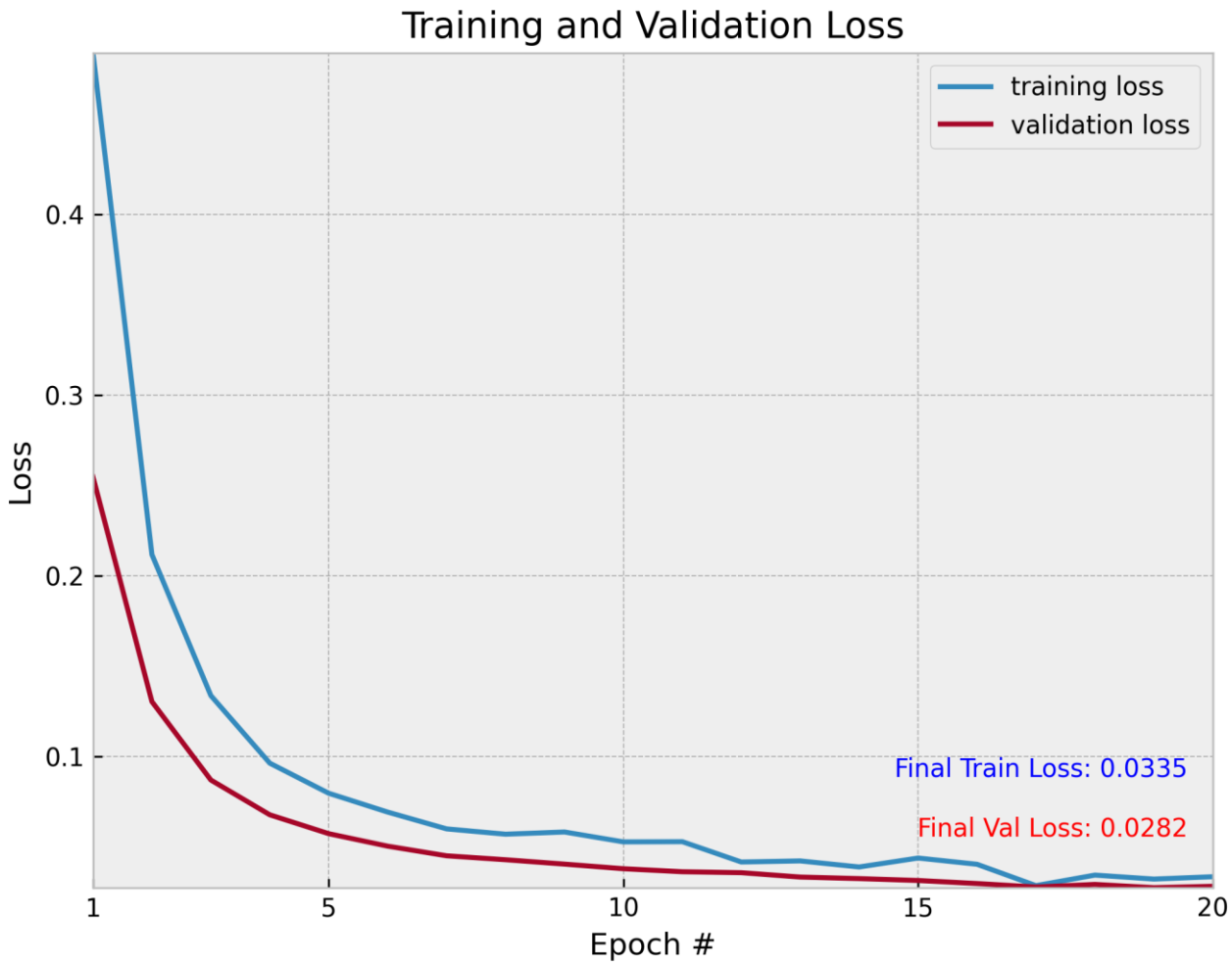
classification_report.txt

	precision	recall	f1-score	support
with_mask	0.99	0.99	0.99	383
without_mask	0.99	0.99	0.99	384
accuracy			0.99	767
macro avg	0.99	0.99	0.99	767
weighted avg	0.99	0.99	0.99	767

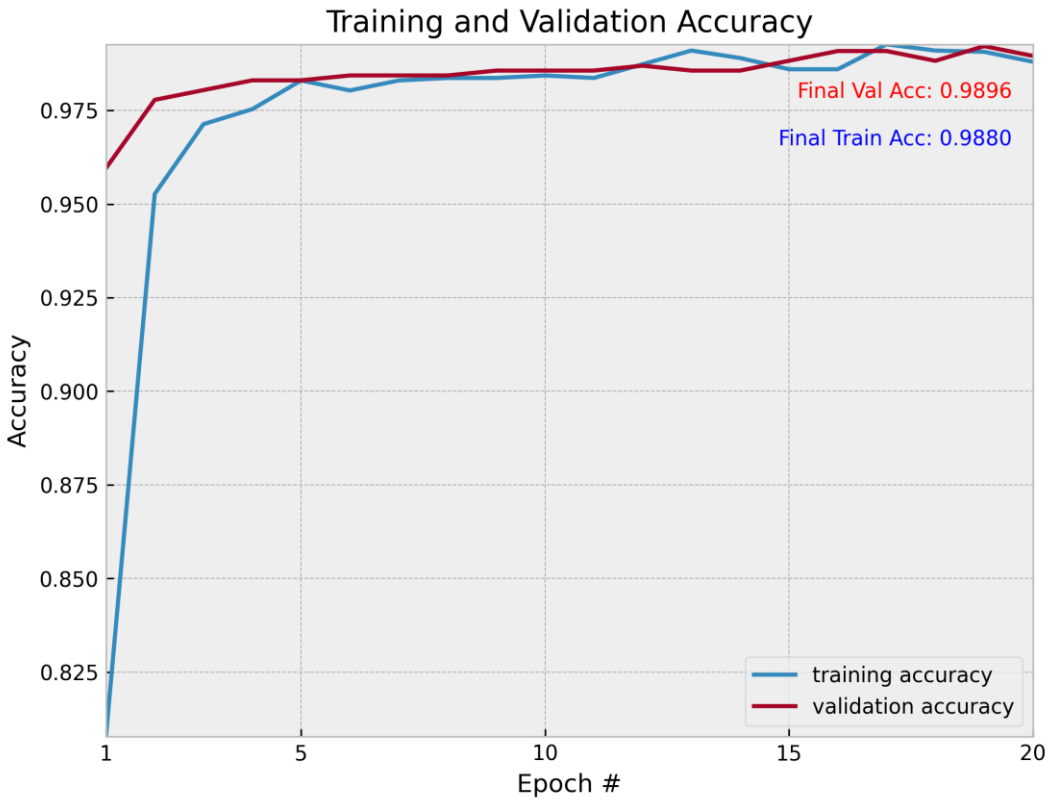
Εικόνα 3.57 Αναφορά μετρικών αξιολόγησης της δοκιμής 4, περίπτωσης Α

3.3.5.2 Περίπτωση Β: Αυξημένο BS = 64

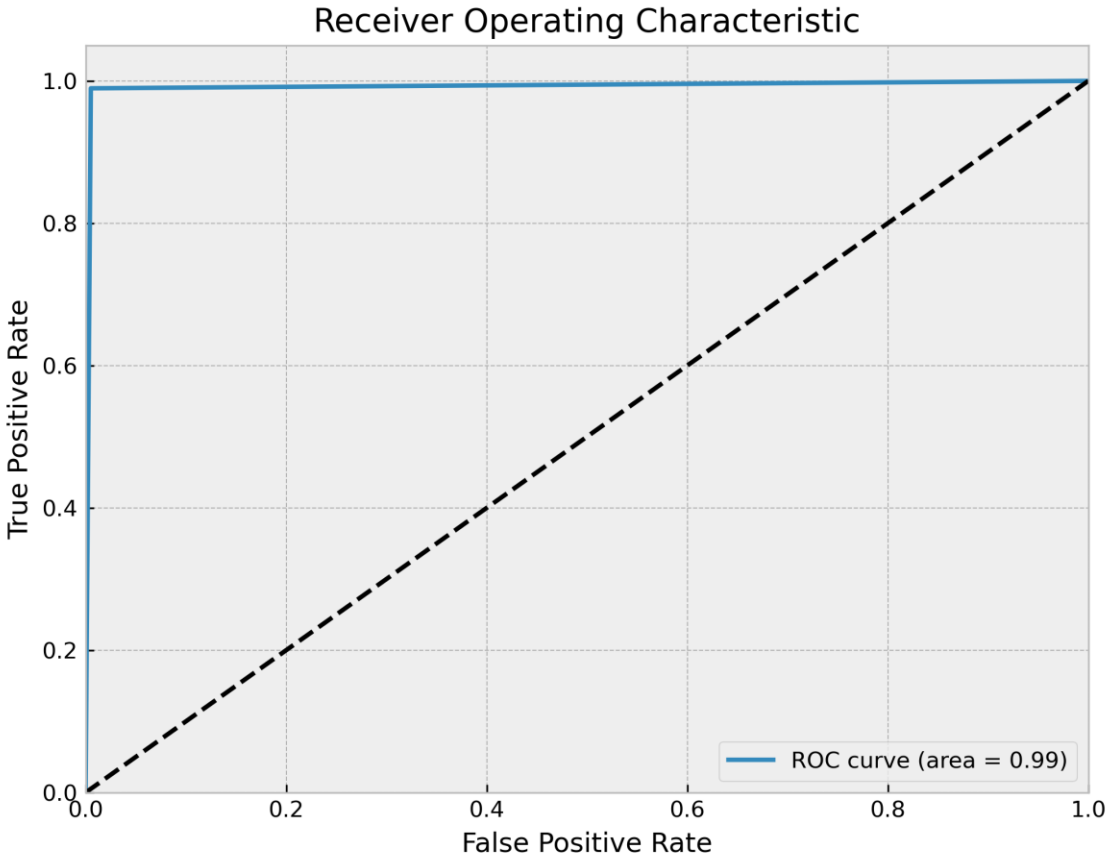
Στην περίπτωση του αυξημένου μεγέθους υποσυνόλου δεδομένων η εκπαίδευση του μοντέλου είναι αρκετά πιο γρήγορη από το βέλτιστο μοντέλο αλλά η απόδοσή του όχι καλύτερη. Αυτό διακρίνεται από τις υψηλότερες τιμές απώλειας και τις χαμηλότερες τιμές ορθότητας σε σχέση με το ιδανικό μοντέλο. Αν και η εκπαίδευση είναι αρκετά ομαλή χωρίς απότομες εναλλαγές απωλειών ή ορθοτήτων, το μοντέλο θα μπορούσε να εκπαιδευτεί ακόμα καλύτερα.



Εικόνα 3.58 Διάγραμμα απωλειών δοκιμής 4, περίπτωσης Β



Εικόνα 3.59 Διάγραμμα ορθοτήτων δοκιμής 4, περίπτωσης Β



Εικόνα 3.60 Διάγραμμα καμπύλης ROC δοκιμής 4, περίπτωσης Β

classification_report.txt

	precision	recall	f1-score	support
with_mask	0.98	0.99	0.99	383
without_mask	0.99	0.98	0.99	384
accuracy			0.99	767
macro avg	0.99	0.99	0.99	767
weighted avg	0.99	0.99	0.99	767

Εικόνα 3.61 Αναφορά μετρικών αξιολόγησης της δοκιμής 4, περίπτωσης Β

3.3.6 Πίνακας σύνοψης όλων των αποτελεσμάτων

Συνολικά όλα τα μοντέλα εκτός από αυτό της αύξησης του ρυθμού εκπαίδευσης, λειτουργούν πολύ καλά αφού έχουν όλα τις τιμές των μετρικών αξιολόγησης κοντά στη μονάδα. Θα μπορούσε να θεωρηθεί ως βέλτιστο όλων το μοντέλο της δοκιμής 2, περίπτωσης Β αλλά όπως προαναφέρθηκε σκοπός μας είναι να εκπαιδεύσουμε ένα ιδανικό μοντέλο το οποίο εκτός από υψηλή απόδοση και χαμηλές απώλειες να εκπαιδεύεται και σχετικά γρήγορα. Οπότε το μοντέλο που θεωρήσαμε βέλτιστο περιέχει λίγο από όλες αυτές τις προδιαγραφές που θέλουμε και μας ικανοποιεί πλήρως στο πρακτικό κομμάτι που είναι η πραγματική ανίχνευση μάσκας στα πρόσωπα ανθρώπων μέσω ζωντανής μετάδοσης βίντεο, όπως θα δούμε στο επόμενο κεφάλαιο. Παρακάτω απεικονίζεται ο πίνακας που περιέχει όλα τα μοντέλα που εκπαιδεύτηκαν για αυτήν την διπλωματική εργασία, μαζί με τις υπερπαραμέτρους τους, τις τελικές τιμές απωλειών, τις τελικές τιμές ορθότητας καθώς και την τιμή της περιοχής κάτω από τη καμπύλη ROC.

Πίνακας 3.1 Αποτελέσματα εκπαίδευσης των μοντέλων

ΜΟΝΤΕΛΑ	ΕΠΟΧΕΣ	ΡΥΘΜΟΣ ΕΚΠΑΙΔΕΥΣΗΣ	ΜΕΓΕΘΟΣ ΥΟΠΟΣΥΝΟΛΟΥ ΔΕΔΟΜΕΝΩΝ	ΤΕΛΙΚΗ ΤΙΜΗ ΑΠΩΛΕΙΩΝ ΕΚΠΑΙΔΕΥΣΗΣ	ΤΕΛΙΚΗ ΤΙΜΗ ΑΠΩΛΕΙΩΝ ΕΠΙΚΥΡΩΣΗΣ	ΤΕΛΙΚΗ ΤΙΜΗ ΟΡΘΟΤΗΤΑΣ ΕΚΠΑΙΔΕΥΣΗΣ	ΤΕΛΙΚΗ ΤΙΜΗ ΟΡΘΟΤΗΤΑΣ ΕΠΙΚΥΡΩΣΗΣ	ΤΙΜΗ ΠΕΡΙΟΧΗΣ ΚΑΤΩ ΑΠΟ ΤΗΝ ΚΑΜΠΥΛΗ ROC
ΒΕΛΤΙΣΤΟ ΜΟΝΤΕΛΟ	20	0,0001	32	0,0259	0,0237	0,9927	0,9909	0,99
ΔΟΚΙΜΗ 1: ΜΕΙΩΜΕΝΑ ΔΕΔΟΜΕΝΑ ΕΚΠΑΙΔΕΥΣΗΣ	20	0,0001	32	0,0494	0,0384	0,988	0,9896	0,99
ΔΟΚΙΜΗ 2, ΠΕΡΙΠΤΩΣΗ Α: ΜΕΙΩΣΗ EPOCHS	10	0,0001	32	0,0469	0,0321	0,9855	0,9896	0,99
ΔΟΚΙΜΗ 2 ΠΕΡΙΠΤΩΣΗ Β: ΑΥΞΗΣΗ EPOCHS	40	0,0001	32	0,0132	0,0149	0,9964	0,9974	1
ΔΟΚΙΜΗ 3 ΠΕΡΙΠΤΩΣΗ Α: ΜΕΙΩΣΗ INIT_LR	20	0,00005	32	0,0382	0,0331	0,9881	0,9896	0,99
ΔΟΚΙΜΗ 3 ΠΕΡΙΠΤΩΣΗ Β: ΑΥΞΗΣΗ INIT_LR	20	0,001	32	0,0173	0,0334	0,9941	0,9896	0,99
ΔΟΚΙΜΗ 4 ΠΕΡΙΠΤΩΣΗ Α: ΜΕΙΩΣΗ BS	20	0,0001	16	0,0215	0,0201	0,9928	0,9948	0,99
ΔΟΚΙΜΗ 4 ΠΕΡΙΠΤΩΣΗ Β: ΑΥΞΗΣΗ BS	20	0,0001	64	0,0335	0,0282	0,988	0,9896	0,99

4 ΚΕΦΑΛΑΙΟ 4^{ο} : Ανίχνευση μάσκας σε ανθρώπινα πρόσωπα με χρήση του μοντέλου που εκπαιδεύτηκε μέσω του Raspberry Pi 4

Σε αυτό το κεφάλαιο γίνεται αρχικά ανάλυση του κώδικα που δημιουργήθηκε για την ανίχνευση μάσκας σε ανθρώπινα πρόσωπα. Όπως και στην ανάλυση του κώδικα εκπαίδευσης μοντέλου έτσι και εδώ η περιγραφή θα είναι γενική με σκοπό να μπορεί ο οποιοσδήποτε να κατανοήσει την λειτουργία των εντολών και όχι τις ίδιες τις εντολές. Για τις ίδιες τις εντολές έχει γίνει λεπτομερής ανάλυση στο παράρτημα Β. Η εφαρμογή έχει φτιαχτεί για να εκτελείται μέσα σε ένα Raspberry Pi 4, το οποίο μέσω της κάμερας του θα εμφανίζει ζωντανή μετάδοση βίντεο και τα αποτελέσματα της ανίχνευσης μάσκας θα εμφανίζονται στο ίδιο το βίντεο αυτό.

Πιο συγκεκριμένα, κατά την εκτέλεση του κώδικα γίνεται χρήση του βέλτιστου μοντέλου ανίχνευσης μάσκας που δημιουργήθηκε με τον κώδικα που περιγράψαμε στο κεφάλαιο 3. Πρώτα με τη χρήση ενός προεκπαιδευμένου μοντέλου ανίχνευσης προσώπων, αναγνωρίζονται ανθρώπινα πρόσωπα και έπειτα γίνεται η ανίχνευση μάσκας με βάση αυτά τα πρόσωπα μέσω του βέλτιστου μοντέλου. Έτσι δημιουργούνται περιγράμματα γύρω από τα πρόσωπα των ανθρώπων με το κατάλληλο χρώμα που εξαρτάται από το εάν φορούν ή όχι μάσκα. Μαζί με το παραλληλόγραμμο αυτό πλαίσιο εμφανίζεται και ένα μήνυμα σχετικά με το αν φοράει ο συγκεκριμένος άνθρωπος μάσκα ή όχι καθώς και γίνεται εμφάνιση της πιθανότητας ως ποσοστό για την πρόβλεψη που έγινε. Όλες αυτές οι πληροφορίες εμφανίζονται τελικά μέσα στο γραφικό παράθυρο της ζωντανής μετάδοσης βίντεο, μέχρι ο χρήστης να το κλείσει.

Στο τέλος του κεφαλαίου αυτού, παρουσιάζονται κάποιες εικόνες με τα αποτελέσματα της εφαρμογής αυτής. Γίνεται παράθεση κάποιων στιγμιότυπων από τη ζωντανή μετάδοση βίντεο δείχνοντας το πως η εφαρμογή αυτή λειτουργεί και κατά πόσο είναι επιτυχής.

4.1 Ανάλυση του κώδικα

4.1.1 Εισαγωγή βιβλιοθηκών

```
import tensorflow as tf
from tensorflow.keras.applications.mobilenet_v2 import preprocess_input
from tensorflow.keras.preprocessing.image import img_to_array
from imutils.video import VideoStream
import numpy as np
import imutils
import cv2
```

Εικόνα 4.1 Εισαγωγή των απαραίτητων βιβλιοθηκών

Ο κώδικας της εφαρμογής ξεκινάει με την δήλωση όλων των βιβλιοθηκών ή ενότητες αυτών που θα χρειαστούν και περιέχουν τις απαραίτητες εντολές, συναρτήσεις, μεθόδους, κλάσεις κλπ. Κάποιες από αυτές τις βιβλιοθήκες είναι ίδιες με εκείνες που χρησιμοποιήθηκαν για τον κώδικα εκπαίδευσης μοντέλου, ενώ κάποιες άλλες χρησιμοποιούνται μόνο σε αυτόν τον κώδικα όπως για παράδειγμα η OpenCV που χρησιμοποιεί εντολές για τον χειρισμό εικόνας και βίντεο.

Οι βασικές βιβλιοθήκες που χρησιμοποιούνται σε αυτόν τον κώδικα είναι:

- **tensorflow**
- **imutils**
- **numpy**
- **cv2 (OpenCV)**

4.1.2 Η συνάρτηση για την ανίχνευση προσώπων και μάσκας

4.1.2.1 Εισαγωγή στη συνάρτηση και ανίχνευση προσώπων

```
9   def detect_and_predict_mask(frame, faceNet, maskNet):
10
11      (h, w) = frame.shape[:2]
12      blob = cv2.dnn.blobFromImage(frame, 1.0, (224, 224), (104.0, 177.0, 123.0))
13
14      faceNet.setInput(blob)
15      detections = faceNet.forward()
16      print(detections.shape)
17
18      faces = []
19      locs = []
20      preds = []
```

Εικόνα 4.2 Η αρχή της συνάρτησης ανίχνευσης μάσκας σε ανθρώπινα πρόσωπα

Μετά την εισαγωγή των απαραίτητων βιβλιοθηκών, ο κώδικας συνεχίζει με την δημιουργία της συνάρτησης που περιέχει όλο το σύνολο των εντολών οι οποίες συμβάλλουν στον έλεγχο ανίχνευσης ανθρώπινων προσώπων και έπειτα μάσκας.

Η συνάρτηση αυτή έχει ονομαστεί “detect_and_predict_mask” και εισάγει τρία αντικείμενα κατά την κλήση της από τον κύριο κώδικα. Το πρώτο είναι το “frame” και αφορά ένα στιγμιότυπο από τη ζωντανή μετάδοση βίντεο το οποίο θα εξεταστεί για να δούμε εάν περιέχει ανθρώπους που φοράνε ή δε φοράνε μάσκα. Το δεύτερο αντικείμενο είναι το μοντέλο για την εύρεση ανθρώπινων προσώπων μέσα στο “frame” και ονομάζεται “faceNet”. Το τρίτο ονομάζεται “maskNet” και είναι το βέλτιστο μοντέλο εντοπισμού μάσκας.

Αφού η συνάρτηση κληθεί από τον κύριο κώδικα ξεκινάει όπως φαίνεται στην εικόνα 4.2, εισάγοντας αρχικά τα τρία αντικείμενα που αναφέραμε και προχωρώντας στην δημιουργία δύο μεταβλητών, της “h” και “w” που περιέχουν πληροφορίες για το ύψος και το πλάτος του στιγμιότυπου “frame”.

Αμέσως μετά δημιουργείται το αντικείμενο “blob” από το στιγμιότυπο “frame”, το οποίο είναι ένα αντικείμενο με μορφή συμβατή για το μοντέλο που θα ανιχνεύσει τα πρόσωπα ανθρώπων. Οπότε για να ελέγξουμε το “frame” πρέπει πρώτα να το προσαρμόσουμε στην κατάλληλη μορφή και εκεί βοηθάει το “blob”. Ειδικότερα αυτό που κάνει το “blob” είναι ότι εισάγει το ίδιο το “frame” μέσα σε αυτό προσαρμόζοντάς το κατάλληλα για να το κάνει συμβατό. Οι προσαρμογές που γίνονται είναι η αλλαγή των διαστάσεων της εικόνας σε 224x224 pixels και η μείωση των μέσων τιμών “(104.0, 177.0, 123.0)” από τα pixels του “frame”. Οι μέσες τιμές αυτές αντιστοιχούν στα κανάλια της εικόνας RGB (RED GREEN BLUE) και σκοπός τους είναι η κανονικοποίηση (normalization) της εικόνας.

Μετά την δημιουργία του “blob”, αυτό εισάγεται στο μοντέλο ανίχνευσης προσώπων “faceNet” με την εντολή “.setInput” και αμέσως γίνεται έλεγχος με την εντολή “.forward()” για πιθανά πρόσωπα στο “frame” Με το πέρας του ελέγχου, εάν υπάρχουν προβλέψεις, τότε αποθηκεύονται στο διάνυσμα “detections”. Επίσης δημιουργούνται οι κενές λίστες “faces”, “locs” και “preds” που θα χρησιμοποιηθούν αργότερα στον κώδικα για τις διάφορες προβλέψεις.

4.1.2.2 Δημιουργία εικόνας κάθε ανθρώπινου προσώπου και εύρεση συντεταγμένων

```
for i in range(0, detections.shape[2]):
    confidence = detections[0, 0, i, 2]

    if confidence > 0.5:
        box = detections[0, 0, i, 3:7] * np.array([w, h, w, h])
        (startX, startY, endX, endY) = box.astype("int")

        (startX, startY) = (max(0, startX), max(0, startY))
        (endX, endY) = (min(w - 1, endX), min(h - 1, endY))

        face = frame[startY:endY, startX:endX]
        face = cv2.cvtColor(face, cv2.COLOR_BGR2RGB)
        face = cv2.resize(face, (224, 224))
        face = img_to_array(face)
        face = preprocess_input(face)

        faces.append(face)
        locs.append((startX, startY, endX, endY))
```

Εικόνα 4.3 Εξαγωγή εικόνας ανθρώπινου προσώπου και αποθήκευση των συντεταγμένων του πάνω στο “frame”

Ο κώδικας συνεχίζει με τη δημιουργία ενός βρόγχου επανάληψης “for” ο οποίος θα ελέγχει με τη σειρά κάθε μία από τις προβλέψεις που περιέχει το “predictions“ σχετικά με τα ανθρώπινα πρόσωπα. Έστω ότι το “for” εισάγει την πρώτη πρόβλεψη του διανύσματος “detections”, από αυτήν την πρόβλεψη αποθηκεύεται στην μεταβλητή “confidence” η πιθανότητα που θεωρεί το μοντέλο “faceNet” ότι αυτή η πρόβλεψη όντως περιέχει κάποιο ανθρώπινο πρόσωπο. Έτσι γίνεται έλεγχος με μια συνθήκη “if” και εάν η πιθανότητα είναι μεγαλύτερη από το 0,5, δηλαδή το 50%, τότε ξεκινάει η διαδικασία δημιουργίας μιας νέας εικόνας που θα περιέχει μόνο το πρόσωπο του ανθρώπου που εντοπίστηκε.

Αρχικά πρέπει να αναφερθεί ότι το “detections” δεν είναι ένα μονοδιάστατο διάνυσμα, αλλά πολυδιάστατο και περιέχει διάφορες πληροφορίες για την κάθε πρόβλεψη, όπως τις συντεταγμένες της περιοχής που βρίσκεται το ανθρώπινο πρόσωπο στο “frame”. Αυτές λοιπόν τις συντεταγμένες που βρίσκονται αποθηκευμένες στην τέταρτη διάσταση του διανύσματος και συγκεκριμένα στις θέσεις 3,4,5 και 6 τις εναποθέτουμε όλες σε ένα διάνυσμα, το “box”. Να επισημανθεί πως αυτές οι συντεταγμένες προσαρμόζονται πρώτα στις διαστάσεις του “frame” πολλαπλασιάζοντάς αυτές με ένα διάνυσμα που περιέχει τις πραγματικές διαστάσεις του. Από το “box”. Αυτές τις συντεταγμένες τις τοποθετούμε τελικά σε 4 ξεχωριστές μεταβλητές, αφού τις μετατρέψουμε σε ακέραιους αριθμούς. Οι μεταβλητές ονομάζονται “startX”, “startY”, “endX”, “endY” και οι δύο πρώτες αντιπροσωπεύουν τις συντεταγμένες του σημείου της πάνω αριστερά γωνίας του νοητού παραλληλόγραμμου που περιέχει το ανθρώπινο πρόσωπο ενώ οι δύο τελευταίες την κάτω δεξιά.

Αυτές οι συντεταγμένες θα χρειαστούν διότι αργότερα θα δημιουργήσουμε χρωματιστά πλαίσια γύρω από τα πρόσωπα των ανθρώπων που εντοπίστηκε ή και όχι η μάσκα έτσι ώστε να δείξουμε τα αποτελέσματα της ανίχνευσης τόσο των προσώπων όσο και της μάσκας στην οθόνη. Για να λειτουργήσουν όμως σωστά αυτά τα πλαίσια πρέπει να προσαρμοστούν σε αυτό το σημείο οι συντεταγμένες, έτσι ώστε να μην βρεθεί αργότερα κάποιο πλαίσιο «έξω» από το παράθυρο του βίντεο. Έτσι γίνεται η προσαρμογή τους σε τιμές εντός των ορίων των διαστάσεων του “frame” και έπειτα δημιουργείται το διάνυσμα “face” που πλέον αποτελεί την μεμονωμένη εικόνα του ανθρώπινου προσώπου. Η δημιουργία του “face” πραγματοποιείται από την εξαγωγή των pixels που περιέχουν το ανθρώπινο πρόσωπο μέσα στην εικόνα του “frame”, με τη βοήθεια των συντεταγμένων που αναφέραμε.

Αφού φτιάχτηκε το “face” με επιτυχία, γίνεται προσαρμογή των καναλιών χρώματός του από μορφή BGR (BLUE GREEN RED) σε RGB (RED GREEN BLUE), αλλαγή των διαστάσεών του σε 224x224 pixels και η πιο εξειδικευμένη προσαρμογή του από την εντολή “preprocess_input” της βιβλιοθήκης “tensorflow” και συγκεκριμένα της “keras”. Αυτή η εντολή προετοιμάζει το “face” σε συμβατή μορφή για το μοντέλο ανίχνευσης μάσκας αργότερα.

Ο βρόγχος επανάληψης “for” τερματίζει με την αποθήκευση του διανύσματος “face” στη λίστα “faces” η οποία θα περιέχει όλες τις εικόνες των προσώπων που εντοπίστηκαν μετά την πλήρη ολοκλήρωση του ελέγχου προβλέψεων σχετικά με τα πρόσωπα. Επίσης αποθηκεύονται για κάθε εικόνα προσώπου αντίστοιχα και οι συντεταγμένες που υπολογίστηκαν για αυτές στη λίστα “locs”.

4.1.2.3 Ανίχνευση μάσκας

```
41    if len(faces) > 0:
42
43        for face in faces:
44            face = np.expand_dims(face, axis=0)
45            maskNet.set_tensor(input_details[0]['index'], face)
46
47            maskNet.invoke()
48
49            pred = maskNet.get_tensor(output_details[0]['index'])
50            preds.append(pred)
51
52    return (locs, preds)
```

Εικόνα 4.4 Κώδικας για την πρόβλεψη ύπαρξης ή μη μάσκας στα ανθρώπινα πρόσωπα

Σε αυτό το σημείο ξεκινά το πιο σημαντικό κομμάτι ίσως του κώδικα, που είναι η πρόβλεψη της πιθανότητας να υπάρχει ή όχι μάσκα πάνω στα πρόσωπα που προβλέφθηκαν νωρίτερα.

Στην αρχή γίνεται ένας έλεγχος με τη συνθήκη “if” για την ύπαρξη ή μη ανθρώπινων προσώπων μέσα στο “frame”. Αυτό συμβαίνει ελέγχοντας εάν η λίστα “faces”, που θα έπρεπε να περιέχει τις εικόνες με τα ανθρώπινα πρόσωπα, είναι κενή. Εάν είναι κενή τότε δεν γίνεται κάποιος έλεγχος για μάσκα οπότε η συνάρτηση “detect_and_predict_mask” τερματίζει, στην περίπτωση όμως που δεν είναι κενή τότε ο κώδικας προχωράει και μπαίνει σε ένα βρόγχο επανάληψης “for”.

Ο βρόγχος “for” θα ελέγξει με τη σειρά όλα τα στοιχεία που περιλαμβάνει η λίστα “faces” , δηλαδή όλα τα ανθρώπινα πρόσωπα που εντοπίστηκαν. Έστω ότι γίνεται έλεγχος για ένα από τα στοιχεία του “faces”, αρχικά αυτό το μεμονωμένο στοιχείο τοποθετείται σε ένα νέο διάνυσμα με ονομασία “face”. Το “face” επειδή σε αυτή την φάση δεν είναι συμβατό με το μοντέλο ανίχνευσης μάσκας λόγω των τριών διαστάσεών του, μετατρέπεται σε διάνυσμα τεσσάρων διαστάσεων προσθέτοντας μέσα σε αυτό μια νέα κενή διάσταση. Αυτό γίνεται διότι το μοντέλο ανίχνευσης μάσκας είναι φτιαγμένο να δέχεται στην είσοδό του διανύσματα τεσσάρων διαστάσεων και οτιδήποτε άλλο θα προκαλούμε πρόβλημα και τερματισμό του κώδικα.

Αφού το “face” πλέον τηρεί όλες τις προδιαγραφές, εισάγεται στον τανυστή εισόδου του μοντέλου “maskNet” με την εντολή ".set_tensor()" έτσι ώστε να είναι έτοιμο για έλεγχο. Ο τανυστής εισόδου προετοιμάζεται κατάλληλα λόγω του ορίσματος “input_details[0]['index']” που περιέχει όλες τις πληροφορίες σχετικά με αυτόν και την ενεργοποίησή του. Έτσι ακολουθεί η εντολή “.invoke()” που εκτελείται πάνω στο μοντέλο ανίχνευσης μάσκας και ξεκινάει ο έλεγχος της εικόνας του “face”. Το αποτέλεσμα της ταξινόμησης της εικόνας είναι δύο προβλέψεις, η μία έχει να κάνει με την πιθανότητα της κλάσης μη ύπαρξης μάσκας και η άλλη της κλάσης ύπαρξης μάσκας. Αυτές οι προβλέψεις εξάγονται από τον τανυστή εξόδου με την εκτέλεση της εντολής “.get_tensor()” πάνω στο “maskNet” και αποθηκεύονται στο διάνυσμα “pred”. Αυτό με τη σειρά του αποθηκεύεται στην λίστα “preds” που θα περιέχει όλες τις αντίστοιχες προβλέψεις μάσκας για όλη τη λίστα “faces”.

Η συνάρτηση ανίχνευσης μάσκας σε ανθρώπινα πρόσωπα τελειώνει σε αυτό το σημείο και επιστρέφονται στον κύριο κώδικα οι λίστες “locs” και “preds”, σε μορφή πλειάδας, που περιέχουν τις τοποθεσίες των ανθρώπινων προσώπων και τις προβλέψεις μάσκας.

4.1.3 Έναρξη κώδικα: Φόρτωση μοντέλων και ενεργοποίηση κάμερας

```
54  prototxtPath = r"face_detector_model/deploy.prototxt"
55  weightsPath = r"face_detector_model/res10_300x300_ssd_iter_140000.caffemodel"
56  faceNet = cv2.dnn.readNet(prototxtPath, weightsPath)
57
58  mask_string = r"mask_detection_model_optim.tflite"
59  maskNet = tf.lite.Interpreter(model_path=mask_string)
60  maskNet.allocate_tensors()
61  input_details = maskNet.get_input_details()
62  output_details = maskNet.get_output_details()
63
64  print("[ΕΝΗΜΕΡΩΣΗ] Το βίντεο ξεκίνησε...")
65  vs = VideoStream(src=0).start()
66  fl = 0
```

Εικόνα 4.5 Φόρτωση των μοντέλων ανίχνευσης προσώπων και μάσκας, ρύθμιση του μοντέλου μάσκας και ενεργοποίηση της κάμερας

Όταν εκτελείται η εφαρμογή ολόκληρου του κώδικα από το rpi4, πρώτα εισάγονται οι βιβλιοθήκες και η πρώτη εντολή μετά από αυτές που θα εφαρμοστεί είναι η γραμμή 54 της εικόνας 4.5. Η συνάρτηση που είδαμε στην ενότητα 4.1.2, παρόλο που δηλώνεται στον κώδικα μετά τις βιβλιοθήκες και πριν από τις γραμμές εντολών της εικόνας 4.5, δεν εκτελείται πρώτα αλλά δηλώνεται απλά στον κώδικα για να την αναγνωρίζει αργότερα από όποιο σημείο του κώδικα αν αυτή καλείται.

Έτσι πρακτικά η εφαρμογή ξεκινά από τη γραμμή 54, όπου δημιουργούνται δύο αλφαριθμητικά με τις τοποθεσίες των δύο αρχείων που αποτελούν το μοντέλο εντοπισμού ανθρώπινων προσώπων. Αυτά τα αλφαριθμητικά δηλώνονται ως είσοδοι στην συνάρτηση “cv2.dnn.readNet()” της βιβλιοθήκης OpenCV, η οποία εντοπίζει τα αρχεία με τις πληροφορίες σχετικά με την αρχιτεκτονική του μοντέλου ή τις παραμέτρους του και τις συνδυάζει για να δημιουργήσει το μοντέλο. Έτσι το μοντέλο φορτώνεται στο αντικείμενο ”faceNet” και αυτό είναι έτοιμο για χρήση όποτε το καλέσει ο κώδικας.

Ακολουθεί η δημιουργία αλφαριθμητικού με την τοποθεσία του βέλτιστου μοντέλου ανίχνευσης μάσκας της μορφής ‘.tflite” και τη φόρτωσή του μοντέλου αυτού μέσα στο αντικείμενο “maskNet” με τη βοήθεια της εντολής “tf.lite.Interpreter()” από τη βιβλιοθήκη του “Tensorflow lite”. Αμέσως μετά εκτελείται η εντολή της γραμμής 60 από την εικόνα 4.5, η οποία είναι πολύ σημαντική και φορτώνει στη μνήμη του rpi4 τους τανυστές εισόδου και εξόδου του μοντέλου. Αφού αυτοί φορτωθούν, παρέχονται πληροφορίες στις μεταβλητές “input_details” και “output_details” σχετικά με την είσοδο και έξοδο του νευρωνικού δικτύου “maskNet”. Αυτές οι πληροφορίες χρησιμοποιούνται αργότερα μέσα στη συνάρτηση “detect_and_predict_mask” που είδαμε συγκεκριμένα στην ενότητα 4.1.2.3.

Μετά από όλα αυτά ενεργοποιείται η ζωντανή μετάδοση βίντεο, χωρίς να εμφανίζεται αυτό ακόμα, μέσω της κάμερας του rpi4 η οποία έχει ως αναγνωριστικό τον αριθμό 0 και για αυτό επιλέγεται ως “src=0” μέσα στην εντολή “VideoStream()”. Εάν επιλεγόταν διαφορετική τιμή για το “src” τότε η ζωντανή μετάδοση θα ξεκινούσε από κάποια άλλη κάμερα συνδεδεμένη πάνω στο rpi4, εάν αυτή υπήρχε. Επίσης γίνεται αρχικοποίηση της μεταβλητής “fl” με τον αριθμό μηδέν που θα χρειαστεί στην επόμενη ενότητα.

4.1.4 Κυρίος κώδικας: Ζωντανή μετάδοση βίντεο και προβολή αποτελεσμάτων

4.1.4.1 Εισαγωγή στον κύριο κώδικα

```
68  while True:
69
70      if cv2.getWindowProperty("Mask Detection", cv2.WND_PROP_VISIBLE) < 1 & fl == 1:
71          break
72      else:
73          fl = 1
74
75      frame = vs.read()
76      frame = imutils.resize(frame, width=400)
```

Εικόνα 4.6 Η αρχή του κύριου κώδικα της εφαρμογής ανίχνευσης μάσκας σε ανθρώπινα πρόσωπα

Ο κύριος κώδικας ξεκινά με τη δημιουργία ενός ατέρμονου βρόγχου while, ο οποίος θα επαναλαμβάνεται μόνιμα μέχρι να αποφασίσει ο ίδιος ο χρήστης τον τερματισμό της εφαρμογής με τρόπους που θα αναλύσουμε στη συνέχεια.

Η πρώτη εντολή που εκτελείται σε κάθε επανάληψη του βρόγχου while είναι μια συνθήκη ελέγχου “if” που ελέγχει αν το παράθυρο, με ονομασία “Mask Detection” της προβολής των στιγμιότυπων από το βίντεο, είναι ακόμα ανοιχτό. Εδώ πρέπει να αναφερθεί πως το παράθυρο αυτό θα δημιουργηθεί αργότερα στον κώδικα και θα είναι το παράθυρο που θα προβάλει τα αποτελέσματα ανίχνευσης μάσκας. Σίγουρα θα αναρωτηθεί κανείς γιατί ελέγχουμε κάτι στην αρχή του κώδικα και όχι στο τέλος του αφού δεν υπάρχει ακόμα. Η απάντηση σε αυτό το ερώτημα είναι πως ο έλεγχος πρέπει να γίνεται στην αρχή του κώδικα διότι μετά από δοκιμές που έγιναν η εκτέλεση της ενέργειας που συμβαίνει μέσα στη συνθήκη “if” δεν λειτουργούσε όταν η συνθήκη αυτή τοποθετούταν στο τέλος του κώδικα. Αυτό έχει να κάνει με την ταχύτητα εκτέλεσης του κώδικα η οποία είναι αρκετά γρήγορη ώστε να ελέγξει αυτό που θέλουμε στο τέλος του βρόγχου επανάληψης “while”. Επίσης η τοποθέτηση έγινε στην αρχή του κώδικα για θέματα ασφαλείας και την σωστή εκτέλεση του κώδικα χωρίς σφάλματα.

Έτσι λοιπόν η συνθήκη αυτή ελέγχει αν ο χρήστης έκλεισε το παράθυρο πατώντας το σύμβολο “x” στην πάνω δεξιά γωνία του παραθύρου της ζωντανής μετάδοσης βίντεο. Εάν το παράθυρο είναι κλειστό τότε επιστρέφεται η τιμή 0 ενώ αν το παράθυρο είναι ανοικτό η τιμή 1. Η συνθήκη “if” για να εκτελέσει την εμφωλευμένη της εντολή “break” πρέπει να δεχτεί την τιμή “true” η οποία προκύπτει εάν και το παράθυρο είναι κλειστό αλλά και η μεταβλητή “fl” είναι ίση με το 1. Εφόσον κατά την πρώτη εκτέλεση του βρόγχου επανάληψης “while” το “fl” είναι ίσο με 0, όπως δηλαδή το είχαμε ορίσει στην ενότητα 4.1.3, θα επιστραφεί η τιμή “false” και η εμφωλευμένη εντολή της “if“ δεν θα εκτελεστεί. Παρόλα αυτά θα εκτελεστεί η εμφωλευμένη εντολή της συνθήκης “else” η οποία αλλάζει το περιεχόμενο του “fl” σε 1. Αυτό σημαίνει πως από την επόμενη φορά που θα ξανά εκτελεστεί ο βρόγχος “while”, η συνθήκη “if” δεν θα εξαρτάται από τη μεταβλητή “fl” αφού αυτή πλέον θα ικανοποιεί την συνθήκη. Από την άλλη όμως θα εξαρτάται μόνο από το γεγονός εάν ο χρήστης έκλεισε ή όχι το παράθυρο. Εάν ο χρήστης έκλεισε το παράθυρο από τη στιγμή που αυτό δημιουργήθηκε, τότε δίνεται η τιμή 0 σαν απάντηση και εκτελείται η εμφωλευμένη εντολή της “if“ που είναι η “break”. Αυτή η εντολή πραγματοποιεί την άμεση έξοδο από την “while” και τον τερματισμό της εφαρμογής.

Μετά από αυτόν τον έλεγχο γίνεται αποθήκευση ενός στιγμιότυπου από το βίντεο μέσα στο διάνυσμα “frame” και αλλαγή των διαστάσεων αυτού σε 400x400 pixels οι οποίες θα είναι ίδιες με τις διαστάσεις του παραθύρου που θα προβάλλονται τα αποτελέσματα.

4.1.4.2 Κλήση της συνάρτησης ανίχνευσης και προετοιμασία αποτελεσμάτων

```
78      (locs, preds) = detect_and_predict_mask(frame, faceNet, maskNet)
79
80      for (box, pred) in zip(locs, preds):
81          (startX, startY, endX, endY) = box
82          (mask, withoutMask) = pred[0]
83
84          label = "Mask" if mask > withoutMask else "No Mask"
85          color = (0, 255, 0) if label == "Mask" else (0, 0, 255)
86          label = "{}: {:.2f}%".format(label, max(mask, withoutMask) * 100)
87
88          cv2.putText(frame, label, (startX, startY - 10),
89                      cv2.FONT_HERSHEY_SIMPLEX, 0.45, color, 2)
90          cv2.rectangle(frame, (startX, startY), (endX, endY), color, 2)
```

Εικόνα 4.7 Κλήση της συνάρτησης ανίχνευσης μάσκας σε ανθρώπινα πρόσωπα και δημιουργία πλαισίων γύρω από τις προβλέψεις

Το κομμάτι του κώδικα της εικόνας 4.7 ξεκινάει με την κλήση της συνάρτησης που αναλύθηκε στην ενότητα 4.1.2 και την επιστροφή των αποτελεσμάτων της σε μια πλειάδα με τις ονομασίες “locs” και “preds”. Αυτά τα αποτελέσματα επειδή μπορεί να περιέχουν παραπάνω από μια πρόβλεψη, ελέγχονται μέσα σε μια συνθήκη “for” μεμονωμένα. Δηλαδή δημιουργούνται κάθε φορά μέσα στο βρόγχο επανάληψης οι μεταβλητές “box” και “pred” οι οποίες περιέχουν τις τιμές σχετικά με τις συντεταγμένες και τις πιθανότητες μιας συγκεκριμένης πρόβλεψης. Οι συντεταγμένες αυτές που πλέον βρίσκονται στο “box”, τοποθετούνται σε τέσσερις επιμέρους μεταβλητές, τις “startX”, “startY”, “endX”, “endY” και οι πιθανότητες για κάθε κλάση της πρόβλεψης τοποθετούνται από το διάνυσμα “pred[0]” στις μεταβλητές “mask”, “withoutMask”.

Ο κώδικας συνεχίζει δημιουργώντας το αλφαριθμητικό “label” που θα περιέχει την απάντηση για το αν φοράει ο άνθρωπος της συγκεκριμένης πρόβλεψης μάσκα ή όχι. Αυτό αποφασίζεται συγκρίνοντας τις πιθανότητες κάθε κλάσης και όποια είναι μεγαλύτερη υπερτερεί διότι αυτή είναι πιο σίγουρη για το αποτέλεσμα. Το “label” αυτό πέρα από το κείμενο που θα περιέχει, δηλαδή “Mask” ή “No mask” ανάλογα με τις πιθανότητες, θα συνοδεύεται και από την πιθανότητα της πρόβλεψης αυτής σε μορφή ποσοστού.

Επιπλέον δημιουργείται η μεταβλητή “color” με περιεχόμενο τις τιμές των καναλιών του χρώματος ,σε μορφή BGR, που θα έχει το πλαίσιο το οποίο θα τοποθετηθεί γύρω από τα πρόσωπα των ανθρώπων που εντοπίστηκαν. Το “color” θα έχει τις τιμές του πράσινου χρώματος αν ο άνθρωπος φοράει μάσκα ενώ κόκκινο αν δεν φοράει.

Έτσι οι πληροφορίες του “label”, του “color” και των συντεταγμένων προσώπου δίνονται μαζί με το στιγμιότυπο “frame” στις εντολές των γραμμών 88-90 της εικόνας 4.7, οι οποίες επεξεργάζονται το στιγμιότυπο και εμφανίζουν τις πληροφορίες αυτές πάνω του. Δηλαδή το κείμενο του “label”

εμφανίζεται πάνω από το κεφάλι του ανθρώπου που εντοπίστηκε ενώ οι συντεταγμένες μαζί με το "color" συμβάλλουν στο σχεδιασμό του χρωματισμένου πλαισίου γύρω από το πρόσωπό του.

4.1.4.3 Εμφάνιση αποτελεσμάτων και τερματισμός κώδικα

```
92      cv2.imshow("Mask Detection", frame)
93      key = cv2.waitKey(1) & 0xFF
94
95      if key == 27:
96          break
97
98  vs.stream.release()
99  cv2.destroyAllWindows()
```

Εικόνα 4.8 Το τελικό κομμάτι εντολών του κώδικα ανίχνευσης μάσκας σε ανθρώπινα πρόσωπα

Ο βρόγχος επανάληψης "while" φτάνει στο τέλος της μίας επανάληψής του όπου όπως φαίνεται στην εικόνα 4.8, εμφανίζεται επιτέλους στην οθόνη ένα παράθυρο που περιέχει το στιγμιότυπο "frame" επεξεργασμένο καταλλήλως όπως περιγράφηκε στην προηγούμενη ενότητα 4.1.4.2. Στο παράθυρο δίνεται η ονομασία "Mask Detection" που θα εμφανίζεται ψηλά στην πάνω μεριά του.

Πριν τερματιστεί η συγκεκριμένη επανάληψη της "while" και αυτή ξανά ξεκινήσει από την αρχή, ελέγχεται με τη συνθήκη "if" εάν ο χρήστης πάτησε το πλήκτρο "ESC" κάποια στιγμή κατά την εκτέλεση του κώδικα. Εάν το πάτησε τότε ο κώδικας βγαίνει κατευθείαν από το βρόγχο επανάληψης "while" και συνεχίζει από την πρώτη εντολή που θα συναντήσει έξω από αυτόν, δηλαδή στη συγκεκριμένη περίπτωση από τη γραμμή 98 της εικόνας 4.8. Αν ο χρήστης δεν πατήσει "ESC" τότε η while τερματίζει και ο κώδικας που περιέχει ξεκινάει να εκτελείται από την αρχή εμφανίζοντας πιθανά το επόμενο στιγμιότυπο στο παράθυρο. Τα στιγμιότυπα αυτά που εμφανίζονται στο παράθυρο το ένα μετά το άλλο, προβάλλονται τόσο γρήγορα που σε εμάς φαίνεται ως βίντεο το αποτέλεσμα.

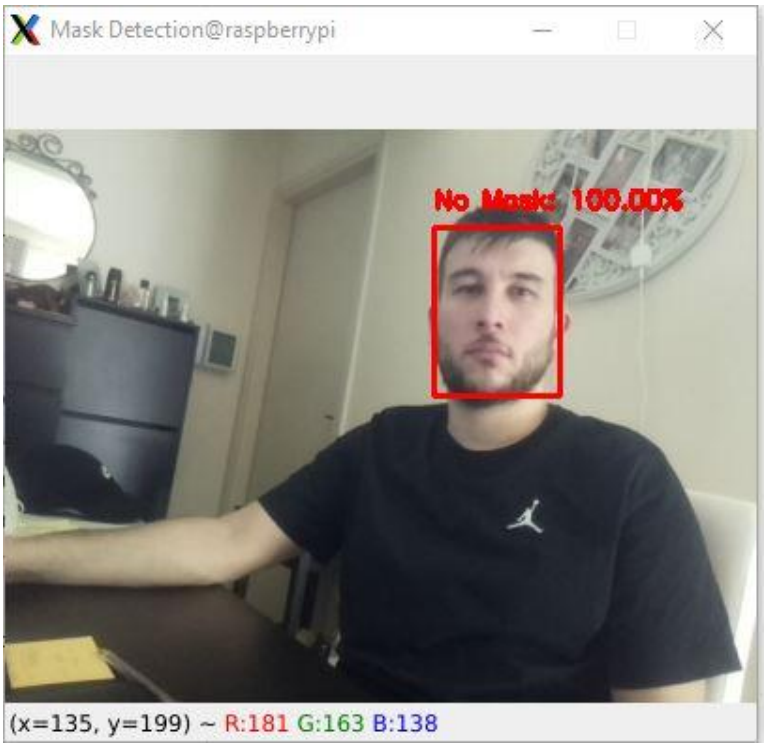
Αν λοιπόν ο χρήστης επιλέξει να κλείσει το παράθυρο του βίντεο, που αναφέραμε και στην ενότητα 4.1.4.1, ή πατήσει το πλήκτρο "ESC" ο κώδικας θα συνεχίσει έξω από τη "while" και θα εκτελέσει τις γραμμές 98,99 της εικόνας 4.8 όπου απενεργοποιείται η κάμερα και κλείνουν τα ανοιχτά παράθυρα που δημιουργήθηκαν από τον κώδικα.

Σε αυτό το σημείο ο κώδικας ολοκληρώθηκε και η εφαρμογή τερματίστηκε.

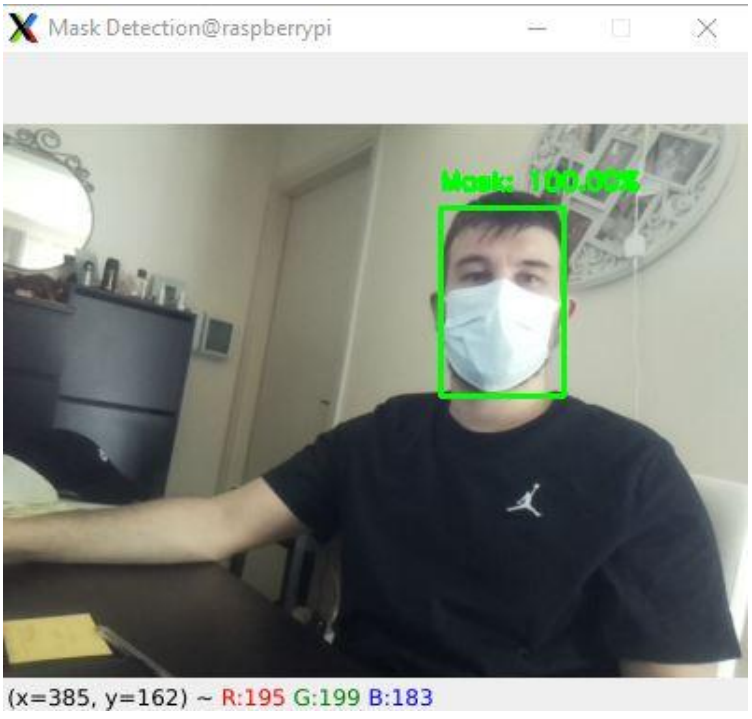
4.2 Αποτελέσματα

Σε αυτήν την ενότητα παρουσιάζονται κάποια στιγμιότυπα από τη ζωντανή μετάδοση βίντεο κατά την εκτέλεση της εφαρμογής ανίχνευσης μάσκας σε ανθρώπινα πρόσωπα. Οι εικόνες έχουν τραβηχτεί σε διάφορες καταστάσεις για να υπάρξει ποικιλία σε σχέση με τα αποτελέσματα. Οι περισσότερες ενότητες εικόνων περιέχουν ένα παράδειγμα με μάσκα και ένα χωρίς.

4.2.1 Στιγμιότυπα μπροστινού προσώπου

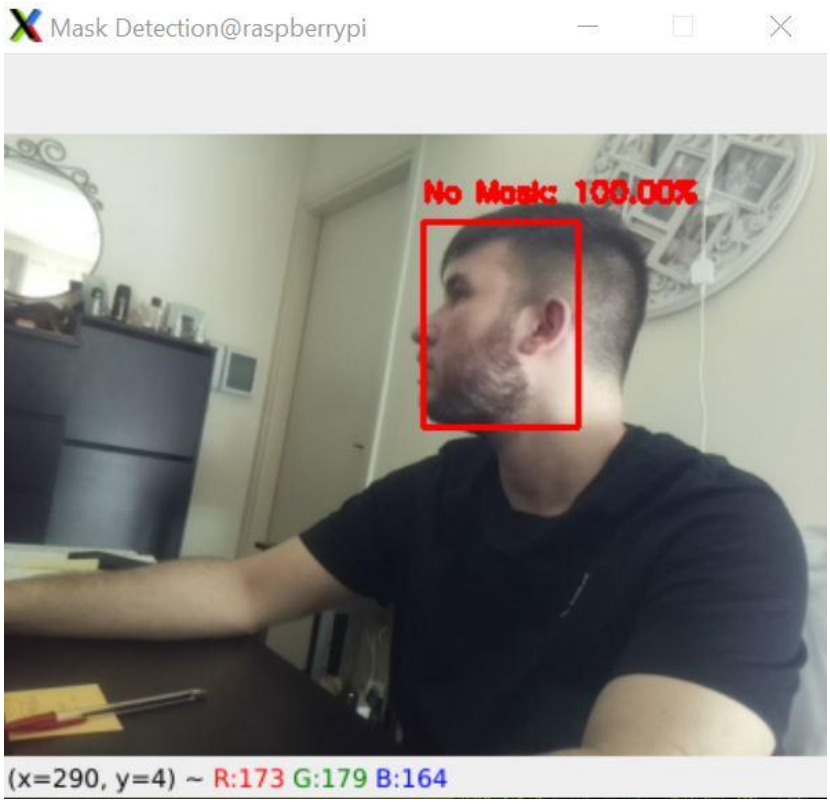


Εικόνα 4.9 Στιγμιότυπο μπροστινού προσώπου χωρίς μάσκα

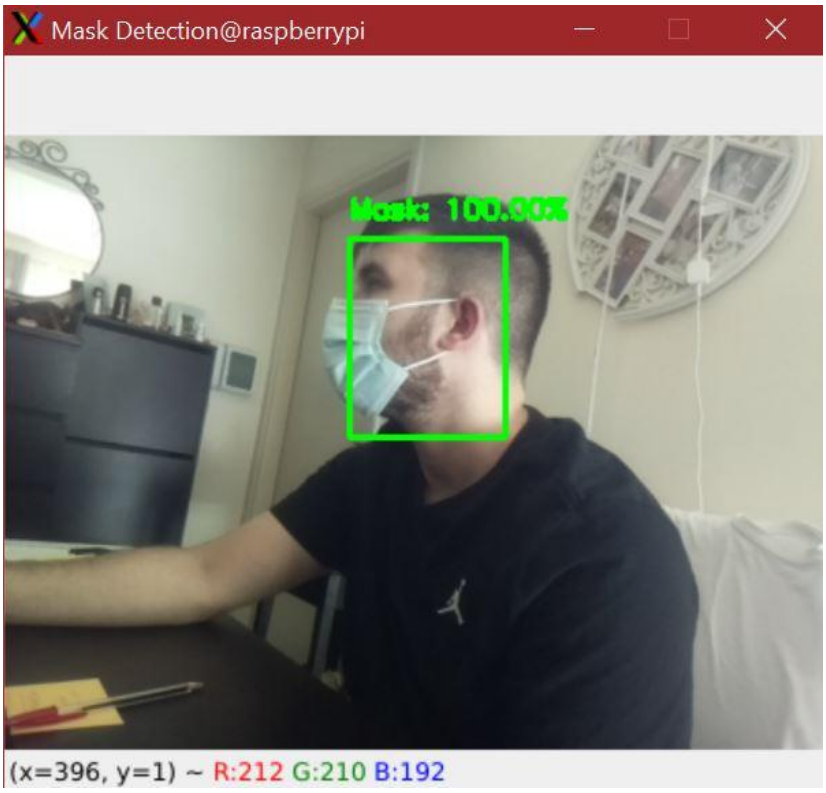


Εικόνα 4.10 Στιγμιότυπο μπροστινού προσώπου με μάσκα

4.2.2 Στιγμιότυπα προσώπου από πλάγια

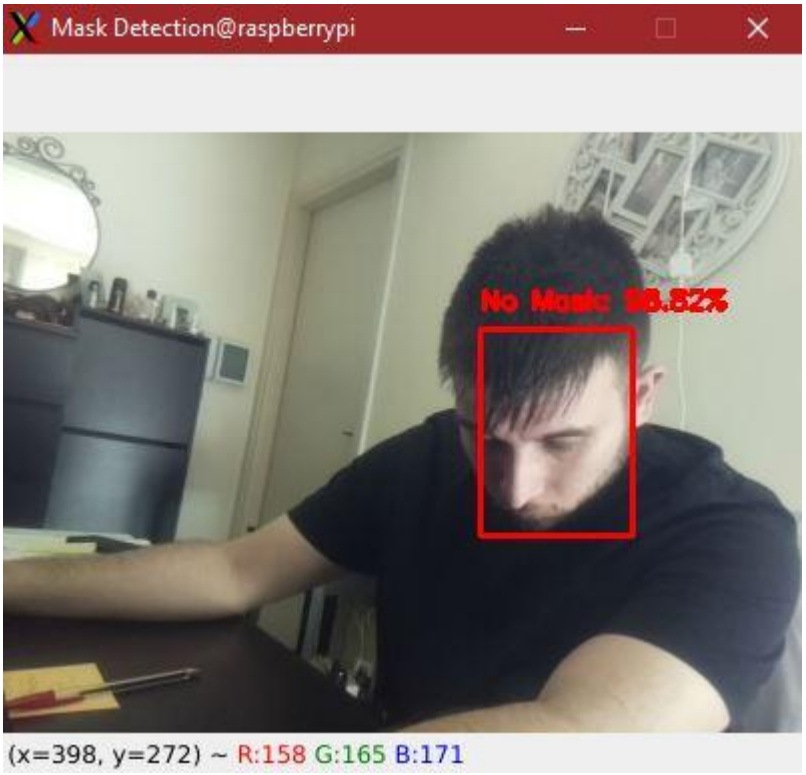


Εικόνα 4.11 Στιγμιότυπο προσώπου από πλάγια χωρίς μάσκα

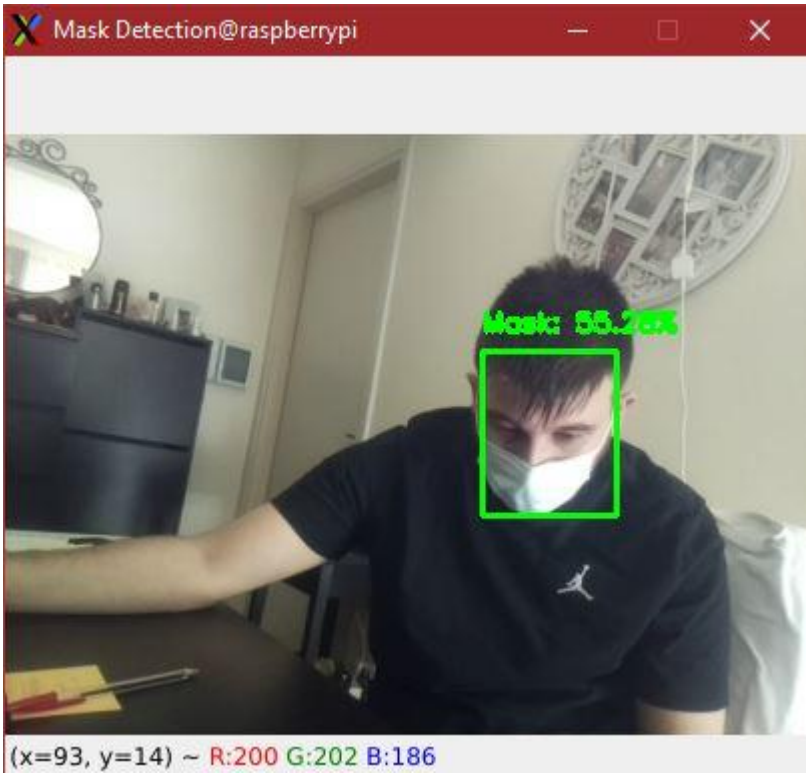


Εικόνα 4.12 Στιγμιότυπο προσώπου από πλάγια με μάσκα

4.2.3 Στιγμιότυπα προσώπου που κοιτάει προς τα κάτω

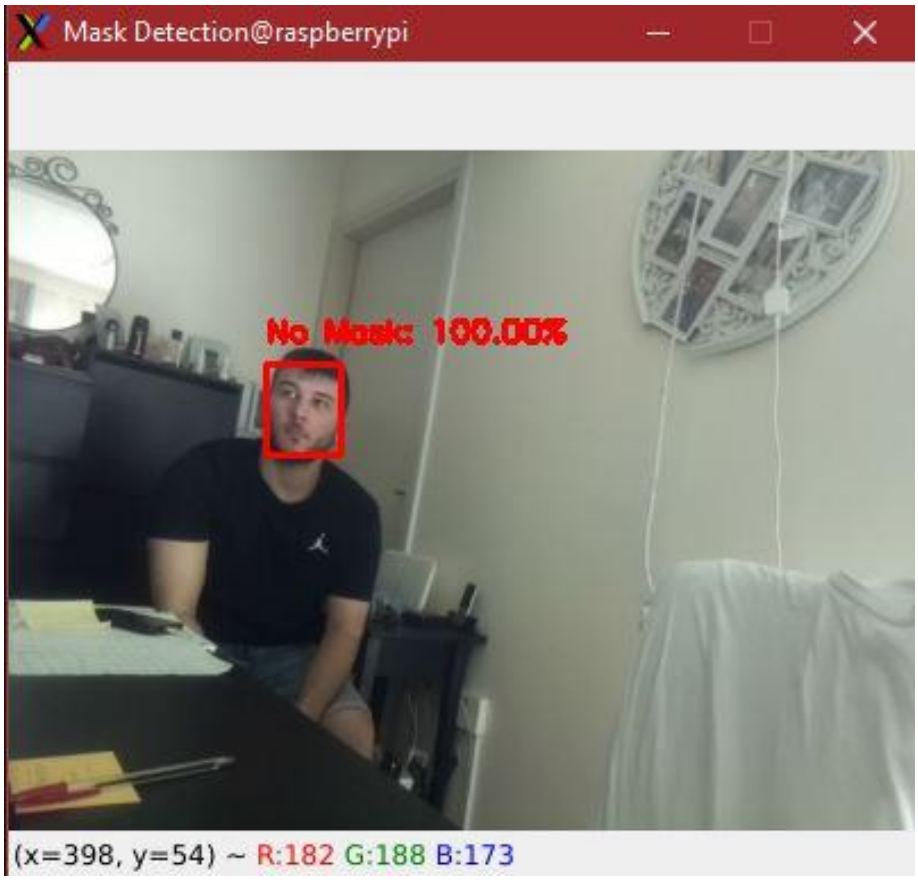


Εικόνα 4.13 Στιγμιότυπο προσώπου που κοιτάει προς τα κάτω χωρίς μάσκα

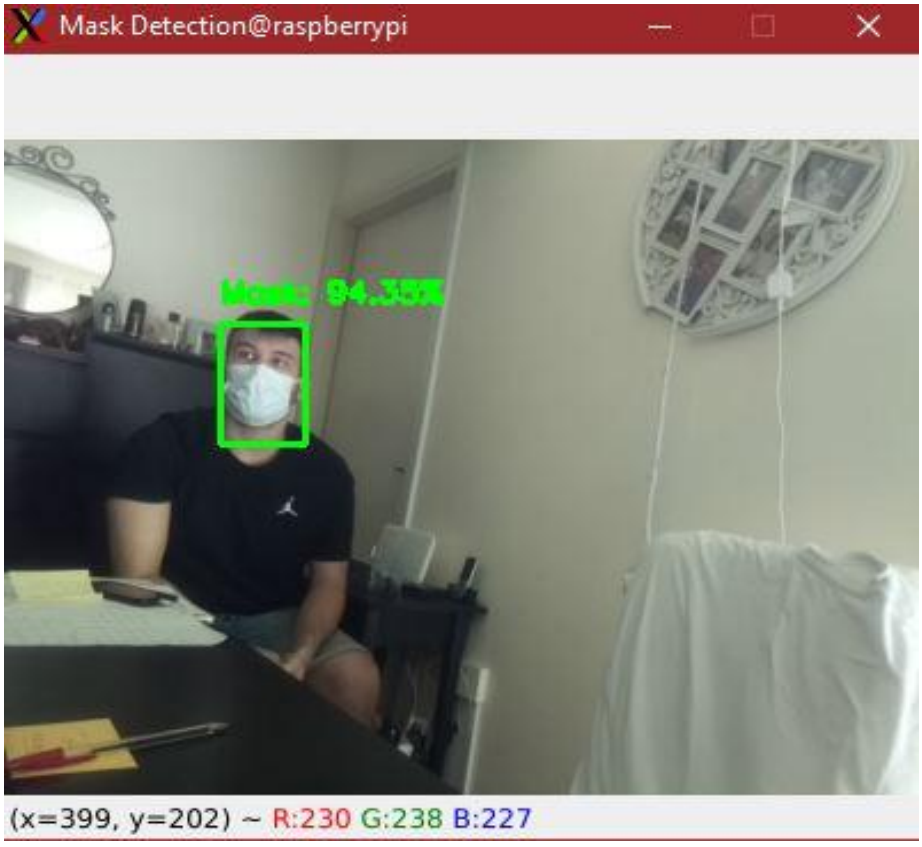


Εικόνα 4.14 Στιγμιότυπο προσώπου που κοιτάει προς τα κάτω με μάσκα

4.2.4 Στιγμιότυπα προσώπου από μακριά

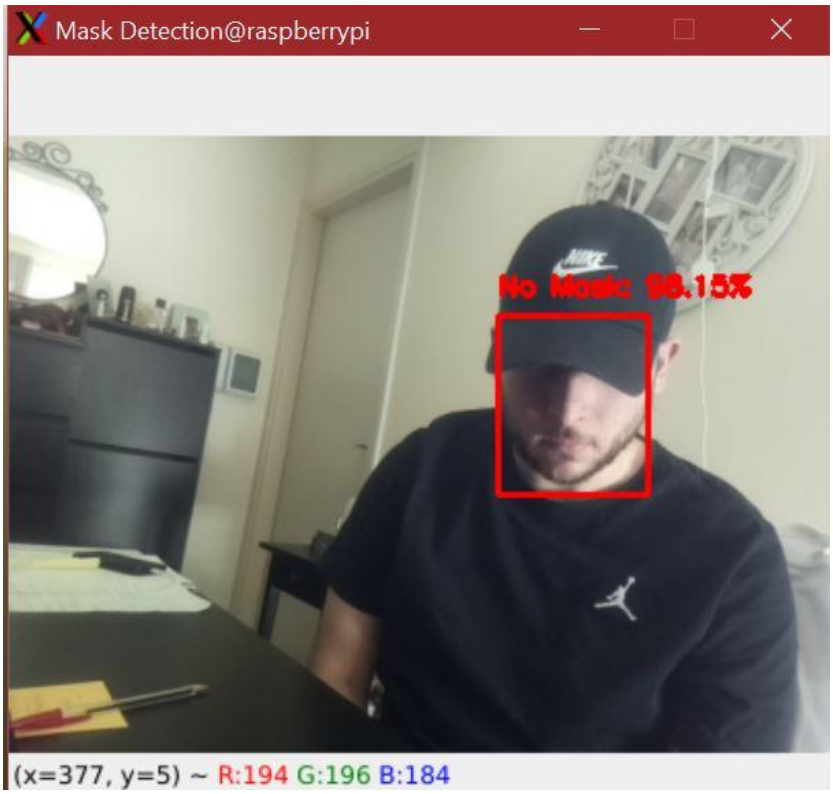


Εικόνα 4.15 Στιγμιότυπο προσώπου από μακριά χωρίς μάσκα

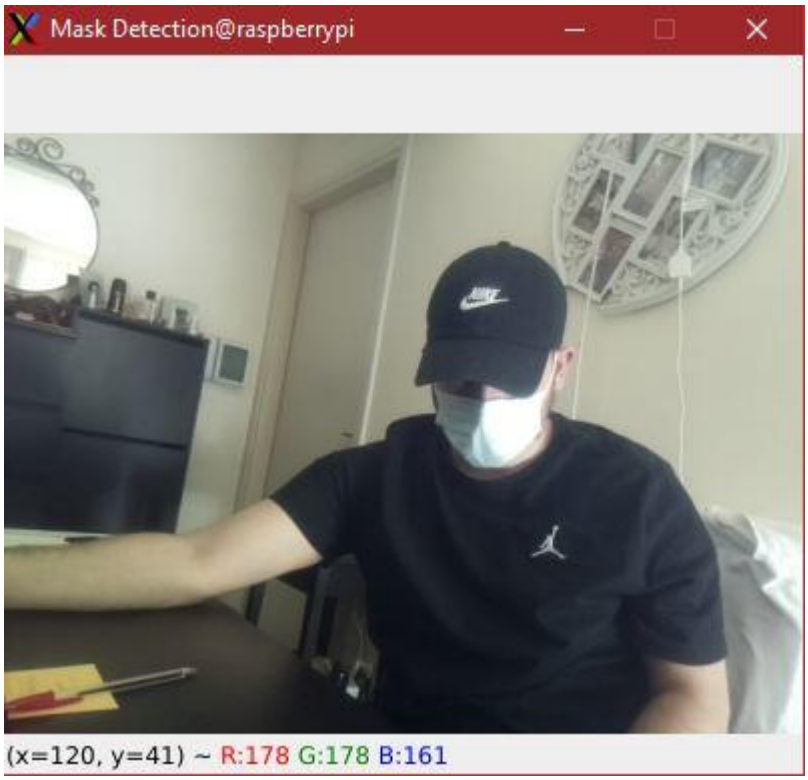


Εικόνα 4.16 Στιγμιότυπο προσώπου από μακριά με μάσκα

4.2.5 Στιγμιότυπα προσώπου με καλυμμένα χαρακτηριστικά

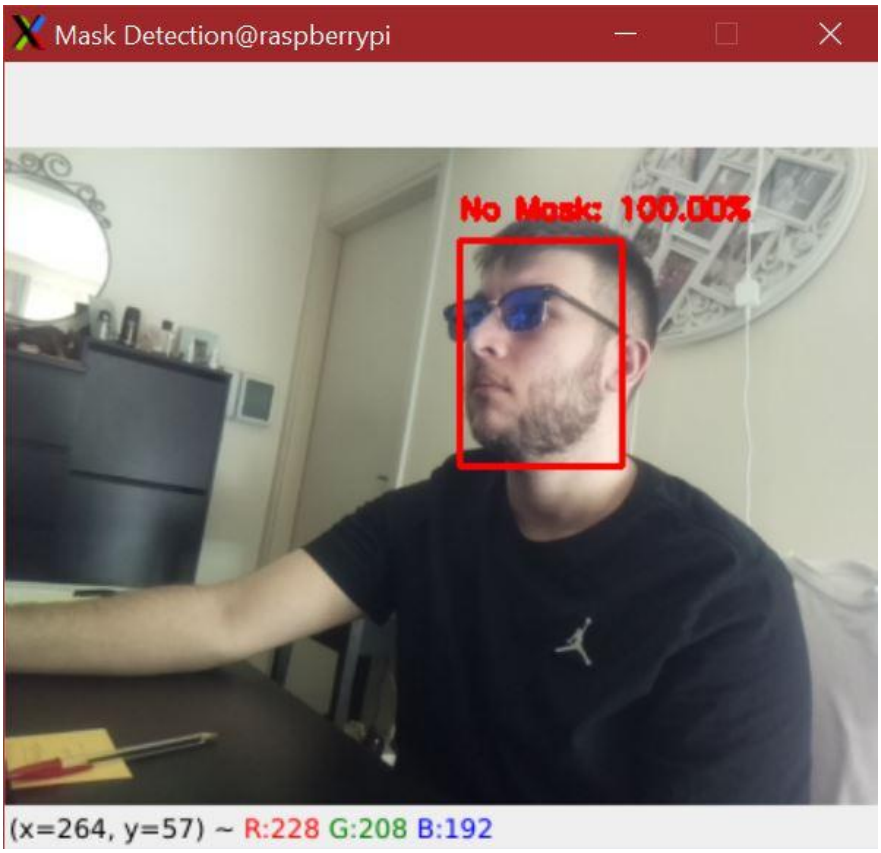


Εικόνα 4.17 Στιγμιότυπο προσώπου με καλυμμένα χαρακτηριστικά χωρίς μάσκα

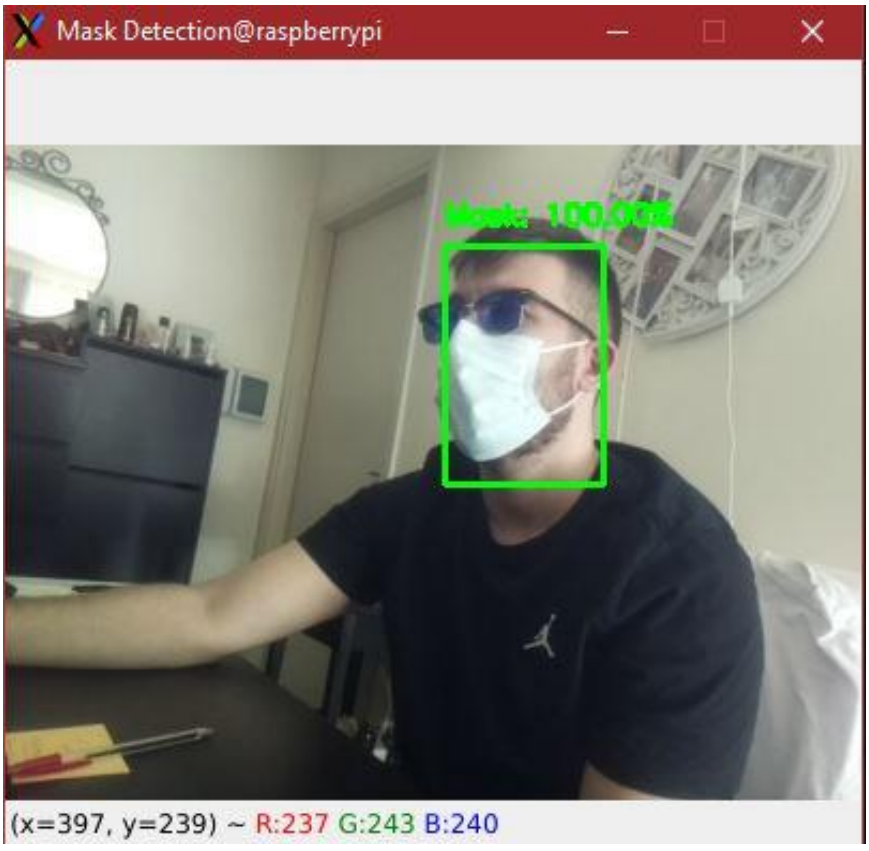


Εικόνα 4.18 Στιγμιότυπο προσώπου με καλυμμένα χαρακτηριστικά με μάσκα

4.2.6 Στιγμιότυπα προσώπου με γυαλιά

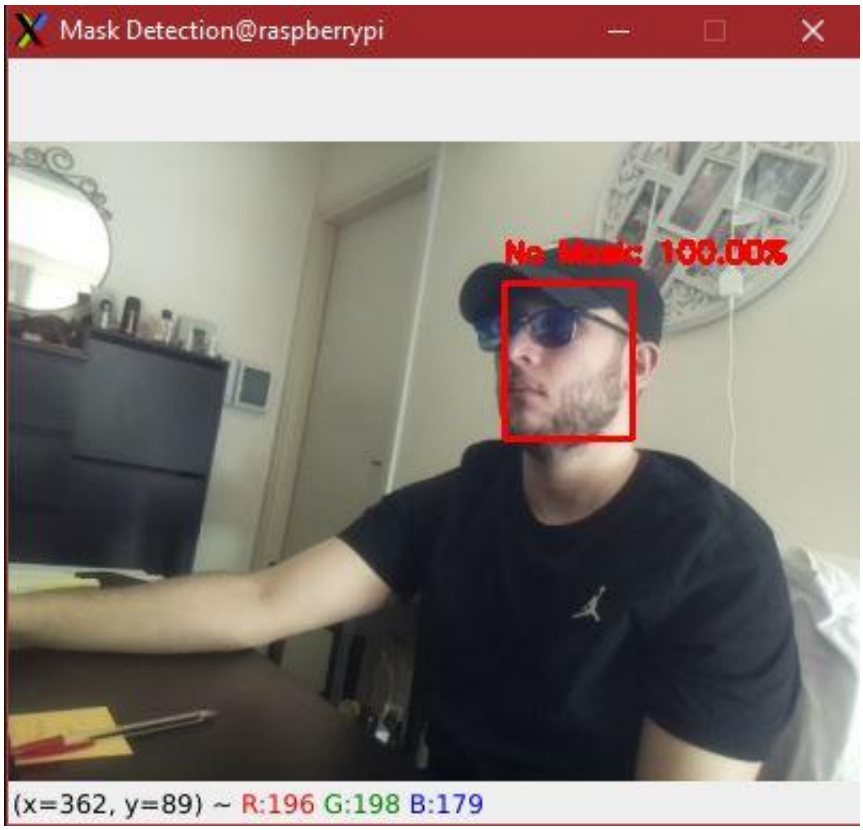


Εικόνα 4.19 Στιγμιότυπο προσώπου με γυαλιά χωρίς μάσκα

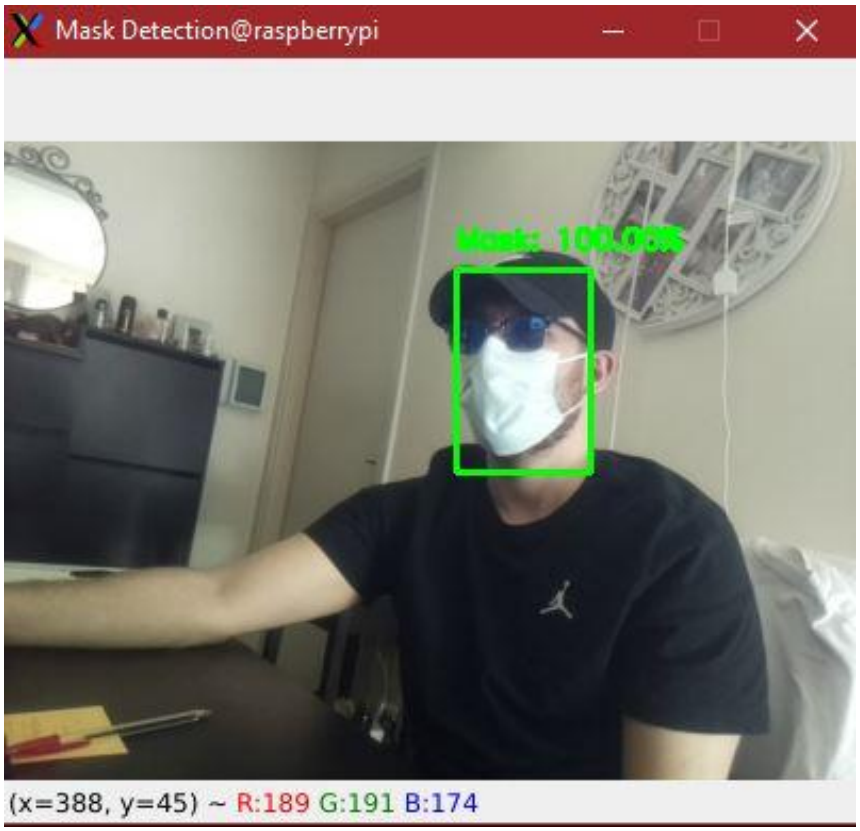


Εικόνα 4.20 Στιγμιότυπο προσώπου με γυαλιά με μάσκα

4.2.7 Στιγμιότυπα προσώπου με καπέλο και γυαλιά

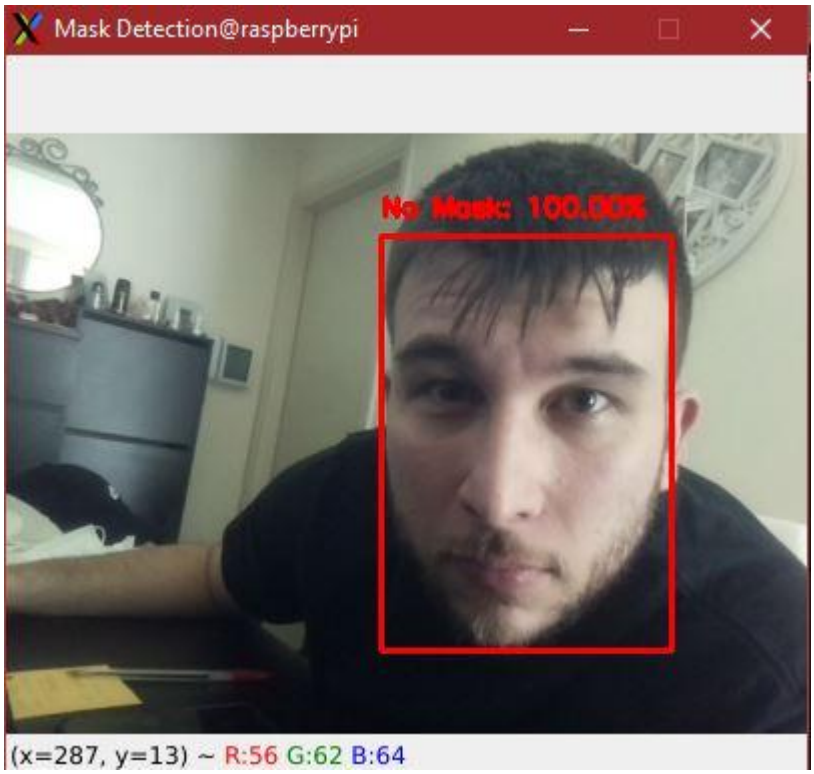


Εικόνα 4.21 Στιγμιότυπο προσώπου με καπέλο και γυαλιά χωρίς μάσκα

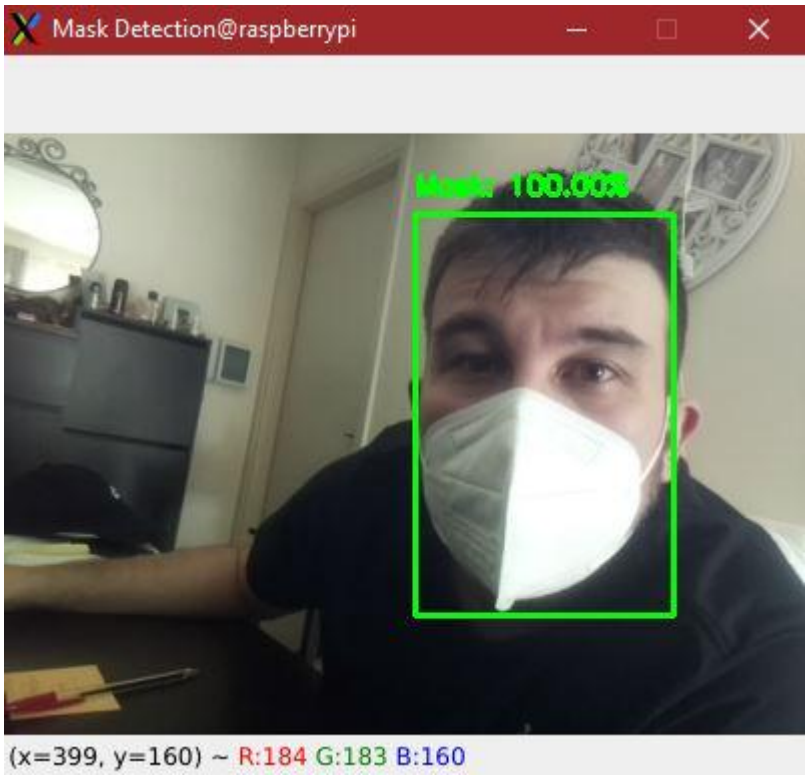


Εικόνα 4.22 Στιγμιότυπο προσώπου με καπέλο και γυαλιά με μάσκα

4.2.8 Στιγμιότυπα προσώπου με διαφορετική μάσκα και πιο κοντινή λήψη

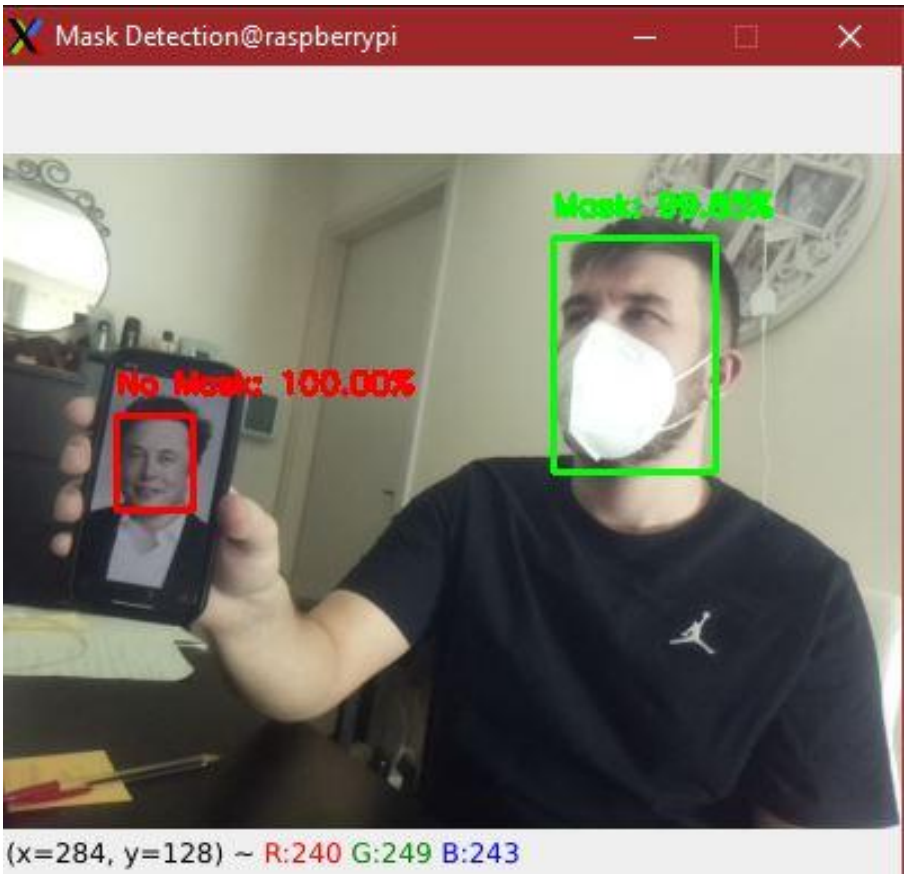


Εικόνα 4.23 Στιγμιότυπο προσώπου κοντινής λήψης χωρίς μάσκα



Εικόνα 4.24 Στιγμιότυπο προσώπου κοντινής λήψης και με διαφορετική μάσκα

4.2.9 Στιγμιότυπα πολλαπλών προσώπων



Εικόνα 4.25 Στιγμιότυπο δύο προσώπων, το ένα με μάσκα το άλλο χωρίς

Εικόνα 4.26 Στιγμιότυπο τριών προσώπων, το ένα με μάσκα και τα άλλα δύο χωρίς

5 ΚΕΦΑΛΑΙΟ 5^{ο} : Επίλογος

Στο τελευταίο κεφάλαιο αυτό παρουσιάζονται τα συμπεράσματα και οι προτάσεις σχετικά με μελλοντικές βελτιώσεις ή ιδέες πάνω στην διπλωματική εργασία αυτή. Στα συμπεράσματα γίνεται αρχικά μια ανασκόπηση στη διαδικασία που ακολουθήθηκε για την ολοκλήρωση της εργασίας και σχολιάζονται τα αποτελέσματα και οι δυσκολίες που αντιμετωπίστηκαν. Όσον αφορά τις μελλοντικές βελτιώσεις και ιδέες, αυτές έχουν να κάνουν με την περεταίρω αναβάθμιση της εφαρμογής που δημιουργήθηκε σε αυτήν την διπλωματική εργασία.

5.1 Συμπεράσματα

Κατά την διάρκεια εκπόνησης της παρούσας διπλωματικής εργασίας αναπτύχθηκε ο κώδικας εκπαίδευσης μοντέλου μηχανικής μάθησης για την ταξινόμηση εικόνων με κριτήριο εάν οι άνθρωποι μέσα σε αυτές φορούν ή δε φορούν μάσκα καθώς αναπτύχθηκε και ο κώδικας για την εφαρμογή του εκπαιδευμένου μοντέλου και χρήση του στη συσκευή Raspberry Pi 4.

Και οι δύο κώδικες χωρίζονται σε διάφορα στάδια όπου κάθε στάδιο εκτελεί ένα συγκεκριμένο σκοπό και για να δουλέψει με επιτυχία χρειάστηκαν αλλαγές εντολών, διάφορες δοκιμές και αρκετή έρευνα.

Ένα βασικό πρόβλημα του πρώτου κώδικα ήταν πως για την εκπαίδευση ενός μοντέλου χρειαζόντουσαν πολλοί πόροι του υπολογιστή και μια καλή κάρτα γραφικών, κάτι το οποίο δυσκόλεψε την κατάσταση όσον αφορά τους χρόνους εκπαίδευσης. Η εκπαίδευση στον υπολογιστή που χρησιμοποιήθηκε μπορεί να έπαιρνε 20 λεπτά, μία ώρα ή και παραπάνω ανάλογα με τις υπερπαραμέτρους που τέθηκαν. Για αυτό το λόγο χρησιμοποιήθηκε η βοήθεια απομακρυσμένων πόρων υπολογιστή με καλή κάρτα γραφικών μέσω της ιστοσελίδας “Google Colab”. Ένα άλλο πρόβλημα που σχετίζεται και αυτό με τους πόρους και τον χρόνο εκπαίδευσης, είναι πως δεν υπήρχε η δυνατότητα παροχής πολλών περισσότερων δεδομένων εκπαίδευσης στο μοντέλο, διότι αυτό θα έπαιρνε πολύ χρόνο παρόλο που ίσως το εκπαίδευε ακόμα καλύτερα

Όσον αφορά τα αποτελέσματα του πρώτου κώδικα, αυτά συγκρίθηκαν ως προς τις τιμές και τα διαγράμματά τους και επιλέχθηκε το βέλτιστο μοντέλο με κριτήριο το να συνδυάζει χαμηλές τιμές απωλειών, υψηλές τιμές απόδοσης και έναν σχετικά γρήγορο χρόνο εκπαίδευσης. Υπήρχαν τρία υποψήφια βέλτιστα μοντέλα με πολύ καλές επιδόσεις και παραπλήσιες τιμές αλλά επιλέχθηκε ως βέλτιστο εκείνο που χρειάστηκε τις λιγότερες επαναλήψεις να εκπαιδευτεί διότι στην πράξη και τα τρία αυτά μοντέλα είχαν το ίδιο ποσοστό επιτυχίας.

Για την υλοποίηση του πρώτου κώδικα δημιουργήθηκε ο δεύτερος, που αποτέλεσε την εφαρμογή της ανίχνευσης μάσκας σε ανθρώπινα πρόσωπα χρησιμοποιώντας το βέλτιστο μοντέλο που δημιουργήθηκε από τον κώδικα εκπαίδευσης. Ο δεύτερος κώδικας χρειάστηκε αρκετές προσαρμογές και αλλαγές έτσι ώστε να λειτουργήσει σωστά με το προσαρμοσμένο μοντέλο της μορφής “.tflite”. Συναντήθηκαν προβλήματα όπως η μη αναγνώριση της κάμερας V2 του Raspberry Pi 4 με σκοπό την επανεγκατάσταση των βιβλιοθηκών της ή η μη αναγνώριση από τον κώδικα πάνω από δύο προσώπων στο βίντεο με αποτέλεσμα η εφαρμογή να τερματίζει. Παρόλα αυτά μετά από έρευνα και πολλές προσπάθειες η εφαρμογή δούλεψε χωρίς κανένα πρόβλημα.

Τα αποτελέσματα της εφαρμογής παρατηρήθηκαν μέσω της ζωντανής μετάδοσης βίντεο που προβάλλεται σε ένα γραφικό παράθυρο κατά την εκτέλεσή του κώδικα. Έγιναν διάφορες δοκιμές

με και χωρίς μάσκα, για την παρατήρηση της ευστοχίας και της λειτουργικότητας της εφαρμογής και κατά συνέπεια του μοντέλου, όπως η αλλαγή γωνίας του προσώπου, η κάλυψη των ματιών φορώντας καπέλο ή η τοποθέτηση διαφορετικής μάσκας. Κατά τον έλεγχο των αποτελεσμάτων παρατηρήθηκε γενικώς πως η εφαρμογή λειτουργεί πολύ καλά κατά μέσο όρο και αναγνωρίζει τις διάφορες καταστάσεις που της δόθηκαν, αλλά στην περίπτωση της κάλυψης των ματιών από το καπέλο και ταυτόχρονα του προσώπου από τη μάσκα δεν αναγνωρίστηκε η κατάσταση σωστά. Αυτό συμβαίνει διότι δεν υπήρχαν πολλά χαρακτηριστικά προσώπου φανερά οπότε δεν αναγνωρίστηκε πρώτα το πρόσωπο από το προεκπαιδευμένο μοντέλο ανίχνευσης προσώπων έτσι ώστε να μπορέσει στην συνέχεια να ταξινομήσει το βέλτιστο μοντέλο ανίχνευσης μάσκας σωστά τα δεδομένα.

Μέσα από όλη αυτή την έρευνα, τις διαδικασίες και την πρακτική υλοποίηση της διπλωματικής εργασίας, μπορεί κάποιος να κατανοήσει τα οφέλη της τεχνητής νοημοσύνης για τον άνθρωπο με τη σωστή χρήση της πάντα ώστε να δημιουργηθούν τα επιθυμητά αποτελέσματα. Η τεχνητή νοημοσύνη και η μηχανική μάθηση μπορούν να βοηθήσουν τον άνθρωπο έτσι ώστε να μην χρειάζεται να εκτελεί ενέργειες οι οποίες μπορούν να εκτελεστούν αυτοματοποιημένα και με μεγαλύτερη ασφάλεια. Στη συγκεκριμένη περίπτωση της ανίχνευσης μάσκας, δεν χρειάζεται η ανθρώπινη παρατήρηση για την διεξαγωγή συμπερασμάτων σχετικά με το ποιος φοράει ή όχι μάσκα αφού όλα καταγράφονται από την εφαρμογή. Επίσης ο άνθρωπος δεν είναι φτιαγμένος να παρατηρεί πολλά πράγματα ταυτόχρονα και να βγάζει κάποιο συμπέρασμα για όλα αυτά ταυτόχρονα, οπότε η εφαρμογή που υλοποιήθηκε θα μπορούσε με ευκολία να αναγνωρίσει τις καταστάσεις των προσώπων για παράδειγμα 10 ανθρώπων την ίδια στιγμή και σε ασύλληπτο χρόνο.

5.2 Μελλοντικές βελτιώσεις και ιδέες

Κατά την διάρκεια αυτής της διπλωματικής εργασίας δημιουργήθηκαν ιδέες για μελλοντική βελτίωση της τελικής εφαρμογής που αναπτύχθηκε.

Μία από αυτές τις ιδέες είναι η προσθήκη αισθητήρα υπέρυθρων στο Raspberry Pi 4 για την καταμέτρηση των ανθρώπων που βρίσκονται στον ίδιο χώρο και την προβολή κατάλληλου μηνύματος στην οθόνη ή την παραγωγή ενός ήχου προειδοποίησης.

Σε συνδυασμό με τον αισθητήρα υπέρυθρων θα μπορούσε να προστεθεί και ένας αισθητήρας θερμοκρασίας ο οποίος με βάση ένα συγκεκριμένο όριο να προειδοποιεί πάλι τους χρήστες ή να κλείνει την πόρτα εισόδου π.χ. ενός ιατρείου μέσω ενός βοηθητικού μοτέρ για να μην μπορούν να εισέλθουν παραπάνω άτομα στον χώρο και πιθανότατα να μολυνθούν από τον ιό του covid-19.

Μία ακόμα μικρή ιδέα που θα μπορούσε να υλοποιηθεί στο μέλλον είναι η προσθήκη εντολών στην εφαρμογή ανίχνευσης μάσκας για την εμφάνιση του αριθμού των καρέ ανά δευτερόλεπτο στο βίντεο με τα αποτελέσματα, έτσι ώστε να μπορέσει να υπάρξει σύγκριση μεταξύ των μοντέλων σχετικά με το πιο από αυτά κάνει πιο χρονοβόρο το σύστημα.

Επίσης μετά από έρευνα ανακαλύφθηκε πως για την επιτάχυνση των εφαρμογών σε ένα Raspberry Pi, που σχετίζονται με τη μηχανική μάθηση, υπάρχει διαθέσιμη προς αγορά μια συσκευή με υποδοχή θύρας usb που ονομάζεται “Coral”. Η συσκευή αυτή κάνει τρισεκατομμύρια

υπολογιστικούς υπολογισμούς το δευτερόλεπτο, αφού χρησιμοποιεί έναν επεξεργαστή της "Google", αλλά δυστυχώς η τιμή της είναι σχετικά ακριβή.

Η τελευταία και αρκετά προκλητική ιδέα για το μέλλον είναι η ανάπτυξη των αντίστοιχων κωδίκων που δημιουργήθηκαν σε αυτήν την διπλωματική εργασία αλλά με τη βιβλιοθήκη "Pytorch" και όχι με την "Tensorflow". Η "Pytorch" έχει γίνει αρκετά γνωστή τον τελευταίο καιρό και προσφέρει πολλές δυνατότητες σε ότι αφορά τον προγραμματισμό στο πεδίο της μηχανικής μάθησης. Έτσι, αυτή θα ήταν μια καλή δοκιμή στο μέλλον για να συγκριθούν οι κώδικες και να παρατηρηθούν οι διαφορές των δύο βιβλιοθηκών.

Βιβλιογραφία – Αναφορές - Διαδικτυακές Πηγές

[1] M. Neelam, Neelam MahaLakshmi (2021) Aspects of Artificial Intelligence In Karthikeyan.J, Su-Hie Ting and Yu-Jin Ng (eds), "Learning Outcomes of Classroom Research" p:250-256, L' Ordine Nuovo Publication, India. 978-93-92995-15-6. 2022.

[2] K. C. A. Khanzode, 'ADVANTAGES AND DISADVANTAGES OF ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING: A LITERATURE REVIEW'.

[3] S. Das, A. Dey, A. Pal, και N. Roy, 'Applications of Artificial Intelligence in Machine Learning: Review and Prospect', Int. J. Comput. Appl., τ. 115, τχ. 9, σσ. 31–41, Απριλίου 2015, doi: 10.5120/20182-2402.

[4] 'Machine Learning Algorithms - Javatpoint', www.javatpoint.com. https://www.javatpoint.com/machine-learning-algorithms

[5] 'Supervised Machine learning - Javatpoint', www.javatpoint.com. https://www.javatpoint.com/supervised-machine-learning

[6] 'Unsupervised Machine learning - Javatpoint'. https://www.javatpoint.com/unsupervised-machine-learning

[7] 'Reinforcement Learning Tutorial - Javatpoint', www.javatpoint.com. https://www.javatpoint.com/reinforcement-learning

[8] F. Pourafshin, 'Big Data Mining in Internet of Things Using Fusion of Deep Features', Απριλίου 2021.

[9] 'What Is Deep Learning? | How It Works, Techniques & Applications'. https://www.mathworks.com/discovery/deep-learning.html

[10] I. Arraut και D. Diaz, 'On the Loss of Learning Capability Inside an Arrangement of Neural Networks: The Bottleneck Effect in Black-Holes', Symmetry, τ. 12, σ. 1484, Σεπτεμβρίου 2020, doi: 10.3390/sym12091484.

[11] M. Nielsen, 'Neural Networks and Deep Learning'.

[12] 'What is a Neural Network?', TIBCO Software. https://www.tibco.com/reference-center/what-is-a-neural-network

[13] J. P. Bharadiya, 'Convolutional Neural Networks for Image Classification', Int. J. Innov. Sci. Res. Technol., τ. 8, τχ. 5, σσ. 673-677., Μαΐου 2023, doi: 10.5281/zenodo.7952031.

[14] Dharmaraj, 'Convolutional Neural Networks (CNN) — Architectures Explained', Medium, 1 Ιούνιος 2022. https://medium.com/@draj0718/convolutional-neural-networks-cnn-architectures-explained-716fb197b243

[15] 'Basic CNN Architecture: Explaining 5 Layers of Convolutional Neural Network', upGrad blog. https://www.upgrad.com/blog/basic-cnn-architecture/

[16] S. Sharma, S. Sharma, και A. Athaiya, 'ACTIVATION FUNCTIONS IN NEURAL NETWORKS', Int. J. Eng. Appl. Sci. Technol., τ. 04, τχ. 12, σσ. 310–316, Μαΐου 2020, doi: 10.33564/IJEAST.2020.v04i12.054.

[17] 'Intro to optimization in deep learning: Gradient Descent', Paperspace Blog, 2 Ιούνιος 2018. https://blog.paperspace.com/intro-to-optimization-in-deep-learning-gradient-descent/

[18] R. Roy, 'Neural Networks: Forward pass and Backpropagation', Medium, 17 Νοέμβριος 2022. https://towardsdatascience.com/neural-networks-forward-pass-and-backpropagation-be3b75a1cfcc

[19] 'Root-Mean-Square Error in R Programming', GeeksforGeeks, 20 Ιούλιος 2020. https://www.geeksforgeeks.org/root-mean-square-error-in-r-programming/

[20] V. Sze, Y.-H. Chen, T.-J. Yang, και J. S. Emer, 'Efficient Processing of Deep Neural Networks: A Tutorial and Survey', Proc. IEEE, τ. 105, τχ. 12, σσ. 2295–2329, Δεκεμβρίου 2017, doi: 10.1109/JPROC.2017.2761740.

[21] D. Pawar, 'Improving Performance of Convolutional Neural Network!', Medium, 3 Οκτώβριος 2020. https://medium.com/@dipti.rohan.pawar/improving-performance-of-convolutional-neural-network-2ecfe0207de7

[22] C. Kiourt, G. Pavlidis, και S. Markantonatou, Deep learning approaches in food recognition. 2020.

[23] 'Machine Learning Models: What They Are and How to Build Them', Coursera, 16 Ιούνιος 2023. https://www.coursera.org/articles/machine-learning-models

[24] 'What is Epoch in Machine Learning? | Intellipaat'. https://intellipaat.com/blog/epoch-in-machine-learning/?US

[25] E. Zvornicanin, 'Relation Between Learning Rate and Batch Size | Baeldung on Computer Science', 21 Ιανουάριος 2022. https://www.baeldung.com/cs/learning-rate-batch-size

[26] G. L. Team, 'An Easy Guide to Gradient Descent in Machine Learning', Great Learning Blog: Free Resources what Matters to shape your Career!, 18 Ιανουάριος 2022. https://www.mygreatlearning.com/blog/gradient-descent/

[27] Y. Ho και S. Wookey, 'The Real-World-Weight Cross-Entropy Loss Function: Modeling the Costs of Mislabeling', IEEE Access, τ. 8, σσ. 4806–4813, 2020, doi: 10.1109/ACCESS.2019.2962617.

[28] J. Brownlee, 'Gentle Introduction to the Adam Optimization Algorithm for Deep Learning', MachineLearningMastery.com, 2 Ιούλιος 2017. https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/

[29] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, και L.-C. Chen, 'MobileNetV2: Inverted Residuals and Linear Bottlenecks'. arXiv, 21 Μάρτιος 2019. [Έκδοση σε ψηφιακή μορφή]. Διαθέσιμο στο: http://arxiv.org/abs/1801.04381

[30] S. Dou, L. Wang, F. Donglin, L. Miao, J. Yan, και H. He, 'Classification of Citrus Huanglongbing Degree Based on CBAM-MobileNetV2 and Transfer Learning', Sensors, τ. 23, σ. 5587, Ιουνίου 2023, doi: 10.3390/s23125587.

[31] Y. Gulzar, 'Fruit Image Classification Model Based on MobileNetV2 with Deep Transfer Learning Technique', Sustainability, τ. 15, σ. 1906, Ιανουαρίου 2023, doi: 10.3390/su15031906.

[32] Z. Vujovic, 'Classification Model Evaluation Metrics', Int. J. Adv. Comput. Sci. Appl., τ. Volume 12, σσ. 599–606, Ιουλίου 2021, doi: 10.14569/IJACSA.2021.0120670.

[33] https://www.facebook.com/ahmed.f.gadd, 'Accuracy, Precision, and Recall in Deep Learning', Paperspace Blog, 12 Οκτώβριος 2020. https://blog.paperspace.com/deep-learning-metrics-precision-recall-accuracy/

[34] H. Ferreira, 'Confusion matrix and other metrics in machine learning', Hugo Ferreira's blog, 4 Απρίλιος 2018. https://medium.com/hugo-ferreiras-blog/confusion-matrix-and-other-metrics-in-machine-learning-894688cb1c0a

[35] K. Leung, 'Micro, Macro & Weighted Averages of F1 Score, Clearly Explained', Medium, 13 Σεπτέμβριος 2022. https://towardsdatascience.com/micro-macro-weighted-averages-of-f1-score-clearly-explained-b603420b292f

[36] 'Welcome to Python.org', Python.org, 1 Ιούλιος 2023. https://www.python.org/

[37] 'Python (programming language)', Wikipedia. 9 Ιούλιος 2023. [Έκδοση σε ψηφιακή μορφή]. Διαθέσιμο στο: https://en.wikipedia.org/w/index.php?title=Python_(programming_language)&oldid=1164530008

[38] 'PyCharm', Wikipedia. 30 Ιούνιος 2023. [Έκδοση σε ψηφιακή μορφή]. Διαθέσιμο στο: https://en.wikipedia.org/w/index.php?title=PyCharm&oldid=1162663992

[39] 'Anaconda (Python distribution)', Wikipedia. 9 Ιούλιος 2023. [Έκδοση σε ψηφιακή μορφή]. Διαθέσιμο στο: https://en.wikipedia.org/w/index.php?title=Anaconda_(Python_distribution)&oldid=1164405291

[40] 'TensorFlow', Wikipedia. 15 Μάιος 2023. [Έκδοση σε ψηφιακή μορφή]. Διαθέσιμο στο: https://en.wikipedia.org/w/index.php?title=TensorFlow&oldid=1154948861

[41] 'Keras', Wikipedia. 9 Ιούνιος 2023. [Έκδοση σε ψηφιακή μορφή]. Διαθέσιμο στο: https://en.wikipedia.org/w/index.php?title=Keras&oldid=1159311667

[42] R. Khandelwal, 'A Basic Introduction to TensorFlow Lite', Medium, 15 Ιούνιος 2020. https://towardsdatascience.com/a-basic-introduction-to-tensorflow-lite-59e480c57292

[43] 'scikit-learn', Wikipedia. 24 Απρίλιος 2023. [Έκδοση σε ψηφιακή μορφή]. Διαθέσιμο στο: https://en.wikipedia.org/w/index.php?title=Scikit-learn&oldid=1151444875

[44] 'What is NumPy? — NumPy v1.25 Manual'. https://numpy.org/doc/stable/user/whatisnumpy.html

[45] 'What Is Matplotlib In Python? How to use it for plotting?', ActiveState. https://www.activestate.com/resources/quick-reads/what-is-matplotlib-in-python-how-to-use-it-for-plotting/

[46] A. Rosebrock, 'imutils: A series of convenience functions to make basic image processing functions such as translation, rotation, resizing, skeletonization, displaying Matplotlib images, sorting contours, detecting edges, and much more easier with OpenCV and both Python 2.7 and Python 3. [Έκδοση σε ψηφιακή μορφή]. Διαθέσιμο στο: https://github.com/jrosebr1/imutils

[47] 'About', OpenCV. https://opencv.org/about/

[48] 'keyurr2/face-detection: Face detection implementation with different methods and applications'. https://github.com/keyurr2/face-detection/tree/master

[49] 'What is GitHub And How To Use It? [Updated]', Simplilearn.com. https://www.simplilearn.com/tutorials/git-tutorial/what-is-github

[50] 'Google Colab'. https://research.google.com/colaboratory/faq.html

[51] Mobatek, 'MobaXterm free Xserver and tabbed SSH client for Windows'. https://mobaxterm.mobatek.net/

[52] R. P. Ltd, 'Raspberry Pi OS', Raspberry Pi. https://www.raspberrypi.com/software/

[53] R. P. Ltd, 'Buy a Raspberry Pi 4 Model B', Raspberry Pi. https://www.raspberrypi.com/products/raspberry-pi-4-model-b/

[54] R. P. Ltd, 'Raspberry Pi 4 Model B specifications', Raspberry Pi. https://www.raspberrypi.com/products/raspberry-pi-4-model-b/specifications/

[55] 'Raspberry Pi Documentation - Raspberry Pi hardware'. https://www.raspberrypi.com/documentation/computers/raspberry-pi.html

[56] 'ABS Case for Raspberry Pi 4B Support Fan'. https://nettop.gr/index.php/raspberry-pi/thikes/abs-case-for-raspberry-pi-4b-support-fan.html

[57] R. P. Ltd, 'Buy a Raspberry Pi 4 Case Fan', Raspberry Pi. https://www.raspberrypi.com/products/raspberry-pi-4-case-fan/

[58] R. P. Ltd, 'Buy a Raspberry Pi Camera Module 2', Raspberry Pi. https://www.raspberrypi.com/products/camera-module-v2/

[59] 'Find Open Datasets and Machine Learning Projects | Kaggle'. https://www.kaggle.com/datasets

Παράρτημα Α : Κώδικας εκπαίδευσης μοντέλου ανίχνευσης μάσκας

```
# Προσθήκη όλων των απαραίτητων πακέτων δηλαδή των βιβλιοθηκών (libraries), ενοτήτων (modules), πλαισίων (frameworks)  ή μέρος αυτών για να μπορέσει να εκτελεστεί ο κώδικας. Κάποια πακέτα εισάγονται ολόκληρα απλά με την εντολή "import όνομα_επιθυμητού_πακέτου". Σε άλλες περιπτώσεις εισάγονται μέρη αυτών που θα χρειαστούν ή κάποιες συναρτήσεις (functions) τους ή μεθόδους (methods) τους. Για την εισαγωγή ενός μόνο μέρους του πακέτου χρησιμοποιείται η σύνταξη "from όνομα_επιθυμητού_πακέτου import όνομα_επιθυμητού_μέρους_του". Επίσης με την προσθήκη της εντολής "as επιθυμητό_νέο_όνομα" δίπλα από την εντολή εισαγωγής κάποιου  πακέτου ή μέρους αυτού, δίνεται η δυνατότητα στον κώδικα να απευθύνεται σε αυτό με ένα νέο όνομα, συνήθως πιο σαφές, απλό και σύντομο.

# Η ενότητα os ανήκει στην βιβλιοθήκη της Python και περιέχει εντολές που αλληλοεπιδρούν με το λειτουργικό σύστημα όπως για παράδειγμα η πλοήγηση σε φακέλους ή η πρόσβαση σε αρχεία.
import os

# Η ενότητα "sys" ανήκει στην βιβλιοθήκη της Python και περιέχει εντολές που επιτρέπουν την πρόσβαση σε ορίσματα της γραμμής εντολών (command line), ειδικές παραμέτρους του συστήματος και συναρτήσεις για τον χειρισμό του περιβάλλοντος της Python.
import sys

# Το "tensorflow" είναι ένα πλαίσιο ανοικτού κώδικα (open-source code) που περιέχει εντολές με σκοπό τη δημιουργία και εκπαίδευση μοντέλων μηχανικής μάθησης (machine learning models).
import tensorflow

# Εισαγωγή της συνάρτησης "load_img" απο την "tensorflow.keras.preprocessing.image" ενότητα. Η συνάρτηση αυτή φορτώνει ένα αρχείο εικόνας από τον τοπικό δίσκο του υπολογιστή μέσα στον κώδικα.
from tensorflow.keras.preprocessing.image import load_img

# Εισαγωγή της συνάρτησης "img_to_array" από την "tensorflow.keras.preprocessing.image" ενότητα. Η συνάρτηση αυτή, μετατρέπει μια εικόνα σε διάνυσμα της βιβλιοθήκης NumPy έτσι ώστε αυτή να μπορεί να υποβληθεί σε επεξεργασία από μοντέλα μηχανικής εκμάθησης.
from tensorflow.keras.preprocessing.image import img_to_array

# Εισαγωγή της συνάρτησης "preprocess_input" από την "tensorflow.keras.applications.mobilenet_v2" ενότητα. Η συνάρτηση αυτή προεπεξεργάζεται τις εισαγόμενες εικόνες και τις προετοιμάζει με βάση την αρχιτεκτονική του μοντέλου MobileNetV2 που αποτελεί το συνελικτικό νευρωνικό δίκτυο του μοντέλου που θα εκπαιδευτεί.
from tensorflow.keras.applications.mobilenet_v2 import preprocess_input

# Εισαγωγή της κλάσης (class) "LabelBinarizer" από την "sklearn.preprocessing" ενότητα. Η κλάση αυτή χρησιμοποιείται για τη δυαδοποίηση (binarizing) των ετικετών (labels) ή τη μετατροπή κατηγορικών ετικετών (categorical labels) σε δυαδικά διανύσματα.
from sklearn.preprocessing import LabelBinarizer

# Εισαγωγή της συνάρτησης "to_categorical" από την "tensorflow.keras.utils" ενότητα. Η συνάρτηση αυτή μετατρέπει ακέραιες ετικέτες σε κωδικοποιημένα διανύσματα, της μεθόδου one-hot encoding.
from tensorflow.keras.utils import to_categorical

# Εισαγωγή της βιβλιοθήκης "numpy" με το ψευδώνυμο "np" που χρησιμεύει για αριθμητικούς υπολογισμούς στην Python. Επίσης παρέχει υποστήριξη για μεγάλους πολυδιάστατους πίνακες ή πίνακες διανυσμάτων, αφού περιέχει μια συλλογή μαθηματικών συναρτήσεων για την αποτελεσματική επεξεργασία τους.
import numpy as np

# Εισαγωγή της συνάρτησης "train_test_split" από την "sklearn.model_selection" ενότητα. Η συνάρτηση αυτή χωρίζει το σύνολο δεδομένων (image dataset) σε υποσύνολα εκπαίδευσης (training set/subset) και δοκιμής (testing set/subset).
from sklearn.model_selection import train_test_split

# Εισαγωγή της κλάσης "ImageDataGenerator" από την "tensorflow.keras.preprocessing.image" ενότητα. Η κλάση αυτή αυξάνει τα δεδομένα, δημιουργώντας ομάδες αυξημένων εικόνων για την εκπαίδευση μοντέλων βαθιάς μάθησης (deep learning models).
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

```
# Εισαγωγή της αρχιτεκτονικής του μοντέλου "MobileNetV2" από την "tensorflow.keras.applications" ενότητα. Το MobileNetV2 είναι μια προεκπαιδευμένη αρχιτεκτονική συνελικτικού νευρωνικού δικτύου που μπορεί να χρησιμοποιηθεί σε περιπτώσεις ταξινόμησης εικόνων.
from tensorflow.keras.applications import MobileNetV2

# Εισαγωγή του επιπέδου (layer) "Input" από την "tensorflow.keras.layers" ενότητα. Το επίπεδο αυτό αντιπροσωπεύει την είσοδο στα κρυφά επίπεδα του νευρωνικού δικτύου του μοντέλου, ορίζοντας το σχήμα και τον τύπο των δεδομένων εισόδου.
from tensorflow.keras.layers import Input

# Εισαγωγή του επιπέδου "AveragePooling2D" από την "tensorflow.keras.layers" ενότητα. Το επίπεδο αυτό εκτελεί μέση συγκέντρωση (Average Pooling), η οποία μειώνει τις χωρικές διαστάσεις των δεδομένων εισόδου λαμβάνοντας το μέσο όρο κάθε παραθύρου συγκέντρωσης.
from tensorflow.keras.layers import AveragePooling2D

# Εισαγωγή του επιπέδου "Flatten" από την "tensorflow.keras.layers" ενότητα. Το επίπεδο αυτό μετατρέπει την πολυδιάστατη είσοδο σε ένα μονοδιάστατο διάνυσμα, σε κατάλληλη δηλαδή μορφή για την εισαγωγή του στο επόμενο επίπεδο που είναι πλήρως συνδεδεμένο με το γειτονικό του.
from tensorflow.keras.layers import Flatten

# Εισαγωγή του επιπέδου "Dense" από την "tensorflow.keras.layers" ενότητα. Το επίπεδο αυτό αντιπροσωπεύει ένα πλήρως συνδεδεμένο επίπεδο, όπου κάθε νευρώνας συνδέεται με κάθε νευρώνα του προηγούμενου επιπέδου.
from tensorflow.keras.layers import Dense

# Εισαγωγή του επιπέδου "Dropout" από την "tensorflow.keras.layers" ενότητα. Το επίπεδο αυτό ρυθμίζει τυχαία ένα μέρος των εισόδων του προηγούμενου επιπέδου στο 0 κατά τη διάρκεια της εκπαίδευσης, κάτι που βοηθά στην αποφυγή της υπερπροσαρμογής (overfitting).
from tensorflow.keras.layers import Dropout

# Εισαγωγή της κλάσης "Model" από την "tensorflow.keras.models" ενότητα. Η κλάση αυτή δημιουργεί ένα αντικείμενο μοντέλου που καθορίζει την αρχιτεκτονική ενός νευρωνικού δικτύου.
from tensorflow.keras.models import Model

# Εισαγωγή του βελτιστοποιητή (optimizer) "Adam" από την "tensorflow.keras.optimizers" ενότητα. Ο Adam είναι ένας αλγόριθμος βελτιστοποίησης που χρησιμοποιείται συνήθως για την εκπαίδευση μοντέλων βαθιάς μάθησης (deep learning).
from tensorflow.keras.optimizers import Adam

# Εισαγωγή των συναρτήσεων "roc_curve" και "auc" από την "sklearn.metrics" ενότηα. Οι συναρτήσεις αυτές υπολογίζουν την καμπύλη ROC και την περιοχή AUC (AUC area) κάτω από την καμπύλη αυτή για την αξιολόγηση μοντέλων ταξινόμησης.
from sklearn.metrics import roc_curve, auc

# Εισαγωγή της συνάρτησης "classification_report" από την "sklearn.metrics" ενότητα. Η συνάρτηση αυτή δημιουργεί μια αναφορά με διάφορες μετρικές αξιολόγησης, όπως η  ακρίβεια, η ανάκληση και η βαθμολογία F1, για την αξιολόγηση της απόδοσης ενός μοντέλου ταξινόμησης.
from sklearn.metrics import classification_report

# Εισαγωγή της ενότητας "pyplot" από τη βιβλιοθήκη "matplotlib" με το ψευδώνυμο "plt". Το Matplotlib είναι μια βιβλιοθήκη σχεδίασης για την γλώσσα προγραμματισμού Python και το pyplot παρέχει μια απλή διεπαφή για τη δημιουργία διαφόρων τύπων γραφημάτων και απεικονίσεων.
import matplotlib.pyplot as plt

# Αρχικοποίηση των υπερπαραμέτρων ρυθμός εκπαίδευσης (learning rate), εποχές (epochs), μέγεθος υποσυνόλου δεδομένων (batch size) και της παραμέτρου νέων διαστάσεων εικόνας (image size). Κάνοντας αλλαγές στις παραμέτρους αυτές, μπορούμε να εκπαιδεύσουμε διαφορετικά μοντέλα και να τα συγκρίνουμε ώστε στο τέλος να κρατήσουμε εκείνο με τα καλύτερα δυνατά αποτελέσματα και το μεγαλύτερο ποσοστό επιτυχίας. 1)Ρυθμός εκπαίδευσης (INIT_LR): Είναι μια υπερπαράμετρος που καθορίζει πόσο γρήγορα ή αργά η συνάρτηση βελτιστοποίησης του σφάλματος που έχουμε επιλέξει (Adam), κατεβαίνει την καμπύλη σφάλματος. Συνήθως η τιμή της βρίσκεται ανάμεσα στο 0.0001 and 0.01. 2)Εποχές (EPOCHS): Είναι μια υπερπαράμετρος η οποία ορίζει τον αριθμό των επαναλήψεων που θα χρειαστεί να εκτελεστούν για την εκπαίδευση του μοντέλου. 3)Μέγεθος υποσυνόλου δεδομένων (BS): Είναι μια υπερπαράμετρος η οποία ομαδοποιεί τα δεδομένα εκπαίδευσης (train_images, train_labels) που χρησιμοποιούμε σε μια εποχή (epoch) για να εκπαιδεύσουμε το νευρωνικό δίκτυο. Συνήθως για τα CNN επιλέγεται το 32. 4)Image size(IMAGE_SIZE): Είναι μια παράμετρος η οποία
```

```
ορίζει τις νέες διαστάσεις που θα έχουν οι εικόνες για την εκπαίδευση του μοντέλου.

INIT_LR = 1e-4
EPOCHS = 20
BS = 32
IMAGE_SIZE = 224


                                """Μέρος 1ο - Data Preprocessing"""


# Παρακάτω ακολουθεί η διαδικασία δημιουργίας μοναδικού φακέλου για το μοντέλο που θα εκπαιδευτεί, ο οποίος θα περιέχει όλες τις απαραίτητες πληροφορίες για αυτό. Αρχικά, δίνεται το όνομα του φακέλου στη μεταβλητή "folder_of_model" που είναι ο συνδυασμός των τριών βασικών υπερπαραμέτρων που θα χρησιμοποιηθούν για την εκπαίδευση του μοντέλου και δηλώθηκαν προηγουμένως. Επίσης το αλφαριθμητικό (string) θα περιέχει το όνομα του φακέλου "models/", που θα περιέχει τον νέο υποφάκελο, σε συνδυασμό με το όνομα του υποφακέλου αυτού, έτσι ώστε να ελεγχθεί παρακάτω η ακριβής τοποθεσία του υποφακέλου.
folder_of_model = f"models/{INIT_LR}_{EPOCHS}_{BS}"

#Έλεγχος εάν υπάρχει ο φάκελος με το όνομα του "folder_of_model" μέσα στον φάκελο "models/".
if os.path.exists(folder_of_model):

# Εάν ο φάκελος υπάρχει τότε εκτυπώνεται ενημερωτικό μήνυμα στο τερματικό, που φαίνεται παρακάτω, όπου το "{folder_of_model[7:]}" θα αντικατασταθεί με την ονομασία αυτού του φακέλου. Το "[7:]" λέγεται string slicing και καταργεί τους πρώτους επτά χαρακτήρες του αλφαριθμητικού "folder_of_model" ώστε να μην εμφανιστεί στην οθόνη το "models/", αλλιώς θα εμφανιζόταν η τοποθεσία του υποφακέλου και όχι η ονομασία του.
  print(f"Αυτή η έκδοση του μοντέλου που προσπαθήσατε να εκπαιδεύσετε ({folder_of_model[7:]}) έχει ήδη δημιουργηθεί."
  f" Το πρόγραμμα θα τερματιστεί τώρα...")
  sys.exit()

#Εκτύπωση ενημερωτικού μηνύματος στην οθόνη.
print(f"[ΕΝΗΜΕΡΩΣΗ] Η εκπαίδευση του μοντέλου με ονομασία έκδοσης '{folder_of_model[7:]}' ξεκίνησε...")

#Δημιουργία του νέου φακέλου μέσα στον φάκελο "models/" με την εντολή "makedirs()" της βιβλιοθήκης os.
os.makedirs(folder_of_model)

# Δημιουργία μιας μεταβλητής (αλφαριθμητικού περιεχομένου) με περιεχόμενο την τοποθεσία των εικόνων, που θα χρησιμοποιηθούν για την εκπαίδευση του μοντέλου.
dataset_location = r"D:\projects\face_mask_detection\dataset"

#Δημιουργία μιας λίστας με δύο περιεχόμενα, το "with_mask" και το "without_mask".
dataset_classes = ["with_mask", "without_mask"]

# Δημιουργία μιας λίστας που αργότερα θα περιέχει όλες τις εικόνες ως αριθμούς και πιο συγκεκριμένα ως διανύσματα.
data = []

# Δημιουργία μιας λίστας, που αργότερα θα περιέχει για κάθε μία από τις εικόνες της λίστας "data" αντίστοιχα έναν αριθμό, ο οποίος θα αντιπροσωπεύει την ετικέτα "with_mask" ή το "without_mask".
labels = []

#Εκτύπωση ενημερωτικού μηνύματος στην οθόνη.
print("[ΕΝΗΜΕΡΩΣΗ] Η φόρτωση των εικόνων ξεκίνησε...")

# Δημιουργία βρόγχου επανάληψης που την πρώτη φορά το "dataset_class" θα έχει την τιμή "with_mask" και τη δεύτερη/τελευταία φορά θα είναι "dataset_class" = "without_mask".
for dataset_class in dataset_classes:

# Δημιουργία μεταβλητής αλφαριθμητικού τύπου η οποία περιέχει το αποτέλεσμα της ένωσης δύο επιμέρους αλφαριθμητικών, του "dataset_location" και του "dataset_class" παραδείγματος χάριν: dataset_location = r"D:\projects\face_mask_detection\dataset\" dataset_class = "with_mask" Άρα path = "D:\projects\face_mask_detection\dataset\with_mask"
```

```
  path = os.path.join(dataset_location, dataset_class)

# Δημιουργία βρόγχου επανάληψης, που σε κάθε επανάληψή του το περιεχόμενο της μεταβλητής "img_name" θα είναι η ονομασία μίας από τις εικόνες του φακέλου που δείχνει το "path". Πιο συγκεκριμένα, το "os.listdir(path)" δημιουργεί μία λίστα με όλες τις ονομασίες των εικόνων που περιέχει ο φάκελος που δείχνει το "path".
  for img_name in os.listdir(path):

# Δημιουργία μεταβλητής αλφαριθμητικού τύπου η οποία περιέχει το αποτέλεσμα της ένωσης δύο επιμέρους αλφαριθμητικών, του "path" και του "img_name" παραδείγματος χάριν: path = r"D:\projects\face_mask_detection\dataset\with_mask\" img_name = "0_0_21.jpg" Άρα img_path = "D:\projects\face_mask_detection\dataset\with_mask\0_0_21.jpg"
    img_path = os.path.join(path, img_name)

# Εφόσον το "img_path" δείχνει στην τοποθεσία του αρχείου μιας συγκεκριμένης εικόνας, το "load_img" φορτώνει στην μεταβλητή "image" την εικόνα αυτή με τις συγκεκριμένες διαστάσεις που ορίζει το "target_size". Άρα με αυτήν την εντολή προσαρμόζονται όλες οι εικόνες έτσι ώστε να έχουν την ίδια διάσταση αλλά με την αναλογία (aspect ratio) που είχαν.
    image = load_img(img_path, target_size=(IMAGE_SIZE, IMAGE_SIZE))

# Μετατροπή της εικόνας που βρίσκεται στο "image" σε μορφή διανύσματος για να μπορεί να επεξεργαστεί αργότερα πιο εύκολα. Ένα κομμάτι του διανύσματος αυτού θα έχει για παράδειγμα την παρακάτω μορφή:
#[84. 58. 45.]
#[84. 58. 45.]
#[84. 58. 45.]
    image = img_to_array(image)

# Επειδή για το CNN μοντέλο μας θα χρησιμοποιήσουμε την αρχιτεκτονική του μοντέλου mobilenet_v2, χρειάζεται να χρησιμοποιήσουμε την εντολή "preprocess_input" πάνω στο διάνυσμα της εικόνας μας. Η εντολή αυτή κάνει κάποιες μετατροπές και προσαρμογές στο διάνυσμα έτσι ώστε αυτό να είναι συμβατό κατά την εκπαίδευση του μοντέλου μας. Λαμβάνοντας υπόψιν το παράδειγμα απεικόνισης ενός μέρους του διανύσματος από την προηγούμενη εντολή μπορούμε να δούμε την διαφορά του παρακάτω, αφού δηλαδή υποστεί την εντολή του "preprocess_input()":
#[-0.34117645 -0.54509807 -0.64705884]
#[-0.34117645 -0.54509807 -0.64705884]
#[-0.34117645 -0.54509807 -0.64705884]
    image = preprocess_input(image)

# Προσθήκη με τη σειρά, όλων των διανυσμάτων των εικόνων στη λίστα "data" ώστε να είναι αποθηκευμένες με αυτή τη μορφή σε ένα μέρος που θα μας διευκολύνει στην μετέπειτα επεξεργασία τους.
    data.append(image)

# Για κάθε μία από τις εικόνες αποθηκεύεται αντίστοιχα και μια ετικέτα με την κατάσταση της εικόνας. Δηλαδή, είτε "with mask" είτε "without_mask". Όλες οι ετικέτες αποθηκεύονται σε μία λίστα, όπως και όλες οι εικόνες στην προηγούμενη εντολή, για να μπορούν να επεξεργαστούν αργότερα τα δεδομένα με μεγαλύτερη ευκολία.
    labels.append(dataset_class)

# Επειδή τα deep learning μοντέλα λειτουργούν σωστά με δεδομένα μέσα σε διανύσματα, οι ετικέτες παρακάτω από λίστα μετατρέπονται και αυτές σε διάνυσμα και ειδικότερα τα δεδομένα του που ήταν προηγουμένως αλφαριθμητικά, πλέον θα είναι αριθμοί. Αρχικά καλείται η κλάση "LabelBinarizer()" η οποία δημιουργεί το αντικείμενο (object) "lb". Αυτό γίνεται για να μπορούμε να χρησιμοποιήσουμε τις μεθόδους της προαναφερόμενης κλάσης πιο εύκολα.
lb = LabelBinarizer()

# Το object "lb" καλεί την μέθοδο "fit_transform" πάνω στη λίστα "labels" και μετατρέπει όλα τα δεδομένα της σε 0 και 1. Δηλαδή το αλφαριθμητικό "with_mask" αντικαταστάθηκε με τον αριθμό 0 και το "without_mask" με τον αριθμό 1. Επίσης το "labels" απο λίστα μετατρέπεται σε διάνυσμα (class numpy.ndarray int32) με δεδομένα της μορφής[0] ή [1].
labels = lb.fit_transform(labels)

# Η μέθοδος "to_categorial" χρησιμοποιεί την κωδικοποίηση One-hot encoding, η οποία μετατρέπει μια λίστα/διάνυσμα που περιέχει κατηγορίες όπως το "labels" σε μορφή τέτοια που μπορεί να χρησιμοποιηθεί εύκολα από αλγόριθμους μηχανικής μάθησης. Η βασική ιδέα της κωδικοποίησης αυτής είναι η δημιουργία νέων μεταβλητών που λαμβάνουν τις τιμές 0 και 1 για να αντιπροσωπεύουν τις αρχικές κατηγορικές τιμές. Το "labels"
```

```
μετά την κωδικοποίηση θα περιέχει δεδομένα της μορφής [1. 0.] για "with_mask" ή [0. 1.] για "without_mask" και θα έχει τα εξής χαρακτηριστικά (class numpy.ndarray float32).
labels = to_categorical(labels)

# Μετατροπή, με τη βοήθεια της βιβλιοθήκης numpy (np), της λίστας "data" που περιέχει όλα τα διάνυσμα των εικόνων, σε ένα διάνυσμα και συγκεκριμένα τύπου float32.
data = np.array(data, dtype="float32")

# Το "labels" επειδή μετατράπηκε προηγουμένως σε διάνυσμα τύπου float32 δεν χρειάζεται θεωρητικά να εκτελεστεί η παρακάτω εντολή, αλλά για λόγους τυπικότητας και ασφάλειας πρέπει να εκτελεστεί.
labels = np.array(labels)

# Η μέθοδος "train_test_split" της βιβλιοθήκης "scikit-learn(sklearn.model_selection)" χωρίζει τα δεδομένα των "data" και "labels" σε τέσσερα διανύσματα δύο κατηγοριών. Ουσιαστικά τα δεδομένα τους θα χωριστούν σύμφωνα με το ποσοστό που ορίζει η ιδιότητα "test_size" όπου στην προκειμένη περίπτωση είναι 0.2 ή αλλιώς 20%. Άρα το 20% των δεδομένων θα αποθηκευτεί στα διανύσματα "test_images" και "test_labels" που αφορούν τη κατηγορία testing (επικύρωσης) και το 80% θα αποθηκευτεί στα "train_images" και "train_labels" της κατηγορίας training. Τα "train_images" και "train_labels" θα χρησιμοποιηθούν αργότερα για την εκπαίδευση του μοντέλου, το οποίο αφού εκπαιδευτεί θα δοκιμαστεί με τα "test_images" και "test_labels", για την απόδοση, την αποτελεσματικότητα και το ποσοστό επιτυχίας του. Η ιδιότητα "stratify" δείχνει στο διάνυσμα των "labels" έτσι ώστε ο διαχωρισμός των δεδομένων στα "train_images", "test_images", "train_labels", "test_labels" να γίνει με ομοιόμορφη κατανομή και να μην έχουμε σφάλματα κατά την εκπαίδευση. Εάν δεν ορίζαμε το "stratify" ως προς το labels, τότε το πρόγραμμα μπορεί να αποθήκευε τυχαία στα training διανύσματα μόνο τα δεδομένα που αντιστοιχούν σε labels=[0. 1] και έτσι αργότερα το μοντέλο να έχει εκπαιδευτεί μόνο για ανθρώπους που δε φοράνε μάσκα. Η ιδιότητα "random_state" παίρνει ως όρισμα έναν αριθμό, ο οποίος συνήθως είναι το 42. Αυτός ο αριθμός ορίζει την τυχαιότητα που θα χωριστούν τα δεδομένα στα διανύσματα "train_images", "test_images", "train_labels", "test_labels". Αυτή η ιδιότητα βοηθάει έτσι ώστε αν μελλοντικά θέλουμε να συγκρίνουμε διαφορετικά μοντέλα μεταξύ τους να είμαστε σίγουροι ότι τα δεδομένα μας χωρίστηκαν με τον ίδιο τρόπο (λογική τυχαιότητας ν. 42) για την εκπαίδευση όλων των υπόλοιπων μοντέλων μας.
(train_images, test_images, train_labels, test_labels) = train_test_split(data, labels, test_size=0.20, stratify=labels, random_state=42)


"""Μέρος 2ο - Data augmentation-Αύξηση δεδομένων"""


# Δημιουργία του αντικειμένου "aug_gen" από τη κλάση "ImageDataGenerator" της ενότητας t"ensorflow.keras.preprocessing.image". Το "ImageDataGenerator()" είναι μια κλάση που βοηθάει στην αύξηση των δεδομένων (data augmentation) και συγκεκριμένα των εικόνων που θα εκπαιδευτεί το μοντέλο. Το βασικό πλεονέκτημα της αύξησης αυτής είναι πως δεν χρειάζεται να αναζητήσει κάποιος χειροκίνητα νέες εικόνες στο διαδίκτυο για να εμπλουτίσει το σύνολο δεδομένων. Η κλάση αυτή λαμβάνει κάθε εικόνα του "train_images" με τη σειρά κατά την εκπαίδευση του μοντέλου, την αντιγράφει μερικές φορές και επεξεργάζεται αυτά τα αντίγραφά της με τέτοιο τρόπο ώστε να φαίνονται σαν να είναι νέες, διαφορετικές εικόνες από το πρωτότυπο. Έτσι το dataset μας θα έχει μεγαλύτερη ποικιλία εικόνων. Οι επεξεργασίες που θα δεχτούν τα αντίγραφα είναι συγκεκριμένες και εξαρτώνται από τα ορίσματα/ιδιότητες της κλάσης αυτής που θα επιλεχθούν.
#Συγκεκριμένα επιλέχθηκαν:
#1) rotation_range=20: Τυχαία περιστροφή των αντιγράφων ανάμεσα στα όρια των -20 με 20 μοιρών.
# 2)zoom_range=0.15: Τυχαία εφαρμογή μεγέθυνσης ή σμίκρυνσης στα αντίγραφα. Όταν η τιμή είναι μικρότερη από 1,0, η εικόνα σμικρύνεται, με αποτέλεσμα να φαίνεται μικρότερη. Αντίθετα, μία τιμή μεγαλύτερη από 1,0 κάνει ζουμ στην εικόνα, κάνοντάς τη να φαίνεται μεγαλύτερη. Για παράδειγμα, εάν το εύρος ζουμ έχει οριστεί σε [0,8, 1,2], οι εικόνες μπορούν να υποστούν τυχαίο ζουμ μεταξύ 80% και 120% του αρχικού τους μεγέθους, είτε δηλαδή να μικραίνουν είτε να μεγαλώνουν. Στην περίπτωσή μας οι εικόνες σμικρύνονται μόνο.
# 3)width_shift_range=0.2: Τυχαία μετατόπιση των αντιγράφων μέχρι και ένα ποσοστό του πλάτους τους. Απο 0% έως 20% στη συγκεκριμένη περίπτωση.
# 4)height_shift_range=0.2: Τυχαία μετατόπιση των αντιγράφων μέχρι και ένα ποσοστό του ύψους τους. Από 0% έως 20% στη συγκεκριμένη περίπτωση.
# 5)shear_range=0.15: Τυχαία διαστρέβλωση των αντιγράφων ως προς τον οριζόντιο ή τον κάθετο άξονα. Η γωνία της διαστρέβλωσης στη συγκεκριμένη περίπτωση έχει ορισθεί από -0.15 έως 0.15 ακτίνια (radians) και όχι μοίρες όπως στην παράμετρο "rotation_range" όπου προαναφέρθηκε.
# 6)horizontal_flip=True: Ενεργοποίηση της οριζόντιας τυχαίας αναστροφής των αντιγράφων. Η αναστροφή γίνεται ως προς τον κατακόρυφο άξονα όπου η αριστερή μεριά της εικόνας μεταφέρεται δεξιά και η δεξιά στα αριστερά. Αυτό είναι χρήσιμο εάν σε κάποιες εικόνες υπάρχει συμμετρία μεταξύ της πάνω και κάτω μεριάς της εικόνας, οπότε έτσι πετυχαίνουμε να κάνουμε ένα αντίγραφο να φαίνεται σαν μία εντελώς διαφορετική εικόνα.
# 7)fill_mode="nearest": Αυτή η παράμετρος είναι αρκετά σημαντική διότι «επιδιορθώνει» τα νέα αντίγραφα
```

που υπέστησαν όλες τις αλλαγές που αναφέραμε στις προηγούμενες παραμέτρους. Ειδικότερα, λόγω των μεταμορφώσεών (transformation) τους, τα αντίγραφα ενδέχεται να περιέχουν κάποια νέα pixel ή να έχουν δημιουργήσει κενές περιοχές. Η παράμετρος "fill_mode" τα διορθώνει, με τη μεθοδολογία "nearest" στη συγκεκριμένη περίπτωση, όπου γεμίζει αυτά τα νέα ή κενά pixel με τιμές χρώματος που έχει το pixel στην αντίστοιχη θέση της αυθεντικής εικόνας.

```
aug_gen = ImageDataGenerator(
rotation_range=20,
zoom_range=0.15,
width_shift_range=0.2,
height_shift_range=0.2,
shear_range=0.15,
horizontal_flip=True,
fill_mode="nearest")


"""Μέρος 3ο - Κατασκευή του μοντέλου με τη μέθοδο TRANSFER LEARNING """


# Δημιουργία ενός από τα δύο μέρη του μοντέλου που θα εκπαιδευτεί. Το πρώτο αυτό μέρος ονομάζεται βασικό μοντέλο (baseModel) και χρησιμοποιεί την αρχιτεκτονική του συνελικτικού νευρωνικού δικτύου MobileNetV2 που έχει ήδη εκπαιδευτεί (pre-trained model) στο παρελθόν. Η μεταβλητή "baseModel" μετατρέπεται σε συνελικτικό νευρωνικό δίκτυο άρα και σε αντικείμενο (object) της ενότητας keras.engine.functional.Functional, με χαρακτηριστικά που δηλώνονται στις παρακάτω ιδιότητες.
# 1)"weights": Εδώ αρχικοποιούνται τα βάρη του νευρωνικού δικτύου όπου επιλέχθηκαν να είναι ίσα με τα προεκπαιδευμένα (pre-trained) βάρη που προέκυψαν από την εκπαίδευση του MobileNetV2 πάνω στο σύνολο δεδομένων του "Imagenet" στο παρελθόν. Το "Imagenet" είναι ένα τεράστιο σύνολο δεδομένων που περιέχει εκκατομύρια ετικέτες εικόνων διαφορετικών κατηγοριών. Εκπαιδεύοντας το μοντέλο μας με αυτά τα βάρη, επιτυγχάνεται μείωση του χρόνου εκπαίδευσης και καλύτερα αποτελέσματα αφού αυτά τα βάρη μπορούν να αναγνωρίζουν κάποια γενικά οπτικά μοτίβα (general visual patterns) μέσα σε μία εικόνα.
# 2)include_top: Με τη boolean τιμή False αφαιρούνται από την κορυφή του συνελικτικού νευρωνικού δικτύου τα κρυφά επίπεδα που περιέχουν τα πλήρως συνδεδεμένα επίπεδα (Fully Connected Layers), τα οποία ήταν υπεύθυνα για την ταξινόμηση (classification) των τελικών αποτελεσμάτων. Αυτά τα συγκεκριμένα κρυφά επίπεδα καταργούνται διότι αργότερα θα προσθέσουμε τα δικά μας κρυφά επίπεδα που θα εκπαιδεύσουμε στο "headModel" και θα ταξινομούν τα δεδομένα στις κατηγορίες που θέλουμε εμείς για την συγκεκριμένη εργασία.
# 3)input_tensor: Εδώ δηλώνονται οι διαστάσεις των εικόνων του συνόλου δεδομένων που θα εκπαιδευτεί το μοντέλο καθώς και το είδος των εικόνων όπου στην συγκεκριμένη περίπτωση θα είναι έγχρωμες τριών επιπέδων (RGB).
baseModel = MobileNetV2(weights="imagenet", include_top=False, input_tensor=Input(shape=(IMAGE_SIZE,
IMAGE_SIZE, 3)))

# Δημιουργία του δεύτερου μέρους του μοντέλου που θα εκπαιδευτεί. Αυτό ονομάζεται "headModel", το οποίο αποτελείται από τα κρυφά επίπεδα που προαναφέρθηκαν και είναι υπεύθυνο για την ταξινόμηση των δεδομένων και την σωστή επιλογή των τελικών αποτελεσμάτων (with mask/without mask). Αρχικά με την εντολή "baseModel.output" δηλώνεται ότι η μεταβλητή "headModel" θα είναι η έξοδος του "baseModel", άρα θα δέχεται τα δεδομένα (εξαγόμενα χαρακτηριστικά μιας εικόνας) που επεξεργάστηκε το "baseModel" και με τη σειρά του θα τα περνάει και αυτό από επεξεργασία για την τελική τους ταξινόμηση. Πιο συγκεκριμένα αυτό το σύνολο δεδομένων που θα δεχτεί το "headModel" ονομάζεται χάρτης χαρακτηριστικών (feature map) και έχει τρισδιάστατη μορφή 7x7x1280. Ο χάρτης χαρακτηριστικών περιέχει πληροφορίες για τα χαρακτηριστικά μιας εικόνας. Επίσης τα επίπεδα που θα χρησιμοποιηθούν βρίσκονται στην ενότητα "tensorflow.keras.layers".
headModel = baseModel.output

# Ενσωμάτωση του "AveragePooling2D" επιπέδου στο "headModel". Αυτό το επίπεδο μετατρέπει την μορφή του χάρτη χαρακτηριστικών από 7x7x1280 σε 1x1x1280 εάν το pool_size επιλεχθεί (7,7). Αυτός ο μετασχηματισμός δέχεται κάθε κανάλι (channel) του 7x7x1280 με τη σειρά και από 49(7x7) pixels τα αλλάζει σε 1(1x1). Συγκεκριμένα υπολογίζεται ο μέσος όρος των τιμών που περιέχουν τα 49 pixels και αυτός αποθηκεύεται στη νέα μορφή του χάρτη χαρακτηριστικών. Κάποια από τα πλεονεκτήματα αυτής της διαδικασίας είναι η μείωση των χωρικών διαστάσεων (spatial dimensions), η μείωση του θορύβου και η βελτίωση της υπολογιστικής απόδοσης.
headModel = AveragePooling2D(pool_size=(7, 7))(headModel)

# Ενσωμάτωση του "Flatten" στο "headModel". Το επίπεδο αυτό δέχεται το αποτέλεσμα του "AveragePooling2D" καλώντας δίπλα από την εντολή "Flatten(name="flatten")" το "(headModel)". Συγκεκριμένα δέχεται το χάρτη χαρακτηριστικών της μορφής 1x1x1280 το οποίο έχει 1280 κανάλια και το μετατρέπει σε ένα διάνυσμα (τύπου vector) μίας μόνο διάστασης, δηλαδή 1x1280. Αυτή η διαδικασία βοηθάει
```

```
κυρίως στη συμβατότητα, αφού από το επόμενο βήμα που θα ακολουθήσουν τα πλήρως συνδεδεμένα επίπεδα, αυτά θα δεχτούν στην είσοδό τους των χάρτη χαρακτηριστικών που θα πρέπει να έχει τη μορφή μονοδιάστατου διανύσματος. Ειδικότερα το επίπεδο "Flatten" παίζει κρίσιμο ρόλο στη μετάβαση από τα επίπεδα του συνελικτικού νευρωνικού δικτύου στα κρυφά επίπεδα του νευρωνικού δικτύου. Ακόμα, κάθε στοιχείο του διανύσματος πλέον θεωρείται ως ένας νευρώνας, άρα τα πλήρως συνδεδεμένα επίπεδα θα δεχτούν 1280 τιμές ως εισόδους σε κάθε νευρώνα που περιέχει το επόμενο επίπεδο.
headModel = Flatten(name="flatten")(headModel)

# Δημιουργία ενός πλήρως συνδεδεμένου επιπέδου. Το "Dense" δέχεται ως είσοδο τις 1280 τιμές του διανύσματος που δημιούργησε το "Flatten" αναγνωρίζοντας αυτές ως αρχικούς νευρώνες και δημιουργεί ένα πλήρως συνδεδεμένο επίπεδο που αποτελείται από 128 νέους νευρώνες. Κάθε νέος νευρώνας από τους 128 δέχεται την τιμή που περιέχει κάθε νευρώνας από τους 1280 την οποία πολλαπλασιάζει με τυχαία βάρη (weights) αφού αυτά θα πάρουν τις τελικές τους σωστές τιμές μετά την εκπαίδευση του τελικού μοντέλου. Στη συνέχεια, όλα αυτά τα σύνολα γινομένων προστίθενται μεταξύ τους και δίνουν μία τιμή ως αποτέλεσμα, η οποία είναι πιθανόν να έχει αρνητική τιμή πέρα από μηδενική ή θετική. Επιπλέον, στο προηγούμενο άθροισμα μπορεί να προστεθεί και μία τιμή που ονομάζεται πόλωση (bias) η οποία επίσης θα αλλάξει με την εκπαίδευση. Επειδή οι τιμές θέλουμε να είναι θετικές χρησιμοποιείται η συνάρτηση ενεργοποίησης Rectified Linear Units (Relu) η οποία ανιχνεύει την τιμή που έχει ο νέος νευρώνας και αν αυτή είναι αρνητική την μετατρέπει σε μηδέν. Σε κάθε άλλη περίπτωση η τιμή παραμένει ίδια.
headModel = Dense(128, activation="relu")(headModel)

# Εφαρμογή της μεθόδου "Dropout" στις τιμές που παρήγαγαν οι νευρώνες του προηγούμενου επιπέδου. Το dropout είναι μία τεχνική τακτοποίησης (regularization technique) που απορρίπτει τυχαία το 50% των εξόδων των νευρώνων από το προηγούμενο επίπεδο κατά τη διάρκεια της εκπαίδευσης του τελικού μοντέλου. Αυτό βοηθά στην αποφυγή της υπερπροσαρμογής (overfitting) που είναι η κατάσταση στην οποία το μοντέλο έχει καταφέρει να εκπαιδευτεί πολύ καλά πάνω στο dataset που ορίστηκε για την εκπαίδευση του, με αποτέλεσμα να μην ανταποκρίνεται σωστά σε άλλα δεδομένα που του δίνονται. Πιο συγκεκριμένα, με το "Dropout" γίνεται αποφυγή της υπερεκπαίδευσης κάνοντας το δίκτυο να μη βασίζεται μόνο σε συγκεκριμένους νευρώνες αλλά να εκπαιδεύεται με πιο γενικευμένα χαρακτηριστικά (generalized features).
headModel = Dropout(0.5)(headModel)

# Δημιουργία του δεύτερου και τελευταίου πλήρως συνδεδεμένου επιπέδου. Το "Dense" αυτή τη φορά δέχεται στην είσοδό του τις εξόδους του προηγούμενου πλήρως συνδεδεμένου επιπέδου με τους 128 νευρώνες, των οποίων οι μισές τιμές θα είναι πλέον μηδενικές λόγω του "Dropout". Το "Dense" εδώ δημιουργεί τους δύο τελικούς νευρώνες του δικτύου και αναλόγως την τιμή τους το μοντέλο θα έχει πάρει την τελική απόφαση, δηλαδή εάν η εικόνα που επεξεργάστηκε ανήκει στην κατηγορία "with mask" ή "without mask". Όπως και στο προηγούμενο πλήρως συνδεδεμένο επίπεδο, έτσι και εδώ κάθε ένας από τους δύο νευρώνες θα λαμβάνει στην είσοδό του όλες τις εξόδους των προηγούμενων 128 νευρώνων. Αυτές οι τιμές θα πολλαπλασιάζονται και εδώ με κάποια βάρη και έπειτα υπολογίζεται το άθροισμα όλων αυτών των γινομένων και ενδεχομένως η πρόσθεση σε αυτό το άθροισμα μίας τιμής του bias. Έτσι οι δύο νευρώνες θα έχουν από μία τιμή ο καθένας, η οποία όμως δεν θα έχει τη σωστή μορφή οπότε θα πρέπει να την μετατρέψουμε σε μορφή πιθανότητας, δηλαδή ανάμεσα στο 0 και το 1. Αυτό το επιτυγχάνουμε με την συνάρτηση ενεργοποίησης "softmax" η οποία προσαρμόζει τις τιμές στο πεδίο που θέλουμε[0-1]. Η πιθανότητα αυτή θα είναι το τελικό αποτέλεσμα που θα κρίνει το μοντέλο εάν κατάφερε να αναγνωρίσει την κατάσταση της εικόνας που δέχτηκε σαν είσοδο κατά την εκπαίδευση και ποια είναι η πιθανότητα/το ποσοστό που το κατάφερε. Πρέπει να αναφερθεί πως η "softmax" επηρεάζει και τις δύο τιμές των τελικών νευρώνων, οι οποίες η μία συμπληρώνει την άλλη και έχουν άθροισμα το 1. Αναλόγως την πιθανότητα και το ποσοστό επιτυχίας του, το νευρωνικό δίκτυο ενημερώνει τα βάρη και τις πολώσεις του σε κάθε επανάληψη εκπαίδευσης του, με σκοπό να μειώσει όσο περισσότερο μπορεί το ποσοστό λάθους.
headModel = Dense(2, activation="softmax")(headModel)

#Παρακάτω φαίνονται αναλυτικά όλα τα επίπεδα του headModel και τα χαρακτηριστικά τους:
# KerasTensor(type_spec=TensorSpec(shape=(None, 7, 7, 1280), dtype=tf.float32, name=None),
name='out_relu/Relu6:0', description="created by layer 'out_relu'")
# KerasTensor(type_spec=TensorSpec(shape=(None, 1, 1, 1280), dtype=tf.float32, name=None),
name='average_pooling2d/AvgPool:0', description="created by layer 'average_pooling2d'")
# KerasTensor(type_spec=TensorSpec(shape=(None, 1280), dtype=tf.float32, name=None),
name='flatten/Reshape:0', description="created by layer 'flatten'")
# KerasTensor(type_spec=TensorSpec(shape=(None, 128), dtype=tf.float32, name=None), name='dense/Relu:0',
description="created by layer 'dense'")
# KerasTensor(type_spec=TensorSpec(shape=(None, 128), dtype=tf.float32, name=None),
name='dropout/Identity:0', description="created by layer 'dropout'")
# KerasTensor(type_spec=TensorSpec(shape=(None, 2), dtype=tf.float32, name=None),
name='dense_1/Softmax:0', description="created by layer 'dense_1'")
```

```
# Ενοποίηση των δύο επιμέρους νευρωνικών δικτύων που δημιουργήθηκαν σε 1 (model), δηλαδή του "baseModel" που είναι το CNN για την δημιουργία του χάρτη χαρακτηριστικών από τις εικόνες του training dataset και του "headModel" που περιέχει τα κρυφά επίπεδα, τα οποία είναι υπεύθυνα για την ταξινόμηση του χάρτη χαρακτηριστικών στις κατηγορίες "mask" ή "without mask". Μετά την σύνδεση του "headModel" στην κορυφή του "baseModel" το ολοκληρωμένο νευρωνικό δίκτυο που θα εκπαιδεύσουμε θα έχει την ονομασία "model". Η ένωση επιτυγχάνεται με την εντολή "Model()" που βρίσκεται στην ενότητα "tensorflow.keras.models".
model = Model(inputs=baseModel.input, outputs=headModel)

# Απενεργοποίηση της δυνατότητας εκπαίδευσης όλων των επιπέδων του "baseModel" αφού είναι ήδη εκπαιδευμένο και δεν θέλουμε να υποστεί κάποια αλλαγή στα βάρη ή γενικότερα στις παραμέτρους του. Αντίθετα, το "headModel" είναι αυτό που θα εκπαιδευτεί εξ' ολοκλήρου και από την αρχή γιατί εκείνο αφορά την ταξινόμηση στις δύο κλάσεις που επιθυμούμε για αυτή τη διπλωματική εργασία.
for layer in baseModel.layers:
	layer.trainable = False

# Δημιουργία του αντικειμένου "adam_optim" το οποίο θα περιέχει μία παραλλαγή του αλγορίθμου βελτιστοποίησης gradient descent (κάθοδος βασισμένη στην κλίση) που ονομάζεται Adam (Adaptive Moment Estimation). Κατά την εκπαίδευση του μοντέλου, στο τέλος κάθε επανάληψης υπολογίζεται η συνάρτηση απωλειών (loss function) συγκρίνοντας το αποτέλεσμα που υπολογίστηκε σε σχέση με το πραγματικό αποτέλεσμα. Έπειτα υπολογίζονται οι κλίσεις (gradients) για κάθε παράμετρο της συνάρτησης απωλειών με μία μέθοδο που ονομάζεται οπισθοδιάδοση (backprobagation). Αυτή η μέθοδος περιλαμβάνει τον υπολογισμό των μερικών παραγώγων όλων των παραμέτρων (weights και bias) πάνω στη συνάρτηση απωλειών χρησιμοποιώντας τον κανόνα της αλυσίδας (chain rule). Αφού υπολογιστεί η συνάρτηση απωλειών και οι κλίσεις, τα δέχεται ο Adam σαν δεδομένα και με τη σειρά του υπολογίζει τις νέες τιμές που θα πάρουν οι παράμετροι κατά την επόμενη επανάληψη της εκπαίδευσης. Ο Adam συνδυάζει ιδέες τόσο από μεθόδους που βασίζονται στην ορμή (momentum) όσο και από μεθόδους προσαρμοστικού ρυθμού εκπαίδευσης (initial learning rate) για να ενημερώσει τις παραμέτρους ενός μοντέλου. Επίσης σαν είσοδο ο Adam δέχεται την υπερπαράμετρο "INIT_LR" που έχουμε ορίσει στην αρχή του κώδικα και του δείχνει το μέγεθος του βήματος που πρέπει να κάνει για την διόρθωση του σφάλματος σε κάθε επανάληψη κατά την εκπαίδευση.
adam_optim = Adam(learning_rate=INIT_LR)

#Εκτύπωση ενημερωτικού μηνύματος στην οθόνη.
print("[ΕΝΗΜΕΡΩΣΗ] Γίνεται μεταγλώττιση του μοντέλου...")

# Μεταγλώττιση (compile) και προετοιμασία του μοντέλου πριν από την εκπαίδευση. Εδώ το μοντέλο δέχεται την τελευταία ενημέρωσή του, όπου δίνονται πληροφορίες για το είδος της συνάρτησης απωλειών (binary_crossentropy) που επιθυμούμε να χρησιμοποιήσει, τον αλγόριθμο βελτιστοποίησης (Adam) καθώς και το είδος των μετρήσεων (metrics) που επιθυμούμε να υπολογίζονται και να εμφανίζονται στην οθόνη κατά την διαδικασία εκπαίδευσης. Η συνάρτηση απώλειας "binary_crossentropy" μετρά την απόκλιση μεταξύ του προβλεπόμενου αποτελέσματος του μοντέλου και του πραγματικού στόχου. Τα "metrics" έχουν οριστεί ως "accuracy" για να γίνεται ενημέρωση ως προς την ακρίβεια του μοντέλου σε κάθε επανάληψη. Επίσης, λόγω του "metrics=["accuracy"]" αργότερα θα αποθηκευτούν στην μεταβλητή "history" πληροφορίες για κάθε επανάληψη εκπαίδευσης σχετικά: 1)Με τις απώλειες κατά την εκπαίδευση με το training set (loss) 2)Με την ακρίβεια κατά την εκπαίδευση με το training set (accuracy) 3)Με τις απώλειες κατά την εκπαίδευση με το testing set (val_loss) 4)Με την ακρίβεια κατά την εκπαίδευση με το testing set (val_accuracy)

model.compile(loss="binary_crossentropy", optimizer=adam_optim, metrics=["accuracy"])


"""Μέρος 4ο - Εκπαίδευση του μοντέλου """


#Εκτύπωση ενημερωτικού μηνύματος στην οθόνη.
print("[ΕΝΗΜΕΡΩΣΗ] Η εκπαίδευση του μοντέλου (head μέρος) ξεκίνησε...")

# Εκπαίδευση του head μέρους του μοντέλου, δηλαδή του "headModel" που περιέχει τα κρυφά επίπεδα, με την μέθοδο ".fit()". Κάποιες πληροφορίες σχετικά με την εκπαίδευση θα αποθηκευτούν στην μεταβλητή "HISTORY" όπου θα περιέχει κατηγορίες δεδομένων που ορίστηκαν κατά τη μεταγλώττιση του μοντέλου με το "metrics=["accuracy"]". Κατά την κλήση της μεθόδου ".fit()" δίνονται τιμές σε κάποιες ιδιότητες προκειμένου η εκπαίδευση να γίνει με συγκεκριμένο τρόπο.
# 1) Αρχικά καλείται η μέθοδος ".flow()" πάνω στο αντικείμενο "aug_gen" στην οποία εισάγονται τα δεδομένα που θα εκπαιδευτεί το μοντέλο (training set) μαζί με το μέγεθος υποσυνόλου δεδομένων (BS) που έχουμε ορίσει στην αρχή του κώδικα και είναι ο αριθμός που θα χωρίσει αυτό το σύνολο δεδομένων σε ομάδες για την
```

```
καλύτερη δυνατή εκπαίδευση. Το "aug_gen" δίνει πληροφορίες στην μέθοδο ".flow" σχετικά με τις δυνατότητες τροποποίησης των εικόνων, όπως του είχαν ορισθεί, και η ".flow()" εκτελεί την αύξηση των δεδομένων (data augmentation) εφαρμόζοντάς τους αυτές τις τροποποιήσεις. Έτσι θα δεχτεί το ".fit()" τα νέα "train_images" και "train_labels" set τα οποία θα περιέχουν μεγαλύτερη ποικιλία και πλήθος εικόνων.
# 2) Η δεύτερη ιδιότητα είναι η "steps_per_epoch" στην οποία δίνουμε το ακέραιο αποτέλεσμα της διαίρεσης του αριθμού όλων των εικόνων του training set με το "BS". Αυτό διασφαλίζει ότι ολόκληρο το training set θα χρησιμοποιείται σε κάθε επανάληψη. Πιο συγκεκριμένα, αυτή η ιδιότητα καθορίζει τον αριθμό των βημάτων (steps/batches) που πρέπει να υποστούν επεξεργασία σε κάθε επανάληψη.
# 3) Στην τρίτη ιδιότητα εισάγονται τα δεδομένα του testing set. Το testing set περιέχει εικόνες που το νευρωνικό δίκτυο δεν έχει ξαναδεί κατά την εκπαίδευση, οπότε γίνεται έλεγχος της απόδοσής του καθώς και παρακολούθηση της ικανότητας γενίκευσής(generalization ability) του.
# 4) Η λογική της τέταρτης ιδιότητας είναι ίδια με αυτή της δεύτερης με την διαφορά ότι εδώ λαμβάνονται υπόψιν τα δεδομένα του testing set και όχι του training set. Δηλαδή για την ιδιότητα υπολογίζεται το ακέραιο αποτέλεσμα της διαίρεσης του αριθμού όλων των εικόνων του testing set με το "BS". Και εδώ γίνεται διασφάλιση του ότι θα χρησιμοποιηθούν όλα τα δεδομένα του testing set σε κάθε επανάληψη.
# 5) Στην πέμπτη και τελευταία ιδιότητα δίνεται η τιμή που περιέχει η υπερπαράμετρος "EPOCHS" που ορίστηκε στην αρχή του κώδικα. Αυτό σημαίνει ότι το μοντέλο θα επαναλάβει τη διαδικασία εκπαίδευσης, πάνω σε ολόκληρο το training set, τόσες φορές όσο η τιμή των "EPOCHS" (εποχές). Κάθε εποχή αποτελείται από βήματα που ορίζονται από τα "step_per_epoch" για εκπαίδευση (training) και "validation_steps" για έλεγχο (testing).
HISTORY = model.fit( aug_gen.flow(train_images, train_labels, batch_size=BS), steps_per_epoch=len(train_images) // BS, validation_data=(test_images, test_labels), validation_steps=len(test_images) // BS, epochs=EPOCHS)

#Εκτύπωση ενημερωτικού μηνύματος στην οθόνη.
print("[ΕΝΗΜΕΡΩΣΗ] Αποθήκευση του μοντέλου ανίχνευσης μάσκας στον φάκελο...")

# Αποθήκευση του μοντέλου στον υπολογιστή σε μορφή ".h5" αρχείου. Αυτό γίνεται για να μην χρειαστεί να εκπαιδευτεί ξανά από την αρχή το μοντέλο, κάθε φορά που το χρειαζόμαστε, κάτι το οποίο είναι συνήθως χρονοβόρο. Επίσης το αποθηκευμένο μοντέλο μπορεί να κληθεί μέσα από κάποιο άλλο πρόγραμμα και να χρησιμοποιηθεί όποτε είναι αναγκαίο με την εντολή "load_model" από την ενότητα "tensorflow.keras.models".
model.save(f"{folder_of_model}/mask_detection_model.h5")


"""Μέρος 5ο - Μετατροπή του μοντέλου από ".h5" σε .tflite """


#Εκτύπωση ενημερωτικού μηνύματος στην οθόνη.
print("[ΕΝΗΜΕΡΩΣΗ] Μετατροπή του μοντέλου από .h5 σε μορφή .tflite...")

# Σε αυτό το σημείο ξεκινά η διαδικασία μετατροπής του μοντέλου που δημιουργήθηκε σε πιο ελαφριά έκδοση, δηλαδή σε μορφή ".tflite". Τα μοντέλα ".tflite" φτιάχνονται για να λειτουργούν πιο γρήγορα και πιο αποτελεσματικά στα λειτουργικά συστήματα των συσκευών άκρης (edge devices), όπως για παράδειγμα ένα Raspberry Pi 4. Γι' αυτό ένα μοντέλο ".tflite" εάν χρησιμοποιηθεί π.χ. σε υπολογιστή με Windows 10, θα αργεί περισσότερο τον κώδικα σε αντίθεση με ένα ".h5" μοντέλο, παρόλο που το πρώτο σαν αρχείο είναι πολύ πιο ελαφρύ. Από την άλλη, ένα ".tflite" μοντέλο εκτελεί τον κώδικα πολύ πιο γρήγορα σε ένα raspberry pi απ' ότι ένα ".h5" μοντέλο. Επίσης πρέπει να αναφερθεί πως ένα μοντέλο δεν μπορεί να δημιουργηθεί κατευθείαν σε μορφή .tflte αλλά μπορεί μόνο να μετατραπεί σε αυτό αφού πρώτα δημιουργηθεί το μοντέλο σε μορφή ".h5" ή μορφή φακέλου. Η βασική διαφορά της μορφής ".h5"  από τη μορφή φακέλου είναι πως στην πρώτη όλα τα αρχεία που δημιουργούνται και αποτελούν το μοντέλο τοποθετούνται μέσα σε ένα μόνο αρχείο με κατάληξη ".h5", ενώ στην δεύτερη μορφή όλα αυτά τα αρχεία τοποθετούνται σε έναν φάκελο. Έτσι η μορφή ".h5" είναι πιο βολική στην περίπτωση της συγκεκριμένης διπλωματικής εργασίας. Αρχικά γίνεται φόρτωση του μοντέλου που αποθηκεύτηκε στη μεταβλητή "loaded_model" για χρήση του στα επόμενα βήματα και τη μετατροπή του σε μορφή ".tflite". Η φόρτωση γίνεται καλώντας την μέθοδο "load_model" της ενότητας "tensorflow.keras.models" και δίνοντάς της ως όρισμα την διεύθυνση της τοποθεσίας που έχει αποθηκευτεί το αρχείο του μοντέλου στον υπολογιστή.
loaded_model = tensorflow.keras.models.load_model(f"{folder_of_model}/mask_detection_model.h5")

# Δημιουργία της μεταβλητής/του αντικειμένου "converter" που θα περιέχει όλες τις πληροφορίες σχετικά με την μετατροπή. Συγκεκριμένα το "converter" ενημερώνεται σχετικά με το μοντέλο που θα γίνει η επεξεργασία, τα βάρη του κλπ. Η βιβλιοθήκη που χρησιμοποιείται είναι η "tensorflow.lite" που περιέχει διάφορες συναρτήσεις χρήσιμες, είτε για την δημιουργία ενός ".tflite" αρχείου, είτε για την επεξεργασία του.
converter = tensorflow.lite.TFLiteConverter.from_keras_model(loaded_model)

# Το "converter.optimizations" ενεργοποιεί τη βελτιστοποίηση του νέου μοντέλου, που είναι μία διαδικασία
```

```
πολύ σημαντική και χρήσιμη. Ενώ είναι προαιρετική εντολή, είναι πρακτικά απαραίτητη αφού μειώνει το μέγεθος του μοντέλου αρκετά σε σχέση με το αρχικό μοντέλο ".h5" και το κάνει πιο γρήγορο. Χωρίς αυτή την εντολή, το νέο μοντέλο πάλι θα ήταν πιο ελαφρύ, αλλά όχι τόσο πολύ.
converter.optimizations = [tensorflow.lite.Optimize.DEFAULT]

# Μετατροπή του μοντέλου από την μορφή ".h5" σε μορφή ".tflite" συμβατή με το Raspberry Pi χρησιμοποιώντας την μέθοδο ".convert()" πάνω στο αντικείμενο "converter". Όλες οι πληροφορίες για το νέο μοντέλο αποθηκεύονται προσωρινά στη μεταβλητή "tflite_model".
tflite_model = converter.convert()

# Αποθήκευση του νέου μοντέλου με όνομα "mask_detection_model_optim.tflite" σε μορφή αρχείου, στον φάκελο που έχει δημιουργηθεί ήδη για την συγκεκριμένη εκτέλεση του κώδικα.
open(f"{folder_of_model}/mask_detection_model_optim.tflite", "wb").write(tflite_model)


"""Μέρος 6ο - Αξιολόγηση του μοντέλου """


#Εκτύπωση ενημερωτικού μηνύματος στην οθόνη.
print("[ΕΝΗΜΕΡΩΣΗ] Αξιολόγηση του νευρωνικού δικτύου...")

# Πρόβλεψη αποτελεσμάτων του μοντέλου που εκπαιδεύτηκε χρησιμοποιώντας το testing set. Η μέθοδος "predict()" εισάγει τις εικόνες του testing set χωρίζοντας αυτές σε ίσες ομάδες (Batches) και αποθηκεύει στη μεταβλητή "predictions" τις προβλέψεις ως πιθανότητες.
predictions = model.predict(test_images, batch_size=BS)

# Δημιουργία διανύσματος, που θα περιέχει τις ετικέτες του testing set, σε μορφή τέτοια έτσι ώστε να την επεξεργαστεί παρακάτω η μέθοδος "roc_curve" χωρίς προβλήματα συμβατότητας.
test_labels_binary = np.argmax(test_labels, axis=1)

# Δημιουργία διανύσματος, που θα περιέχει τις προβλέψεις που έγιναν για το μοντέλο πάνω στις εικόνες του testing set, σε μορφή τέτοια έτσι ώστε να το επεξεργαστεί παρακάτω η μέθοδος "roc_curve" χωρίς προβλήματα συμβατότητας.
predictions_binary = np.argmax(predictions, axis=1)

# Υπολογισμός των FPR (False Positive Rate), TPR (True Positive Rate), και thresholds (τα αντίστοιχα κατώφλια τους) χρησιμοποιώντας την μέθοδο "roc_curve()" της βιβλιοθήκης NumPy. Τα ορίσματα που δέχεται η μέθοδος είναι οι μεταβλητές "test_labels_binary" και "predictions_binary" που δημιουργήθηκαν παραπάνω και περιέχουν τις εικόνες του testing set καθώς και τις ετικέτες τους σε δυαδική μορφή. Τα FPR, TPR και τα thresholds τους θα δημιουργήσουν παρακάτω στον κώδικα το διάγραμμα της καμπύλης ROC (Receiver Operating Characteristic).
fpr, tpr, thresholds = roc_curve(test_labels_binary, predictions_binary)

# Υπολογισμός του AUC (Area Under the ROC Curve), που είναι η περιοχή κάτω από την ROC καμπύλη, από τα "fpr" και "tpr" χρησιμοποιώντας την μέθοδο "auc()" της βιβλιοθήκης scikit-learn. Ο αριθμός ή βαθμολογία AUC δείχνει την απόδοση του μοντέλου.
auc_score = auc(fpr, tpr)

# Η μέθοδος ".argmax()" της βιβλιοθήκης numpy, μετατρέπει τις προβλεπόμενες πιθανότητες, του "predictions" σε ετικέτες κλάσεων (class labels) επιλέγοντας τον δείκτη (index) με την υψηλότερη πιθανότητα για κάθε πρόβλεψη. Ο άξονας που αναζητά η μέθοδος τον δείκτη της υψηλότερης πιθανότητας είναι ο νούμερο ένα, δηλαδή οι γραμμές (row) όπου κάθε γραμμή είναι και μία πρόβλεψη. Για παράδειγμα, έστω ότι το "predictions" περιέχει τα δεδομένα [0.3 0.7] στην πρώτη του γραμμή. Αυτό σημαίνει πως για την πρώτη εικόνα της πρώτης ομάδας (batch) του "test_images" έγινε η πρόβλεψη με αποτέλεσμα 0.3 (30%) πιθανότητα για το "with_mask" και 0,7(70%) πιθανότητα για το "without_mask". Έτσι το "argmax()" θα επιστρέψει τη θέση του δείκτη με τη μεγαλύτερη πιθανότητα που στην προκειμένη περίπτωση θα ήταν ο 1 και όχι ο 0, αφού στη γλώσσα προγραμματισμού Python η πρώτη στήλη αριθμείται ως μηδέν. Επίσης το "predictions" μετά την εκτέλεση της εντολής "np.argmax(predictions, axis=1)" μετατρέπεται από ένα δισδιάστατο διάνυσμα (n γραμμών και 2 στηλών), σε μονοδιάστατο (1 γραμμής και n στηλών).
predictions = np.argmax(predictions, axis=1)

# Εμφάνιση στην οθόνη του τερματικού μίας αναφοράς ταξινόμησης (classification report) που παρέχει μετρήσεις όπως η ανάκληση (recall), η βαθμολογία F1 (F1-score), η ακρίβεια (precision), η ορθότητα (accuracy),ο μέσος όρος (macro average) αυτών σε όλες τις κλάσεις καθώς και ο σταθμισμένος μέσος όρος
```

```
(weighted average) αυτών. Για τη διαδικασία παραγωγής όλων των παραπάνω πληροφοριών είναι υπεύθυνη η εντολή "classification_report()" της ενότητας "sklearn.metrics". Αυτή η εντολή δέχεται αρχικά ως δεδομένα τις πραγματικές τιμές των ετικετών που αντιστοιχούν στο "test_images" σύνολο δεδομένων. Τα δεδομένα αυτά βρίσκονται θεωρητικά στο διάνυσμα "test_labels" αλλά επειδή αυτά τα δεδομένα είναι κωδικοποιημένα με τη μορφή one-hot encoding, θα πρέπει πρώτα να μετατραπούν στην μορφή μονοδιάστατου διανύσματος. Για αυτήν τη διαδικασία χρησιμοποιείται η μέθοδος ".argmax()" η οποία συμπεριφέρεται με τον ίδιο τρόπο που εκτελέστηκε και για το "predictions" προηγουμένως. Πιο συγκεκριμένα, η ".argmax()" θα ελέγξει κάθε γραμμή του διανύσματος "test_labels" αφού το "axis" έχει ορισθεί ως 1 και θα επιστρέψει τον δείκτη με τη μεγαλύτερη τιμή. Παίρνοντας ως παράδειγμα μία σειρά του "test_labels" που έχει τη μορφή [0. 1.] αυτή θα ελεγχθεί και θα επιστρέψει την τιμή 1 στο νέο μονοδιάστατο διάνυσμα του "test_labels", αφού μεταξύ του 0 και 1, μεγαλύτερο είναι το 1 που βρίσκεται στη στήλη 1. Ακόμα, το "classification_report" δέχεται ως δεδομένα τις τιμές "predictions" που προβλέφθηκαν και προορίζονται για σύγκριση με τις πραγματικές τιμές του "test_labels". Τέλος ενεργοποιείται η ιδιότητα "target_names" στην οποία δίνεται η τιμή lb.classes_ που επιστρέφει το διάνυσμα με τις αλφαριθμητικές ονομασίες των δύο ομάδων από ετικέτες (with_mask και without_mask). Αυτές δίνονται στο "classification_report" έτσι ώστε αυτό να προβάλλει τις πληροφορίες με περισσότερη αναγνωσιμότητα. Παρακάτω αναλύονται οι μετρικές αξιολόγησης που θα εκτυπωθούν:
# 1) ακρίβεια (precision): Υπολογίζεται ως η αναλογία μεταξύ του αριθμού των Θετικών δειγμάτων (π.χ. with_mask) που ταξινομήθηκαν σωστά προς τον συνολικό αριθμό των δειγμάτων που ταξινομήθηκαν ως Θετικά (είτε σωστά είτε λανθασμένα) και μετρά την ακρίβεια του μοντέλου στην ταξινόμηση ενός δείγματος ως θετικού. Έτσι, προβάλλει πόσο αξιόπιστο είναι το μοντέλο στην ταξινόμηση των δειγμάτων ως Θετικών.
# 2) ανάκληση (recall):  Υπολογίζεται ως η αναλογία μεταξύ του αριθμού των Θετικών δειγμάτων (π.χ. with_mask) που ταξινομήθηκαν σωστά ως Θετικά προς τον συνολικό αριθμό των Θετικών δειγμάτων. Η ανάκληση μετράει την ικανότητα του μοντέλου να ανιχνεύει Θετικά δείγματα. Όταν η ανάκληση είναι υψηλή τότε το μοντέλο μπορεί να ταξινομήσει σωστά όλα τα θετικά δείγματα ως Θετικά και θεωρείται αξιόπιστο ως προς την ικανότητά του να ανιχνεύει θετικά δείγματα.
# 3) βαθμολογία F1 (F1-score): Μετρά την ακρίβεια ενός μοντέλου συνδυάζοντας τις τιμές της ανάκλησης και της ακρίβειας που έχουν ήδη υπολογιστεί. Ειδικότερα, παρέχει μία ισορροπημένη μέτρηση της απόδοσης του μοντέλου, ειδικά όταν υπάρχει ανισορροπία μεταξύ του αριθμού των δειγμάτων σε διαφορετικές κλάσεις.
# 4) Ορθότητα (accuracy): Υπολογίζεται ως η αναλογία μεταξύ του αριθμού των σωστών προβλέψεων (θετικών και αρνητικών σωστών δειγμάτων) προς τον συνολικό αριθμό των προβλέψεων και περιγράφει την απόδοση του μοντέλου σε όλες τις κλάσεις. Γενικότερα μετρά τη συνολική ορθότητα των προβλέψεων του μοντέλου.
#5) Μακρύς μέσος όρος (macro average): Βρίσκει τον μέσο όρο κάθε παραπάνω μετρικού μεταξύ όλων των κλάσεων.
# 6) Σταθμισμένος μέσος όρος (weighted average): Και αυτός υπολογίζει τον μέσο όρο τον μετρικών αλλά δίνει βαρύτητα στην ποσότητα των δεδομένων κάθε κλάσης έτσι ώστε να γίνει μια πιο δίκαιη μέτρηση.
report = classification_report(test_labels.argmax(axis=1), predictions, target_names=lb.classes_)
print(report)

# Δημιουργία ενός νέου αρχείου κειμένου με όνομα "classification_report.txt" και άνοιγμά του για εγγραφή πληροφοριών μέσα σε αυτό.
with open(f'{folder_of_model}/classification_report.txt', 'w') as file:
  #Αποθήκευση των πληροφοριών που παρήχθησαν από το "classification_report" σε αρχείο .txt .
  file.write(report)
  #Κλείσιμο του αρχείου αφού τελείωσε η επεξεργασία του.
  file.close()

                    """Μέρος 7ο - Δημιουργία διαγραμμάτων loss, accuracy και ROC"""

#Εκτύπωση ενημερωτικού μηνύματος στην οθόνη.
print("[ΕΝΗΜΕΡΩΣΗ] Η γραφική απεικόνιση των μετρήσεων ξεκίνησε...")

# Υπολογισμός των τελικών τιμών απωλειών και ορθοτήτων για την γραφική απεικόνισή τους πάνω στα διαγράμματα που θα φτιαχτούν. Συνολικά θα δημιουργηθούν τέσσερις μεταβλητές αφού υπάρχουν δύο τελικές τιμές για τις απώλειες, μια για τα δεδομένα εκπαίδευσης (training set) και μία για τα δεδομένα ελέγχου/επικύρωσης (testing/validating set), καθώς και δύο για την ορθότητα, μια για τα δεδομένα εκπαίδευσης και μια για αυτά της επικύρωσης. Οι τιμές αυτές υπολογίζονται χρησιμοποιώντας το αντικείμενο "HISTORY" που δημιουργήθηκε κατά την εκπαίδευση του μοντέλου και εκτελώντας πάνω του την μέθοδο ".history" με τα αντίστοιχα ορίσματα (loss, accuracy, val_loss, val_accuracy). Επίσης δίπλα από κάθε όρισμα υπάρχει το [-1] που δηλώνει ποια από όλες τις τιμές της κάθε μέτρησης θέλουμε να υπολογίσουμε. Επειδή η κάθε μέτρηση έχει τη μορφή διανύσματος, για ένα διάνυσμα το [-1] σημαίνει η τελευταία τιμή του στη γλώσσα προγραμματισμού Python.
final_train_loss = HISTORY.history["loss"][-1]
final_train_acc = HISTORY.history["accuracy"][-1]
final_val_loss = HISTORY.history["val_loss"][-1]
```

```
final_val_acc = HISTORY.history["val_accuracy"][-1]

#Διαδικασία δημιουργίας του 1ου διαγράμματος που θα περιέχει τις απώλειες εκπαίδευσης και τις απώλειες επικύρωσης.

# Επιλογή του στυλ (style) που θα έχει το διάγραμμα, δηλαδή η φωτεινότητά του, το χρώμα στο παρασκήνιο, τα χρώματα των γραμμών κ.λπ. Εδώ επιλέχθηκε το στυλ "bmh" (Bayesian Methods for Hackers), διότι παρουσιάζει τα δεδομένα στα γραφήματα με μεγαλύτερη ευκρίνεια και πιο ωραίο τρόπο. Όλες οι μέθοδοι που θα χρειαστούν για τη δημιουργία του διαγράμματος βρίσκονται στη βιβλιοθήκη "matplotlib" ή αλλιώς "plt" όπως ονομάστηκε για συντομία. Οπότε για το στυλ χρησιμοποιήθηκε η μέθοδος ".style()" με όρισμα το "bmh" στυλ.
plt.style.use("bmh")

# Ορισμός μεγέθους του πλαισίου (figure) που θα περιέχει το διάγραμμα και συγκεκριμένα με τη μέθοδο ".figure()" και ορίσματα "8,6". Αυτά τα ορίσματα μετριούνται σε ίντσες και επιλέχθηκαν μετά από διάφορες δοκιμές διότι εμφάνιζαν καλύτερα τα δεδομένα και τους άξονες.
plt.figure(figsize=(8, 6))

# Εισαγωγή των δεδομένων της μέτρησης "loss" (που αφορά την εκπαίδευση) στον άξονα y του διαγράμματος και στις τιμές του άξονα x τοποθετούνται οι αριθμοί 1 έως "EPOCHS", δηλαδή κάθε εποχή/επανάληψη εκπαίδευσης. Έτσι για κάθε εποχή απεικονίζεται η αντίστοιχη τιμή του "loss". Επιπλέον δίνεται μία ονομασία για τα δεδομένα αυτά με την ιδιότητα "label" και όνομα το "training loss".
plt.plot(np.arange(1, EPOCHS+1), HISTORY.history["loss"], label="training loss")

# Εδώ γίνεται ακριβώς η ίδια διαδικασία με την προηγούμενη εντολή, με τη διαφορά ότι προσθέτουμε στον άξονα y του διαγράμματος τα δεδομένα της μέτρησης "val_loss" (αφορούν τα δεδομένα επικύρωσης) και τους δίνουμε την ονομασία "validation loss".
plt.plot(np.arange(1, EPOCHS+1), HISTORY.history["val_loss"], label="validation loss")

#Με τη μέθοδο ".title()" δίνεται ο τίτλος του διαγράμματος ο οποίος θα εμφανίζεται πάνω από το διάγραμμα.
plt.title("Training and Validation Loss")

#Η μέθοδος ".xlabel()" θέτει τον τίτλο του x άξονα που θα εμφανίζεται κάτω από αυτόν.
plt.xlabel("Epoch #")

#Η μέθοδος ".ylabel()" αντίστοιχα θέτει τον τίτλο του y άξονα που θα εμφανίζεται αριστερά του.
plt.ylabel("Loss")

# Δημιουργία με τη μέθοδο ".legend()" του υπομνήματος που θα περιέχει τις ονομασίες των δύο διαφορετικών γραμμών για το "loss" (εκπαίδευσης και επικύρωσης) καθώς και το χρώμα που αντιπροσωπεύει κάθε γραμμή. Το υπόμνημα με την ιδιότητα "loc" τοποθετείται πάνω δεξιά (upper right) στο διάγραμμα.
plt.legend(loc="upper right")

# Με τη μέθοδο ".annotate" επισημαίνεται πάνω στο διάγραμμα το μήνυμα "Final Train Loss:" μαζί με την αριθμητική τιμή της μεταβλητής "final_train_loss" με τέσσερα δεκαδικά ψηφία (.4f). Το κείμενο επισήμανσης αυτό τοποθετείται στις συντεταγμένες όπου το x είναι ίσο με την τελευταία εποχή της εκπαίδευσης και το y ίσο με την τιμή της μεταβλητής "final_train_loss". Επίσης με το "textcoords" και την τιμή του "offset points" μετατοπίζεται η επισήμανση από το σημείο των συντεταγμένων που δώσαμε κατά τόσο όσο ορίζεται στην ιδιότητα "xytext". Η ιδιότητα "ha" παίρνει την τιμή "right" η οποία ορίζει τη στοίχιση του κειμένου της επισήμανσης ως δεξιά στοιχισμένο και η "color" δέχεται την τιμή "blue" που κάνει το κείμενο μπλε για να ταιριάζει με το χρώμα της γραμμής των τιμών απωλειών εκπαίδευσης.
plt.annotate(f"Final Train Loss: {final_train_loss:.4f}", (EPOCHS, final_train_loss), textcoords="offset points", xytext=(-10, 40), ha='right', color='blue')

# Εδώ γίνεται επισήμανση πάνω στο διάγραμμα σχετικά με την τελευταία τιμή των απωλειών επικύρωσης (final_val_loss) ακριβώς με τον ίδιο τρόπο όπως και για την τελική τιμή των απωλειών εκπαίδευσης.
plt.annotate(f"Final Val Loss: {final_val_loss:.4f}", (EPOCHS, final_val_loss), textcoords="offset points", xytext=(-10, 20), ha='right', color='red')

# Η μέθοδος ".margind()" θέτει τα περιθώρια του διαγράμματος ως 0 και για τους δύο άξονες (x και y). Αυτό διασφαλίζει ότι οι γραμμές με τα δεδομένα φτάνουν μέχρι τις άκρες του διαγράμματος, εκμεταλλεύοντας έτσι τον διαθέσιμο χώρο πάνω στο διάγραμμα.
plt.margins(x=0, y=0)

# Μετατροπή των τιμών του x άξονα σε ακέραιους αριθμούς, διότι οι τιμές είναι καλό να μη φαίνονται δεκαδικές
```

```
αφού οι επαναλήψεις αντιπροσωπεύουν ακέραιους αριθμούς. Επίσης δεν είναι ευανάγνωστο να έχουμε όλες τις τιμές των επαναλήψεων στον άξονα x αν αυτές είναι πολλές π.χ. 40 οπότε στη μεταβλητή "tick_locations" αποθηκεύονται οι τιμές του x άξονα ανά 5, δηλαδή 0,5,10,15 κλπ.
tick_locations = np.arange(0, EPOCHS+1, 5)

# Επειδή στην προηγούμενη εντολή ορίστηκαν οι τιμές των επαναλήψεων να ξεκινάνε από το 0 γίνεται τροποποίηση της πρώτης τιμής του άξονα και ορίζεται το 1 ως τιμή έναρξης.
tick_locations[0] = 1

#Τοποθέτηση των παραπάνω δεδομένων στον πραγματικό άξονα του διαγράμματος και ενημέρωσή του.
plt.xticks(tick_locations, tick_locations)

# Αποθήκευση του διαγράμματος με την μέθοδο ".savefig()" στον επιθυμητό φάκελο σε μορφή εικόνας τύπου ".png", με ανάλυση 300 dpi (dots per inch) εξασφαλίζοντας έτσι υψηλή ανάλυση. Η ανάλυση ορίστηκε με την ιδιότητα "dpi". Επίσης η ιδιότητα "bbox_inches" με όρισμα το "tight" δηλώνει ότι η αποθηκευμένη εικόνα θα περιλαμβάνει μόνο την πραγματική περιοχή του διαγράμματος χωρίς περιττά κενά.
plt.savefig(f"{folder_of_model}/loss_plot.png", dpi=300, bbox_inches="tight")

# Διαδικασία δημιουργίας του 2ου διαγράμματος που θα περιέχει τις ορθότητες εκπαίδευσης και τις ορθότητες επικύρωσης. Για τη δημιουργία του 2ου διαγράμματος ακολουθείται ακριβώς η ίδια διαδικασία με το πρώτο διάγραμμα αλλάζοντας μόνο τα δεδομένα του διαγράμματος και το όνομα του αρχείου που θα αποθηκευτεί.
plt.style.use("bmh")
plt.figure(figsize=(8, 6))
plt.plot(np.arange(1, EPOCHS+1), HISTORY.history["accuracy"], label="training accuracy")
plt.plot(np.arange(1, EPOCHS+1), HISTORY.history["val_accuracy"], label="validation accuracy")
plt.title("Training and Validation Accuracy")
plt.xlabel("Epoch #")
plt.ylabel("Accuracy")
plt.legend(loc="lower right")
plt.annotate(f"Final Train Acc: {final_train_acc:.4f}", (EPOCHS, final_train_acc), textcoords="offset points", xytext=(-10, -40), ha='right', color='blue')
plt.annotate(f"Final Val Acc: {final_val_acc:.4f}", (EPOCHS, final_val_acc), textcoords="offset points", xytext=(-10, -20), ha='right', color='red')
plt.margins(x=0, y=0)
tick_locations = np.arange(0, EPOCHS+1, 5)
tick_locations[0] = 1
plt.xticks(tick_locations, tick_locations)
plt.savefig(f"{folder_of_model}/accuracy_plot.png", dpi=300, bbox_inches="tight")

# Διαγράμματα με σκούρο παρασκήνιο. Τα παρακάτω δύο διαγράμματα (3ο και 4ο) είναι ακριβώς ίδια με το 1ο και 2ο, με τη μόνη διαφορά στο στυλ. Εδώ, συγκεκριμένα, το στυλ είναι σκοτεινό και αυτό μπορεί να είναι πιο φιλικό στο μάτι κάποιων χρηστών. Η εναλλακτική αυτή επιλογή έγινε καθαρά για θέματα που αφορούν την καλύτερη και πιο άνετη αναγνωσιμότητα των αποτελεσμάτων από τον χρήστη.

# Διαδικασία δημιουργίας του 3ου διαγράμματος που θα περιέχει τα τις απώλειες εκπαίδευσης και απώλειες επικύρωσης. Για τη δημιουργία του 3ου διαγράμματος ακολουθείται ακριβώς η ίδια διαδικασία με το πρώτο διάγραμμα αλλάζοντας μόνο το στυλ του διαγράμματος, τα δεδομένα του και το όνομα του αρχείου που θα αποθηκευτεί.
plt.style.use("dark_background")
plt.figure(figsize=(8, 6))
plt.plot(np.arange(1, EPOCHS+1), HISTORY.history["loss"], label="training loss", color='blue')
plt.plot(np.arange(1, EPOCHS+1), HISTORY.history["val_loss"], label="validation loss", color='yellow')
plt.title("Training and Validation Loss")
plt.xlabel("Epoch #")
plt.ylabel("Loss")
plt.legend(loc="upper right")
plt.annotate(f"Final Train Loss: {final_train_loss:.4f}", (EPOCHS, final_train_loss), textcoords="offset points", xytext=(-10, 40), ha='right', color='blue')
plt.annotate(f"Final Val Loss: {final_val_loss:.4f}", (EPOCHS, final_val_loss), textcoords="offset points", xytext=(-10, 20), ha='right', color='yellow')
plt.margins(x=0, y=0)
tick_locations = np.arange(0, EPOCHS+1, 5)
tick_locations[0] = 1
plt.xticks(tick_locations, tick_locations)
```

```
plt.savefig(f"{folder_of_model}/loss_plot_dark_background.png", dpi=300, bbox_inches="tight")

# Διαδικασία δημιουργίας του 4ου διαγράμματος που θα περιέχει τις ορθότητες εκπαίδευσης και τις ορθότητες επικύρωσης. Για τη δημιουργία του 4ου διαγράμματος ακολουθείται ακριβώς η ίδια διαδικασία με το πρώτο διάγραμμα αλλάζοντας μόνο το στυλ του διαγράμματος, τα δεδομένα του και το όνομα του αρχείου που θα αποθηκευτεί.
plt.style.use("dark_background")
plt.figure(figsize=(8, 6))
plt.plot(np.arange(1, EPOCHS+1), HISTORY.history["accuracy"], label="training accuracy", color='blue')
plt.plot(np.arange(1, EPOCHS+1), HISTORY.history["val_accuracy"], label="validation accuracy", color='yellow')
plt.title("Training and Validation Accuracy")
plt.xlabel("Epoch #")
plt.ylabel("Accuracy")
plt.legend(loc="lower right")
plt.annotate(f"Final Train Acc: {final_train_acc:.4f}", (EPOCHS, final_train_acc), textcoords="offset points", xytext=(-10, -40), ha='right', color='blue')
plt.annotate(f"Final Val Acc: {final_val_acc:.4f}", (EPOCHS, final_val_acc), textcoords="offset points", xytext=(-10, -20), ha='right', color='yellow')
plt.margins(x=0, y=0)
tick_locations = np.arange(0, EPOCHS+1, 5)
tick_locations[0] = 1
plt.xticks(tick_locations, tick_locations)
plt.savefig(f"{folder_of_model}/accuracy_plot_dark_background.png", dpi=300, bbox_inches="tight")

#Δημιουργία διαγράμματος σχετικά με την καμπύλη ROC.

# Για τη δημιουργία αυτού του διαγράμματος ακολουθείται η ίδια διαδικασία με το 1ο διάγραμμα αλλάζοντας μόνο τα δεδομένα που θα εμφανιστούν. Επιπλέον γίνεται ορισμός των τιμών που θα εμφανίζει ο άξονας x και y με τις μεθόδους "xlim" και "ylim" αντίστοιχα.
plt.style.use("bmh")
plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, label='ROC curve (area = %0.2f)' % auc_score)
plt.plot([0, 1], [0, 1], 'k--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic')
plt.legend(loc="lower right")
plt.margins(x=0, y=0)
plt.savefig(f"{folder_of_model}/ROC_plot.png", dpi=300, bbox_inches="tight")

#Εκτύπωση ενημερωτικού μηνύματος στην οθόνη.
print("[ΕΝΗΜΕΡΩΣΗ] Τέλος εκπαίδευσης του μοντέλου. Τερματισμός προγράμματος...")
```

Παράρτημα Β : Κώδικας ανίχνευσης μάσκας σε ανθρώπινα πρόσωπα μέσω του Raspberry Pi 4

```
# Προσθήκη όλων των απαραίτητων πακέτων δηλαδή των βιβλιοθηκών (libraries), ενοτήτων (modules), πλαισίων (frameworks)  ή μέρος αυτών για να μπορέσει να εκτελεστεί ο κώδικας. Κάποια πακέτα εισάγονται ολόκληρα απλά με την εντολή "import όνομα_επιθυμητού_πακέτου". Σε άλλες περιπτώσεις εισάγονται μέρη αυτών που θα χρειαστούν ή κάποιες συναρτήσεις (functions) τους ή μεθόδους (methods) τους. Για την εισαγωγή ενός μόνο μέρους του πακέτου χρησιμοποιείται η σύνταξη "from όνομα_επιθυμητού_πακέτου import όνομα_επιθυμητού_μέρους_του". Επίσης με την προσθήκη της εντολής "as επιθυμητό_νέο_όνομα" δίπλα από την εντολή εισαγωγής κάποιου  πακέτου ή μέρους αυτού, δίνεται η δυνατότητα στον κώδικα να απευθύνεται σε αυτό με ένα νέο όνομα, συνήθως πιο σαφές, απλό και σύντομο.

# Το "tensorflow" είναι ένα πλαίσιο ανοικτού κώδικα (open-source code) όπου η βιβλιοθήκη του περιέχει εντολές με σκοπό τη δημιουργία και εκπαίδευση μοντέλων μηχανικής μάθησης (machine learning models). Επίσης όταν το "tensorflow" εισάγεται, του δίνεται το ψευδώνυμο "tf" για να καλείται μέσα στον κώδικα με μεγαλύτερη ευκολία.
import tensorflow as tf

#Εισαγωγή της συνάρτησης "preprocess_input" από την "tensorflow.keras.applications.mobilenet_v2" ενότητα. Η συνάρτηση αυτή
#προεπεξεργάζεται τις εισαγόμενες εικόνες και τις προετοιμάζει με βάση την αρχιτεκτονική του μοντέλου MobileNetV2 που
#αποτελεί το συνελικτικό νευρωνικό δίκτυο του μοντέλου που θα εκπαιδευτεί.
from tensorflow.keras.applications.mobilenet_v2 import preprocess_input

#Εισαγωγή της συνάρτησης "img_to_array" από την "tensorflow.keras.preprocessing.image" ενότητα. Η συνάρτηση αυτή, μετατρέπει
#μία εικόνα σε διάνυσμα της βιβλιοθήκης NumPy έτσι ώστε αυτή να μπορεί να υποβληθεί σε επεξεργασία από μοντέλα μηχανικής
#εκμάθησης.
from tensorflow.keras.preprocessing.image import img_to_array

#Εισαγωγή της κλάσης "VideoStream" από την "imutils.video" ενότητα. Η κλάση αυτή παρέχει μία απλοποιημένη διεπαφή για πρόσβαση
#σε ροές βίντεο από διάφορες πηγές, όπως web κάμερες ή αρχεία βίντεο.
from imutils.video import VideoStream

#Εισαγωγή της βιβλιοθήκης "numpy" με το ψευδώνυμο "np" που χρησιμεύει για αριθμητικούς υπολογισμούς στην Python. Επίσης
#παρέχει υποστήριξη για μεγάλους πολυδιάστατους πίνακες ή πίνακες διανυσμάτων, αφού περιέχει μία συλλογή μαθηματικών
#συναρτήσεων για την αποτελεσματική επεξεργασία τους.
import numpy as np

#Το "imutils" είναι μια βιβλιοθήκη που περιέχει συναρτήσεις για την διευκόλυνση της χρήσης της βιβλιοθήκης OpenCV (Open Source
#Computer Vision). Κάποιες από τις λειτουργίες των συναρτήσεων αυτών είναι η αλλαγή μεγέθους των εικόνων και η περιστροφή τους.
import imutils

#Το "cv2" αναφέρεται στην βιβλιοθήκη ανοικτού κώδικα "OpenCV" η οποία χρησιμοποιείται ευρέως για θέματα
#όρασης υπολογιστή (computer vision) και μηχανικής μάθησης (machine learning).
import cv2


"""Μέρος 1ο - Συνάρτηση για την ανίχνευση μάσκας"""

# Δημιουργία της συνάρτησης, που όταν καλείται από τον κύριο κώδικα, λαμβάνει το στιγμιότυπο/την εικόνα του βίντεο εκείνης της δεδομένης στιγμής και το επεξεργάζεται για να επιστρέψει ξανά στον κύριο κώδικα κάποιες πληροφορίες. Αρχικά, όταν καλείται δέχεται ως ορίσματα την εικόνα από το βίντεο, το μοντέλο
```

```
εντοπισμού ανθρώπινου προσώπου και το μοντέλο εντοπισμού μάσκας. Έπειτα γίνονται κάποιες επεξεργασίες
πάνω στην εικόνα για να είναι συμβατή με τα δύο παραπάνω μοντέλα που αναφέρθηκαν και γίνεται πρώτα ο
εντοπισμός προσώπων σε αυτή. Το πρώτο μοντέλο ελέγχει την πιθανότητα ύπαρξης κάποιου προσώπου ή
προσώπων και αν υπάρχει τότε αποθηκεύει αυτές τις πιθανότητες στην μεταβλητή "detections". Έπειτα γίνεται
σύγκριση αυτών των πιθανοτήτων με τον αριθμό 0.5, δηλαδή το 50%, όπου αν αυτές είναι μεγαλύτερες από
αυτόν τότε θεωρείται ότι βρέθηκε πρόσωπο, αλλιώς θεωρείται ότι το πρόγραμμα δεν βρήκε πρόσωπο και απλά
ανίχνευσε κάτι παρόμοιο με αυτό. Για τα πρόσωπα που εντοπίστηκαν υπολογίζονται οι συντεταγμένες τους πάνω
στο στιγμιότυπο και δημιουργείται για κάθε πρόσωπο μία νέα εικόνα μόνο με αυτό, η οποία δέχεται κάποιες
επεξεργασίες. Οι επεξεργασίες αυτές είναι αρχικά η μετατροπή της από μορφή BGR (Blue Green Red) σε RGB
(Red Blue Green), αλλαγή των διαστάσεων σε 224x224, μετατροπή σε διάνυσμα και κάποιες άλλες ειδικές
επεξεργασίες από την μέθοδο "preprocess_input". Όλες αυτές οι επεξεργασίες γίνονται για να είναι συμβατή η
εικόνα κατά την εισαγωγή της στο μοντέλο ανίχνευσης μάσκας. Πέρα από την εικόνα προσώπου υπολογίζονται
και οι συντεταγμένες που θα τοποθετηθεί αργότερα το παραλληλόγραμμο γύρω από το πρόσωπο του ανθρώπου
που εντοπίστηκε. Στη συνέχεια γίνεται ο έλεγχος ύπαρξης μάσκας ή όχι με βάση την εικόνα προσώπου και τα
αποτελέσματα που είναι σε μορφή πιθανοτήτων από το 0 έως το 1 αποθηκεύονται στη λίστα "preds". Για κάθε
εικόνα προσώπου αποθηκεύονται δύο πιθανότητες στη λίστα "preds", μία για την ύπαρξη μάσκας και μία για
την μη ύπαρξη μάσκας. Τέλος, τα αποτελέσματα αυτά μαζί με τις συντεταγμένες του παραλληλογράμμου,
επιστρέφονται στον κύριο κώδικα για την περαιτέρω επεξεργασία τους.

# Με το "def" γίνεται ορισμός της συνάρτησης όπου ακριβώς δίπλα του ακολουθεί το όνομα που της δόθηκε,
δηλαδή το "detect_and_predict_mask". Αμέσως μετά το όνομα, μέσα σε παρενθέσεις, δηλώνονται τα ορίσματα
που θα δεχτεί η συνάρτηση από τον κύριο κώδικα. Τα ορίσματα αυτά είναι ένα στιγμιότυπο (frame), το μοντέλο
ανίχνευσης προσώπου (faceNet) και το μοντέλο ανίχνευσης μάσκας (maskNet).
def detect_and_predict_mask(frame, faceNet, maskNet):

# Το χαρακτηριστικό (attribute) ".shape" επιστρέφει πληροφορίες σχετικά με το "frame" πάνω στο οποίο
εκτελείται. Με το slicing "[:2]" στη λίστα δίνεται εντολή να επιστραφούν μόνο οι πληροφορίες σχετικά με το
ύψος και το πλάτος του "frame". Αυτά αποθηκεύονται σε ένα tuple (πλειάδα) που περιέχει τις μεταβλητές "h"
και "w" τα οποία αντιστοιχούν στο ύψος και το πλάτος.
  (h, w) = frame.shape[:2]

# Δημιουργία ενός "blob" (Binary Large Object) που θα περιέχει πληροφορίες σχετικά με το "frame"
χρησιμοποιώντας τη συνάρτηση "cv2.dnn.blobFromImage" της βιβλιοθήκης OpenCV. Ειδικότερα, γίνεται
προεπεξεργασία του "frame" για είσοδο στο μοντέλο ανίχνευσης προσώπου δίνοντας στη συνάρτηση κάποιες
προδιαγραφές ως ορίσματα. Το πρώτο όρισμα  είναι το ίδιο το "frame", το δεύτερο είναι η τιμή 1 που
αντιπροσωπεύει στη συγκεκριμένη περίπτωση τη μη σμίκρυνση ή μεγέθυνση του "frame", το τρίτο είναι οι νέες
διαστάσεις που θα πάρει το "frame", δηλαδή 224x224 και το τέταρτο αντιπροσωπεύει τις μέσες τιμές pixel που
αφαιρούνται από την εικόνα. Σε αυτήν την περίπτωση, το "(104.0, 177.0, 123.0)" αντιστοιχεί στις μέσες τιμές για
τα κανάλια κόκκινο, πράσινο και μπλε. Το αντικείμενο "blob" που θα δημιουργηθεί θα είναι τεσσάρων
διαστάσεων όπου η πρώτη είναι η εικόνα που περιέχει, άρα ένα αφού του εισάγουμε μόνο ένα frame. Η δεύτερη
περιέχει τα τρία κανάλια χρώματος που έχει το "frame" και τα αναγνωρίζει ως BGR (Blue Green Red). Η τρίτη
περιέχει το ύψος, δηλαδή 224 και η τέταρτη το πλάτος όπου και αυτό είναι 224.
  blob = cv2.dnn.blobFromImage(frame, 1.0, (224, 224), (104.0, 177.0, 123.0))

# Εισαγωγή του "blob" στο μοντέλο ανίχνευσης προσώπου με την εντολή ".setInput()".
  faceNet.setInput(blob)

# Επεξεργασία του "blob" και ανίχνευση πιθανών προσώπων. Τα αποτελέσματα ανίχνευσης αποθηκεύονται στο
διάνυσμα "detections" ως πιθανότητες από το 0 έως το 1.
  detections = faceNet.forward()

# Δημιουργία της κενής λίστας "faces" στην οποία θα αποθηκευτούν οι εικόνες των προσώπων που μπορεί να
βρέθηκαν.
  faces = []

# Δημιουργία της κενής λίστας "locs" στην οποία θα αποθηκευτούν οι συντεταγμένες των παραλληλογράμμων
που θα τοποθετηθούν γύρω από τα πρόσωπα που μπορεί να βρέθηκαν.
  locs = []

# Δημιουργία της κενής λίστας "preds" στην οποία θα αποθηκευτούν οι προβλέψεις για το αν εντοπίστηκε
μάσκα ή όχι στα πρόσωπα που μπορεί να βρέθηκαν.
  preds = []

# Δημιουργία βρόγχου επανάληψης "for" όπου η μεταβλητή "i" θα παίρνει με τη σειρά τις τιμές από 0 έως τον
```

```
αριθμό του συνόλου των προσώπων που πιθανά βρέθηκαν. Το "range" ορίζει τις τιμές που θα πάρει το "i", δηλαδή από 0 έως "detections.shape[2]" με βήμα 1. Το "detection.shape[2]" επιστρέφει τον αριθμό των προσώπων που πιθανά βρέθηκαν, όπου αυτός ο αριθμός βρίσκεται συγκεκριμένα αποθηκευμένος στην τρίτη διάσταση του "detections". Η πρόσβαση στην τρίτη διάσταση αυτή γίνεται με τη χρήση του ".shape[2]" όπου το όρισμα "2" αντιστοιχεί στην τρίτη διάσταση, αφού στη γλώσσα προγραμματισμού Python οι δείκτες μίας λίστας ξεκινούν από το 0 και όχι το 1.
   for i in range(0, detections.shape[2]):

# Αποθήκευση της πιθανότητας (confidence score) που περιέχει η νούμερο "i" πρόβλεψη ανίχνευσης προσώπου στη μεταβλητή "confidence". Συγκεκριμένα για την αναζήτηση της αποθηκευμένης αυτής πιθανότητας εκτελείται το "detections[0, 0, i, 2]" όπου το πρώτο "0" σημαίνει ότι θέλουμε την πρόβλεψη από την πρώτη εικόνα, άρα και τη μοναδική έτσι κι αλλιώς αφού μόνο πάνω στο "frame" έγιναν προβλέψεις. Το δεύτερο "0" αντιπροσωπεύει ποιας κλάσης την πιθανότητα θέλουμε, αλλά αφού μόνο μια κλάση υπάρχει και αυτή είναι το ανθρώπινο πρόσωπο χρησιμοποιείται το "0" που αντιστοιχεί στην πρώτη κλάση. Το "i" αντιστοιχεί σε μία από τις προβλέψεις. Το "2" είναι ο δείκτης που περιέχει την πιθανότητα που χρειάζεται να αποθηκευτεί στο "confidence".
      confidence = detections[0, 0, i, 2]

# Έλεγχος του "confidence" εάν είναι μεγαλύτερο από 0.5 δηλαδή 50% πιθανότητα. Εάν είναι, τότε σημαίνει ότι η συγκεκριμένη πρόβλεψη είναι αληθής και υπάρχει πρόσωπο στο "frame", οπότε το πρόγραμμα θα συνεχίσει μέσα στο "if". Ο κώδικας μέσα στο "if" πραγματοποιεί εύρεση των συντεταγμένων του παραλληλογράμμου που θα τοποθετηθεί γύρω από το πρόσωπο και δημιουργεί μία νέα εικόνα με το πρόσωπο που βρέθηκε.
      if confidence > 0.5:

# Γίνεται αποθήκευση των συντεταγμένων του νοητού παραλληλογράμμου, γύρω από το πρόσωπο που εντοπίστηκε, οι οποίες είναι αποθηκευμένες στην τέταρτη διάσταση του "detections" με δείκτες 3, 4, 5, 6 αφού το slicing του διανύσματος ορίζεται ως "3:7". Επίσης αυτές οι συντεταγμένες πολλαπλασιάζονται με ένα διάνυσμα που περιέχει τις διαστάσεις του "frame" για να προσαρμοστούν ως προς το μέγεθός του.
         box = detections[0, 0, i, 3:7] * np.array([w, h, w, h])

# Οι τέσσερις νέες συντεταγμένες που αποθηκεύτηκαν στο "box" μετατρέπονται αρχικά σε ακέραιες (integers) τιμές μέσω της μεθόδου ".astype("int")" και έπειτα αποθηκεύονται σε μια πλειάδα και ειδικότερα η κάθε μία σε μία αντίστοιχη μεταβλητή.
         (startX, startY, endX, endY) = box.astype("int")

# Επειδή υπάρχει η περίπτωση, θεωρητικά, οι συντεταγμένες να έχουν τιμές εκτός της πραγματικής εικόνας που θα εμφανιστούν, γίνεται προσαρμογή τους να μην ξεπερνάνε τα όρια της εικόνας. Οπότε αρχικά, για τις συντεταγμένες της πάνω αριστερά γωνίας του νοητού παραλληλογράμμου υπολογίζεται η νέα τιμή τους, βρίσκοντας την μέγιστη τιμή μεταξύ του 0 και της πραγματικής τους τιμής. Εάν δηλαδή η πραγματική τους τιμή είναι αρνητική, αυτή αναγκαστικά θα μετατραπεί σε 0 για να ξεκινήσει να φαίνεται το παραλληλόγραμμο από την κάτω αριστερή γωνία του "frame".
         (startX, startY) = (max(0, startX), max(0, startY))

# Αντίστοιχα εδώ, υπολογίζονται οι νέες τιμές της κάτω δεξιά γωνίας όπου βρίσκεται η μέγιστη τιμή μεταξύ των πραγματικών τιμών των συντεταγμένων "endX", "endY" και των ορίων του πλάτους ή του ύψους του "frame" αντίστοιχα "-1". Το μείον ένα χρειάζεται επειδή το ανώτατο όριο της εικόνας είτε στο πλάτος είτε στο ύψος δεν είναι το 224 αλλά το 223 λόγω του προγραμματισμού σε γλώσσα Python. Αυτό μπορεί να γίνει πιο ξεκάθαρο αν σκεφτεί κανείς πως στην πάνω εντολή τα όρια προσαρμόζονται από το σημείο 0 και όχι το 1 άρα τα όρια είναι 0-223 και όχι 1-224.
         (endX, endY) = (min(w - 1, endX), min(h - 1, endY))

# Εδώ γίνεται εξαγωγή ενός κομματιού του "frame" και συγκεκριμένα αυτού που περιέχει το πρόσωπο που εντοπίστηκε. Οι συντεταγμένες που υπολογίστηκαν προηγουμένως συμβάλλουν σε αυτή τη διαδικασία και η νέα εικόνα του προσώπου αποθηκεύεται στη μεταβλητή "face".
         face = frame[startY:endY, startX:endX]

# Επειδή το "frame" ήταν της μορφής BGR έτσι και το "face" δημιουργήθηκε ως BGR. Επειδή όμως αυτή η μορφή εικόνας δεν είναι συμβατή με αυτή που θα πρέπει να ελέγξει το μοντέλο ανίχνευσης μάσκας, πρέπει να αλλάξει και να μετατραπεί σε μορφή RGB. Για αυτό αναλαμβάνει η συνάρτηση ".cvtColor()" που δέχεται ως πρώτο όρισμα την εικόνα "face" και σαν δεύτερο όρισμα την μορφή που πρέπει να γίνει η μετατροπή.
         face = cv2.cvtColor(face, cv2.COLOR_BGR2RGB)

# Προσαρμογή της εικόνας "face" στις διαστάσεις 224 x 224, διότι το μοντέλο ανίχνευσης μάσκας έχει εκπαιδευτεί για να δέχεται εικόνες αυτών των συγκεκριμένων διαστάσεων. Αυτό επιτυγχάνεται με την
```

```
συνάρτηση ".resize()" της βιβλιοθήκης OpenCV η οποία δέχεται ως ορίσματα την εικόνα "face" και τις διαστάσεις προσαρμογής.
        face = cv2.resize(face, (224, 224))

# Μετατροπή της εικόνας "face" σε μορφή διανύσματος με τη χρήση της συνάρτησης "img_to_array()" από την "tensorflow.keras.preprocessing.image" ενότητα. Αυτή η μετατροπή είναι απαραίτητη για την προετοιμασία του "face" για περαιτέρω επεξεργασία από το μοντέλο ανίχνευσης μάσκας.
        face = img_to_array(face)

# Με τη συνάρτηση "preprocess_input()" από την "tensorflow.keras.applications.mobilenet_v2" ενότητα γίνονται κάποιες τελικές μετατροπές στο "face" για να μπορεί να είναι συμβατό με την αρχιτεκτονική της "mobilenet_v2" που είναι αυτή που χρησιμοποιήθηκε για την κατασκευή του μοντέλου ανίχνευσης μάσκας.
        face = preprocess_input(face)

# Προσθήκη του διανύσματος "face" στη λίστα "faces" με την συνάρτηση ".append()" που προσθέτει ένα νέο διάνυσμα κάθε φορά στο τέλος της λίστας.
        faces.append(face)

# Προσθήκη στη λίστα "locs", με τη βοήθεια πάλι της εντολής ".append()", των συντεταγμένων του παραλληλογράμου που θα σχεδιαστεί αργότερα γύρω από το πρόσωπο που αντιστοιχεί στο "face" που αποθηκέυτηκε προηγουμένως στην λίστα "faces".
        locs.append((startX, startY, endX, endY))

# Έλεγχος με τη συνθήκη "if" εάν υπάρχει έστω ένα εντοπισμένο πρόσωπο στο "frame". Εάν υπάρχει, τότε ο κώδικας συνεχίζει μέσα στο "if" όπου ξεκινάει η ανίχνευση μάσκας. Αν δεν υπάρχει, τότε ο κώδικας επιστρέφει τις κενές λίστες "locs" και "preds" στον κύριο κώδικα. Στη συνθήκη "if" ελέγχεται ουσιαστικά το μήκος της λίστας "faces" εάν είναι μεγαλύτερο του μηδενός, αφού εάν υπήρχε έστω και ένα "face" μέσα του, το μήκος θα ήταν τουλάχιστον 1. Ο υπολογισμός του μήκους της λίστας γίνεται με την εντολή "len()".
    if len(faces) > 0:

# Δημιουργία βρόγχου επανάληψης "for" που θα ελέγξει ξεχωριστά όλα τα πρόσωπα που εντοπίστηκαν και βρίσκονται μέσα στην λίστα "faces".
        for face in faces:

# Προσθήκη στο "face" μίας επιπλέον διάστασης στην αρχή των άλλων διαστάσεων που θα αντιπροσωπεύει τον αριθμό των εικόνων που περιέχει αυτό το διάνυσμα. Αυτή η μετατροπή γίνεται μόνο για λόγους συμβατότητας με το μοντέλο ανίχνευσης μάσκας επειδή είναι σχεδιασμένο να δέχεται διανύσματα τεσσάρων διαστάσεων, ενώ το "face" έχει τρείς. Για παράδειγμα εάν το "face" έχει τη μορφή (224, 224, 3) τότε μετά την μετατροπή θα μοιάζει έτσι (1, 224, 224, 3). Για την προσθήκη της νέας διάστασης χρησιμοποιείται η εντολή "np.expand_dims()" της βιβλιοθήκης "numpy" που δέχεται ως πρώτο όρισμα το "face" και ως δεύτερο την τοποθεσία της νέας διάστασης με δείκτη "0" δηλαδή στην πρώτη θέση.
            face = np.expand_dims(face, axis=0)

# Με την εκτέλεση της εντολής ".set_tensor()" πάνω στο μοντέλο ανίχνευσης μάσκας ή "maskNet" γίνεται ανάθεση του διανύσματος "face" στον τανυστή εισόδου του μοντέλου. Πιο συγκεκριμένα, το μοντέλο θα δεχτεί στην είσοδό του το "face". Το πρώτο όρισμα που δέχεται η εντολή είναι το "input_details[0]['index']" το οποίο είναι μία λίστα λεξικών (Python Dictionaries) που περιέχουν πληροφορίες σχετικά με τους τανυστές εισόδου του μοντέλου. Ειδικότερα, με τον δείκτη "0" παρέχονται πληροφορίες από το λεξικό για τον πρώτο τανυστή εισόδου. Το "'index'" είναι υπεύθυνο για την αναγνώριση και προετοιμασία του συγκεκριμένου τανυστή εισόδου. Το δεύτερο όρισμα είναι το διάνυσμα "face".
            maskNet.set_tensor(input_details[0]['index'], face)

# Με την εντολή ".invoke()" ξεκινά η διαδικασία πρόβλεψης με βάση το διάνυσμα "face" που περιέχει το πρόσωπο κάποιου ανθρώπου. Τα αποτελέσματα αποθηκεύονται στο μοντέλο.
            maskNet.invoke()

# Με την εντολή ".get_tensor()" πάνω στο "maskNet" λαμβάνονται τα αποτελέσματα, δηλαδή οι προβλέψεις από τον τανυστή εξόδου του μοντέλου. Αυτά τα αποτελέσματα αποθηκεύονται στη μεταβλητή "pred" και περιέχουν την πιθανότητα για την ύπαρξη μάσκας άλλα και την πιθανότητα για την μη ύπαρξη μάσκας. Το ".get_tensor()" δέχεται ένα μόνο όρισμα το οποίο είναι το "output_details[0]['index']" που παρομοίως με τον τανυστή εισόδου παρέχει πληροφορίες και προετοιμάζει τον πρώτο τανυστή εξόδου.
            pred = maskNet.get_tensor(output_details[0]['index'])

# Η πρόβλεψη "pred" που έγινε προστίθεται στη λίστα "preds" που θα περιέχει όλες τις προβλέψεις για όλα τα
```

```
πρόσωπα που εντοπίστηκαν.
        preds.append(pred)

# Τέλος, γίνεται επιστροφή στον κύριο κώδικα των "locs" και "preds" ως tuple για την τελική επεξεργασία τους και εμφάνιση των αποτελεσμάτων σε live video stream.
    return (locs, preds)


    """Μέρος 2ο - Κύριος κώδικας ενεργοποίησης video stream και απεικόνηση των αποτελεσμάτων σε παράθυρο"""


# Αποθήκευση της τοποθεσίας του αρχείου "deploy.prototxt" ως "r" (raw) αλφαριθμητικό στη μεταβλητή "prototxtPath". Το "deploy.prototxt" είναι το πρώτο από τα δύο αρχεία που αποτελούν το μοντέλο ανίχνευσης προσώπου και περιέχει πληροφορίες σχετικά με την αρχιτεκτονική του, τις παραμέτρους του, τα επίπεδά του και τη σύνδεση μεταξύ αυτών. Το μοντέλο που χρησιμοποιείται έχει σχεδιαστεί με βάση των αλγόριθμο Single Shot MultiBox Detector (SSD) και περιέχει στοιχεία από την αρχιτεκτονική του "MobileNetV1".
prototxtPath = r"face_detector_model/deploy.prototxt"

# Αποθήκευση της τοποθεσίας του δεύτερου αρχείου του μοντέλου ανίχνευσης προσώπου "res10_300x300_ssd_iter_140000.caffemodel" ως "r" (raw) αλφαριθμητικό στη μεταβλητή "weightsPath". Αυτό το αρχείο περιέχει πληροφορίες σχετικά με τα βάρη του μοντέλου και τις εκπαιδευμένες παραμέτρους του. Τα βάρη αντιπροσωπεύουν τις γνώσεις του μοντέλου για την καλή αναγνώριση διάφορων χαρακτηριστικών σε εικόνες. Αυτά έχουν διαμορφωθεί μετά από εκπαίδευση του μοντέλου πάνω σε ένα τεράστιο σύνολο δεδομένων σχετικά με την αναγνώριση προσώπων.
weightsPath = r"face_detector_model/res10_300x300_ssd_iter_140000.caffemodel"

# Με την εντολή "cv2.dnn.readNet()" της βιβλιοθήκης openCV γίνεται φόρτωση του μοντέλου αναγνώρισης προσώπου στο αντικείμενο "faceNet". Η εντολή δέχεται ως ορίσματα τις δύο τοποθεσίες των αρχείων που προαναφέρθηκαν, ώστε αυτά να δημιουργήσουν το μοντέλο μετά την ένωση και την κατάλληλη επεξεργασία τους.
faceNet = cv2.dnn.readNet(prototxtPath, weightsPath)

# Αποθήκευση της τοποθεσίας του αρχείου "mask_detection_model_optim.tflite" ως "r" (raw) αλφαριθμητικό στη μεταβλητή "mask_string". Το αρχείο αυτό αντιπροσωπεύει το μοντέλο που εκπαιδεύτηκε για την ανίχνευση μάσκας και έχει τη μορφή ".tflite".
mask_string = r"mask_detection_model_optim.tflite"

# Φόρτωση του αρχείου "mask_detection_model_optim.tflite" και δημιουργία του μοντέλου ανίχνευσης μάσκας με όνομα αντικειμένου "maskNet". Η δημιουργία του μοντέλου οφείλεται στην εντολή "tf.lite.Interpreter()" της βιβλιοθήκης "tensorflow" και συγκεκριμένα του μέρους του που ονομάζεται "tensorflow lite". Η εντολή αυτή δέχεται ως όρισμα την τοποθεσία του αρχείου του μοντέλου στον υπολογιστή δηλαδή το "mask_string". Η σύνταξη του ορίσματος είναι η εξής "model_path = mask_string".
maskNet = tf.lite.Interpreter(model_path = mask_string)

# Η εντολή ".allocate_tensors()" είναι απαραίτητη πριν από οποιαδήποτε χρήση του μοντέλου στον κώδικα διότι κατανέμει στη μνήμη τους τανυστές εισόδου και εξόδου. Μετά από αυτήν την εντολή το μοντέλο "maskNet" είναι έτοιμο να δεχτεί εισόδους και να παρέχει εξόδους.
maskNet.allocate_tensors()

# Η εντολή ".get_input_details()" παρέχει πληροφορίες σχετικά με τους τανυστές εισόδου του μοντέλου τις οποίες αποθηκεύει στο αντικείμενο "input_details". Πιο συγκεκριμένα, θα αποθηκευτεί μία λίστα απο λεξικά (dictionaries) όπου κάθε λεξικό (dictionary) αντιστοιχεί σε ένα τανυστή εισόδου παρέχοντας πληροφορίες, όπως όνομα και τύπος δεδομένων.
input_details = maskNet.get_input_details()

# Η εντολή ".get_output_details()" παρέχει πληροφορίες σχετικά με τους τανυστές εξόδου του μοντέλου τις οποίες αποθηκεύει στο αντικείμενο "output_details". Όπως και για τους τανυστές εισόδου αντίστοιχα θα αποθηκεύεται μία λίστα από λεξικά όπου κάθε λεξικό αντιστοιχεί σε έναν τανυστή εξόδου.
output_details = maskNet.get_output_details()

#Εκτύπωση ενημερωτικού μηνύματος στην οθόνη.
print("[ΕΝΗΜΕΡΩΣΗ] Το βίντεο ξεκίνησε...")

# Η κλάση "VideoStream()"  του "imutils.video" module, προετοιμάζει τη ροή βίντεο (video stream) από την
```

```
προκαθορισμένη κάμερα του Raspberry Pi 4. Η επιλογή της κάμερας γίνεται μέσα στις παρενθέσεις του "VideoStream()" όπου δέχεται το όρισμα "src = 0". Το "0" σε αυτή την περίπτωση αντιστοιχεί στο camera module V2 που έχει συνδεθεί στο Raspberry. Εάν κάποιος θελήσει να χρησιμοποιήσει ροή βίντεο από κάποια άλλη κάμερα που είναι συνδεδεμένη πάνω στο Raspberry, τότε αρκεί να επιλέξει τον αριθμό που αντιστοιχεί σε αυτήν την κάμερα. Με την μέθοδο ".start()" ενεργοποιείται η ροή βίντεο και η κάμερα ξεκινά να λαμβάνει στιγμιότυπα (Frames).
vs = VideoStream(src = 0).start()

# Αρχικοποίηση της μεταβλητής "fl" με την τιμή "0" για να χρησιμοποιηθεί στο κομμάτι του κώδικα που θα ελέγχει εάν το παράθυρο της ροής βίντεο έκλεισε.
fl = 0

# Δημιουργία ατέρμονα βρόγχου επανάληψης "while" αφού κατά τον έλεγχό του δέχεται πάντα την απάντηση της συνθήκης ως "True". Μέσα σε αυτόν τον βρόγχο πραγματοποιείται αρχικά σε κάθε επανάληψή του η λήψη ενός στιγμιότυπου "frame" από την κάμερα. Έπειτα αυτό προσαρμόζεται στις διαστάσεις που έχουν επιλεχθεί για το παράθυρο στο οποίο θα εμφανιστεί. Έτσι, γίνεται κλήση της συνάρτησης "detect_and_predict_mask()" όπου δέχεται αυτό το "frame" και επιστρέφει απαντήσεις σχετικά με το εάν ανιχνεύθηκε κάποια μάσκα σε πρόσωπο ή όχι. Έπειτα σε περίπτωση που κάποιο πρόσωπο με μάσκα ή χωρίς ανιχνεύθηκε, ελέγχεται και δημιουργείται το νέο "frame" που θα εμφανιστεί στην οθόνη. Αυτό το νέο "frame" θα περιέχει ένα ορθογώνιο παραλληλόγραμμο γύρω από το ανθρώπινο πρόσωπο με χρώμα κόκκινο εάν δεν φοράει μάσκα ή πράσινο εάν φοράει. Επιπλέον, πάνω ακριβώς από αυτό το πλαίσιο θα εμφανίζεται κατάλληλο μήνυμα σχετικά με την ύπαρξη ή μη μάσκας, συνοδευόμενο από την πιθανότητα, σε ποσοστό, να είναι σωστή αυτή η πρόβλεψη. Στη συνέχεια γίνεται εμφάνιση αυτού του νέου "frame" στην οθόνη του υπολογιστή μέσα σε ένα παράθυρο με διαστάσεις 400x400 και γίνεται έλεγχος εάν ο χρήστης πάτησε το πλήκτρο "ESC" ή το σύμβολο "x" στο πάνω δεξιά σημείο του παραθύρου. Εφόσον ένα από τα δύο πατήθηκε, τότε το παράθυρο κλείνει και το πρόγραμμα τερματίζεται. Σε περίπτωση που δεν κλείσει το παράθυρο τότε ο κώδικας μέσα στον βρόγχο επανάληψης συνεχίζει να εκτελείται λαμβάνοντας το επόμενο στιγμιότυπο.
while True:

# Έλεγχος με τη συνθήκη "if" εάν η μεταβλητή "fl" είναι ίση με το "1". Αυτός ο έλεγχος γίνεται ώστε την πρώτη φορά που θα εκτελεστεί ο κώδικας μέσα στον βρόγχο επανάληψης "while", να μην εκτελεστεί το κομμάτι κώδικα μέσα στην συνθήκη "if" αυτή. Οπότε αφού έχει οριστεί το "fl" ως "0" έξω από το "while", αυτή η συνθήκη δεν θα εκτελεστεί την πρώτη φορά.
  if fl == 1:

# Απο τη δεύτερη επανάληψη της εκτέλεσης του κώδικα μέσα στο "while" το παρακάτω κομμάτι κώδικα με τη συνθήκη "if" θα ελέγχεται κάθε φορά. Εδώ γίνεται έλεγχος εάν το παράθυρο, που εμφανίζονται τα "frames" ως βίντεο, έκλεισε από την επιλογή του χρήστη να πατήσει πάνω στο σύμβολο "x" του παραθύρου. Σε αυτό βοηθάει η εντολή ".getWindowProperty()" της βιβλιοθήκης OpenCV η οποία δέχεται σαν πρώτο όρισμα την ονομασία του παραθύρου που θέλει να ελέγξει και σαν δεύτερο την εντολή "WND_PROP_VISIBLE" που δείχνει τι θέλουμε να ελέγξουμε. Έτσι, η εντολή ".getWindowProperty()" επιστρέφει μία τιμή κάτω του "1" αν το παράθυρο έχει κλείσει ή μεγαλύτερο ή και ίσο με το "1" άν παραμένει ανοικτό. Εάν λοιπόν είναι μικρότερο του "1" τότε ο κώδικας μέσα σε αυτό το "if" παραβλέπεται και η εκτέλεση του προγράμματος συνεχίζει παρακάτω. Σε άλλη περίπτωση ο εμφωλευμένος κώδικας εκτελείται.
    if cv2.getWindowProperty("Mask Detection", cv2.WND_PROP_VISIBLE) < 1:

# Εάν εκτελεστεί ο εμφωλευμένος αυτός κώδικας και συγκεκριμένα η εντολή "break", τότε ο κώδικας μέσα στον βρόγχο επανάληψης "while" σταματάει να εκτελείται σε αυτό το σημείο και συνεχίζει να εκτελείται από το σημείο που η "while" τελειώνει. Η έννοια «τελειώνει» εννοεί ότι ο κώδικας συνεχίζει να εκτελείται μετά την τελευταία εμφωλευμένη εντολή που ανήκει στην "while".
      break
# Η "else" συνθήκη θα εκτελεστεί μόνο μία φορά και συγκεκριμένα στην πρώτη επανάληψη του βρόγχου "while" επειδή κατά τον έλεγχο του "fl" αυτό θα είναι διάφορο του "1" και συγκεκριμένα "0".
  else:
# Η εντολή που θα εκτελεστεί μία φορά μόνο, είναι η αλλαγή της τιμής της μεταβλητής "fl" από "0" σε "1", άρα η ενεργοποίηση από την επόμενη επανάληψη του βρόγχου "while" του ελέγχου για το κλείσιμο ή μη του παραθύρου που προβάλλει τα "frames".
    fl = 1

# Με την εντολή ".read()" πάνω στο αντικείμενο "vs" λαμβάνεται το στιγμιότυπο της κάμερας εκείνης της δεδομένης στιγμής ως εικόνα και αποθηκεύεται στη μεταβλητή "frame".
  frame = vs.read()

# Με την συνάρτηση ".resize()" της ενότητας "imutils" αλλάζουν οι διαστάσεις της εικόνας "frame" σε τέτοιες
```

```
ώστε να είναι ευδιάκριτη όταν προβληθεί αργότερα στο παράθυρο εμφάνισης του βίντεο. Το ".resize()" παίρνει ως πρώτο όρισμα το ίδιο το "frame" και σαν δεύτερο την διάσταση του νέου πλάτους που θέλουμε να προσαρμοστεί, δηλαδή "width=400". Το ύψος θα αλλάξει ανάλογα με το πλάτος, έτσι ώστε η νέα εικόνα να διατηρήσει την αναλογία (aspect ratio) της.
  frame = imutils.resize(frame, width=400)

# Γίνεται κλήση της συνάρτησης "detect_and_predict_mask()" η οποία δέχεται ως ορίσματα το "frame", το μοντέλο ανίχνευσης προσώπου (faceNet) και το μοντέλο ανίχνευσης μάσκας (maskNet). Όταν τερματίσει η εκτέλεση της συνάρτησης, αυτή επιστρέφει σε μορφή πλειάδας τις λίστες "locs" και "preds" που περιέχουν αντίστοιχα τις συντεταγμένες των νοητών παραλληλογράμων γύρω από τα ανθρώπινα πρόσωπα, καθώς και τις προβλέψεις ανίχνευσης ή μη μάσκας.
  (locs, preds) = detect_and_predict_mask(frame, faceNet, maskNet)

# Δημιουργία βρόγχου επανάληψης "for" που θα ελέγχει κάθε περίπτωση εντοπισμού προσώπου μέσα στο "frame" ξεχωριστά, άρα και πρόβλεψη ανίχνευσης ή μη μάσκας σε αυτό, θα εμφανίζει τις απαραίτητες πληροφορίες για αυτά. Ειδικότερα, ο εμφωλευμένος κώδικας της "for" θα πλαισιώνει με ένα ορθογώνιο παραλληλόγραμμο συγκεκριμένου χρώματος το ανθρώπινο πρόσωπο και θα εμφανίζει πληροφορίες σε μορφή κειμένου για την ύπαρξη ή μη μάσκας και την πιθανότητα της πρόβλεψης αυτής σαν ποσοστό. Στην αρχή κάθε επανάληψης του "for" θα αποθηκεύονται στις μεταβλητές "box" και "pred" οι αντίστοιχες τιμές από το "locs" και "preds". Η εντολή "zip()" χρησιμοποιείται πάνω στα "(locs, preds)" έτσι ώστε οι τιμές που θα καταχωρούνται από αυτά στα "box" και "pred" να είναι αντιστοιχισμένες μεταξύ τους και να μην μπερδεύονται.
  for (box, pred) in zip(locs, preds):

# Αποθήκευση των τεσσάρων συντεταγμένων που βρίσκονται στο "box" σε τέσσερις επιμέρους μεταβλητές. Οι συντεταγμένες "startX" και "startY" αντιπροσωπεύουν την πάνω αριστερή γωνία του παραλληλογράμμου που θα περιέχει ένα ανθρώπινο πρόσωπο ενώ οι "endX" και "endY" αντιπροσωπεύουν την κάτω δεξιά γωνία.
    (startX, startY, endX, endY) = box

# Εναπόθεση των δύο πιθανοτήτων που περιέχει η λίστα "pred[0]" στις μεταβλητές "mask" και "withoutMask". Λόγω του ότι οι πιθανότητες αυτές παράχθηκαν μέσω ενός τανυστή εξόδου, πρέπει το "pred" να «δείχνει» στον δείκτη "0" της λίστας όπου εκεί συγκεκριμένα βρίσκονται αποθηκευμένες οι πιθανότητες που χρειάζονται.
    (mask, withoutMask) = pred[0]

# Έλεγχος με τη συνθήκη "if-else" εάν η πιθανότητα να φοράει το πρόσωπο που εντοπίστηκε μάσκα είναι μεγαλύτερη της πιθανότητας να μη φοράει μάσκα. Εάν φοράει τότε δίνεται στη μεταβλητή "label" το string "Mask" αλλιώς δίνεται το string "No Mask". Το "label" είναι η μεταβλητή που θα εμφανίσει το αλφαριθμητικό που περιέχει στην οθόνη ως κείμενο αργότερα.
    label = "Mask" if mask > withoutMask else "No Mask"

# Έλεγχος του περιεχομένου της μεταβλητής "label" όπου εάν αυτό είναι "Mask" αποθηκεύεται στην μεταβλητή "color" το χρώμα πράσινο με τη μορφή BGR (Blue, Green, Red) δηλαδή "(0, 255, 0)". Εάν είναι "No Mask" γίνεται αποθήκευση του κόκκινου χρώματος, δηλαδή "(0, 0, 255)". Το "color" περιέχει το χρώμα του παραλληλογράμμου που θα σχεδιαστεί γύρω από το ανθρώπινο πρόσωπο.
    color = (0, 255, 0) if label == "Mask" else (0, 0, 255)

# Επεξεργασία του ήδη υπάρχοντος "label" και προσθήκη μέσα σε αυτό της πιθανότητας που αντιστοιχεί στο "label" με μορφή ποσοστού. Το "{}: {:.2f}%" είναι το string που θα εμφανίζεται στην οθόνη πάνω από κάθε ανθρώπινο πρόσωπο όπου το "{}" θα περιέχει το περιεχόμενο του πρώτου ορίσματος της εντολής ".format()" ενώ το "{:.2f}" θα περιέχει το περιεχόμενο του δεύτερου ορίσματος με δύο δεκαδικά ψηφία μόνο. Το ".format()" περιέχει ως πρώτο όρισμα την μεταβλητή "label" ενώ ως δεύτερο όρισμα την μεγαλύτερη πιθανότητα από τις "mask" και "withoutMask" πολλαπλασιασμένη επί "100".
    label = "{}: {:.2f}%".format(label, max(mask, withoutMask) * 100)

# Με την εντολή ".putText()" της βιβλιοθήκης OpenCV εμφανίζεται πάνω στο "frame" το κείμενο που περιέχει το "label". Επειδή όμως το κείμενο δεν μπορεί να εμφανιστεί σε οποιοδήποτε σημείο με οποιοδήποτε χρώμα κ.λπ., αυτά όλα ρυθμίζονται από τα ορίσματα που δίνονται στο ".putText()". Το πρώτο όρισμα είναι το "frame" στο οποίο θα εμφανιστεί το κείμενο. το δεύτερο είναι το κείμενο που θα εμφανιστεί και περιέχεται στο "label". Το τρίτο είναι οι συντεταγμένες που θα ξεκινάει το κείμενο δηλαδή "(startX, startY - 10)" που σημαίνει ότι αυτό θα ξεκινάει 10 pixel πιο ψηλά από την πάνω αριστερά γωνία του ορθογώνιου παραλληλογράμμου. Το τέταρτο είναι η επιλογή της γραμματοσειράς του κειμένου όπου έχει επιλεχθεί να είναι η "FONT_HERSHEY_SIMPLEX". Το πέμπτο είναι το μέγεθος της γραμματοσειράς όπου έχει οριστεί στα 0.45. Το έκτο είναι το χρώμα του κειμένου και το έβδομο το πόσο έντονη θα είναι η γραμματοσειρά.
    cv2.putText(frame, label, (startX, startY - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.45, color, 2)
```

```
# Η εντολή ".rectangle()" της OpenCV λειτουργεί με την ίδια λογική όπως η εντολή ".putText()", απλά σκοπός της είναι η εμφάνιση του ορθογώνιου παραλληλογράμμου περιγράμματος γύρω από ένα ανθρώπινο πρόσωπο. Εδώ το όρισμα κειμένου έχει αφαιρεθεί, αφού δεν υπάρχει, όπως και η γραμματοσειρά μαζί με το μέγεθός της. Παρ' όλα αυτά έχει επιλεχθεί το ίδιο χρώμα με το κείμενο και το πόσο έντονες/παχιές θα είναι οι πλευρές του σχήματος.
    cv2.rectangle(frame, (startX, startY), (endX, endY), color, 2)

# Αφού το "frame" έχει υποστεί όλες τις απαραίτητες επεξεργασίες και έχουν σχεδιαστεί πάνω του όλα όσα έχει ανιχνεύσει, εμφανίζεται στην οθόνη με την εντολή "imshow()" της βιβλιοθήκης OpenCV. Τα ορίσματα της εντολής είναι το όνομα του παραθύρου που δημιουργείται και η εικόνα, δηλαδή το "frame", που θα εμφανιστεί μέσα σε αυτό. Επειδή όλος ο κώδικας μέσα στον βρόγχο επανάληψης "while" εκτελείται συνεχώς και πολύ γρήγορα, τα frames που εμφανίζονται μέσα στο παράθυρο το ένα μετά το άλλο φαίνονται σαν βίντεο και όχι σαν μεμονωμένες εικόνες.
  cv2.imshow("Mask Detection", frame)

# Με την εντολή "waitKey(1)" της βιβλιοθήκης OpenCV το πρόγραμμα περιμένει για ένα millisecond και ελέγχει εάν πατήθηκε κάποιο πλήκτρο. Εάν πατήθηκε, τότε η μοναδική δυαδική τιμή που αντιστοιχεί στο συγκεκριμένο πλήκτρο θα υποστεί την λογική πράξη AND με τον δυαδικό αριθμό 0xFF (255 στο δεκαδικό). Το αποτέλεσμα αυτής της πράξης θα αποθηκευτεί στην μεταβλητή "key". Αυτή η πράξη ουσιαστικά μετατρέπει τον δυαδικό αριθμό σε δεκαδικό μεταξύ του 0 και του 255 και γίνεται για λόγους συμβατότητας με άλλες πλατφόρμες.
  key = cv2.waitKey(1) & 0xFF

# Γίνεται έλεγχος μέσω της συνθήκης "if", μεταξύ της τιμής του "key" και του αριθμού "27" που αντιστοιχεί στο πλήκτρο "ESC". Εάν οι δύο τιμές είναι ίδιες τότε η εμφωλευμένη εντολή του "if" εκτελείται, αλλιώς ο κώδικας συνεχίζει και ο βρόγχος επανάληψης "while" ξεκινά από την αρχή.
  if key == 27:

# Εάν η συνθήκη "if" αποδειχθεί αληθής, τότε εκτελείται η εντολή "break" και ο κώδικας μέσα στον βρόγχο επανάληψης "while" σταματάει να εκτελείται σε αυτό το σημείο και συνεχίζει να εκτελείται από το επόμενο σημείο έξω από αυτόν τον βρόγχο και συγκεκριμένα με την εντολή "vs.stream.release()".
    break

# Αφού ο χρήστης κλείσει το παράθυρο πληκτρολογώντας το "ESC" ή πατώντας το σύμβολο "X" πάνω δεξιά του παραθύρου, εκτελείται η εντολή ".stream.release()" πάνω στο αντικείμενο της ροής βίντεο και την απενεργοποιεί. Έτσι η κάμερα παύει να χρησιμοποιείται.
vs.stream.release()

# Η εντολή ".destroyAllWindows()" της OpenCV κλείνει ό,τι παράθυρο υπάρχει ανοικτό στην μνήμη του υπολογιστή και έπειτα το πρόγραμμα τερματίζεται επιτυχώς.
cv2.destroyAllWindows()
```