ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ
ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ
Τμήμα Ηλεκτρολόγων & Ηλεκτρονικών Μηχανικών
www.uniwa.gr
Θηβών 250, Αθήνα-Αιγάλεω 12244
Τηλ.+30 210 538-1225, Fax.+30 210 538-1226

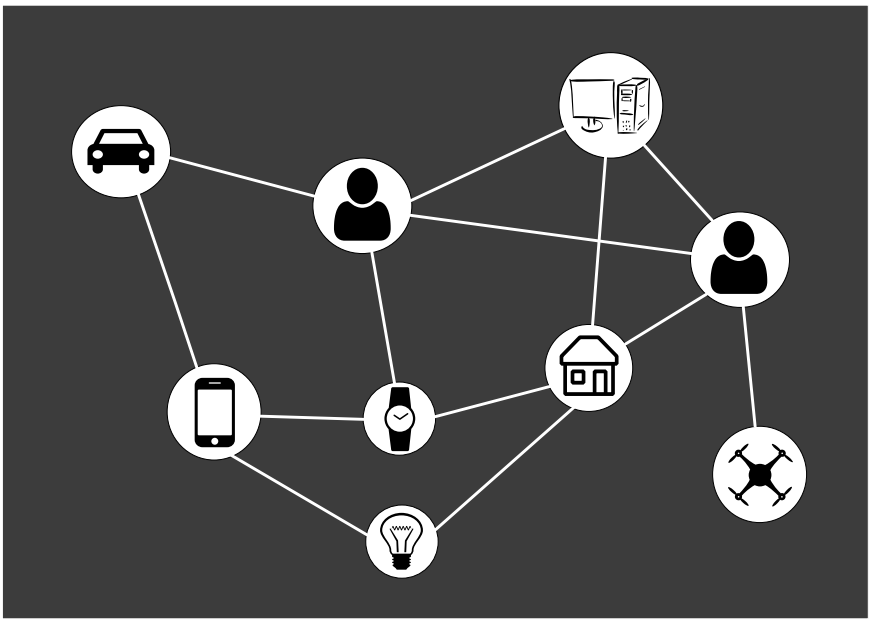
UNIVERSITY of WEST ATTICA
FACULTY OF ENGINEERING
Department of Electrical & Electronics Engineering
www.uniwa.gr
250, Thivon Str., Athens, GR-12244, Greece
Tel:+30 210 538-1225, Fax:+30 210 538-1226

Πρόγραμμα Μεταπτυχιακών Σπουδών
Ηλεκτρικές & Ηλεκτρονικές Επιστήμες μέσω Έρευνας

Master of Science By Research in
Electrical & Electronics Engineering

ΜΕΤΑΠΤΥΧΙΑΚΗ ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Ομότιμη, συνεργατική αλληλεπίδραση ανθρώπου – μηχανής



Μεταπτυχιακός Φοιτητής: Λάζαρος Τουμανίδης, ΑΜ MSCRES-0018
Επιβλέπων: Πατρικάκης, Χαράλαμπος, Καθηγητής

ΑΙΓΑΛΕΩ, Απρίλιος 2020

ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ
ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ
Τμήμα Ηλεκτρολόγων & Ηλεκτρονικών Μηχανικών
www.uniwa.gr
Θηβών 250, Αθήνα-Αιγάλεω 12244
Τηλ.+30 210 538-1225, Fax.+30 210 538-1226

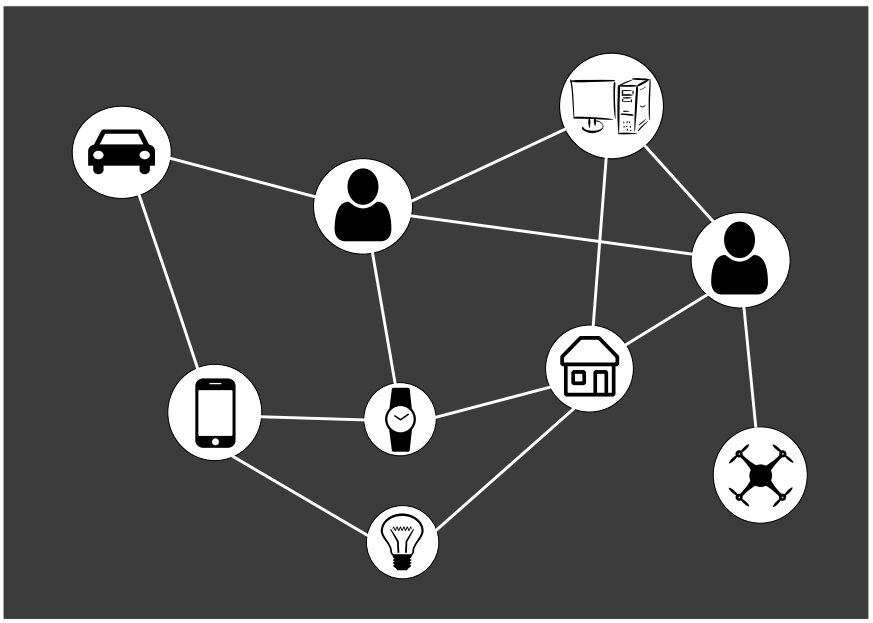
UNIVERSITY of WEST ATTICA
FACULTY OF ENGINEERING
Department of Electrical & Electronics Engineering
www.uniwa.gr
250, Thivon Str., Athens, GR-12244, Greece
Tel:+30 210 538-1225, Fax:+30 210 538-1226

Πρόγραμμα Μεταπτυχιακών Σπουδών
Ηλεκτρικές & Ηλεκτρονικές Επιστήμες μέσω Έρευνας

Master of Science By Research in
Electrical & Electronics Engineering

MSC Thesis

Peer-to-peer, collaborative human-machine interaction



Student: Lazaros Toumanidis, Registration Number MSCRES-0018
MSc Thesis Supervisor: Patrikakis, Charalampos, Professor

ATHENS-EGALEO, April 2020

Η Μεταπτυχιακή Διπλωματική Εργασία έγινε αποδεκτή, εξετάστηκε και βαθμολογήθηκε από την εξής τριμελή εξεταστική επιτροπή:

Επιβλέπων	Μέλος	Μέλος
Digitally signed by Charalampos Patrikakis Date: 2022.02.02 12:26:06 +02'00'	Digitally signed by Grigorios Kaltsas Date: 2022.02.02 13:03:29 +02'00'	Evangelos Valamontes Digitally signed by Evangelos Valamontes Date: 2022.02.02 13:09:45 +02'00'
Πατρικάκης Χαράλαμπος	Καλτσάς Γρηγόριος	Βαλαμόντες Ευάγγελος
Καθηγητής	Καθηγητής	Καθηγητής

ΔΗΛΩΣΗ ΣΥΓΓΡΑΦΕΑ ΜΕΤΑΠΤΥΧΙΑΚΗΣ ΔΙΠΛΩΜΑΤΙΚΗΣ ΕΡΓΑΣΙΑΣ

Ο κάτωθι υπογεγραμμένος Τουμανίδης Λάζαρος του Γεωργίου, με αριθμό μητρώου MSCRES-0018 φοιτητής του Προγράμματος Μεταπτυχιακών Σπουδών «Ηλεκτρικές και Ηλεκτρονικές Επιστήμες μέσω Έρευνας» του Τμήματος Ηλεκτρολόγων και Ηλεκτρονικών Μηχανικών της Σχολής Μηχανικών του Πανεπιστημίου Δυτικής Αττικής, δηλώνω ότι:

«Είμαι συγγραφέας αυτής της μεταπτυχιακής 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 εμένα 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 πτυχίου μου».

Ο Δηλών

Τουμανίδης Λάζαρος

ΠΕΡΙΛΗΨΗ

Μια συνήθης πρακτική σε λύσεις μηχανικής μάθησης, είναι η συνεχής χρήση της ανθρώπινης ευφυίας, με σκοπό την βελτίωση της ποιότητας και αποδοτικότητάς τους. Το αντικείμενο της παρούσας εργασίας είναι η μελέτη των τρόπων που μπορούμε να συνδυάσουμε την ανθρώπινη και της μηχανική ευφυΐα με σκοπό την βελτίωση των λύσεων που προσφέρουν τα συστήματα μηχανικής μάθησης. Μελετάται το πρόβλημα της ανάγκης για συλλογή και τιτλοφόρηση μεγάλου όγκου δεδομένων στα συστήματα αυτά, οι τεχνικές της μεταφοράς μάθησης και της ενεργητικής μάθησης, που χρησιμοποιούνται για την μείωση του όγκου αυτού καθώς και ο ρόλος που παίζει η ανθρώπινη κρίση κατά τις διαδικασίες αυτές.

Παρουσιάζουμε μια ολοκληρωμένη πρακτική υλοποίηση, που περιλαμβάνει των συνδυασμού των τεχνικών αυτών, με την εκμετάλλευση προ-εκπαιδευμένων μοντέλων και την επιλεκτική δειγματοληψία δειγμάτων προς τιτλοφόρηση, καθώς και μια διαπλατφορμική εφαρμογή κινητών συσκευών για της διαδικασία της τιτλοφόρησης των επιλεγμένων δειγμάτων. Η υλοποίηση αποτελείται από τρία διακριτά μέρη, αυτό της εφαρμογής των τεχνικών μηχανικής μάθησης, αυτό της εφαρμογής για κινητές συσκευές και αυτό της διαδικτυακής εφαρμογής που χρησιμοποιείται ως διεπαφή των υπόλοιπων δύο, αλλά και για την αποθήκευση τόσο των παραμέτρων των διεργασιών αλλά και των εκάστοτε σταδίων αυτών.

Αξιολογούμε την υλοποίηση αυτή εφαρμόζοντας ένα πρόβλημα δυαδικής κατηγοριοποίησης στον τομέα της μηχανικής όρασης, με χρήση ενός προ-εκπαιδευμένου μοντέλου για αρχικοποίηση της διαδικασίας, το κριτήριο της μέγιστης εντροπίας σαν κριτήριο επιλογής νέων δειγμάτων στο στάδιο της μεταφοράς μάθησης, συγκρινόμενο με την τυχαία επιλογή δειγμάτων, και μηχανισμό ειδικής πλειοψηφίας σαν κριτήριο αξιοπιστίας των απαντήσεων στην διαδικασία τιτλοφόρησης νέων δεδομένων για την προσθήκη τους στο σύνολο δεδομένων εκπαίδευσης. Εξασφαλίζουμε ασφαλή επικοινωνία ανάμεσα στην εφαρμογή κινητών συσκευών και της διαδικτυακής εφαρμογής, και αποστέλλουμε στην δεύτερη, τις απαντήσεις που έδωσαν οι χρήστες στην πρώτη.

Παρουσιάζουμε τα αποτελέσματα από τα οποία εύλογα διαφαίνεται η μείωση των απαιτούμενων δεδομένων για την βελτίωση του εκπαιδευμένου μοντέλου και συζητάμε σχετικά με τις δυσκολίες που αντιμετωπίσαμε αλλά και τις προοπτικές που ανοίγονται για την περαιτέρω αξιοποίηση της υλοποίησής μας.

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: Ενεργητική Μάθηση, Κινητή Υπολογιστική, Μεταφορά Μάθησης, Μηχανική Μάθηση, Πληθοπορισμός

ABSTRACT

A widespread practice in machine learning solutions is the continuous use of human intelligence to increase their quality and efficiency. The subject of this work is the study of the ways we can combine human and machine intelligence in order to improve machine learning related systems. We review the requirement of a large amount of data that is needed in such systems, the techniques of transfer and active learning that are used to reduce this amount, as well as the role that human play in these techniques.

We present a complete practical implementation that includes the combination of these techniques, utilizing existing pre-trained models for the tasks initialization, sophisticated data sampling to use for annotation as well as a cross-platform mobile application that is used for the annotation of the selected samples. The implementation consists of the three separate modules, one responsible for machine learning related tasks, one being the mobile client application and a web application being the interface between the other two, also responsible for storing both the tasks' parameters and their current states.

We evaluate the implementation using a pre-trained computer vision model for task initialization, max entropy strategy as the uncertainty sampling method, compared to random sampling for the active learning part and majority voting as the quality assurance requirement for new human-annotated images to be added on the train set. We ensure secure communication between the mobile client and the web application and submit to the latter the answers that were given by the users on the former.

We present the achieved results, where the reduction of the required samples for the improvement of the trained model is clear, and discuss about the difficulties we faces during the whole process as well as the opportunities that are opening for further usage of our implementation.

KEYWORDS: Active Learning, Crowdsourcing, Machine Learning, Mobile Computing, Transfer Learning

ΕΥΧΑΡΙΣΤΙΕΣ

Η παρούσα εργασία αποτελεί την τελική αναφορά και ολοκλήρωση της εργασίας μου στα πλαίσια του Προγράμματος Μεταπτυχιακού Σπουδών “Ηλεκτρικές & Ηλεκτρονικές Επιστήμες μέσω Έρευνας”. Κατά τη διάρκεια της εργασίας είχα την τιμή να συνεργαστώ με αξιόλογους ανθρώπους, τους οποίους θα ήθελα να ευχαριστήσω για την πολύτιμη βοήθειά τους, καθώς και τη συνεχή στήριξη και καθοδήγηση που μου παρείχαν.

Αρχικά θα ήθελα να εκφράσω την τεράστια ευγνωμοσύνη μου στον επιβλέποντα καθηγητή της εργασίας μου όχι μόνο για την υποδειγματική του καθοδήγηση στα πλαίσια τς εργασίας και γενικότερα στην διαδικασία της έρευνας, αλλά και για την αμέριστη εμπιστοσύνη του καθ όλη τη διάρκεια της συνεργασίας μου μαζί του. Ένα τεράστιο ευχαριστώ οφείλω επίσης στον Παναγιώτη Κασνέση, χωρίς την καθοδήγηση και τις συμβουλές του οποίου η εργασία αυτή δεν θα είχε ολοκληρωθεί επιτυχώς. Ιδιαίτερα ευγνώμων είμαι και στους συνήθεις υπόπτους του χώρου στον οποίο πραγματοποιήθηκε το πειραματικό μέρος της εργασίας και είχα και έχω την χαρά να συναναστρέφομαι, Χριστίνα, Μαρία Κ, Μαρία Π, Έντα, Χρήστο Χ, Βασίλη, Δημήτρη, Μιχάλη Φ, Μιχάλη Ξ, Βαγγέλη, Χρήστο Γ, Νίκο.

Τέλος, θα ήθελα να ευχαριστήσω τους γονείς μου Γιώργο και Ευδοξία, την αδελφή μου Δέσποινα, τον σύντροφό της ζωής της Γιώργο, για την υπομονή τους και την ολόψυχη υποστήριξή τους και τις δύο μικρές μεγάλες μου αδυναμίες, τις ανιψιές μου Δανάη και Λυδία.

ΠΙΝΑΚΑΣ ΣΥΜΒΟΛΩΝ-ΑΚΡΩΝΥΜΙΩΝ-ΣΥΝΤΟΜΟΓΡΑΦΙΩΝ

Ακρωνύμιο	**Όρος στα Αγγλικά**	**Όρος / Επεξήγηση στα Ελληνικά**
AI	Artificial Intelligence	Τεχνητή Νοημοσύνη
ANN(s)	Artificial Neural Network(s),	Τεχνητό(ά) Νευρωνικό(ά) Δίκτυο(α)
API	Application Programming Interface,	Διεπαφή Προγραμματισμού Εφαρμογών
CNN	Convolutional Neural Network	Συνελικτικό Νευρωνικό Δίκτυο
ConvNet	Convolutional Network	Συνελικτικό Δίκτυο
CV	Computer Vision	Μηχανική Όραση
DNN	Deep Neural Network	Βαθύ Νευρωνικό Δίκτυο
GPU	Graphical Processing Unit	Μονάδα Επεξεργασίας Γραφικών
HTTP	Hypertext Transfer Protocol	Πρωτόκολλο μεταφοράς υπερκειμένου
ILSVRC	ImageNet Large Scale Visual Recognition Challenge	Διαγωνισμός οπτικής αναγνώρισης μεγάλης κλίμακας στην βάση οπτικών δεδομένων ImageNet
JSON	JavaScript Object Notation	Σημειογραφία Αντικειμένων στην γλώσσα JavaScript
ML	Machine Learning	Μηχανική Μάθηση
RL	Representational Learning	Μάθηση Χαρακτηριστικών
TLS	Transport Layer Security	Ασφάλεια Επιπέδου Μεταφοράς

ΚΑΤΑΛΟΓΟΣ ΕΙΚΟΝΩΝ

ΚΑΤΑΛΟΓΟΣ ΠΙΝΑΚΩΝ

ΚΑΤΑΛΟΓΟΣ ΑΡΧΕΙΩΝ

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

ΕΙΣΑΓΩΓΗ:
Αντικείμενο, ερευνητικά ερωτήματα και διάρθρωση της εργασίας

Χωρίς αμφιβολία, η μηχανική μάθηση, ως πεδίο της τεχνητής νοημοσύνης και κατά συνέπεια της τέταρτης βιομηχανικής επανάστασης [4], έχει αλλάξει σε σημαντικό βαθμό την καθημερινότητά μας. Ένα από σημαντικότερα εμπόδια και ενεργός ερευνητικός τομέας σε συστήματα μηχανικής μάθησης, είναι η απαίτηση μεγάλου όγκου συλλογής και τιτλοφόρησης δεδομένων, με τον άνθρωπο να παίζει καταλυτικό ρόλο στην βελτιστοποίηση των συστημάτων αυτών. Η μελέτη του συνδυασμού της ανθρώπινης και μηχανικής ευφυίας με σκοπό την βελτιστοποίηση των συστημάτων μηχανικής μάθησης αποτελεί το αντικείμενο της παρούσας εργασίας.

Μια συχνά χρησιμοποιούμενη τεχνική που μειώνει τον όγκο των δεδομένων που απαιτούνται κατά την διαδικασία της εκπαίδευση και επαλήθευσης των μοντέλων μηχανικής μάθησης είναι η τεχνική της Μεταφοράς Μάθησης (Transfer Learning). Βασικός στόχος της μεταφοράς μάθησης είναι η εκμετάλλευση δεδομένων από μία υπάρχουσα λύση σε συγκεκριμένο πρόβλημα με σκοπό την εξαγωγή πληροφορίας που μπορεί να φανεί χρήσιμη κατά τη διάρκεια εκπαίδευσης ή και πρόβλεψης σε μια ρύθμιση ενός διαφορετικού προβλήματος. Οι Pan και Yang στο [5] κατηγοριοποιούν την μεταφορά μάθησης με βάση τρία βασικά ερευνητικά ερωτήματα: 1) τι να μεταφερθεί, 2) πώς να μεταφερθεί, και 3) πότε να μεταφερθεί. Το "Τι να μεταφερθεί" αναφέρεται στο ποιο μέρος της υπάρχουσας γνώσης πρέπει να μεταφερθεί ανάμεσα στα στις ρυθμίσεις της πηγής και του δέκτη. Το "Πώς να μεταφερθεί" αναφέρεται στην μελέτη των αλγορίθμων και των τεχνικών που χρησιμοποιούνται στην μεταφορά της γνώσης, ενώ το "Πότε να μεταφερθεί", αναφέρεται στην εξέταση των προϋποθέσεων που πρέπει να τηρούνται για την μεταφορά της γνώσης, αποφεύγοντας την αποκαλούμενη "Αρνητική Μεταφορά" (Negative Transfer), όπου η μεταφορά της γνώσης αποφέρει αρνητικά αποτελέσματα ως προς την απόδοση στον αποδέκτη. Μια πρακτική μελέτη στην συμβατότητα και προσαρμογή των ρυθμίσεων της πηγής και του αποδέκτη κατά την μεταφορά μάθησης, παρουσιάζεται από τους Ali, Budka και Gabrys στο [6]. Οι συγγραφείς δίνουν έμφαση στα κοινά χαρακτηριστικά μεταξύ πηγής και δέκτη, καθώς και στην ποσοτικοποίηση συγκεκριμένων απαιτούμενων παραμέτρων για την βελτιστοποίηση της ακρίβειας του μοντέλου του αποδέκτη.

Σε ορισμένους από τους επιμέρους τομείς της μηχανικής μάθησης, επιπλέον μείωση του απαιτούμενου όγκου δεδομένων μπορεί να επιτευχθεί με την χρήση της τεχνικής της Ενεργητικής Μάθησης (Active Learning), μιας τεχνικής που αποσκοπεί στην ελαχιστοποίηση

των δεδομένων που χρειάζονται τιτλοφόρηση. Σε αντίθεση με την Παθητική Μάθηση (Passive Learning), όπου τα δεδομένα τιτλοφορούνται και χρησιμοποιούνται στο σύνολό τους, χωρίς κάποια αναφορά στον αλγόριθμο εκπαίδευσης, εδώ έχουμε την επιλεκτική δειγματοληψία από μία δεξαμενή δειγμάτων, με διάφορα κριτήρια σχετικά με την αξιοπιστία του συστήματος [7], που αποσκοπούν στην βελτίωση του αλγορίθμου εκπαίδευσης, μειώνοντας έτσι τα τελικά χρησιμοποιηθέντα δείγματα. Σε αυτό το σημείο είναι που αξιοποιείται η ανθρώπινη ευφυΐα, αφού ο πιο αξιόπιστος τρόπος τιτλοφόρησης των επιλεγμένων αυτών δειγμάτων είναι από τους ανθρώπους.

Ως πληθοπορισμό καλούμε την τεχνική ανάθεσης μιας εργασίας σε ένα πλήθος ανθρώπων [8]. Αποτελεί την πιο συνήθη και αξιόπιστη τεχνική συλλογής και τιτλοφόρησης δεδομένων που χρησιμοποιούνται σε συστήματα μηχανικής μάθησης. Διάφοροι παραγόμενοι ορισμοί προκύπτουν από την πληθοπορισμό, ανάλογα με το είδος των δεδομένων (κείμενο, πολυμέσα, δεδομένα αισθητηρίων), τον βαθμό συμμετοχής του ανθρώπου, την ομοιογένεια και τον τρόπο εξαγωγής της εξόδου από τις διάφορες συνεισφορές [9]. Οι Tian, Zhu, και Qiaoben στο [10], μελετούν τεχνικές τρόπου επιλεκτικής ανάθεσης εργασιών και προτείνουν μια τεχνική ειδικής πλειοψηφίας (Max-Margin Majority Voting) για την βελτίωση της ικανότητας επιλογής σε συστήματα πληθοπορισμού.

Στην εργασία αυτή, παρουσιάζουμε μια ολοκληρωμένη πρακτική εφαρμογή που συνδυάζει τις τρεις προαναφερθείσες τεχνικές, δηλαδή την μεταφορά μάθησης, την ενεργητική μάθηση και τον πληθοπορισμό με σκοπό την βελτιστοποίηση της διαδικασίας εκπαίδευσης μοντέλων μηχανικής μάθησης με χρήση όσο το δυνατόν μικρότερου αριθμού δεδομένων. Η τελική ανάπτυξη αποτελείται από τρία διακριτά στοιχεία, αυτό του κύκλου μηχανικής μάθησης, μιας εφαρμογής εξυπηρετητή που επικοινωνεί με το στοιχείο αυτό καθώς και μιας δια-πλατφορμικής εφαρμογής για κινητές συσκευές που χρησιμοποιείται στο στάδιο του πληθοπορισμού. Η διάρθρωση του υπόλοιπου μέρους της εργασίας έχει ως εξής:

- Στο Κεφάλαιο 1 θέτουμε το θεωρητικό πλαίσιο στο οποίο κινήθηκε η μελέτη μας, δίνοντας περισσότερες λεπτομέρειες για τις έννοιες που αναφέραμε. Πιο συγκεκριμένα, μετά από μια εισαγωγή στην μηχανική μάθηση και τον ρόλο της στην τεχνητή νοημοσύνη, προχωράμε στις διάφορες τεχνικές που συναντώνται στην μεταφορά μάθησης, καθώς και τις στρατηγικές επιλογής δειγμάτων για την ενεργητική μάθηση. Κλείνουμε το κεφάλαιο με τα βασικά χαρακτηριστικά και τις επιμέρους έννοιες συστημάτων πληθοπορισμού.
- Στο Κεφάλαιο 2 παρουσιάζουμε την ερευνητική προσέγγιση που ακολουθήσαμε, την μέθοδο και τα εργαλεία που χρησιμοποιήσαμε καθ όλη τη διάρκεια της ερευνητικής

διεργασίας, και περιγράφουμε τα χαρακτηριστικά της τελικής υλοποίησής.

- Με βασικό κορμό το θεωρητικό υπόβαθρο και τον τρόπο προσέγγισης που παρουσιάζουμε στα δύο πρώτα κεφάλαια, προχωράμε στον σχεδιασμό και την ανάπτυξη της τελικής υλοποίησης στο Κεφάλαιο 3. Ξεκινάμε με την αρχιτεκτονική του συστήματος στο σύνολό του, συνεχίζουμε εστιάζοντας στα επιμέρους στοιχεία του, τις επιμέρους υλοποιήσεις τους καθώς και την σύνδεση μεταξύ τους για την σύνθεση του τελικού συστήματος.
- Στο Κεφάλαιο 4 παρουσιάζουμε μια πρακτική εφαρμογή στην υλοποιούμενη λύση και συζητάμε για τα αποτελέσματά της καθώς και τα σημεία τα οποία δέχεται βελτιώσεις ή / και επεκτάσεις, και τέλος στο
- Στο Κεφάλαιο 5 παρουσιάζουμε τα συμπεράσματα της έρευνάς μας στο σύνολό της και τα σχετικά πεδία στα οποία υπάρχει ενεργό ερευνητικό ενδιαφέρον και στα οποία μπορεί η εργασία αυτή να φανεί αρωγός για μελλοντική ερευνητική δραστηριότητα.

ΚΕΦΑΛΑΙΟ 1:
Θεωρητικό πλαίσιο του θέματος – Ανασκόπηση του πεδίου

Στο κεφάλαιο αυτό παρουσιάζουμε τις βασικές θεωρητικές έννοιες που χρειάστηκαν κατά την εκπόνηση της ερευνητικής εργασίας. Ξεκινώντας με μια εισαγωγή στην μηχανική μάθηση έναν από τους κλάδους της τεχνητής νοημοσύνης, προχωρούμε στα κλάδο της βαθιάς μάθηση και προχωρούμε στις τεχνικές μεταφοράς μάθησης και ενεργής μάθησης, που αποτελούν τα δύο βασικά εργαλεία που χρησιμοποιήσαμε στην τελική μας υλοποίηση. Τέλος αναφέρουμε τις τεχνικές του πληθοπορισμού και της πληθαίσθησης σαν εργαλεία συλλογής και τιτλοφόρησης δεδομένων για την δημιουργία ικανοποιητικού μεγέθους συνόλου δειγμάτων για εκπαίδευση και δοκιμή μοντέλων μηχανικής μάθησης.

1.1 Μηχανική Μάθηση

Ο Alan Turing([11], [12]) το 1950 ανέφερε πως "Η ιδέα πίσω από τους ψηφιακούς υπολογιστές, μπορεί να εξηγηθεί, λέγοντας πως οι μηχανές αυτές προορίζονται στο να πραγματοποιούν οποιονδήποτε υπολογισμό θα μπορούσε να κάνει ένας ανθρώπινος υπολογιστής". Σύμφωνα με τον Samuel [13], η μηχανική μάθηση μπορεί να θεωρηθεί ως ένα πεδίο μελέτης στο οποίο δίνεται η ικανότητα στους υπολογιστές να αποκτήσουν γνώση χωρίς να είναι ρητά προγραμματισμένοι για κάτι τέτοιο. Ένας σχετικά γενικός ορισμός της Μηχανικής Μάθησης δίνεται από τον Mitchell [14] το 1997: "Ένα πρόγραμμα υπολογιστή λέμε ότι μαθαίνει από την εμπειρία E ως προς κάποια κλάση εργασιών T και μέτρο απόδοσης P, αν η απόδοσή του σε εργασίες από το T, όπως μετριέται από το P, βελτιώνεται μέσω της εμπειρίας E." Ανάλογα την φύση του προβλήματος, οι τεχνικές μηχανικής μάθησης μπορούν να χωριστούν σε τρεις μεγάλες κατηγορίες: την μάθηση με επίβλεψη, την μάθηση χωρίς επίβλεψη και την ενισχυτική μάθηση [1].

1.1.1 Μάθηση με επίβλεψη (Supervised Learning)

Είναι η διαδικασία όπου ο αλγόριθμος κατασκευάζει μια συνάρτηση που απεικονίζει δεδομένες εισόδους (σύνολο εκπαίδευσης) σε γνωστές επιθυμητές εξόδους, με απώτερο στόχο τη γενίκευση της συνάρτησης αυτής και για εισόδους με άγνωστη έξοδο. Χρησιμοποιείται σε προβλήματα:

- **Ταξινόμησης (Classification)**, προβλήματα δηλαδή προσδιορισμού της κλάσης (κατηγορίας) στην οποία ανήκει μια νέα παρατήρηση. Οι πιθανές κλάσεις είναι

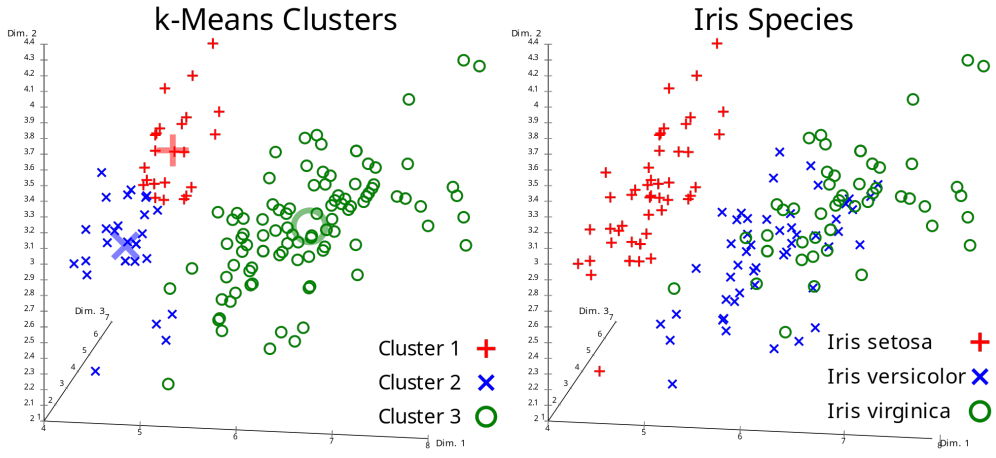
διακριτές και εκ των προτέρων καθορισμένες. Ανάλογα τον αριθμό των κλάσεων εξόδου, τα προβλήματα μπορεί να είναι δύο (binary) ή πολλαπλών κλάσεων (multi-class).

- **Παρεμβολής (Regression)**, όπου δημιουργούνται μοντέλα πρόβλεψης αριθμητικών τιμών. Σε αυτή την περίπτωση οι τιμές εξόδου παίρνουν συνεχείς τιμές. Παράδειγμα παρεμβολής είναι η πρόβλεψη της ταχύτητας του ανέμου (έξοδος), με βάση την θερμοκρασία, την ατμοσφαιρική πίεση την σχετική υγρασία (είσοδοι).

1.1.2 Μάθηση χωρίς επίβλεψη (Unsupervised Learning)

Στην κατηγορία αυτή ο αλγόριθμος κατασκευάζει ένα μοντέλο για κάποιο σύνολο εισόδων υπό μορφή παρατηρήσεων χωρίς να γνωρίζει τις επιθυμητές εξόδους. Χρησιμοποιείται σε προβλήματα:

- **Ανάλυσης Συσχετισμών (Association Analysis)**, στα οποία ανακαλύπτονται σχέσεις μεταξύ των δεδομένων εισόδου, όπως για παράδειγμα η δημιουργία κανόνων με συσχετισμούς αντικειμένων που αγοράζονται σε μια αγορά [15].
- **Ομαδοποίησης (Clustering)**, της εύρεσης δηλαδή κοινών χαρακτηριστικών και δημιουργίας διακριτού αριθμού ομάδων στα οποία ανήκουν τα στοιχεία που έχουν τα χαρακτηριστικά αυτά, όπως για παράδειγμα η ομαδοποίηση των ιριδοειδών φυτών σε σχέση με το μέγεθος των φύλλων και του πέταλου των ανθών τους (Εικόνα 2).



Εικόνα 1: Ομαδοποίηση ιριδοειδών φυτών[1]

1.1.3 Ενισχυτική Μάθηση (Reinforcement Learning)

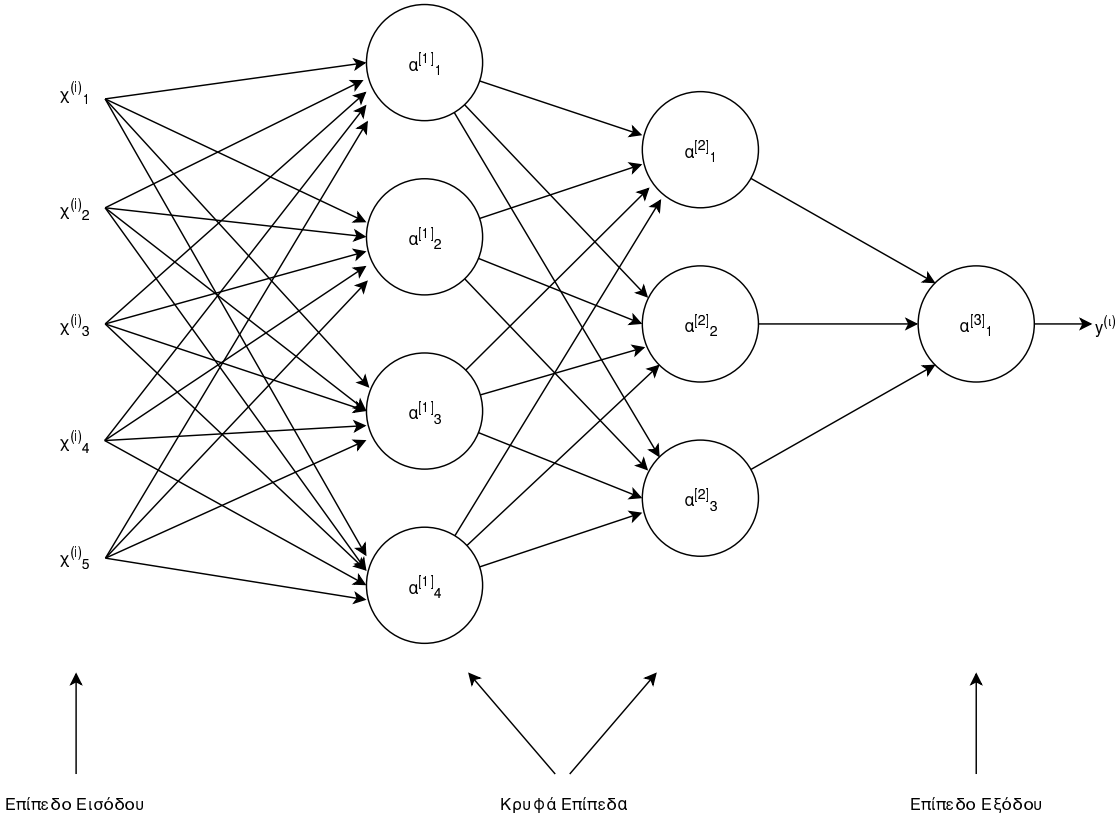
Εδώ ο αλγόριθμος μαθαίνει μια στρατηγική ενεργειών μέσα από άμεση αλληλεπίδραση με το περιβάλλον. Χρησιμοποιείται κυρίως σε προβλήματα Σχεδιασμού (Planning), όπως για

παράδειγμα ο έλεγχος κίνησης ρομπότ και η βελτιστοποίηση εργασιών σε εργοστασιακούς χώρους.

[1] https://commons.wikimedia.org/wiki/File:Iris_Flowers_Clustering_kMeans.svg

1.1.4 Νευρωνικά Δίκτυα

Η διαφορετική αρχιτεκτονική ενός ηλεκτρονικού υπολογιστή και του ανθρώπινου εγκεφάλου είναι κατά γενική πεποίθηση [2] ο λόγος για τον οποίο ο άνθρωπος υπερτερεί σε δεξιότητες όπως η αναγνώριση πραγμάτων ή καταστάσεων και οι συσχετίσεις αυτών ενώ οι υπολογιστές υπερτερούν σε δεξιότητες όπως η ακρίβεια και η ταχύτητα αριθμητικών υπολογισμών.
Συγκεκριμένα, από τη μια μεριά, ένας ηλεκτρονικός υπολογιστής έχει λίγες αλλά πολύπλοκες μονάδες επεξεργασίας πληροφοριών. Από την άλλη μεριά, ο ανθρώπινος εγκέφαλος αποτελείται από πολλά δισεκατομμύρια απλές μονάδες επεξεργασίας πληροφοριών που ονομάζονται νευρώνες. Κάθε νευρώνας μπορεί να επικοινωνεί με χιλιάδες άλλους μέσω αποφυάδων που ονομάζονται δενδρίτες. Κατά τη λειτουργία του εγκεφάλου ηλεκτρική διέγερση ενός νευρώνα μπορεί να μεταφερθεί προς κάθε νευρώνα με τον οποίο αυτός είναι διασυνδεδεμένος. Έτσι, λειτουργεί ο ανθρώπινος εγκέφαλος εκτελώντας, μαζικά, μια παράλληλη και κατανεμημένη επεξεργασία ηλεκτρικών σημάτων. Η μελέτη αυτή της λειτουργίας του ανθρώπινου εγκεφάλου, οδήγησε στον κλάδο της μηχανικής μάθησης που χρησιμοποιεί τεχνητά νευρωνικά δίκτυα(Artificial Neural Networks), επιδιώκοντας έτσι να προσομοιώσει της λειτουργία του εγκεφάλου.

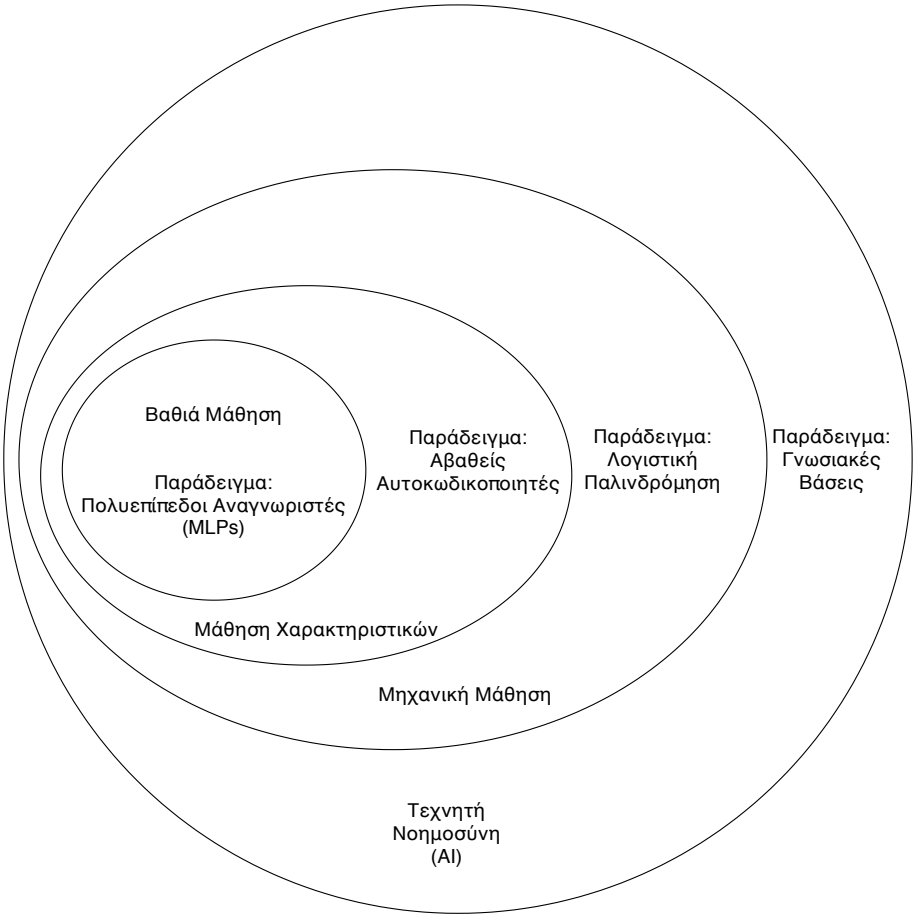


Εικόνα 2: Αναπαράσταση πολυεπίπεδου νευρωνικού δικτύου

Ένας τεχνητός νευρώνας αποτελείται από μία σειρά από εισόδους, οι οποίες πολλαπλασιάζονται με ισάριθμα βάρη και στη συνέχεια αθροίζονται. Το αποτέλεσμα οδηγείται στη συνέχεια σε ένα μη-γραμμικό στοιχείο παραμόρφωσης, την συνάρτηση μεταφοράς, η οποία παράγει την έξοδο του νευρώνα. Η έξοδος του νευρώνα μπορεί να είναι η επιθυμητή προσέγγιση της λύσης του προβλήματος, αλλά μπορεί να είναι και είσοδος σε έναν επόμενο νευρώνα, με αποτέλεσμα την δυνατότητα δημιουργίας ενός δικτύου διασυνδεδεμένων νευρώνων αντίστοιχου αυτού του εγκεφάλου (εικόνα 2). Σε ένα τέτοιο πολυεπίπεδο δίκτυο, τα ενδιάμεσα επίπεδα ανάμεσα στις εισόδους και το τελικό αποτέλεσμα καλούνται κρυφά επίπεδα και επιλογή του αριθμού τους εξαρτάται από το εκάστοτε πρόβλημα. Η μάθηση ενός νευρωνικού δικτύου είναι η διαδικασία υπολογισμού των βαρών των συνδέσμων του και όπως είδαμε προηγουμένως, μπορεί να γίνει είτε με εποπτεία ή αλλιώς επίβλεψη (σε περιπτώσεις όπου η έξοδος είναι μία ή περισσότερες γνωστές διακριτές τιμές, για παράδειγμα στην κατηγοριοποίηση), είτε χωρίς χωρίς εποπτεία (σε περιπτώσεις όπως η ομαδοποίηση).

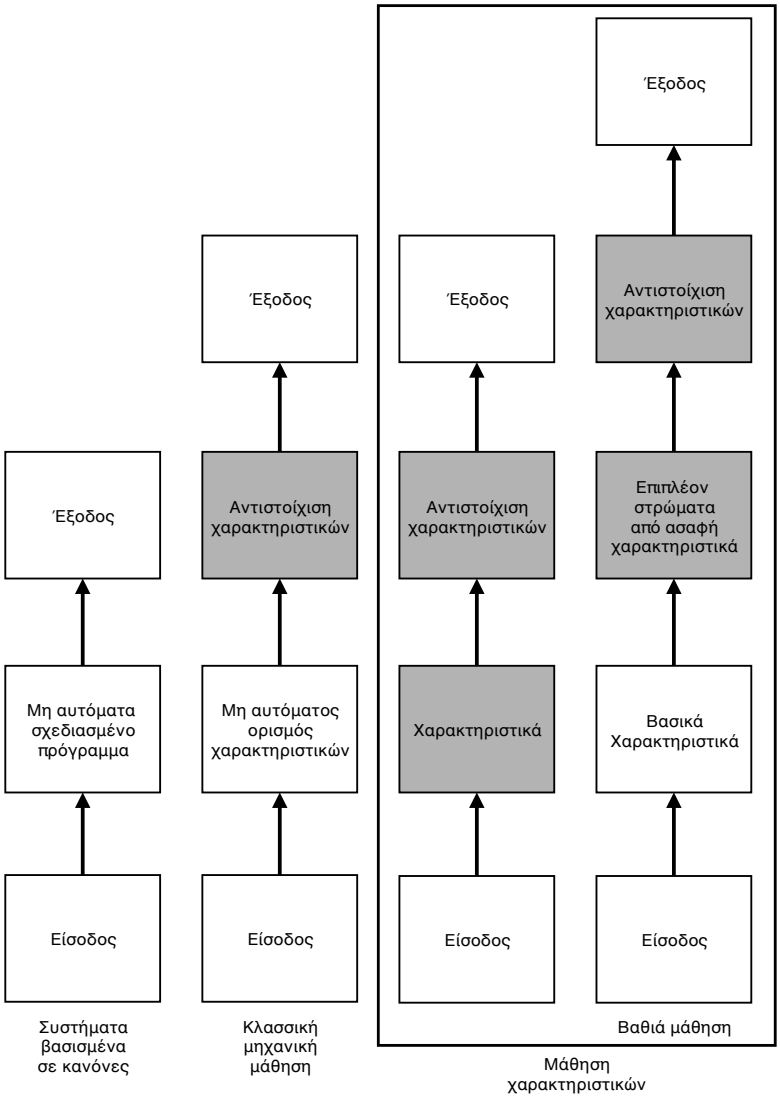
1.1.5 Βαθιά Μάθηση

Η Βαθιά Μάθηση (Deep Learning, DL) είναι ένας κλάδος της Μηχανικής Μάθησης και πιο συγκεκριμένα, ένας όρος που δηλώνει ένα Βαθύ Νευρωνικό Δίκτυο (Deep Neural Network, DNN). Η ονομασία του οφείλεται στο γεγονός ότι σε αντίθεση με τα συμβατικά Τεχνητά Νευρωνικά Δίκτυα (Artificial Neural Networks, ANNs), αποτελείται από κρυφά επίπεδα. Στην εικόνα 3 μπορούμε να δούμε τις εμφωλευμένες σχέσεις μεταξύ της Βαθιάς Μάθησης, της Μάθησης Χαρακτηριστικών (Representational Learning, RL), της Μηχανικής Μάθησης (Machine Learning, ML) και της Τεχνητής Νοημοσύνης (Artificial Inteligence, AI). Όπως βλέπουμε στην εικόνα, η Μηχανική Μάθηση, ως σύνολο αλγορίθμων που τροφοδοτούνται από επεξεργασμένα δεδομένα και κάνουν προβλέψεις για αυτά, αποτελεί έναν από τους κλάδους της τεχνητής νοημοσύνης. Σε αντίθεση με συστήματα που βασίζονται σε κανόνες (Rule based systems), όταν τα πρότυπα που πρέπει να ανακαλυφθούν είναι πολύπλοκα, οι αλγόριθμοι αυτοί έχουν καλύτερα αποτελέσματα.



Εικόνα 3: Venn διάγραμμα που παρουσιάζει τις σχέσεις μεταξύ Τεχνητής Νοημοσύνης, Μηχανικής Μάθησης, Μάθησης Χαρακτηριστικών και Βαθιάς Μάθησης [16]

Στην Μάθηση Χαρακτηριστικών έχουμε εξαγωγή χαρακτηριστικών υψηλού επιπέδου, όπως για παράδειγμα η αναγνώριση των ακμών των αντικειμένων μιας εικόνας. Η διαφοροποίηση της Βαθιάς Μάθησης έγκειται στο γεγονός ότι η εξαγωγή των χαρακτηριστικών αυτών γίνεται αυτόματα και όχι από κάποιους εμπειρογνώμονες. Στην εικόνα 4 μπορούμε να δούμε τα σχετικά διαγράμματα ροής που παρουσιάζουν πώς οι διάφορες διαδικασίες ενός συστήματος τεχνητής νοημοσύνης σχετίζονται μεταξύ τους στο πλαίσιο διαφορετικών επιστημονικών κλάδων αυτής. Τα σκιασμένα κουτιά υποδηλώνουν συστατικά στοιχεία που έχουν τη δυνατότητα να μαθαίνουν, από μόνα τους, κάνοντας χρήση των δεδομένων εισόδου.

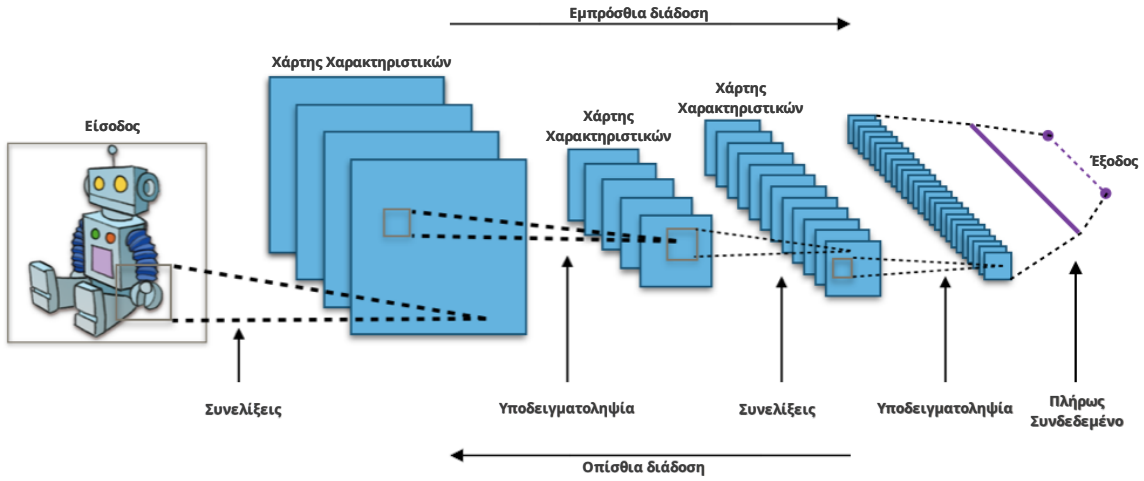


Εικόνα 4: Διαγράμματα ροής που παρουσιάζουν πώς οι διάφορες διαδικασίες ενός συστήματος AI σχετίζονται μεταξύ τους στο πλαίσιο διαφορετικών επιστημονικών κλάδων της AI. [16]

Η Βαθιά Μάθηση παρουσιάστηκε το 2007, υπό τον όρο Αυτοδίδακτη Μάθηση (Self-Taught Learning) [17], ωστόσο μεγάλη προσοχή κέρδισε μόλις τα τελευταία χρόνια. Προσπάθειες προσθήκης κρυφών επιπέδων σε ένα Τεχνητό Νευρωνικό Δίκτυο δεν είχαν ικανοποιητικά αποτελέσματα λόγω του γεγονότος ότι [3] :

- τα σύνολα δεδομένων που ήταν επισημασμένα ήταν πολύ μικρά σε μέγεθος εκείνες τις ημέρες,
- οι δυνατότητες επεξεργασίας των υπολογιστών ήταν πολύ περιορισμένες,
- η αρχικοποίηση των βαρών δεν γινόταν σωστά, και
- οι μη γραμμικές συναρτήσεις που εφαρμόζονταν, όπως είναι η σιγμοειδής συνάρτηση (sigmoid) και η συνάρτηση εφαπτομένης (tanh), ήταν λανθασμένες.

Τα τελευταία δύο γεγονότα οδήγησαν στο πρόβλημα της εξαφάνισης της κλίσης της παραγώγου (vanishing gradient problem), κατά το οποίο το σφάλμα ταξινόμησης που χρησιμοποιείται για την οπίσθια διάδοση (back-propagation) "εξαφανίζεται" (αποκτά σχεδόν μηδενικές τιμές) όταν μεταδίδεται μέσω του δικτύου. Ως αποτέλεσμα, τα βάρη των πρώτων στρωμάτων του δικτύου δεν ενημερώνονται στο βαθμό που θα έπρεπε. Ο πιο συνηθισμένος αλγόριθμος Βαθιάς Μάθησης είναι αυτός των Συνελικτικών Νευρωνικών Δικτύων, (Convolutional Neural Networks, CNN).

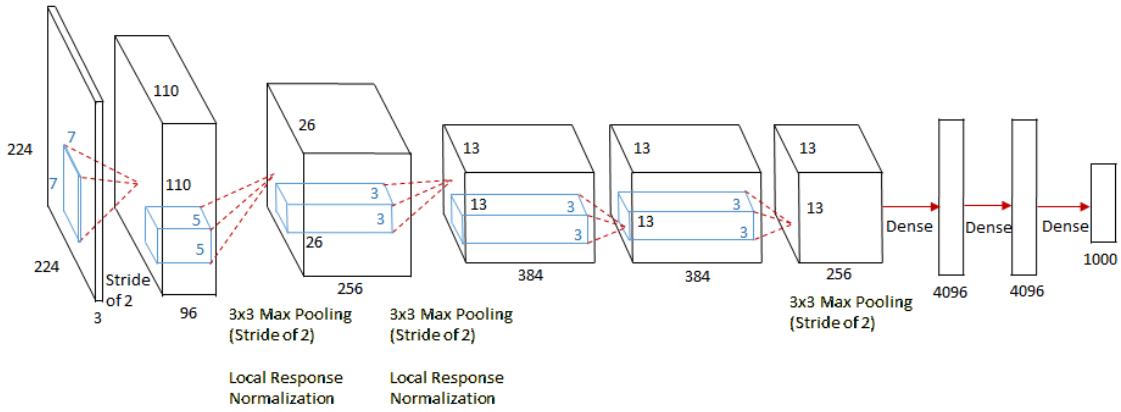


Εικόνα 5: Τυπική Αρχιτεκτονική ενός Συνελικτικού Νευρωνικού Δικτύου, [2]

Με βάση τη μαθηματική λειτουργία της συνέλιξης (του σχηματισμού μίας συνάρτησης ως αποτέλεσμα συνδυασμού δύο άλλων), το μοντέλο LeNet κατάφερε να ξεπεράσει τους άλλους αλγόριθμους ταξινόμησης αναγνωρίζοντας χειρόγραφα ψηφία [18]. Ωστόσο, τα ConvNets (Convolutional Networks) προσέλκυσαν την προσοχή του επιστημονικού κοινού σχεδόν 15 χρόνια αργότερα, όταν το βαθύ ConvNet του A. Krizhevsky και λοιπών [19], που ονομάστηκε AlexNet, κατάφερε χρησιμοποιώντας το σύνολο δεδομένων ImageNet [20], να ξεπεράσει το

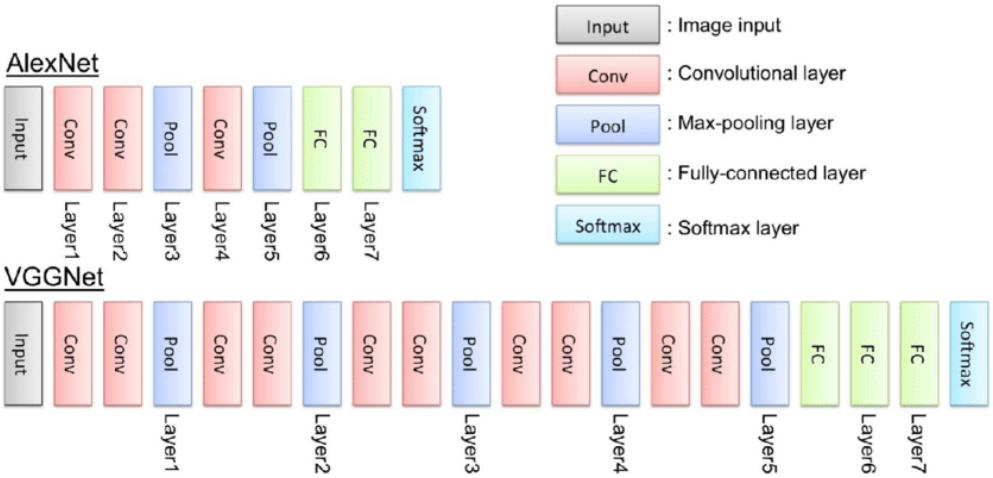
[2]https://commons.wikimedia.org/wiki/File:Typical_cnn.png

δεύτερο στο διαγωνισμό οπτικής αναγνώρισης μεγάλης κλίμακας (ImageNet Large Scale Visual Recognition Challenge, ILSVRC [21]) αλγόριθμο κατά σχεδόν 5%.



Εικόνα 6: Αρχιτεκτονική του δικτύου ZFNet, [3]

Το top-5 ποσοστό σφάλματος αναφέρεται στις εικόνες του συνόλου δεδομένων δοκιμής, για τις οποίες η σωστή κατηγορία δεν είναι ανάμεσα στις 5 πιο πιθανές που έχουν προβλεφθεί από τον αλγόριθμο. Με μερική τροποποίηση του AlexNet, το δίκτυο ZFNet καταφέρνει την επόμενη χρονιά να μειώσει το ποσοστό αυτό σε 14.8%. Ο νικητής της επόμενης χρονιάς ήταν το GoogLeNet ή InceptionV1, με 22 επίπεδα και το θεαματικό top-5 ποσοστό σφάλματος 6.67%, πολύ κοντά σε ανθρώπινες επιδόσεις. [22]. Το 2^0 δίκτυο του διαγωνισμού της ίδιας χρονιάς είναι το VGGNet με ποσοστό 7.3%. Το δίκτυο αποτελείται από 16 επίπεδα και είναι παρόμοιο με το AlexNet (Εικόνα 7).

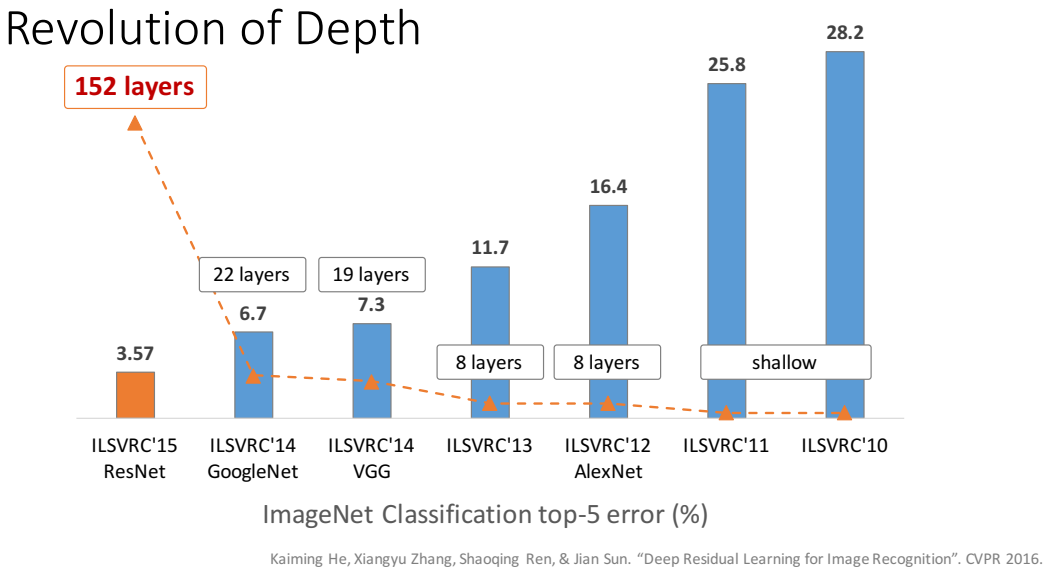


Εικόνα 7: Αρχιτεκτονική του δικτύων AlexNet και VGGNet, [23]

Το ποσοστό ανθρώπινου λάθους για το σύνολο των δεδομένων(5.1%) ξεπεράστηκε την επόμενη χρονιά με το ResNet ([24]) με 3.57% top-5 error rate. Στην εικόνα 8 μπορούμε να δούμε την

[3]https://medium.com/coinmonks/paper-review-of-zfnet-the-winner-of-ilsvlc-2013-image-classification-d1a5a0c45103

εξέλιξη των αρχιτεκτονικών αυτών που συμμετείχαν στον διαγωνισμό, τόσο ως προς τον αριθμό των επιπέδων όσο και ως προς το ποσοστό σφάλματος.



Εικόνα 8: Εξέλιξη των αρχιτεκτονικών που συμμετείχαν στον διαγωνισμό ILSVRC [24]

1.1.6 Μεταφορά Μάθησης (Transfer Learning)

Η μελέτη της μεταφοράς μάθησης, ήρθε ως αποτέλεσμα της παρατήρησης πως οι άνθρωποι μπορούν να χρησιμοποιήσουν ευφυώς την γνώση που έχουν αποκτήσει για την επίλυση νέων προβλημάτων γρηγορότερα ή αποτελεσματικότερα. Οι Karl και λοιποί [12], ορίζουν την μεταφορά μάθησης ως: "Η μεταφορά μάθησης για βαθιά νευρωνικά δίκτυα είναι η διαδικασία της αρχικής εκπαίδευσης ενός βασικού δικτύου σε ένα σύνολο δεδομένων και η εν συνεχεία μεταφορά των γνώσεων που έχουν αποκτηθεί (τα βάρη του δικτύου) σε ένα δεύτερο δίκτυο που θα εκπαιδευτεί σε ένα διαφορετικό σύνολο δεδομένων". Στην μεταφορά μάθησης, υπάρχουν τρία κύρια ερευνητικά ερωτήματα: 1) Τι να μεταφερθεί, 2) Πώς να μεταφερθεί, και 3) πότε να μεταφερθεί. Το ερώτημα "Τι να μεταφερθεί" αναφέρεται στο ποιο μέρος της υπάρχουσας γνώσης θέλουμε να μεταφέρουμε ανάμεσα στους τομείς και τις εργασίες. Μέρος της γνώσης μπορεί να είναι κοινή ανάμεσα σε διάφορους τομείς ή όχι. Αφού βρεθεί η γνώση που μπορεί να μεταφερθεί, χρειάζεται να αναπτυχθούν αλγόριθμοι για την μεταφορά αυτή, που μας οδηγεί στο ερώτημα του "Πώς να μεταφερθεί". Το ερώτημα "Πότε να μεταφερθεί" αναφέρεται στην μελέτη των συνθηκών που χρειάζεται να ικανοποιούνται ώστε να έχει νόημα η μεταφορά. Σε περιπτώσεις όπου τα πεδία ανάμεσα στα οποία μελετάμε την μεταφορά είναι πολύ διαφορετικά μεταξύ τους, η μεταφορά είναι πιθανό όχι μόνο να μην

προσφέρει, αλλά να χειροτερέψει την απόδοση της εκπαίδευσης στο πεδίο προορισμού, μία κατάσταση γνωστή και ως αρνητική μεταφορά. Στην πράξη, λίγες φορές εκπαιδεύουμε ένα Συνελικτικό Δίκτυο (ConvNet) με τυχαία αρχικοποίηση. Αυτό συμβαίνει γιατί είναι δύσκολη και χρονοβόρα η συγκέντρωση δεδομένων ικανοποιητικού μεγέθους. Αντιθέτως, η συνήθης πρακτική είναι η εκπαίδευση ενός ConvNet σε ένα υπάρχον σετ δεδομένων (το ImageNet για παράδειγμα που έχει 1.2 εκατομμύρια εικόνες με 1000 κατηγορίες) και στη συνέχεια να χρησιμοποιηθεί το ConvNet είτε για αρχικοποίηση των βαρών, είτε ως σταθερός εξαγωγέας χαρακτηριστικών (Fixed Feature Extractor) για την εργασία που μας ενδιαφέρει (εικόνα 5). Τα τρία ευρέως χρησιμοποιούμενα σενάρια της μεταφοράς μάθησης είναι [25]:

- **Χρήση του ConvNet ως σταθερός εξαγωγέας χαρακτηριστικών.** Χρησιμοποιείται ένα προ-εκπαιδευμένο ConvNet στο σετ δεδομένων ImageNet και αφού αφαιρεθεί το τελευταίο πλήρως συνδεδεμένο στρώμα (το στρώμα με έξοδο τις 1000 κατηγορίες), χρησιμοποιούμε το υπόλοιπο δίκτυο ως σταθερός εξαγωγέας χαρακτηριστικών για το νέο σετ δεδομένων.
- **Βελτιστοποίηση του ConvNet.** Σε αυτή την περίπτωση, εκτός από την αντικατάσταση και επανεκπαίδευση του ταξινομητή στο νέο σετ δεδομένων, επιδιώκεται και η βελτιστοποίηση των βαρών των ενδιάμεσων στρωμάτων, μέσω της διατήρησής τους κατά το στάδιο της οπίσθια διάδοσης. Εδώ υπάρχει η επιλογή της βελτιστοποίησης όλων των στρωμάτων του δικτύου, ή η διατήρηση των αρχικών στρωμάτων (που περιλαμβάνουν γενικότερα χαρακτηριστικά όπως συνδέσεις άκρων, ή ανίχνευση χρωματισμών) και βελτιστοποίηση των ανώτερων στρωμάτων που περιέχουν περισσότερες λεπτομέρειες για τις κατηγορίες που έχει ως έξοδο το αρχικό ConvNet (για παράδειγμα την κατηγοριοποίηση σκύλων ανά γένος).
- **Χρήση προ-εκπαιδευμένων δικτύων.** Μιας και η εκπαίδευση σε συστοιχία πολλαπλών Μονάδων Επεξεργασίας Γραφικών (Graphics Processing Units, GPUs), ενός ConvNet διαρκεί περίπου 2-3 εβδομάδες για το ImageNet, συνηθίζεται να δημοσιοποιούνται τα αποτελέσματα της εκπαίδευσης με σκοπό την χρήση τους με την στρατηγική της βελτιστοποίησης.

1.1.7 Ενεργητική Μάθηση (Active Learning)

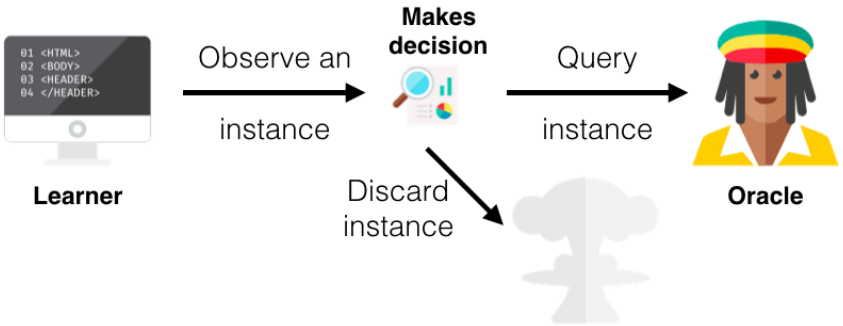
Η ενεργητική μάθηση αποτελεί έναν ακόμα κλάδο της μηχανικής μάθησης. Η βασική υπόθεση είναι ότι, αν ο αλγόριθμος μάθησης μπορέσει να επιλέξει τα δεδομένα από τα οποία μαθαίνει θα αποδώσει καλύτερα με λιγότερη εκπαίδευση. Στα συστήματα αυτά της ενεργητικής μάθησης, επιδιώκεται η τιτλοφόρηση επιλεγμένων μη χαρακτηρισμένων δεδομένων, με στόχο την επίτευξη υψηλής ακρίβειας στην πρόβλεψη, χρησιμοποιώντας όσο

το δυνατόν λιγότερα δεδομένα, ελαχιστοποιώντας έτσι το κόστος απόκτησης τιτλοφόρησης. Τρεις είναι οι τυπικές στρατηγικές με τις οποίες το εκπαιδευόμενο μοντέλο ζητά τιτλοφόρηση δεδομένων:

- **Σύνθεση ερωτήματος ιδιότητας μέλους (Membership Query Synthesis, εικόνα 9)**: Σε αυτή την περίπτωση, το εκπαιδευόμενο μοντέλο συνθέτει δείγματα και ζητάει την τιτλοφόρησή τους. Για παράδειγμα, εάν τα δεδομένα είναι εικόνες ψηφίων, το μοντέλο θα δημιουργούσε μια εικόνα παρόμοια με ένα ψηφίο (ίσως με περιστροφή ή με αποκλεισμό κάποιου από τα κομμάτια του ψηφίου) και αυτή η νέα εικόνα αποστέλλεται για τιτλοφόρηση.

Εικόνα 9: Σύνθεση ερωτήματος ιδιότητας μέλους [26]

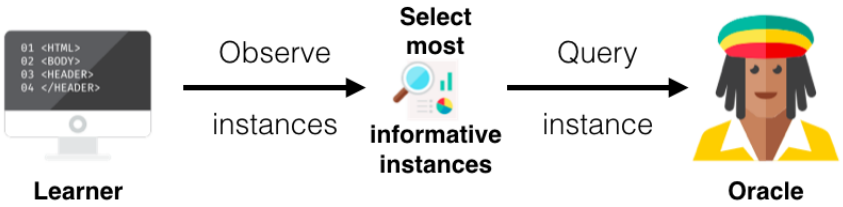
- **Επιλεκτική ή διαδοχική δειγματοληψία (Selective or Sequential Sampling, εικόνα 10)**: Σε αυτή την περίπτωση, το μοντέλο αντλεί διαδοχικά δείγματα από μια υπάρχουσα κατανομή και αποφασίζει κάθε φορά εάν πρέπει ή όχι να τιτλοφορηθούν. Η λήψη της απόφασης για την τιτλοφόρηση ή όχι ενός δείγματος σχετίζεται με το μέγεθος της χρήσιμης πληροφορίας που μπορεί να λάβει από αυτό.



Εικόνα 10: Επιλεκτική ή διαδοχική δειγματοληψία [26]

- **Δειγματοληψία από δεξαμενή δειγμάτων (Pool-based Sampling εικόνα 11)**: Όπως και στην επιλεκτική δειγματοληψία, προϋπόθεση για την στρατηγική αυτή είναι η

ύπαρξη μιας μεγάλης ομάδας μη τιτλοφορημένων δειγμάτων. Εδώ η απόφαση για την τιτλοφόρηση ή μη των δειγμάτων γίνεται για όλο το σύνολο της ομάδας και όχι ανά μέλος ξεχωριστά. Η επιλογή της στρατηγικής για την απόφαση αυτή παρουσιάζει και αυτή ερευνητικό ενδιαφέρον. Παρακάτω αναφέρουμε τις επικρατέστερες αυτές στρατηγικές. Είναι η πιο συνηθισμένη πρακτική και έχει χρησιμοποιηθεί σε αρκετά πραγματικά σενάρια [5] στην Μηχανική Μάθηση, όπως η όπως η ταξινόμηση κειμένου [27], [28], [29], η εξόρυξη πληροφορίας [30], [31], [32] η ταξινόμηση και ανάκτηση εικόνας και βίντεο [33], [34], η αναγνώριση ομιλίας [35] και η διάγνωση του καρκίνου [36].



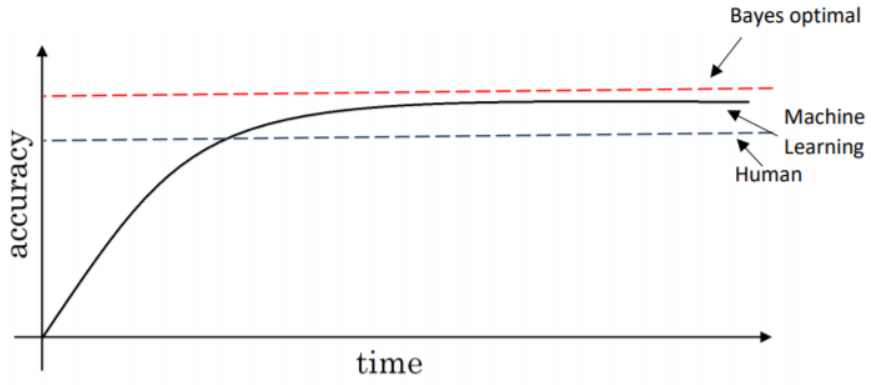
Εικόνα 11: Δειγματοληψία από δεξαμενή δειγμάτων [26]

Στρατηγικές επιλογής δειγμάτων

Ένα κρίσιμο ερώτημα που προκύπτει στην μεταφορά μάθησης αφορά τον τρόπο υπολογισμού της πληροφορίας που παρέχει ένα δείγμα, ώστε αυτό στη συνέχεια να τιτλοφορηθεί. Συχνά χρησιμοποιούμενες στρατηγικές είναι οι εξής [7], [26], [37]:

- **Στρατηγική Ελάχιστης Βεβαιότητας (Least Confidence).** Σε αυτή την στρατηγική, το εκπαιδευόμενο μοντέλο δίνει προτεραιότητα σε δείγματα που παρουσιάζουν την μικρότερη βεβαιότητα στη πιο πιθανή κατηγορία τους. Έτσι για παράδειγμα εάν η πιο πιθανή πρόβλεψη για ένα δείγμα είναι της κατηγορίας Α, τότε θα επιλεχθεί το δείγμα το οποίο έχει την μικρότερη βεβαιότητα να ανήκει στην κατηγορία αυτή.
- **Περιθώριο Βεβαιότητας (Margin of Confidence).** Σε αυτή την περίπτωση το μοντέλο δεν αγνοεί τις πιθανότητες των υπόλοιπων κατηγοριών, όπως παραπάνω, αλλά δίνει προτεραιότητα στην περίπτωση της μικρότερης διαφοράς ανάμεσα στην πιο πιθανή και την δεύτερη πιο πιθανή κατηγορία. Για παράδειγμα, εάν σε ένα δυαδικό πρόβλημα οι προβλέψεις του μοντέλου είναι 0.4 και 0.6 για ένα δείγμα και 0.3 και 0.7 για ένα άλλο, τότε θα προτιμηθεί το πρώτο δείγμα μιας και έχει διαφορά βεβαιότητας 0.2.
- **Εντροπία (Entropy).** Ένα δημοφιλές μέτρο υπολογισμού της βεβαιότητας είναι η εντροπία. Σε αυτή την στρατηγική υπολογίζεται η εντροπία σε κάθε δείγμα και δίνεται προτεραιότητα σε αυτά με την μεγαλύτερη τιμή.

- **Λόγος Βεβαιότητας (Ratio of Confidence)** Ο λόγος βεβαιότητας αποτελεί μια παραλλαγή του περιθωρίου βεβαιότητας, μόνο που σε αυτή την περίπτωση υπολογίζεται ο λόγος των των δύο πιο πιθανών κατηγοριών αντί για την διαφορά.



Εικόνα 12: Απόδοση μηχανική μάθησης, απόδοση ανθρώπου και βέλτιστο λάθος [4]

1.1.8 Ο ανθρώπινος παράγοντας

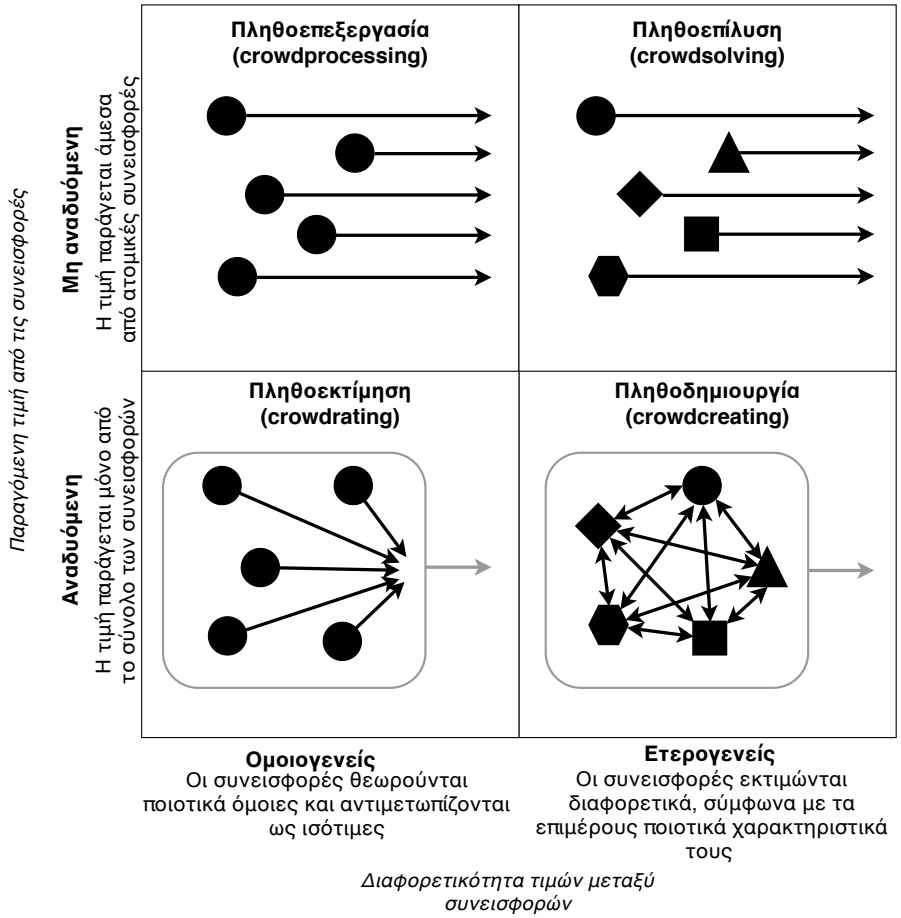
Το βέλτιστο θεωρητικό λάθος που μπορεί να έχει ένας αλγόριθμος μηχανικής μάθησης καλείται Σφάλμα του Bayes (Bayes Optimal Error) Εικόνα 12. Πολλά συστήματα μηχανικής μάθησης ασχολούνται με την αυτοματοποίηση ενεργειών που πραγματοποιούν οι άνθρωποι και έχουν βελτιωθεί τόσο πολύ που τώρα πια ξεπερνούν την απόδοση που έχουν οι άνθρωποι σε ορισμένες από αυτές τις ενέργειες (για παράδειγμα η εύρεση προτεινόμενων βιβλίων σε ένα σύστημα συστάσεων) [38]. Η σύγκριση της απόδοσης ενός συστήματος με αυτή των ανθρώπων μας βοηθάει στην εκτίμηση του βέλτιστου λόγου λάθους και του επιθυμητού λόγου λάθους. Επιπλέον, πολλές φορές είναι εύκολη και η συλλογή σωστά τιτλοφορημένων δεδομένων από ανθρώπους, είτε αυτοί είναι ειδικοί στο αντικείμενο στο οποίο καλούνται να συνεισφέρουν τη γνώση τους, είτε όχι. Τέλος, η ανάλυση σφαλμάτων (error analysis) είναι πολύ αποδοτικότερη όταν αξιοποιείται η ανθρώπινη διορατικότητα (για παράδειγμα στην αναγνώριση ομιλίας, ο διαχωρισμός της φράσης “ήρθαμε δύο από τους τέσσερεις” από την φράση “ήρθα με δύο από τους τέσσερεις”). Στην επόμενη ενότητα βλέπουμε την πιο συνηθισμένη τεχνική εκμετάλλευσης του ανθρώπινου παράγοντα μέσω της ανάθεσης διάφορων εργασιών.

[4] https://towardsdatascience.com/how-to-improve-my-ml-algorithm-lessons-from-andrew-ngs-experience-ii-f66926926f88

1.2 Πληθοπορισμός και πληθαίσθηση

Για την διαδικασία της μάθησης ενός νευρωνικού δικτύου, αλλά και για τον έλεγχο, επαλήθευση και βελτίωση αυτού, χρειάζονται δεδομένα. Παρόλο που συσκευές με αισθητήρες προσφέρουν δεδομένα στο σύστημα, η τιτλοφόρηση και ο χαρακτηρισμός αυτών από τον άνθρωπο μπορούν να επιταχύνουν την διαδικασία και να τροφοδοτήσουν τιτλοφορημένα και χαρακτηρισμένα δεδομένα. Μια τεχνική ανάθεσης σε πολλούς ανθρώπους μιας εργασίας, όπως αυτή της συλλογής ή και της τιτλοφόρησης δεδομένων, είναι η τεχνική του πληθοπορισμού.

Το 2005, ο Surowiecki [39] μας επισημαίνει το φαινόμενο πως η συσσωμάτωση δεδομένων ή πληροφορίας από μια ομάδα ανθρώπων έχει συχνά ως αποτέλεσμα στην λήψη καλύτερων αποφάσεων συγκριτικά με την περίπτωση όπου η απόφαση θα λαμβάνονταν από ένα μόνο άτομο στην ομάδα. Ο όρος πληθοπορισμός (Crowdsourcing) εμφανίστηκε πρώτη φορά στο άρθρο του Howe “The rise of crowdsourcing” [8] το 2006. Ανάλογα τα χαρακτηριστικά της ανατιθέμενης διεργασίας, οι Morschheuser και λοιποί, διακρίνουν συστήματα πληθοπορισμού σε τέσσερις κατηγορίες (εικόνα 13).



Εικόνα 13: Αρχέτυπα Συστημάτων Πληθοπορισμού [9]

Τα συστήματα πληθοεπεξεργασίας (Crowdprocessing systems) βασίζονται στο πλήθος για την εκτέλεση μεγάλων ποσοτήτων ομοιογενών εργασιών. Ίδιες συνεισφορές είναι ένα ποιοτικό χαρακτηριστικό της εγκυρότητα της εργασίας. Το αποτέλεσμα προέρχεται απευθείας από κάθε απομονωμένη συνεισφορά.

Οι προσεγγίσεις πληθεπίλυσης (Crowdsolving approaches) χρησιμοποιούν την ποικιλομορφία του πλήθους ώστε να βρουν ένα μεγάλο αριθμό ετερογενών λύσεων σε ένα συγκεκριμένο πρόβλημα. Το αποτέλεσμα αυτής της προσέγγισης προκύπτει και εδώ απευθείας από κάθε απομονωμένη συνεισφορά.

Τα συστήματα πληθοεκτίμησης (Crowdrating systems) συνήθως επιδιώκουν να αξιοποιήσουν τη σοφία των πλήθους (Wisdom of Crowds, Surowiecki, [39]) για την εκτέλεση συλλογικών εκτιμήσεων ή προβλέψεων. Στην περίπτωση αυτή, το αποτέλεσμα προκύπτει από έναν πολύ μεγάλο αριθμό ομοιογενών “ψήφων”.

Οι λύσεις πληθοδημιουργίας (Crowdcreating solutions) επιδιώκουν να δημιουργήσουν περι-
εκτικά (αναδυόμενα) αντικείμενα που βασίζονται σε ποικίλες ετερογενείς συνεισφορές. Τυπικά παραδείγματα περιλαμβάνουν όλα τα είδη του περιεχομένου που δημιουργείται από το χρήστη (για παράδειγμα η υπηρεσία YouTube[5]) ή της γνώσης που προέρχεται από συνεργατική συσσωμάτωση (για παράδειγμα η διαδικτυακή εγκυκλοπαίδεια Wikipedia[6])

Σύμφωνα με τους Borcea και Talasila [40], η κινητή αίσθηση (mobile sensing) περιλαμβάνει μια εφαρμογή που εκτελείται στο κινητό τηλέφωνο ενός χρήστη, για πρόσβαση σε δεδομένα που ανιχνεύονται από διάφορους ενσωματωμένους αισθητήρες και την χρησιμοποίηση των δεδομένων αυτών, αναφέροντάς τα σε κεντρικούς διακομιστές, είτε για προσωπική ενημέρωση ή για δημόσια χρήση. Η κινητή αίσθηση μπορεί να κατηγοριοποιηθεί ως προσωπική ή ως κοινοτική αίσθηση, ανάλογα τα φαινόμενα τα οποία παρακολουθούνται[41]. Η κοινοτική αίσθηση με τη σειρά της, μπορεί να είναι συμμετοχική (participatory sensing) ή ευκαιριακή (opportunistic sensing). Στην συμμετοχική αίσθηση, απαιτείται η ενεργός συμμετοχή ατόμων για τη συμβολή δεδομένων αισθητήρων (π.χ. λήψη φωτογραφιών, πραγματοποίηση καταγραφής ή αναφορά καταστάσεων κυκλοφορίας) που σχετίζονται με φαινόμενα μεγάλης κλίμακας. Από την άλλη πλευρά, η ευκαιριακή αίσθηση είναι πιο αυτόνομη και η συμμετοχή του χρήστη είναι ελάχιστη (π.χ. συνεχής δειγματοληψία της τοποθεσίας του χρήστη, χωρίς να απαιτείται κάποια ενέργεια από αυτόν).

[5]https://www.youtube.com
[6]https://www.wikipedia.org

Εικόνα 14: Σύγκριση κινητής πληθαίσθησης και σχετικών εννοιών [42]

Το 2011, οι Ganti και λοιποί στο [43], ορίζουν την πληθαίσθηση (Crowdsensing) ως "ένα παράδειγμα προσέγγισης αίσθησης μεγάλης κλίμακας όπου άτομα με συσκευές με αισθητήρια και υπολογιστική ικανότητα, συλλογικά μοιράζονται δεδομένα και εξαγάγουν πληροφορίες για να μετρήσουν και να χαρτογραφήσουν φαινόμενα κοινού ενδιαφέροντος". Μια συγκριτική μελέτη ανάμεσα στην πληθαίσθηση και άλλες σχετικές έννοιες παρουσιάζεται στην εικόνα 14.
Μπορούμε να δούμε πως η "σοφία του πλήθους" και ο πληθοπορισμός βασίζονται στην ανθρώπινη νοημοσύνη, ενώ η πληθαίσθηση και η συμμετοχική αίσθηση στοχεύουν στην συνδυαστική χρήση της ανθρώπινης και της μηχανικής νοημοσύνης.

ΚΕΦΑΛΑΙΟ 2:
Μεθοδολογία της έρευνας

Στο κεφάλαιο αυτό παρουσιάζονται τα βήματα που ακολουθήθηκαν κατά την διάρκεια της ερευνητικής εργασίας. Όπως είδαμε στην Εισαγωγή, η μελέτη αφορά την πρακτική εφαρμογή συνδυασμού τεχνικών μηχανικής μάθησης και πληθοπορισμού. Το πρώτο βήμα κατά την έναρξη της ερευνητικής διεργασίας ήταν η βιβλιογραφική αναζήτηση στους σχετικούς τομείς [44].

Πίνακας 1: Χαρακτηριστικά Ποσοτικών, Ανάμεικτων και Ποιοτικών μεθόδων έρευνας [45].

Ποσοτικές Μέθοδοι	Μεικτές Μέθοδοι	Ποιοτικές Μέθοδοι
Προκαθορισμένες μέθοδοι	Προκαθορισμένες και αναδυόμενες μέθοδοι	Αναδυόμενες μέθοδοι
Ερωτήσεις κλειστού τύπου	Ερωτήσεις ανοικτού και κλειστού τύπου	Ερωτήσεις ανοικτού τύπου
Δεδομένα απόδοσης, συμπεριφοράς, παρατήρησης και δεδομένα αισθητηρίων	Πολλαπλές μορφές δεδομένων που καλύπτουν όλες τις πιθανότητες	Δεδομένα συνεντεύξεων, παρατήρησης, εγγράφων και οπτικοακουστικά δεδομένα
Στατιστική ανάλυση	Στατιστική ανάλυση και ανάλυση κειμένων	Ανάλυση κειμένων και εικόνων
Στατιστική διερμηνεία	Διερμηνεία διαμέσου βάσεων δεδομένων	Διερμηνεία θεμάτων και προτύπων

2.1 Βιβλιογραφική Αναζήτηση

Η βιβλιογραφική αναζήτηση περιείχε τοος ευρύτερους όρους “machine learning” και “crowdsourcing”, αλλά και τους πιο συγκεκριμένους όρους όπως “deep learning”, “transfer learning”, “active learning”, “crowdsensing”, καθώς και τους συνδυασμούς αυτών. Οι βάσεις δεδομένων και μηχανές αναζήτησης που χρησιμοποιήθηκαν είναι:

- **Scopus**, μια βάση δεδομένων με περιλήψεις δημοσιεύσεων και αναφορές, με αρκετές επιλογές φιλτραρίσματος των αποτελεσμάτων αναζήτησης.

- **Association for Computing Machinery (ACM) Digital Library**, μια βιβλιοθήκη με εγγραφές σχετικές με τις επιστήμες πληροφορίας και επικοινωνίας. Προσφέρει εκτός από δημοσιεύσεις, σύνολα δεδομένων, πηγαίο κώδικα και αρχεία πολυμέσων.
- **IEEE Xplore Digital Library**, βάση δεδομένων που προσφέρει και αυτή πληθώρα δημοσιεύσεων σχετικών με την τεχνολογία της πληροφορίας.
- **arXiv**, ένα αποθετήριο επιστημονικών μελετών, κυρίως στις θετικές επιστήμες, που διατηρεί αρχεία πριν την δημοσίευσή τους σε κάποιο ακαδημαϊκό περιοδικό ή σε κάποια πρακτικά συνεδρίου.
- **Google Scholar**, η μεγαλύτερη μηχανή αναζήτησης ακαδημαϊκού περιεχομένου που προσφέρει αναζήτηση σε πλήρη κείμενα και όχι μόνο σε τίτλους δημοσιεύσεων.

Τα αποτελέσματα της βιβλιογραφικής αναζήτησης ήταν περισσότερα από εκατό, και αφορούν κυρίως την μηχανική μάθηση. Από αυτά επιλέχθηκαν αυτά που είτε έφεραν τις περισσότερες βιβλιογραφικές αναφορές, είτε ήταν δημοσιευμένα σε περιοδικά υψηλού δείκτης επιστημονικής ποιότητας h (h-index). Από αυτά τέλος παραλήφθηκαν ορισμένα που είχαν δημοσιευτεί πριν από περισσότερα των δέκα ετών.

2.2 Επιλογή εργαλείων

Παράλληλα με την θεωρητική κατάρτιση στους αναφερθέντες κλάδους, πραγματοποιήθηκε επίσης μελέτη σχετικά με τα εργαλεία που χρησιμοποιούνται για την υλοποίηση των επιμέρους στοιχείων αλλά και για την ανάλυση και αξιολόγηση των αποτελεσμάτων.

2.2.1 Εφαρμογή Μηχανικής Μάθησης (E1)

Για την ανάπτυξη της εφαρμογής μηχανικής μάθησης και συγκεκριμένα της μεταφοράς μάθησης και της ενεργητικής μάθησης, επιλέχθηκε η γλώσσα προγραμματισμού Python[7], η οποία χρησιμοποιείται κατά κόρον σε περιπτώσεις λύσεων μηχανικής μάθησης. Επιπλέον, για την εκπαίδευση,την επαλήθευση, την δοκιμή και την επιλογή νέων δειγμάτων, χρησιμοποιείται η βιβλιοθήκη PyTorch[8]. Ο τομέας στον οποίο έχει αναπτυχθεί η εφαρμογή είναι αυτός της Μηχανικής Όρασης (Computer Vision) και μπορεί να χρησιμοποιηθεί σε προβλήματα δυαδικής κατηγοριοποίησης (Αρχείο 11). Παρόλα αυτά, υπάρχει η δυνατότητα επέκτασης των διαθέσιμων λειτουργιών σε περισσότερα προβλήματα μηχανικής όρασης αλλά και σε προβλήματα διαφορετικών τομέων όπως αυτός της Επεξεργασίας Ήχου (Audio

[7] https://www.python.org
[8] https://pytorch.org

Processing) και της Επεξεργασίας Κειμένου (Text Processing). Για τις υπάρχουσες λειτουργίες χρησιμοποιείται η βιβλιοθήκη torchvsion[9], ενώ για τους αναφερθέντες τομείς, είναι διαθέσιμες οι αντίστοιχες βιβλιοθήκες torchaudio[10] και torchtext.[11] (αρχείο 5). Για αρχικοποίηση του νευρωνικού δικτύου επιλέξαμε το νευρωνικό δίκτυο ResNet [24], που όπως είδαμε είναι ένα από τα αποτελεσματικότερα στον χώρο της μηχανικής όρασης, ως σταθερό εξαγωγέα χαρακτηριστικών.

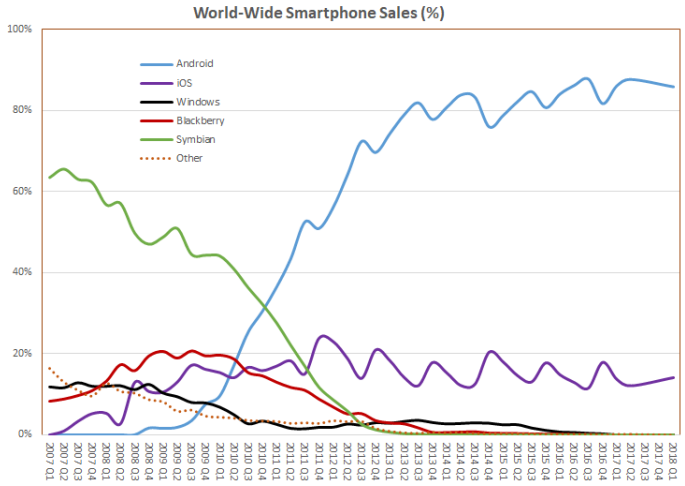
2.2.2 Εφαρμογή Εξυπηρετητή (E2)

Για την υλοποίηση της εφαρμογής του εξυπηρετητή επιλέχθηκε και πάλι η γλώσσα προγραμματισμού Python και η βιβλιοθήκη FastAPI[13] (αρχείο 4). Επιπλέον, για την αποθήκευση των παραμέτρων των διεργασιών χρησιμοποιείται η σχεσιακή βάση δεδομένων PostgreSQL[14]. Για τις διεργασίες που τρέχουν στο παρασκήνιο (ενεργοποίηση κύκλου εκπαίδευσης, έλεγχος κριτηρίου προσθήκης νέων δειγμάτων στο σύνολο δεδομένων εκπαίδευσης), χρησιμοποιείται η βιβλιοθήκη Celery[12] και για την διαχείριση των διεργασιών αυτών χρησιμοποιείται η βάση δεδομένων Redis[13]. Το πρόγραμμα εξυπηρετητή nginx[14] (Αρχείο 8) χρησιμοποιείται ως μεσολαβητής ανάμεσα στους πελάτες και την διαδικτυακή εφαρμογή, ενώ το πρόγραμμα Certbot[15] χρησιμοποιείται για την λήψη και ανανέωση πιστοποιητικών κρυπτογράφησης.Για την ασφαλή μεταφορά δεδομένων από την κινητή εφαρμογή στον εξυπηρετητή, επιλέχθηκε η χρήση του πρωτοκόλλου κρυπτογράφησης Transport Layer Security (TLS), ενώ για έλεγχο της γνησιότητας της εφαρμογής για κινητές συσκευές από τον εξυπηρετητή επιλέχθηκε η χρήση του προτύπου OAuth 2.0. Η ταυτοποίηση των χρηστών από την εφαρμογή παραλήφθηκε και συνεπώς επιτρέπεται ανώνυμη επικοινωνία με τον εξυπηρετητή. Όλες οι μικρο-υπηρεσίες αυτές αποτελούν επιμέρους πακέτα (containers) τα οποία εγκαθίστανται και επικοινωνούν μεταξύ τους με χρήση του προγραμμάτων docker[16] (Αρχείο 6) και docker-compose [17](Αρχείο 7).

[9] https://pytorch.org/docs/stable/torchvision/
[10] https://pytorch.org/audio
[11] https://pytorch.org/text/
[13] https://fastapi.tiangolo.com
[14] https://www.postgresql.org
[12] http://www.celeryproject.org
[13] https://redis.io
[14] https://nginx.org
[15] https://certbot.eff.org
[16] https://www.docker.com
[17] https://docs.docker.com/compose/

2.2.3 Εφαρμογή για κινητές συσκευές (E3)

Για την υλοποίηση της εφαρμογής για κινητές συσκευές, επιλέχθηκε η γλώσσα προγραμματισμού TypeScript[18] και το πλαίσιο ανάπτυξης React Native[19], η οποία επιτρέπει την εγκατάσταση της τελικής εφαρμογής στην συντριπτική πλειοψηφία των διαθέσιμων συσκευών (Εικόνα 15), λειτουργικού συστήματος είτε Android[20] της Google, είτε iOS[21] της Apple. Χρησιμοποιήθηκαν επίσης επιπλέον βιβλιοθήκες απαραίτητες είτε για τον έλεγχο της ροής της εφαρμογής είτε για την επικοινωνία με τον εξυπηρετητή (αρχείο 13).

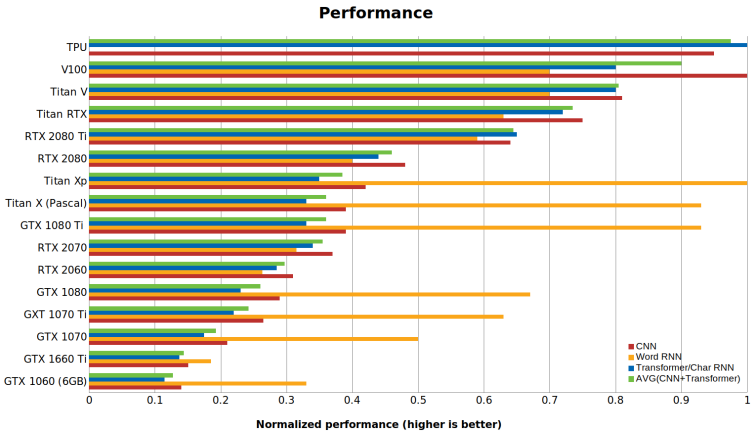


Εικόνα 15: Παγκόσμια αγορά κινητών συσκευών[22]

2.2.4 Υλισμικό

Για την στέγαση των δύο πρώτων εφαρμογών αξιοποιήθηκε ένας υπολογιστής υψηλών επιδόσεων με κεντρική μονάδα επεξεργασίας Ryzen 52 2400G[23] με μνήμη τυχαίας προσπέλασης 16 GB και κάρτας γραφικών NVidia Titan X (Pascal)[24] με 12 GB μνήμη και 3072 παράλληλους επεξεργαστές γραφικών (Compute Unified Device Architecture, CUDA cores) (εικόνα 16).

[18] https://www.typescriptlang.org
[19] https://facebook.github.io/react-native
[20] https://www.android.com
[21] https://www.apple.com/ios
[22] Smartmo - Own work, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=22720629
[23] https://www.amd.com/en/products/apu/amd-ryzen-5-2400g
[24] https://www.geforce.com/hardware/desktop-gpus/geforce-gtx-titan-x/specifications



Εικόνα 16: Επιδόσεις καρτών γραφικών σε προβλήματα μηχανικής μάθησης[25]

2.3 Παρουσίαση υλοποιούμενης λύσης

Η τελική υλοποίηση μπορεί να χωριστεί στα τρεις παραπάνω εφαρμογές. Στην ενότητα αυτή παρουσιάζονται τα χαρακτηριστικά της υλοποίησης καθώς και το πλαίσιο δοκιμής και επαλήθευσης αυτής μέσω πειράματος.

2.3.1 Χαρακτηριστικά

Τα κύρια χαρακτηριστικά υψηλού επιπέδου της υλοποίησης είναι τα παρακάτω:

- **Μεταφορά Μάθησης (X1):** Δυνατότητα χρήσης υπάρχοντας προ-εκπαιδευμένου μοντέλου μηχανικής μάθησης.
- **Ενεργητική Μάθηση (X2):** Κύκλος εκπαίδευσης-επαλήθευσης-δοκιμής-δειγματοληψίας. Χρησιμοποιώντας ως παράμετρο τον αριθμό των δειγμάτων που επιθυμούμε να προστεθούν στο σύνολο δεδομένων εκπαίδευσης, μετά από κάθε προσθήκη νέων δειγμάτων έχουμε νέα εκπαίδευση του μοντέλου, αποθήκευσή του και έλεγχο της ακρίβειας του σε σύνολο δειγμάτων δοκιμής
- **Πληθοπορισμός (X3):** Τιτλοφόρηση των επιλεγμένων δειγμάτων από ανθρώπους, αποθήκευση των απαντήσεων και απόφαση για την κατηγορία των επιμέρους δειγμάτων.
- **Ανατροφοδότηση του συστήματος:** **(X4)** Προσθήκη των νέων τιτλοφορημένων δειγμάτων στο σύνολο δεδομένων εκπαίδευσης και ενεργοποίηση της επανεκπαίδευσης του μοντέλου.

[25] https://timdettmers.com/2019/04/03/which-gpu-for-deep-learning/

2.3.2 Περίπτωση χρήσης και πειράματα

Για δοκιμή και επαλήθευση της λειτουργίας της υλοποίησης, επιλέχθηκε ένα πρόβλημα δυαδικής κατηγοριοποίησης εικόνων και αφορά τον χαρακτηρισμό κτιρίων ως κατεδαφισμένα (collapsed), ή μη (non_collapsed). Το σύνολο των εικόνων που χρησιμοποιείται στα στάδια της ενεργητικής μάθησης χρησιμοποιήθηκε από τους Yeum, Dyke και Ramirez [46] σε μελέτη κτιρίων μετά από φυσικές καταστροφές. Οι συγγραφείς, κάνουν χρήση συνελικτικών νευρωνικών δικτύων με συνδυασμό κατηγοριοποίησης εικόνων (image classification) και εντοπισμού αντικειμένων (object detection) με σκοπό την εξαγωγή περιοχών ενδιαφέροντος σχετικών με τις βλάβες τις οποίες έχουν υποστεί τα κτίρια. Με τον τρόπο αυτό επιτυγχάνουν να διευκολύνουν κατά πολύ την διαδικασία η οποία ακολουθείται από τους μηχανικούς που ειδικεύονται σε σχετικά προβλήματα.

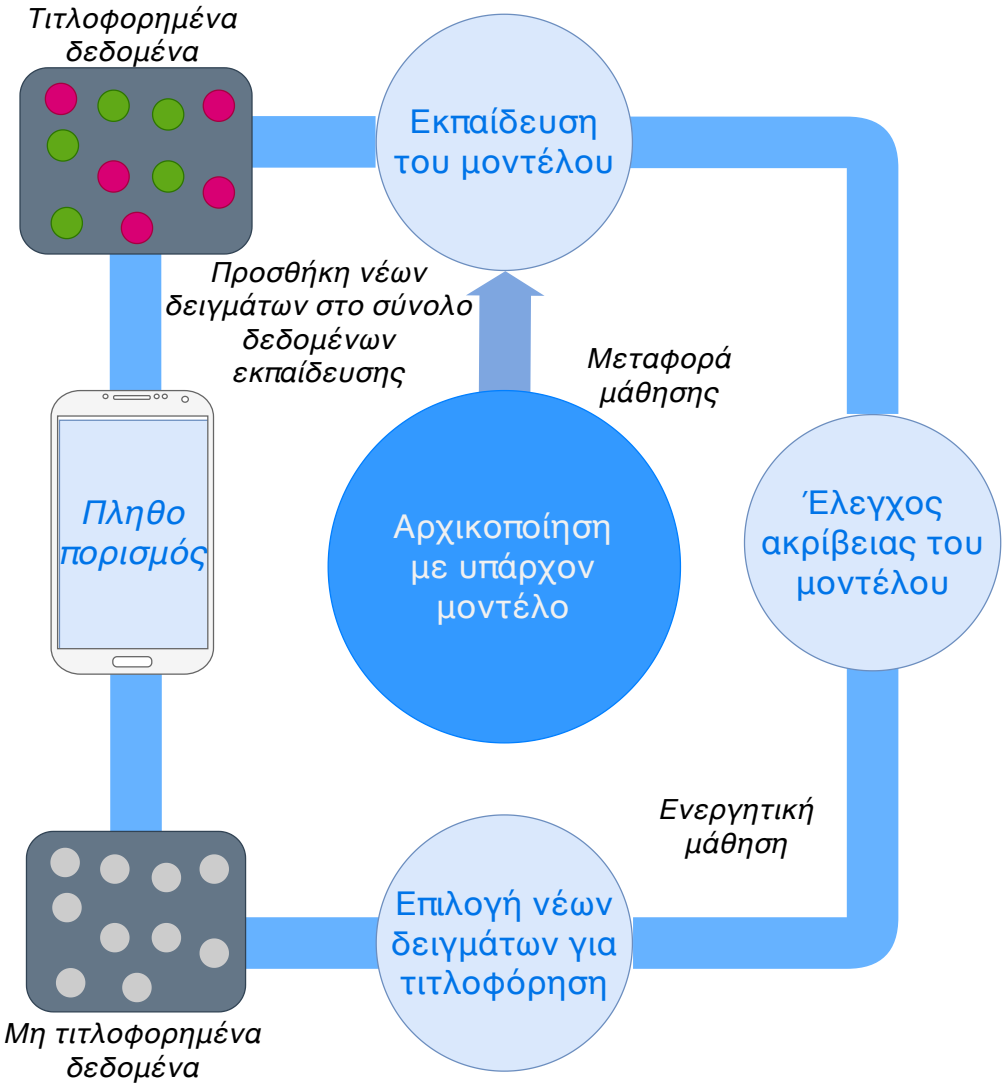
Το επιλεγμένο σύνολο δεδομένων αποτελείται από 5270 εικόνες, ήδη τιτλοφορημένες με τους χαρακτηρισμούς “collapse”, “damaged”, “undamaged” και “irrelevant”, γεγονός που μας επιτρέπει τον έλεγχο της εφαρμογής της ενεργητικής μάθησης. Με εκπαίδευση του συνόλου των δειγμάτων και χρήση της αρχιτεκτονικής AlexNet [47] οι Yeum, Dyke και Ramirez επιτυγχάνουν ακρίβεια πρόβλεψης περίπου 91%. Στην περίπτωση χρήσης της υλοποίησης, οι εικόνες με τον χαρακτηρισμό “collapse” χρησιμοποιήθηκαν για την κλάση “collapsed” ενώ όλες οι υπόλοιπες για την δεύτερη κλάση “non_collapsed”. Για το στάδιο της μεταφοράς μάθησης επιλέχθηκε το μοντέλο ResNet το οποίο όπως είδαμε είναι από τα πιο αποτελεσματικά, ενώ για το στάδιο της μεταφοράς μάθησης και την στρατηγική δειγματοληψίας, δοκιμάζονται τόσο η τυχαία λήψη δειγμάτων όσο και οι μέθοδοι της μέγιστης εντροπίας, της ελάχιστης αβεβαιότητας, του περιθωρίου αβεβαιότητας, μέθοδοι που επαληθεύουμε πως δίνουν το ίδιο αποτέλεσμα στην περίπτωση της δυαδικής κατηγοριοποίησης. Στο τέλος της διαδικασίας εξετάζουμε τα αποτελέσματα που λάβαμε ως προς την ακρίβεια του εκπαιδευμένου μοντέλου μηχανικής μάθησης με χρήση ενός σχετικά μικρού αριθμού δειγμάτων για την εκπαίδευσή του.

Κλείνοντας το κεφάλαιο αυτό, μπορούμε να δούμε πως από τα χαρακτηριστικά του αντικειμένου και των εργαλείων που χρησιμοποιούνται πως η παρούσα έρευνα μπορεί να χαρακτηριστεί ως πειραματική με ποσοτικό χαρακτήρα. (Πίνακας 1).

ΚΕΦΑΛΑΙΟ 3:
Αρχιτεκτονική, Πλαίσιο ανάπτυξης, υλοποίηση

Στο κεφάλαιο αυτό παρουσιάζεται η αρχιτεκτονική της υλοποίησης με βάση τα χαρακτηριστικά του συστήματος που περιγράφηκαν στο προηγούμενο κεφάλαιο καθώς και η τελική υλοποίηση των επιμέρους εφαρμογών.

3.1 Αρχιτεκτονική



Εικόνα 17: Αρχιτεκτονική του συστήματος

Η τελική αρχιτεκτονική φαίνεται στην εικόνα 17 και αποτελείται από τα επιμέρους στοιχεία:

- Αρχικοποίηση με υπάρχον μοντέλο, όπου ορίζονται οι παράμετροι της μεταφοράς μάθησης και χρησιμοποιείται η εφαρμογή εξυπηρετητή για την αποθήκευσής τους.
- Εκπαίδευση του μοντέλου, η οποία ξεκινά με την χρήση μικρού αριθμού δειγμάτων ο οποίος αυξάνεται με χρήση της ενεργητικής μάθησης. Η εφαρμογή εξυπηρετητή είναι αρμόδια για την ενεργοποίηση της εφαρμογής μηχανικής μάθησης.
- Έλεγχος ακρίβειας του μοντέλου, ο οποίος γίνεται μετά από κάθε φορά που ολοκληρώνεται η εκπαίδευση του μοντέλου και γίνεται αυτόματα από την εφαρμογή μηχανικής μάθησης.
- Επιλογή νέων δειγμάτων για τιτλοφόρηση, με χρήση ενός από τα κριτήρια δειγματοληψίας στην ενεργητική μάθηση. Η επιλογή γίνεται μέσω της εφαρμογής μηχανική μάθησης η οποία όμως ενεργοποιείται και πάλι από την εφαρμογή εξυπηρετητή.
- Σύνολο μη τιτλοφορημένων δεδομένων, που αποτελεί την δεξαμενή δειγμάτων πό τα οποία επιλέγονται τα νέα δείγματα για τιτλοφόρηση. Οι πληροφορίες για τα δείγματα μεταφέρονται από την εφαρμογή του εξυπηρετητή στην εφαρμογή για κινητές συσκευές.
- Πληθοπορισμός, με τον οποίο επιτυγχάνεται η τιτλοφόρηση νέων δειγμάτων, αφορά άμεσα την εφαρμογή εξυπηρετητή και την εφαρμογή για κινητές συσκευές και μας δίνει σαν έξοδο το
- Σύνολο τιτλοφορημένων δεδομένων, που προστίθεται στο σύνολο δειγμάτων εκπαίδευσης και σχετίζεται και αυτό με την αλληλεπίδραση της εφαρμογής για κινητές συσκευές με την εφαρμογή εξυπηρετητή.

Στον πίνακα 2 παρουσιάζονται οι σχέσεις μεταξύ των συστατικών της αρχιτεκτονικής, των χαρακτηριστικών της υλοποίησης, και των επιμέρους εφαρμογών αυτής.

Πίνακας 2: Σύνδεση στοιχείων αρχιτεκτονικής με τα χαρακτηριστικά της υλοποίησης και τις επιμέρους εφαρμογές

Στοιχείο αρχιτεκτονικής	**Σχετικά Χαρακτηριστικά**	**Σχετικές Εφαρμογές**
Αρχικοποίηση μοντέλου	X1	E1, E2
Εκπαίδευση μοντέλου	X2, X4	E1, E2
Έλεγχος μοντέλου	X2	E1
Δειγματοληψία	X2	E1
Πληθοπορισμός	X3	E1,E3
Τιτλοφορημένα δεδομένα	X2, X3, X4	E1
Μη τιτλοφορημένα δεδομένα	X2, X3	E1,E3

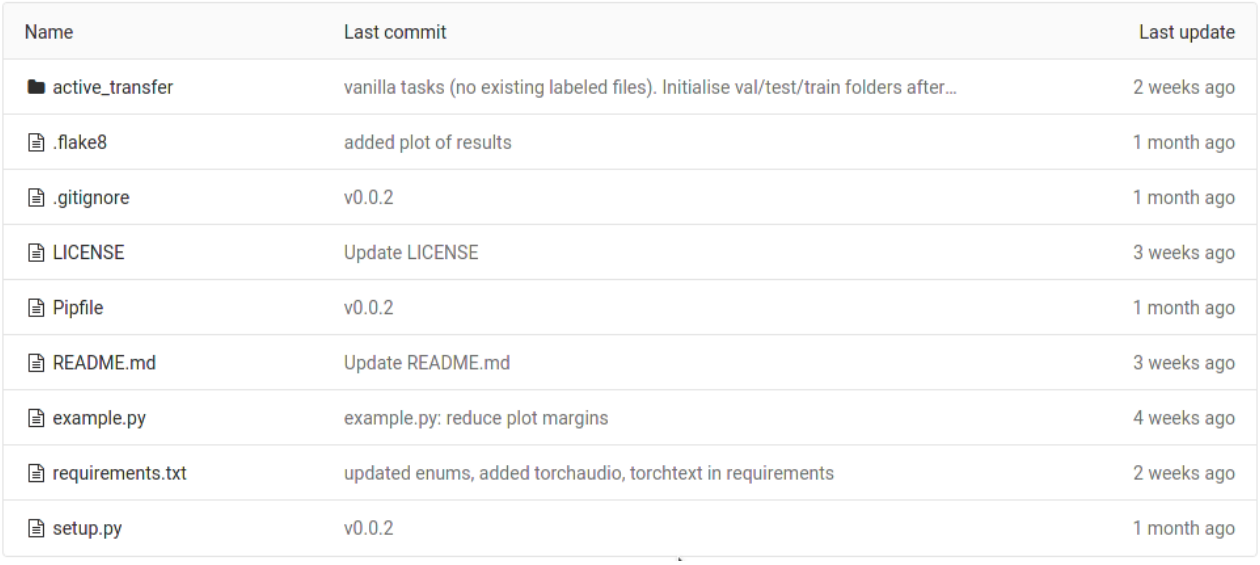
Έχοντας τις συνδέσεις μεταξύ χαρακτηριστικών και εφαρμογών μπορούμε να δούμε και τις αντίστοιχες Διεπαφές Προγραμματισμού Εφαρμογών (Application Programming Interfaces, APIs) που θα υλοποιηθούν και επιτρέπουν την επικοινωνία μεταξύ των επιμέρους εφαρμογών. Η υλοποίηση της εφαρμογής της μηχανικής μάθησης ως ξεχωριστό python πακέτο, μας επιτρέπει την χρήση του σαν προαπαιτούμενη βιβλιοθήκη για την εγκατάσταση της εφαρμογής εξυπηρετητή, ενώ για την επικοινωνία μεταξύ εξυπηρετητή και εφαρμογής για κινητές συσκευές θα υλοποιηθεί μια διεπαφή για επικοινωνία μέσω πρωτοκόλλου Hypertext Transfer Protocol (HTTP).

3.2 Εφαρμογή μηχανικής μάθησης

Αρχικά υλοποιήθηκε η εφαρμογή μηχανικής μάθησης με την δημιουργία ενός python πακέτου το οποίο μπορεί να εγκατασταθεί μεμονωμένα σε οποιονδήποτε υπολογιστή έχει εγκαταστημένη την έκδοση 3.7 της γλώσσας προγραμματισμού Python. Ο πηγαίος κώδικας, συνοδευόμενος με τις οδηγίες εγκατάστασης και ένα αρχείο με παράδειγμα χρήσης αυτού είναι διαθέσιμος στο αποθετήριο ανοικτού κώδικα gitlab [26] (εικόνα 18). Η ανάπτυξή του περιλαμβάνει μία αφηρημένη (abstract) κλάση μηχανικής μάθησης (αρχείο 10), η οποία απαιτεί την υλοποίηση των επιμέρους συναρτήσεων της εκπαίδευσης, δοκιμής και

[26] https://gitlab.com/lazToum/active_transfer

δειγματοληψίας ενός μοντέλου, μία υλοποίησης της κλάσης αυτής για προβλήματα δυαδική κατηγοριοποίησης σε προβλήματα μηχανικής όρασης, καθώς και τα απαραίτητα βοηθητικά αρχεία διαχείρισης των παραμέτρων και των αρχείων που χρησιμοποιούνται.



Name	Last commit	Last update
active_transfer	vanilla tasks (no existing labeled files). Initialise val/test/train folders after...	2 weeks ago
.flake8	added plot of results	1 month ago
.gitignore	v0.0.2	1 month ago
LICENSE	Update LICENSE	3 weeks ago
Pipfile	v0.0.2	1 month ago
README.md	Update README.md	3 weeks ago
example.py	example.py: reduce plot margins	4 weeks ago
requirements.txt	updated enums, added torchaudio, torchtext in requirements	2 weeks ago
setup.py	v0.0.2	1 month ago

Εικόνα 18: Αποθετήριο εφαρμογής μηχανικής μάθησης

3.3 Εφαρμογή εξυπηρετητή

Η ανάπτυξη της διαδικτυακής εφαρμογής, πραγματοποιήθηκε όπως είδαμε με την βιβλιοθήκη FastAPI, η οποία προσφέρει εκ των προτέρων και την δυνατότητα περιήγησης της τεκμηρίωσης του τελικού προϊόντος με γραφικό περιβάλλον. Η λίστα με τις διαθέσιμες μεθόδους την σχετική τοποθεσία και την περιγραφή αυτών παρουσιάζονται στον πίνακα 3. Στα αρχεία 2 και 14 μπορούμε να δούμε την μοντελοποίηση των μηνυμάτων που ανταλλάσσονται μεταξύ του εξυπηρετητή και της εφαρμογής για κινητές συσκευές.

Πίνακας 3: Τεκμηρίωση διαδικτυακής εφαρμογής

Μέθοδος	Σχετική Τοποθεσία	Περιγραφή
GET	/oauth/token	Έλεγχος εγκυρότητας αδειοπλαισίου πρόσβασης
POST	/oauth/token	Λήψη αδειοπλαισίου πρόσβασης
POST	/oauth/token/revoke	Κατάργηση αδειοπλαισίου πρόσβασης
GET	/api/ping	Έλεγχος συνδεσιμότητας
GET	/api/categories	Λήψη διαθέσιμων κατηγοριών
GET	/api/categories/{category}	Λεπτομέρειες συγκεκριμένης κατηγορίας
GET	/api/categories/{category}/subcategories	Υποκατηγορίες συγκεκριμένης κατηγορίας
GET	/api/tasks	Λήψη διαθέσιμων εργασιών με δυνατότητα φιλτραρίσματος αποτελεσμάτων ανά κατηγορία ή υποκατηγορία
POST	/api/tasks	Δημιουργία νέας εργασίας
GET	/api/tasks/{task_id}	Λεπτομέρειες συγκεκριμένης εργασίας
PATCH	/api/tasks/{task_id}	Αλλαγή παραμέτρων συγκεκριμένης εργασίας
DELETE	/api/tasks{task_id}	Διαγραφή συγκεκριμένης εργασίας
POST	/api/upload	Ανέβασμα αρχείου(ων)
GET	/api/tasks/{task_id}/resources	Λήψη δεδομένων για τιτλοφόρηση συγκεκριμένης εργασίας
GET	/api/tasks/{task_id}/values	Λήψη υπαρχουσών απαντήσεων για συγκεκριμένη εργασία
POST	/api/tasks/{task_id}/values	Υποβολή απαντήσεων για συγκεκριμένη εργασία
GET	/api/results/{task_id}	Λήψη αποτελεσμάτων συγκεκριμένης διεργασίας

Κατά την εκκίνηση της εφαρμογής, και εφόσον δεν έχουν ήδη δημιουργηθεί, δημιουργούνται και αποθηκεύονται οι κατηγορίες διεργασιών, “εικόνας”, “ήχου”, “κειμένου” και “άλλο”, καθώς και οι υποκατηγορίες διεργασιών εικόνας “δυαδικής κατηγοριοποίησης”, “κατηγοριοποίησης πολλαπλών κλάσεων”, “τιτλοφόρησης αντικειμένων” και “ανίχνευσης αντικειμένων” (Αρχείο 1).

Αρχείο 1: categories_response.json

Αναπαράσταση σε μορφή JSON της απάντησης του εξυπηρετητή με την λίστα των διαθέσιμων κατηγοριών διεργασιών.

```
[{
    "name": "image",
    "cover": "static/image.png",
    "description": "Image",
    "subcategories": [{
        "name": "binary",
        "cover": "static/image_binary.png",
        "description": "Binary Classification"
      },{
        "name": "multi",
        "cover": "static/image_multi.png",
        "description": "Multi-class Classification"
      },{
        "name": "caption",
        "cover": "static/image_caption.png",
        "description": "Object Captioning"
      },{
        "name": "detect",
        "cover": "static/image_detect.png",
        "description": "Object Detection"
      }]
  }, {
    "name": "text",
    "cover": "static/text.png",
    "description": "Text",
```

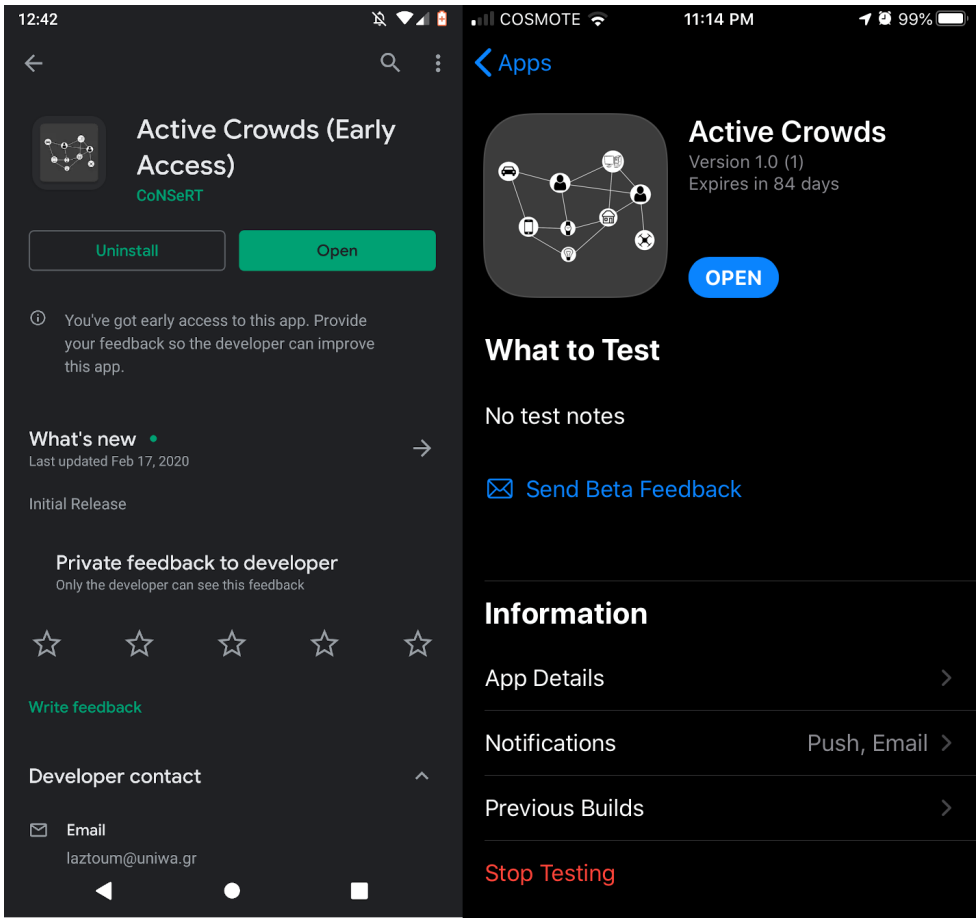
```
      "subcategories": []
    },{
      "name": "audio",
      "cover": "static/audio.png",
      "description": "Audio",
      "subcategories": []
    },{
      "name": "other",
      "cover": "static/other.png",
      "description": "Other",
      "subcategories": []
  }]
```

Όπως μπορούμε να δούμε και στην τεκμηρίωση της διαδικτυακής εφαρμογής, η επικοινωνία με τον εξυπηρετητή, απαιτεί την αυθεντικοποίηση του πελάτη (στην δική μας περίπτωση της εφαρμογής για κινητές συσκευές). Η ανάπτυξη του αντίστοιχου μηχανισμού περιλαμβάνει μία από τις διαθέσιμες ροές αυθεντικοποίησης του προτύπου OAuth 2.0, αυτήν των διαπιστευτηρίων του πελάτη (client credentials). Η ροή αυτή δεν προϋποθέτει την σύνδεση χρηστών στην πλατφόρμα, αλλά χρησιμοποιείται μόνο για την αυθεντικοποίηση μεταξύ υπηρεσιών και περιλαμβάνει χρήση ενός ζεύγους συμβολοσειρών (client_key, client_secret) που απαιτείται για την λήψη ενός αδειοπλαισίου πρόσβασης (access token), το οποίο χρησιμοποιείται από τον πελάτη και επαληθεύεται από τον εξυπηρετητή στις περαιτέρω ανταλλαγές μηνυμάτων μεταξύ τους.

3.4 Εφαρμογή για κινητές συσκευές

Το ζεύγος αυτό των συμβολοσειρών δημιουργείται και αποθηκεύεται στον εξυπηρετητή και χρησιμοποιείται στην εφαρμογή κινητών εφαρμογών. Στην εικόνα 19 μπορούμε να δούμε την προεπισκόπηση της εφαρμογής στις δύο μεγαλύτερες αγορές για κινητές συσκευές. Συσκευές με λειτουργικό σύστημα Android μπορούν να εγκαταστήσουν την beta έκδοση της εφαρμογής από το αντίστοιχο κατάστημα[27], ενώ για συσκευές λειτουργικού συστήματος iOS, απαιτείται η χρήση του βοηθητικού προγράμματος testflight [28].

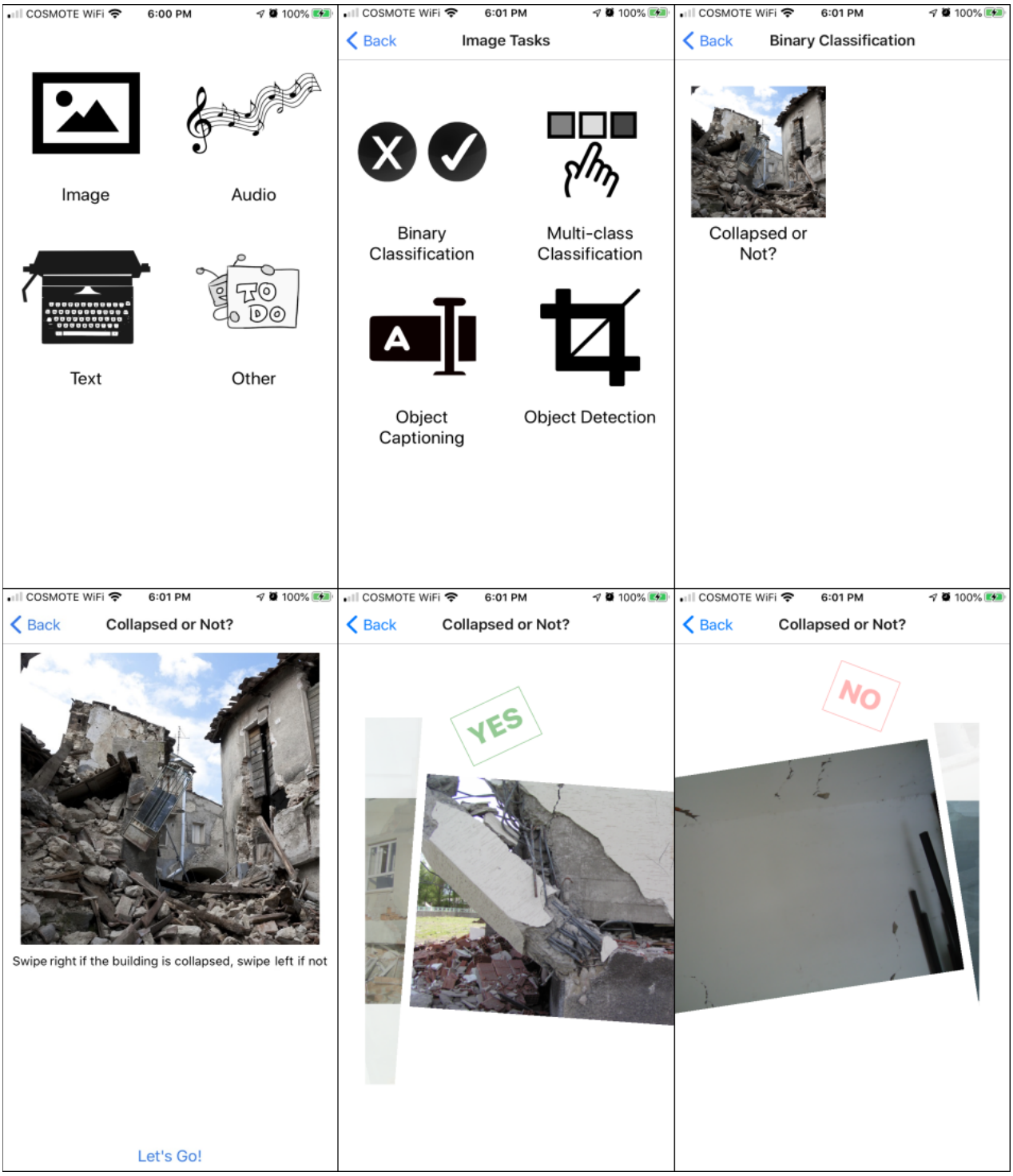
[27] https://play.google.com/store/apps/details?id=gr.uniwa.eee.consert.ac
[28] https://developer.apple.com/testflight/



Εικόνα 19: Προεπισκόπηση εφαρμογής στις αγορές για κινητές συσκευές με λειτουργικά συστήματα Android και iOS

Κατά την εκκίνηση της εφαρμογής, λαμβάνεται το αδειοπλαίσιο πρόσβασης και στη συνέχεια οι κύριες κατηγορίες πληθοπορισμού. Οι χρήστες, επιλέγοντας μία από τις κατηγορίες αυτές μπορούν να δουν τις υποκατηγορίες αυτής (Εικόνα 20). Σημειώνουμε πως παρόλο που μόνο εργασίες σχετικές με εικόνες και μόνο δυαδικής κατηγοριοποίησης έχουν αναπτυχθεί μέχρι στιγμής, ο εξυπηρετητής αποστέλλει επίσης τις κατηγορίες ήχου, κειμένου και λοιπών (π.χ. μετρήσεις από αισθητήρες της συσκευής) καθώς και τις υποκατηγορίες εικόνας πολλαπλής κατηγοριοποίησης, του χαρακτηρισμού εικόνων, καθώς και του εντοπισμού αντικειμένων σε εικόνα, κατηγορίες και υποκατηγορίες η ανάπτυξη των οποίων αποτελεί έναν από τους μελλοντικούς μας στόχους. Στην περίπτωση που ο εξυπηρετητής αποστείλει μια υποκατηγορία με ενεργές διεργασίες, αυτές εμφανίζονται στην επόμενη οθόνη και ο χρήστης μπορεί να δει τον τίτλο και μια ενδεικτική εικόνα αυτών. Επιλέγοντας μία από αυτές τις διεργασίες, μπορεί να δει την περιγραφή της και έχει την επιλογή να συνεισφέρει στην εκτέλεση αυτής. Αφού ο χρήστης ολοκληρώσει την τιτλοφόρηση, μπορεί να αποστείλει τις

απαντήσεις του και να επιστρέψει στην αρχική σελίδα της εφαρμογής. Η εφαρμογή αποθηκεύει τοπικά ένα μοναδικό χαρακτηριστικό της διεργασίας αυτής και σε περίπτωση που αυτό αλλάξει (επιλεχθεί νέο δείγμα για τιτλοφόρηση από την εφαρμογή ενεργητικής μάθησης), εμφανίζει εκ νέου την εργασία αυτή και ο χρήστης έχει την δυνατότητα να συνεισφέρει ξανά τις απαντήσεις του για τα νέα δείγματα.



Εικόνα 20: Οθόνες της εφαρμογής για κινητές συσκευές

ΚΕΦΑΛΑΙΟ 4:
Πείραμα, ανάλυση και αξιολόγηση αποτελεσμάτων

4.1 Πείραμα

Μετά την ολοκλήρωση της διαδικτυακής εφαρμογής, χρησιμοποιήθηκε το προσφερόμενο γραφικό περιβάλλον για την δημιουργία μιας νέας εργασίας τιτλοφόρησης που αφορά δυαδική κατηγοριοποίηση εικόνων και περιλαμβάνει όπως είδαμε στο ενότητα 1.2 τον χαρακτηρισμό εικόνων ως κατεδαφισμένες (“collapsed”) ή μη “non collapsed”. Οι παράμετροι της νέας ανάθεσης εργασίας αφορούν τον τίτλο, την περιγραφή, την κατηγορία και υποκατηγορία της (εικόνα, δυαδική κατηγοριοποίηση), το κριτήριο για την απόφαση ως προς την κατηγορία ενός δείγματος, τον αριθμό των δειγμάτων που χρειάζονται για τις φάσεις της ενεργητικής μάθησης, καθώς και παραμέτρους σχετικές με το μοντέλο εκπαίδευσης (Αρχείο 2). Το κριτήριο που επιλέχθηκε για την απόφαση χαρακτηρισμού ενός νέου δείγματος, είναι η απαίτηση λήψης απαντήσεων από τουλάχιστον 10 χρήστες (min_peers_count=10) και η ανάγκη συμφωνίας στις απαντήσεις από τουλάχιστον του 80% (min_peer_accuracy=0.8) των απαντήσεων (για παράδειγμα αν 8 στους 10 δώσουν την ίδια απάντηση για την κατηγορία ενός δείγματος, τότε αποφασίζεται ότι το δείγμα αυτό ανήκει στην κατηγορία αυτή). Στην εικόνα 20 μπορούμε να δούμε τις οθόνες της εφαρμογής για κινητές συσκευές, κατά την εκκίνησή της και κατά την επιλογή και χαρακτηρισμό των εικόνων της ανάθεσης εργασίας.

Αρχείο 2: task_response.json

Αναπαράσταση σε JSON μορφή της απάντησης του εξυπηρετητή με λεπτομέρειες της συγκεκριμένης διεργασίας.

```
{
  "status": "Ready",
  "title": "Collapsed or Not?",
  "description": "Swipe right if the building is collapsed,
   ↪  swipe left if not",
  "cover": "static/image_binary_collapse.jpg",
  "category": "image",
  "subcategory": "binary",
  "strategy": "entropy",
  "class_names": [
```

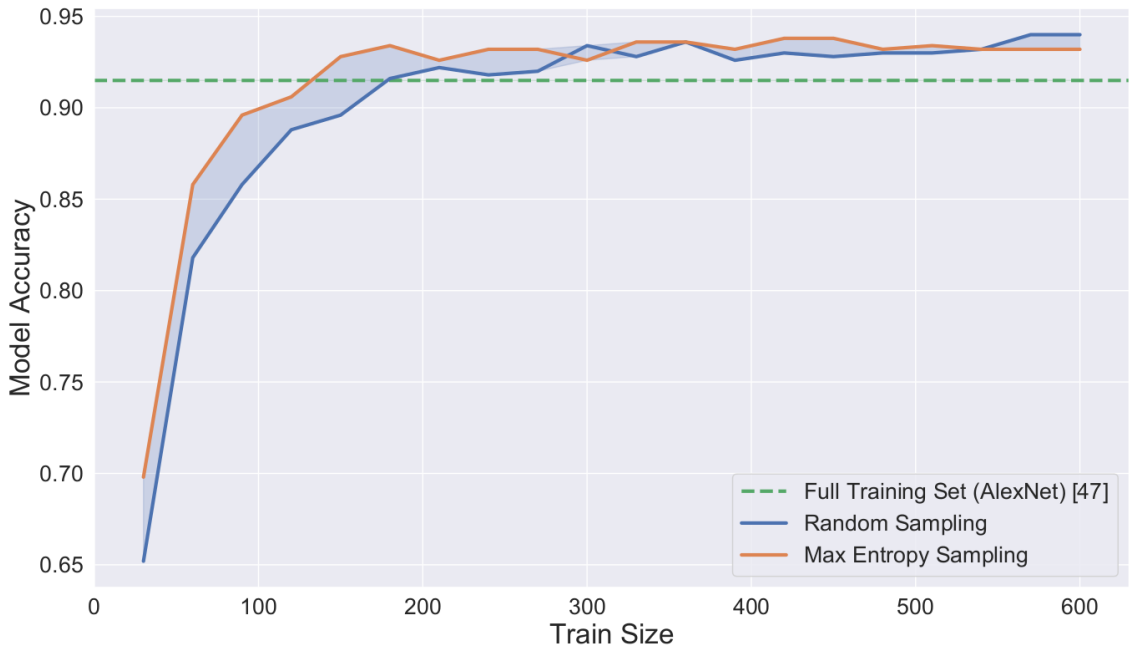
```
      "collapsed",
      "non_collapsed"
    ],
    "epochs": 20,
    "optimizer_lr": 0.001,
    "optimizer_momentum": 0.9,
    "scheduler_step": 7,
    "append_count": 30,
    "val_count": 500,
    "test_count": 500,
    "min_peers_count": 10,
    "peer_accuracy": 0.8,
    "id": "899063e4-477e-401f-8f47-a3a514e20782",
    "resources_id": "e49d96bf-b432-4de0-8bd3-d29da797bcb7",
    "results": null
  }
```


Ο αριθμός των εικόνων που χρησιμοποιούνται για την επαλήθευση και τον έλεγχο του εκπαιδευμένου μοντέλου είναι από 500, και ο αριθμός των νέων δειγμάτων που προστίθενται στο σύνολο δεδομένων εκπαίδευσης είναι 30.

Εικόνα 21: Προεπισκόπηση Δεδομένων Εκπαίδευσης

Εικόνα 22: Αποτελέσματα Ενδιάμεσης Εκτίμησης

Στην εικόνα 21 μπορούμε να δούμε μία προεπισκόπηση των δεδομένων που χρησιμοποιήθηκαν κατά την εκπαίδευση του μοντέλου, ενώ στην εικόνα 22 μπορούμε να δούμε ενδιάμεσα αποτελέσματα μετά από κάποιες επαναλήψεις της διαδικασίας. Η επιλογή νέων δειγμάτων που προστίθενται στο σύνολο δεδομένων εκπαίδευσης έγινε μία φορά με τυχαίο τρόπο και μία φόρα με την στρατηγική της μέγιστης εντροπίας. Δοκιμάστηκαν επίσης οι στρατηγικές ελάχιστης βεβαιότητας περιθωρίου και μέγιστου περιθωρίου βεβαιότητας, για την δοκιμή της υλοποίησης μας εξακριβώνοντας πως δίνουν το ίδιο ακριβώς αποτέλεσμα με την στρατηγική μέγιστης εντροπίας στην περίπτωση δυαδικής κατηγοριοποίησης. Τα αποτελέσματα που λάβαμε μας δίνουν μια ακρίβεια πρόβλεψης κοντά στο 93% με χρήση λιγότερων από 300 δείγματα εκπαίδευσης (Εικόνα 23).



Εικόνα 23: Ακρίβεια του εκπαιδευμένου μοντέλου, σε συνάρτηση με το πλήθος δεδομένων εκπαίδευσης

4.2 Αποτελέσματα

Όπως είδαμε, ο συνδυασμός της χρήσης μεταφοράς μάθησης (προ-εκπαιδευμένο μοντέλο ResNet), με την ενεργητική μάθηση (διαρκής ανατροφοδότηση του συνόλου δεδομένων εκπαίδευσης με επιλεκτική δειγματοληψία), είχε αποτέλεσμα την επίτευξη ακρίβειας πρόβλεψης του μοντέλου μας στα 93% με χρήση ενός περιορισμένου αριθμού δειγμάτων (λιγότερα από 300) κατά την διαδικασία της εκπαίδευσης. Είναι συνεπώς προφανές πως μπορούμε να επιτύχουμε την μείωση του όγκου των δεδομένων που απαιτούνται για την υλοποίηση του προβλήματος αυτού της δυαδικής κατηγοριοποίησης. Όπως μπορούμε να δούμε στην εικόνα 23, η επιλογή της στρατηγικής μέγιστης εντροπίας, σε αντίθεση με την τυχαία επιλογή δειγμάτων επιφέρει ταχύτερη βελτίωση της ακρίβειας του μοντέλου. Μία ακόμα ποσοτική ανάλυση των αποτελεσμάτων μπορούμε να δούμε και στον πίνακα σύγχυσης στην εικόνα 24, όπου κατά την διάρκεια του ελέγχου του μοντέλου επιτυγχάνουμε 469 ορθές προβλέψεις από τις 500 (315 Ορθώς Θετικές, True Positive, και 154 Ορθώς Αρνητικές, True Negative, έναντι 9 Λανθασμένα Θετικών, False Positive και 22 Λανθασμένα Αρνητικών, False Negative) .

Εικόνα 24: Πίνακας συγχύσεως

4.3 Προβλήματα και μελλοντικές επεκτάσεις

Οι απαιτήσεις για τον αριθμό των χρηστών / απαντήσεων στις αναθέσεις εργασίας πληθοπορισμού καθώς και για τν ελάχιστο ποσοστό συμφωνίας στις απαντήσεις, δεν μας επέτρεψαν να δοκιμάσουμε σε πραγματικές συνθήκες την υλοποίησή μας, όπου πιθανές λανθασμένες απαντήσεις θα μπορούσαν να τροφοδοτήσουν το σύνολο εκπαίδευση του μοντέλου μηχανικής μάθησης, γι' αυτό και χρησιμοποιήσαμε το ήδη τιτλοφορημένο σύνολο δεδομένων. Επίσης, η έλλειψη περαιτέρω συνόλων δεδομένων δεν μας επέτρεψε την δοκιμή τόσο σε διαφορετικό είδος τιτλοφόρησης όσο και σε παράλληλη λειτουργία του συστήματος με περισσότερα του ενός προβλήματα. Παρά την δοκιμή μόνο σε πρόβλημα δυαδικής κατηγοριοποίησης όμως, για η δειγματοληψία νέων δεδομένων δοκιμάστηκε και επαληθεύτηκε με χρήση εκτός του κριτηρίου μέγιστης εντροπίας, αλλά και αυτών της ελάχιστης βεβαιότητας και του περιθωρίου βεβαιότητας. Μελλοντικές επεκτάσεις της εφαρμογής μας περιλαμβάνουν λοιπόν τόσο την εφαρμογή προβλημάτων διαφορετικών ειδών τιτλοφόρησης στην μηχανική όραση, όσο και την υλοποίηση διαχείρισης προβλημάτων διαφορετικών τομέων στην μηχανική μάθησης όπως αυτός της επεξεργασίας ήχου και κειμένου. Τέλος, η προσθήκη επιπλέον προ-εκπαιδευμένων μοντέλων αλλά και νέων στρατηγικών δειγματοληψίας για το στάδιο της ενεργητικής μάθησης θα μας επιτρέψει να μελετήσουμε ακόμα καλύτερα την συμπεριφορά της εφαρμογής.

ΚΕΦΑΛΑΙΟ 5:
Συμπεράσματα – Προτάσεις

Στο τελευταίο αυτό κεφάλαιο της εργασίας μας, παρουσιάζονται τα συμπεράσματα που αποκομίστηκαν στο σύνολό της καθώς και τις προοπτικές που ανοίχθηκαν από αυτήν. Σε θεωρητικό επίπεδο μελετήθηκε η ανάγκη της συλλογής και τιτλοφόρησης δεδομένων, μιας χρονοβόρας και δύσκολης διαδικασίας, ενός ένα από τα σημαντικότερα προβλήματα που υπάρχουν σε συστήματα μηχανικής μάθησης καθώς και δύο συχνά χρησιμοποιούμενες μεθόδους μείωσης του τελικού απαιτούμενο όγκου δεδομένων, την τεχνική της μεταφοράς μάθησης, και αυτήν της ενεργητικής μάθησης. Καθοριστικό ρόλο στην διαδικασία αυτή μπορεί να παίξει ο άνθρωπος, γεγονός που εκμεταλλεύονται τα συστήματα πληθοπορισμού.
Παρουσιάστηκε μια πρακτική υλοποίηση συνδυασμού των τριών αυτών τεχνικών, και εφαρμόστηκε σε αυτήν σαν περίπτωση χρήσης ένα πρόβλημα δυαδικής κατηγοριοποίησης στον κλάδο της μηχανικής όρασης. Όπως αναφέρεται και στο τέλος του προηγούμενου κεφαλαίου, η μελέτη εφαρμογής τόσο των επιπλέον στρατηγικών δειγματοληψίας όσο και των τεχνικών μεταφοράς μάθησης, έχουν μεγάλο ερευνητικό ενδιαφέρον, δεν ήταν όμως δυνατόν να καλυφθούν στα πλαίσια της εργασίας αυτής. Παρόλο που η εφαρμογή αυτή καλύπτει ένα πολύ μικρό μέρος των τεχνικών αυτών, παρέχει ωστόσο την δυνατότητα, και αποτελεί επόμενο ερευνητικό στόχο, να χρησιμοποιηθεί ώς κορμός για περαιτέρω επεκτάσεις και σε επιπλέον κλάδους της μηχανικής μάθησης όπως η επεξεργασία ήχου και κειμένου, αλλά και σε επιπλέον προβλήματα στον κλάδο της μηχανικής όρασης, καθώς και στην μελέτη του τρόπου ανάθεσης των εργασιών πληθοπορισμού και την ελαχιστοποίηση των λάθος τιτλοφορήσεων που προκύπτουν από απαντήσεις σε συστήματα πληθοπορισμού.

ΒΙΒΛΙΟΓΡΑΦΙΑ - ΠΗΓΕΣ

Βιβλιογραφία – Πηγές στην ελληνική γλώσσα

[1] Α. Γεωργούλη, "Μηχανική Μάθηση", στο *Τεχνητή νοημοσύνη*. Σύνδεσμος Ελληνικών Ακαδημαϊκών Βιβλιοθηκών, 2015, κεφ. 4.

[2] Β. Καμπουρλάζος και Γ. Παπακώστας, *Εισαγωγή στην υπολογιστική νοημοσύνη*. Σύνδεσμος Ελληνικών Ακαδημαϊκών Βιβλιοθηκών, 2015.

[3] Π. Κασνέσης, "Χρήση γνωσιακών πρακτόρων με επίγνωση πλαισίου για τη δημιουργία κοινωνικού διαδικτύου των πραγμάτων", Διδακτορική διατρ., Εθνικό Μετσόβιο Πολυτεχνείο, 2018.

Βιβλιογραφία – Πηγές σε ξένες γλώσσες

[4] K. Schwab, *The fourth industrial revolution*. Currency, 2017, ISBN: 9780241300756.

[5] S. J. Pan and Q. Yang, "A survey on transfer learning", *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, 2010. DOI: 10.1109/TKDE.2009.191.

[6] A. R. Ali, M. Budka, and B. Gabrys, "Towards meta-learning of deep architectures for efficient domain adaptation", in *PRICAI 2019: Trends in Artificial Intelligence*, A. C. Nayak and A. Sharma, Eds., Cham: Springer International Publishing, 2019, pp. 66–79.

[7] B. Settles, "Active learning literature survey", 2009.

[8] J. Howe, "The rise of crowdsourcing", *Wired magazine*, vol. 14, no. 6, pp. 1–4, Jun. 2006.

[9] B. Morschheuser, J. Hamari, J. Koivisto, and A. Maedche, "Gamified crowdsourcing: Conceptualization, literature review, and future agenda", *International Journal of Human-Computer Studies*, vol. 106, pp. 26–43, Oct. 2017. DOI: 10.1016/j.ijhcs.2017.04.005.

[10] T. Tian, J. Zhu, and Y. Qiaoben, "Max-margin majority voting for learning from crowds", *IEEE transactions on pattern analysis and machine intelligence*, vol. 41, no. 10, pp. 2480–2494, 2018.

[11] A. J. Quinn and B. B. Bederson, "Human computation: A survey and taxonomy of a growing field", New York, NY, USA: ACM, 2011, pp. 1403–1412, ISBN: 978-1-4503-0228-9. DOI: 10.1145/1978942.1979148.

[12] A. M. Turing, "Computing machinery and intelligence", *Mind*, vol. 59, no. 236, pp. 433–460, 1950, ISSN: 00264423, 14602113.

[13] A. L. Samuel, “Some studies in machine learning using the game of checkers”, *IBM Journal of Research and Development*, vol. 3, no. 3, pp. 210–229, 1959. DOI: 10.1147/rd.33.0210.

[14] T. M. Mitchell, *Machine Learning*, 1st ed. New York, NY, USA: McGraw-Hill, Inc., 1997, ISBN: 9780070428072.

[15] R. Agrawal, T. Imieliński, and A. Swami, “Mining association rules between sets of items in large databases”, *SIGMOD Rec.*, SIGMOD ’93, pp. 207–216, Jun. 1993. DOI: 10.1145/170035.170072.

[16] I. Goodfellow, Y. Bengio, A. Courville, and F. Bach, “Deep learning (adaptive computation and machine learning series) mit press”, *Cambridge, MA, USA*, p. 800, 2016.

[17] R. Raina, A. Battle, H. Lee, B. Packer, and A. Y. Ng, “Self-taught learning: Transfer learning from unlabeled data”, in *Proceedings of the 24th International Conference on Machine Learning*, ser. ICML ’07, Corvalis, Oregon, USA: ACM, 2007, pp. 759–766, ISBN: 978-1-59593-793-3. DOI: 10.1145/1273496.1273592.

[18] Y. LeCun, C. Cortes, and C. Burges, “Mnist handwritten digit database”, *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, vol. 2, 2010.

[19] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks”, *Advances in neural information processing systems*, vol. 25, pp. 1097–1105, 2012.

[20] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “Imagenet: A large-scale hierarchical image database”, in *2009 IEEE conference on computer vision and pattern recognition*, Ieee, 2009, pp. 248–255.

[21] O. Russakovsky, J. Deng, H. Su, *et al.*, “Imagenet large scale visual recognition challenge”, *International journal of computer vision*, vol. 115, no. 3, pp. 211–252, 2015.

[22] C. Szegedy, W. Liu, Y. Jia, *et al.*, “Going deeper with convolutions”, in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[23] H. Kataoka, K. Iwata, and Y. Satoh, “Feature evaluation of deep convolutional neural networks for object recognition and detection”, *arXiv preprint arXiv:1509.07627*, 2015.

[24] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition”, in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[25] *Cs231n convolutional neural networks for visual recognition*.

[26] S. Hosein, *A beginner’s guide to active learning*.

[27] D. D. Lewis and W. A. Gale, “A sequential algorithm for training text classifiers”, in *SIGIR’94*, Springer, 1994, pp. 3–12.

[28] A. K. McCallumzy and K. Nigamy, “Employing em and pool-based active learning for text classification”, in *Proc. International Conference on Machine Learning (ICML)*, Citeseer, 1998, pp. 359–367.

[29] S. Hoi, R. Jin, and M. Lyu, “Large-scale text categorization by batch mode learning”, in *Proc 15th Int World Wide Web conference (WWW2006)*, Edinburgh England, UK, 2006.

[30] C. A. Thompson, M. E. Califf, and R. J. Mooney, “Active learning for natural language parsing and information extraction”, in *ICML*, Citeseer, 1999, pp. 406–414.

[31] B. Settles and M. Craven, “An analysis of active learning strategies for sequence labeling tasks”, in *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, 2008, pp. 1070–1079.

[32] C. Zhang and T. Chen, “An active learning framework for content-based information retrieval”, *IEEE transactions on multimedia*, vol. 4, no. 2, pp. 260–268, 2002.

[33] S. Tong and E. Chang, “Support vector machine active learning for image retrieval”, in *Proceedings of the ninth ACM international conference on Multimedia*, 2001, pp. 107–118.

[34] A. G. Hauptmann, W.-H. Lin, R. Yan, J. Yang, and M.-Y. Chen, “Extreme video retrieval: Joint maximization of human and computer performance”, in *Proceedings of the 14th ACM international conference on Multimedia*, 2006, pp. 385–394.

[35] G. Tur, D. Hakkani-Tür, and R. E. Schapire, “Combining active and semi-supervised learning for spoken language understanding”, *Speech Communication*, vol. 45, no. 2, pp. 171–186, 2005.

[36] Y. Liu, “Active learning with support vector machine applied to gene expression data for cancer classification”, *Journal of chemical information and computer sciences*, vol. 44, no. 6, pp. 1936–1941, 2004.

[37] R. Munro, *Human-in-the-Loop Machine Learning: Active Learning and Annotation for Human-centered AI*. Manning Publications, 2020, ISBN: 9781617296741.

[38] A. Ng, “Machine learning yearning”, *URL: http://www. mlyearning. org/(96)*, vol. 139, 2017.

[39] J. Surowiecki, *The Wisdom of Crowds*. Random House LCC US, Sep. 11, 2005, ISBN: 0385721706.

[40] M. Talasila, R. Curtmola, and C. Borcea, “Crowdsensing in the wild with aliens and micropayments”, *IEEE Pervasive Computing*, vol. 15, no. 1, pp. 68–77, Jan. 2015, ISSN: 1536-1268. DOI: 10.1109/MPRV.2016.18.

[41] J. Burke, D. Estrin, M. Hansen, *et al.*, "Participatory sensing", in *In: Workshop on World-Sensor-Web (WSW'06): Mobile Device Centric Sensor Networks and Applications*, 2006, pp. 117–134.

[42] B. Guo, Z. Yu, X. Zhou, and D. Zhang, "From participatory sensing to mobile crowd sensing", *2014 IEEE International Conference on Pervasive Computing and Communication Workshops, Percom Workshops 2014*, pp. 593–598, 2014. DOI: 10.1109/PerComW.2014.6815273. arXiv: 1401.3090.

[43] R. K. Ganti, F. Ye, and H. Lei, "Mobile crowdsensing: Current state and future challenges", *IEEE Communications Magazine*, vol. 49, no. 11, pp. 32–39, 2011, ISSN: 01636804. DOI: 10.1109/MCOM.2011.6069707.

[44] D. Moher, L. Shamseer, M. Clarke, *et al.*, "Preferred reporting items for systematic review and meta-analysis protocols (prisma-p) 2015 statement", *Systematic Reviews*, vol. 4, no. 1, 2015. DOI: 10.1186/2046-4053-4-1.

[45] J. W. Creswell and J. D. Creswell, *Research design: Qualitative, quantitative, and mixed methods approaches*. SAGE Publications, Inc, 2018.

[46] C. M. Yeum, S. J. Dyke, and J. Ramirez, "Visual data classification in post-event building reconnaissance", *Engineering Structures*, vol. 155, pp. 16–24, 2018, ISSN: 0141-0296. DOI: 10.1016/j.engstruct.2017.10.057.

[47] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks", in *Advances in Neural Information Processing Systems*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds., vol. 25, Curran Associates, Inc., 2012.

ΠΑΡΑΡΤΗΜΑΤΑ

Α΄ Δημοσιεύσεις

Δημοσιευμένες Εργασίες

- L. Toumanidis, R. Heartfield, P. Kasnesis, G. Loukas, and C. Patrikakis, "A prototype framework for assessing information provenance in decentralised social media: The eunomia concept", in *E-Democracy – Safeguarding Democracy and Human Rights in the Digital Age*, S. Katsikas and V. Zorkadis, Eds., Cham: Springer International Publishing, 2020, pp. 196–208, ISBN: 978-3-030-37545-4.
- C. Chatzigeorgiou, P. Kasnesis, and L. G. Toumanidis, "Exploiting edge computing for privacy aware tourism demand forecasting", *IEEE IT Professional*, vol. 21, no. 3, pp. 19–25, May 2019, ISSN: 1520-9202. DOI: 10.1109/MITP.2019.2901876, IF: 2.424.
- L. Toumanidis, E. Bocaj, P. Kasnesis, and C. Z. Patrikakis, "Supporting cultural heritage preservation through game-based crowdsourcing", in *Strategic Innovative Marketing and Tourism*, Springer, 2019, pp. 989–997.

Επόμενες υποβολές

- L. Toumanidis, P. Kasnesis, C. Chatzigeorgiou, and C. Patrikakis, "ActiveCrowds: A human-in-the-loop machine learning platform", *MDPI Sensors (ISSN 1424-8220), Special Issue "Advances in Machine Learning for Intelligent Engineering Systems and Applications II"*. IF: 3.031
- Jongseong Choi, Kasnesis Panagiotis, Toumanidis Lazaros, Shirley J. Dyke, Patrikakis Charalampos, Chul Min Yeum, Ali Lenjani, and Xiaoyu Liu, "Automated Graffiti Detection: A Novel Approach for Maintaining Historical Structures in Community", *Journal of Computing and Cultural Heritage*, CiteScore 2018: 1.83

Β΄ Ενδεικτικά Αρχεία Πηγαίου Κώδικα

Αρχείο 3: .env.template

Πρότυπο αρχείο ορισμού μεταβλητών σχετικών με την εγκατάσταση και εκτέλεση των επιμέρους υπηρεσιών.

```
APP_NAME=ActiveCrowds
DOMAIN_NAME=example.com
CERTBOT_EMAIL=you@example.com
MOBILE_CLIENT_ID=randomlongstringwith55chars
MOBILE_CLIENNT_SECRET=randomlongstringwith40chars
REDIS_PASSWORD=redispassword
REDIS_PORT=
APP_ENV=Dev
POSTGRES_HOST=db
POSTGRES_PORT=5432
POSTGRES_USER=dbuser
POSTGRES_PASSWORD=dbpassword
POSTGRES_DB=dbname
SERVICE_USER=ubuntu
CLIENT_PORT=4000
EMAIL_USE_TLS=1
EMAIL_USE_SSL=0
EMAIL_HOST=smtp.domain.com
EMAIL_HOST_USER=youremail@domain.com
EMAIL_HOST_PASSWORD=yourpassword
EMAIL_PORT=587
DOCKERFILE=Dockerfile
```

Αρχείο 4: server/requirements.txt

Προαπαιτούμενες βιβλιοθήκες για την εγκατάσταση της εφαρμογής του εξυπηρετητή.

```
alembic==1.4.0
aiofiles==0.4.0
apispec==3.2.0
black==19.10b0
celery==4.4.0
email_validator==1.0.5
eventlet==0.25.1
```

```
fastapi==0.48.0
gunicorn==20.0.4
mako==1.1.1
oauthlib==3.1.0
pydantic==1.4
python-multipart==0.0.5
psycopg2-binary==2.8.4
redis==3.4.1
ujson==1.35
uvicorn==0.11.2
sqlalchemy==1.3.13
sqlalchemy-utils==0.36.1
```

Αρχείο 5: server/active_transfer/requirements.txt

Προαπαιτούμενες βιβλιοθήκες για την εγκατάσταση της εφαρμογής μηχανικής μάθησης.

```
numpy==1.18.1
torch==1.4.0
torchaudio==0.4.0
torchtext==0.5.0
torchvision==0.5.0
```

Αρχείο 6: Dockerfile.nvidia

Αρχείο δημιουργίας του περιβάλλοντος στο οποίο εγκαθίσταται και ενεργοποιείται η εφαρμογή του εξυπηρετητή. Περιλαμβάνει την εγκατάσταση των προαπαιτούμενων εφαρμογών και βιβλιοθηκών καθώς και την δημιουργία του χρήστη του συστήματος που εκτελεί την εφαρμογή.

```
FROM nvidia/cuda:10.1-base

LABEL maintainer="Lazaros Toumanidis
 ↪  <laztoum@protonmail.com>"
```

```
4
5  ENV PYTHONUNBUFFERED 1
6  ENV NVIDIA_VISIBLE_DEVICES all
7  ENV NVIDIA_DRIVER_CAPABILITIES compute,utility
8
9  RUN apt update && \
10     apt install software-properties-common -y && \
11     apt upgrade -y && \
12     apt install -y python3.7 python3.7-dev
        ↪ python3.7-distutils curl git postgresql-client
        ↪ build-essential libssl-dev
13 RUN curl https://bootstrap.pypa.io/get-pip.py -o get-pip.py
14 RUN python3.7 get-pip.py
15 RUN pip3.7 install --upgrade pip
16 COPY active_transfer/requirements.txt /at_requirements.txt
17 RUN pip3.7 install -r /at_requirements.txt
18 COPY active_transfer .
19 RUN cd ./active_transfer
20 RUN python3.7 ./setup.py install
21 RUN cd ..
22 COPY requirements.txt /requirements.txt
23 RUN pip3.7 install -r /requirements.txt
24 RUN adduser --disabled-login --disabled-password --system
    ↪ --quiet --uid 1000 --group user
25 USER user
```

Αρχείο 7: docker-compose.yml

Αρχείο δήλωσης όλων των επιμέρους εφαρμογών και υπηρεσιών καθώς και της εξάρτησης και συνδεσιμότητας μεταξύ τους.

```
1  version: "3.7"
2  services:
3    nginx:
```

```
    image: nginx:1.17.0-alpine
    container_name: ac_nginx
    volumes:
      - ./nginx/conf.d:/etc/nginx/conf.d
      - ./nginx/certbot/conf:/etc/letsencrypt
      - ./nginx/certbot/www:/var/www/certbot
      - ./nginx/logs:/var/log/nginx
      - ./shared/static:/app/static
    ports:
      - "80:80"
      - "443:443"
    command: "/bin/sh -c 'while :; do sleep 6h & wait
     ↪ $${!}; nginx -s reload; done & nginx -g \"daemon
     ↪ off;\"'"
    restart: unless-stopped
    links:
      - server:server
    depends_on:
      - server
  db:
    container_name: ac_db
    restart: always
    image: postgres:latest
    env_file: .env
    volumes:
      - ./db:/var/lib/postgresql/data
    ports:
      - "5432:5432"
  certbot:
    container_name: ac_certbot
    image: certbot/certbot
    restart: unless-stopped
    volumes:
      - ./nginx/certbot/conf:/etc/letsencrypt
```

```
36        - ./nginx/certbot/www:/var/www/certbot
37      depends_on:
38        - nginx
39      entrypoint: "/bin/sh -c 'trap exit TERM; while :; do
          ↪  certbot renew; sleep 12h & wait $${!}; done;'"
40    server:
41      container_name: ac_server
42      image: "active-crowds:latest"
43      build:
44        context: ./server/
45        dockerfile: ${DOCKERFILE}
46      restart: on-failure
47      env_file: .env
48      volumes: &server_volumes
49        - ./server:/home/user/server
50        - ./shared:/home/user/server/shared
51      links:
52        - redis:redis
53        - db:db
54      depends_on:
55        - db
56        - redis
57        - celery
58      working_dir: /home/user/server
59      healthcheck:
60        test: ["CMD-SHELL", "wget -q --spider --proxy=off
            ↪  localhost:8000/api/ping || exit 1"]
61      command: /bin/bash ./wait-for-postgres-and-run.sh
62    redis:
63      container_name: ac_redis
64      restart: unless-stopped
65      image: redis:5.0.7-alpine
66      env_file: .env
67      ports:
```

```
68        - ${REDIS_PORT}:${REDIS_PORT}
69      entrypoint: redis-server --appendonly yes --port
          ↪ ${REDIS_PORT} --requirepass ${REDIS_PASSWORD}
70      volumes:
71        - ./redis:/data
72      healthcheck:
73        test: ["CMD", "redis-cli", "ping"]
74    celery:
75      container_name: ac_celery
76      image: "active-crowds:latest"
77      build:
78        context: ./server/
79        dockerfile: ${DOCKERFILE}
80      restart: on-failure
81      env_file: .env
82      depends_on:
83        - redis
84      links:
85        - redis:redis
86      volumes:
87        *server_volumes
88      working_dir: /home/user/server
89      command: sh -c "celery --workdir=. -A tasks.celery
          ↪ worker -l info -E -O fair -B -P threads"
```

Αρχείο 8: nginx/conf.d/app.conf

Αρχείο καθορισμού ρυθμίσεων του εξυπηρετητή nginx.

```
1   server {
2       listen 80;
3       server_name example.com;
4       server_tokens off;
5       location /.well-known/acme-challenge/ {
```

```
            root /var/www/certbot;
        }
        location / {
            return 301 https://$host$request_uri;
        }
    }
    server {
        server_tokens off;
        server_name example.com;
        charset utf-8;
        listen [::]:443 ssl ipv6only=on;
        listen 443 ssl;
        ssl_certificate
         ↪ /etc/letsencrypt/live/example.com/fullchain.pem;
        ssl_certificate_key
         ↪ /etc/letsencrypt/live/example.com/privkey.pem;
        include /etc/letsencrypt/options-ssl-nginx.conf;
        ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem;
        add_header Strict-Transport-Security
         ↪ "max-age=15780000; includeSubdomains;",always;
        add_header X-Frame-Options DENY;
        add_header X-Content-Type-Options nosniff;
        client_max_body_size 50M;
        client_body_buffer_size 50M;
        client_body_timeout 60;
        location /static {
            root /app/;
        }
        root /app/static/;
        location / {
            proxy_pass http://server:8000;
            proxy_set_header X-Forwarded-Proto https;
            proxy_set_header X-Url-Scheme $scheme;
            proxy_set_header X-Forwarded-For
             ↪ $proxy_add_x_forwarded_for;
```

```
            proxy_redirect off;
            proxy_set_header Host $host;
            proxy_set_header X-Real-IP $remote_addr;
            proxy_set_header X-Forwarded-Host $server_name;
        }
    }
```

Αρχείο 9: server/app/crud.py

Αρχείο που περιλαμβάνει μεθόδους για την ανάγνωση, δημιουργία, αλλαγή και διαγραφή εγγραφών στην βάση δεδομένων.

```
import os
import copy
import uuid
from datetime import datetime, timedelta
from typing import Optional
from oauthlib.common import generate_token
from sqlalchemy.orm import Session
from active_transfer import (
    Category as ATCategory,
    SubCategory as ATSubCategory,
    QueryStrategy,
)

from . import models, schemas
from .. import STATIC_ROOT


def get_client(db: Session, client: schemas.ClientToken):
    return (
        db.query(models.Client)
        .filter(
            models.Client.client_id == client.client_id,
```

```
            models.Client.client_secret ==
             ↪  client.client_secret,
        )
        .first()
    )


def get_token(db: Session, client: models.Client):
    now = datetime.now()
    token = models.Token(
        token_type="Bearer",
        client_id=client.client_id,
        access_token=generate_token(55),
        refresh_token=generate_token(55),
        expires=now + timedelta(days=30),
    )
    db.add(token)
    db.commit()
    db.refresh(token)
    return token


def refresh_token(db: Session, client: schemas.ClientToken):
    existing: models.Token =
     ↪  db.query(models.Token).filter_by(
        client_id=client.client_id,
         ↪  refresh_token=client.refresh_token
    ).first()
    now = datetime.now()
    if not existing:
        return None
    existing.access_token = generate_token(55)
    existing.refresh_token = generate_token(55)
    existing.expires = now + timedelta(days=30)
```

```
54      db.commit()
55      db.refresh(existing)
56      return existing
57
58
59  def revoke_token(db: Session, instance: schemas.RevokeToken):
60      existing: models.Token =
        ↪  db.query(models.Token).filter_by(
61          client_id=instance.client_id,
            ↪  access_token=instance.token
62      ).first()
63      if not existing:
64          return False
65      db.delete(existing)
66      db.commit()
67      return True
68
69
70  def get_category_by_id(db: Session, category_id: str):
71      return
        ↪  db.query(models.Category).filter(models.Category.id
        ↪  == category_id).first()
72
73
74  def get_category_by_name(db: Session, category_name: str):
75      category = (
76          db.query(models.Category).filter(models.Category.name
            ↪  == category_name).first()
77      )
78      if category:
79          return category
80      return None
81
82
```

```python
def get_subcategories(db: Session, category_name: str):
    category = (
        db.query(models.Category).filter(models.Category.name
        ↪  == category_name).first()
    )
    if category:
        return category.subcategories
    return []


def get_categories(db: Session, skip: int = 0, limit: int =
↪  100):
    return
    ↪  db.query(models.Category).offset(skip).limit(limit).all()


def add_category(db: Session, category:
↪  schemas.CategoryCreate):
    db_item = models.Category(**category.dict())
    db.add(db_item)
    db.commit()
    db.refresh(db_item)
    return db_item


def add_subcategory(
    db: Session, subcategory: schemas.SubCategoryCreate,
    ↪  category: ATCategory
):
    db_item = models.SubCategory(**subcategory.dict(),
    ↪  category_name=category)
    db.add(db_item)
    db.commit()
    db.refresh(db_item)
```

```
111     return db_item
112
113
114 def get_tasks(
115     db: Session,
116     category: Optional[ATCategory] = None,
117     subcategory: Optional[ATSubCategory] = None,
118 ):
119     if category is not None:
120         if subcategory is not None:
121             return (
122                 db.query(models.Task)
123                 .filter(
124                     models.Task.category == category,
125                     models.Task.subcategory == subcategory,
126                 )
127                 .all()
128             )
129         return
            ↪ db.query(models.Task).filter(models.Task.category
            ↪ == category).all()
130     return db.query(models.Task).all()
131
132
133 def add_task(db: Session, task: schemas.TaskCreate):
134     task_dict = task.dict()
135     task_dict["resources_id"] = str(uuid.uuid4())
136     task_dict["subcategory"] =
        ↪ ATSubCategory.from_string(task.subcategory)
137     task_dict["strategy"] =
        ↪ QueryStrategy.from_string(task.strategy)
138     db_item = models.Task(**task_dict)
139     db.add(db_item)
140     db.commit()
```

```
141     db.refresh(db_item)
142     return db_item
143
144
145 def get_task_by_id(db: Session, task_id: str):
146     return db.query(models.Task).filter(models.Task.id ==
        ↪ task_id).first()
147
148
149 def get_task_answers(db: Session, task_id: str):
150     entry: Optional[models.Resource] =
        ↪ db.query(models.Resource).filter(
151         models.Resource.root_path == task_id
152     ).first()
153     if entry is not None:
154         return entry.values
155     return {}
156
157
158 def update_task_by_id(db: Session, task_id: str, task:
    ↪ schemas.TaskUpdate):
159     existing: Optional[models.Task] = get_task_by_id(db,
        ↪ task_id)
160     if existing:
161         updates = 0
162         for field, value in task.dict().items():
163             if value is not None:
164                 _value = value
165                 if field in ["category", "subcategory",
                    ↪ "strategy"]:
166                     _value = value.name
167                 setattr(existing, field, _value)
168                 updates += 1
169         if updates > 0:
```

```
170             db.commit()
171             db.refresh(existing)
172         return existing
173     return None
174 
175 
176 def delete_task_by_id(db: Session, task_id: str):
177     task = get_task_by_id(db, task_id)
178     if task:
179         db.delete(task)
180         db.commit()
181         return True
182     return False
183 
184 
185 def save_task_answers(db: Session, task_id: str, answers:
    ↪ dict):
186     task: Optional[models.Task] = get_task_by_id(db, task_id)
187     if not task:
188         return False
189     existing: Optional[models.Resource] =
      ↪ db.query(models.Resource).filter_by(
190         root_path=task_id
191     ).first()
192     existing_values = copy.deepcopy(existing.values) if
      ↪ existing else {}
193     for file_path, value in answers.items():
194         _path: str = file_path
195         if (
196             not _path.startswith(f"static/{task_id}")
197             and task.category == ATCategory.Image
198             or task.category == ATCategory.Audio
199         ):
200             _path = f"static/{task_id}/{file_path}"
```

```
201             if os.path.exists(os.path.join(STATIC_ROOT, "..",
                ↪  _path)):
202                 if existing_values.get(_path, None) is None:
203                     existing_values[_path] = []
204                 existing_values[_path].append(value)
205         if not existing:
206             existing = models.Resource(root_path=task_id,
                ↪  values=existing_values)
207             db.add(existing)
208         else:
209             existing.values = existing_values
210         db.commit()
211         db.refresh(existing)
212         return True
```

Αρχείο 10: server/active_transfer/ml.py (abstract class)

Αρχείο ορισμού της κλάσης που είναι υπεύθυνη για τις λειτουργίες μηχανικής μάθησης. Περιλαμβάνει την αρχικοποίηση των παραμέτρων, τους ορισμούς των μεθόδων εκπαίδευσης, δοκιμής και επαλήθευσης του μοντέλου μηχανικής μάθησης καθώς και τους ορισμούς των στρατηγικών δειγματοληψίας που χρησιμοποιούνται στο στάδιο της ενεργητικής μάθησης.

```
1  # -*- coding: utf-8 -*-
2
3  import os
4  from abc import abstractmethod, ABCMeta
5  from typing import Tuple, List, Dict, Union, Optional, Any
6  import torch
7  import torchvision
8  from .enums import Phase, QueryStrategy
9  from .storage import AtStorage
10
11
```

```python
class AtMl(metaclass=ABCMeta):
    """ML part(abstract class)."""

    root_dir: Union[str, os.PathLike]
    class_names: List[str]
    _device: torch.device
    _transforms: Dict[Phase,
     ↪  Optional[torchvision.transforms.Compose]] = {
        Phase.Train: None,
        Phase.Val: None,
        Phase.Sample: None,
        Phase.Test: None,
    }
    _random_seed: int = 0
    _batch_size = 32
    _num_workers = 4
    _num_epochs: int = 20
    _notifier: callable = print
    _storage: AtStorage

    def __init__(
        self, root_dir: Union[str, os.PathLike], class_names:
         ↪  List[str], **kwargs
    ):
        """Class Initialization."""
        self.root_dir = root_dir
        self.class_names = class_names
        device_string = kwargs.get("device", None)  # cuda:0,
         ↪  cpu
        if device_string is not None:
            self.device = torch.device(device_string)
        else:
            self._device = torch.device(
                "cuda:0" if torch.cuda.is_available() else
                 ↪  "cpu"
```

```
            )
        self._batch_size = int(kwargs.get("batch_size",
         ↪  self._batch_size))
        self._num_workers = int(kwargs.get("num_workers",
         ↪  self._num_workers))
        self._num_epochs = int(kwargs.get("num_epochs",
         ↪  self._num_epochs))
        self._notifier = kwargs.get("notifier",
         ↪  self._notifier)
        self._random_seed: int =
         ↪  int(kwargs.get("random_seed", self._random_seed))
        transforms: Dict[Phase,
         ↪  Optional[torchvision.transforms.Compose]] =
         ↪  kwargs.get(
            "transforms", {}
        )
        for key, value in self._transforms.items():
            self._transforms[key] = transforms.get(key,
             ↪  value)

    @property
    def storage(self) -> AtStorage:
        """Storage module."""
        return self._storage

    @storage.setter
    def storage(self, value: AtStorage):
        self._storage = value

    @property
    def device(self) -> torch.device:
        """Run in cpu or gpu."""
        return self._device

```

```
69      @device.setter
70      def device(self, value: torch.device):
71          """Run in cpu or gpu."""
72          self._device = value
73
74      @abstractmethod
75      def can_train(self, required: any) -> bool:
76          """Check if we can train our model."""
77          raise NotImplementedError
78
79      @abstractmethod
80      def train(
81          self,
82          model: torch.nn.Module,
83          criterion,
84          optimizer,
85          scheduler,
86          notifier: callable = print,
87          **kwargs
88      ) -> Tuple[
89          torch.nn.Module,
90          Union[
91              Dict[Phase, dict],
92              Dict[Any, Dict[str, Union[Union[float, int, int,
                ↪  float, bool, bool], Any]]],
93          ],
94          float,
95      ]:
96          """Model training.
97
98          :param model The torch model
99          :param criterion: the loss criterion
100         :param optimizer: the model optimizer
101         :param scheduler: learning rate scheduler
```

```
        :param notifier: the notifier function (e.g. print)
        :return: The model, the last epoch and the min loss
        """
        raise NotImplementedError

    @abstractmethod
    def can_test(self, required: any) -> bool:
        """Check if we can test our model.

        :param required: What to compare with
        :return: If we have enough reqs to test the model
        """
        raise NotImplementedError

    @abstractmethod
    def test(
        self, model: torch.nn.Module, notifier: callable =
         ↪  print, **kwargs
    ) -> dict:
        """Test the current state of the model.

        :param model: The model
        :param notifier: Notify about the status
        :return The test results
        """
        raise NotImplementedError

    @abstractmethod
    def have_samples(self, required: any) -> bool:
        """Do we have enough samples."""
        raise NotImplementedError

    @abstractmethod
    def margin_confidence_sampling(
```

```
        self, model: torch.nn.Module, count, **kwargs
    ) -> List[Tuple[str, torch.Tensor]]:
        """Margin Confidence sampling.

        :param model: the torch model
        :param count: How many items
        :param kwargs: Optional extra arguments
        """
        raise NotImplementedError

    @abstractmethod
    def least_confidence_sampling(
        self, model: torch.nn.Module, count: int, **kwargs
    ) -> List[Tuple[str, torch.Tensor]]:
        """Least confidence sampling.

        :param model: the torch model
        :param count: How many items
        :param kwargs: Optional extra arguments
        """
        raise NotImplementedError

    @abstractmethod
    def entropy_sampling(
        self, model: torch.nn.Module, count, **kwargs
    ) -> List[Tuple[str, torch.Tensor]]:
        """Max entropy sampling.

        :param model: the torch model
        :param count: How many items
        :param kwargs: Optional extra arguments
        """
        raise NotImplementedError

```

```
169     @abstractmethod
170     def random_sampling(self, count: int, **kwargs):
171         """Random Sampling.
172 
173         :param count: How many items
174         :param kwargs: Optional extra arguments (e.g. random
    ↪ seed)
175         """
176         raise NotImplementedError
177 
178     @abstractmethod
179     def get_samples(
180         self,
181         count: int,
182         strategy: QueryStrategy,
183         model: torch.nn.Module = None,
184         **kwargs
185     ):
186         """Get samples from the samples pool folder.
187 
188         @:param count: how many samples
189         @:param strategy: how to get these samples
190         """
191         raise NotImplementedError
```

Αρχείο 11: server/active_transfer/handlers/image/binary.py

```
1 # -*- coding: utf-8 -*-s
2 import math
3 import os
4 import time
5 from typing import Tuple, Union, List, Optional, Dict, Any
6 import copy
```

```python
from numpy import random
import torch
import torchvision
import torch.nn.functional as f
from torch.utils.data.dataset import Dataset
from torch.utils.data.dataloader import DataLoader
from torchvision.datasets.folder import default_loader

from ...enums import Phase, QueryStrategy
from ...ml import AtMl


class ImageBinarySamplesDataset(Dataset):
    """Samples dataset (no/unknown classes)."""

    def __init__(self, root_dir, transform=None):
        """Class Initialization."""
        self.root_dir = root_dir
        self.transform = transform
        self.items = [os.path.join(root_dir, x) for x in
         ↪  sorted(os.listdir(root_dir))]
        self.loader = default_loader

    def __len__(self):
        """Items length."""
        return len(self.items)

    def __getitem__(self, index):
        """Get item form index."""
        img_path = self.items[index]
        sample = self.loader(img_path)
        if self.transform is not None:
            sample = self.transform(sample)
        return sample, img_path
```

```
40
41
42  class ImageBinary(AtMl):
43      """Binary Classification Image handler."""
44
45      positive: str
46      negative: str
47
48      def __init__(
49          self,
50          root_dir: Union[str, os.PathLike],
51          class_names: Optional[List[str]],
52          **kwargs
53      ):
54          """Class Initialization."""
55          if class_names is None:
56              if len(os.listdir(root_dir)) != 2:
57                  raise ValueError("Cannot get the two binary
                        ↪ classes")
58          elif len(class_names) != 2:
59              raise ValueError("Only two binary classes are
                    ↪ allowed")
60          _class_names = (
61              class_names if class_names is not None else
                    ↪ sorted(os.listdir(root_dir))
62          )
63          self.class_names = _class_names
64          self.positive = _class_names[0]
65          self.negative = _class_names[1]
66          super(ImageBinary, self).__init__(root_dir,
                ↪ _class_names, **kwargs)
67
68      def _train_epoch(self, model: torch.nn.Module, min_loss,
            ↪ best_model_wts, **kwargs):
```

```
69          phases = kwargs.get("phases")
70          dataloaders = kwargs.get("dataloaders")
71          dataset_sizes = kwargs.get("dataset_sizes")
72          optimizer = kwargs.get("optimizer")
73          scheduler = kwargs.get("scheduler")
74          criterion = kwargs.get("criterion")
75          epoch_stats = {}
76
77          # Each epoch has a training and validation phase
78          for phase in phases:
79              is_train = phase == Phase.Train
80              if is_train:
81                  model.train()  # Set model to training mode
82              else:
83                  model.eval()  # Set model to evaluate mode
84
85              running_loss: float = 0.0
86              running_corrects: torch.Tensor = torch.tensor(0)
87
88              # Iterate over data.
89              for _, (inputs, labels) in
                 ↪  enumerate(dataloaders[phase]):
90                  inputs = inputs.to(self._device)
91                  labels = labels.to(self._device)
92
93                  # zero the parameter gradients
94                  optimizer.zero_grad()
95                  with torch.set_grad_enabled(is_train):
96                      outputs = model(inputs)
97                      _, predictions = torch.max(outputs, 1)
98                      loss = criterion(outputs, labels)
99                      # print(loss)
100
101                     # backward + optimize only if in train
```

```
102                     if is_train:
103                         loss.backward()
104                         optimizer.step()
105                 # statistics
106                 running_loss += loss.item() * inputs.size(0)
107                 running_corrects += torch.sum(predictions ==
                     ↪ labels.data)
108             epoch_loss = running_loss / dataset_sizes[phase]
109             epoch_acc = running_corrects.double() /
                 ↪ dataset_sizes[phase]
110             _acc = epoch_acc.item()
111             self._notifier(
112                 "{} loss: {:.4f} acc: {:.4f}".format(phase,
                     ↪ epoch_loss, _acc),
113                 phase=phase.name,
114                 epoch_loss=epoch_loss,
115                 epoch_acc=_acc,
116             )
117 
118             if is_train:
119                 scheduler.step()
120             elif epoch_loss < min_loss:
121                 min_loss = float(epoch_loss)
122                 # deep copy the model
123                 best_model_wts =
                     ↪ copy.deepcopy(model.state_dict())
124             epoch_stats[phase] = {"loss": epoch_loss, "acc":
                 ↪ _acc}
125         return model, optimizer, scheduler, best_model_wts,
             ↪ min_loss, epoch_stats
126 
127     @staticmethod
128     def _update_test_stats(data, target, predictions,
         ↪ **kwargs):
```

```
        true_pos = kwargs.get("true_pos")
        true_neg = kwargs.get("true_neg")
        false_pos = kwargs.get("false_pos")
        false_neg = kwargs.get("false_neg")
        for j in range(data.size(0)):
            expectation = target[j].item()
            prediction = predictions[j].item()
            positive = prediction == 1
            true = prediction == expectation
            if true and positive:
                true_pos += 1
            if true and not positive:
                true_neg += 1
            if not true and positive:
                false_pos += 1
            if not true and not positive:
                false_neg += 1
        return true_pos, true_neg, false_pos, false_neg

    def can_test(self, required: int) -> bool:
        """Check if we can test our model.

        :return: If we have enough reqs to test the model
        """
        test_files = self.storage.test_dir()
        return (
            sum(
                len(os.listdir(os.path.join(self.root_dir,
                 ↪  Phase.Train.value, x)))
                for x in test_files
            )
            > required
        )

```

```
162     def can_train(self, required: int) -> bool:
163         """Check if we can train our model.
164 
165         :return: If we have enough reqs to start training
166         """
167         train_files = self.storage.train_dir()
168         val_files = self.storage.val_dir()
169         test_files = self.storage.test_dir()
170         train_size = sum(
171             len(os.listdir(os.path.join(self.root_dir,
                ↪  Phase.Train.value, x)))
172             for x in train_files
173         )
174         val_size = sum(
175             len(os.listdir(os.path.join(self.root_dir,
                ↪  Phase.Val.value, x)))
176             for x in val_files
177         )
178         test_size = sum(
179             len(os.listdir(os.path.join(self.root_dir,
                ↪  Phase.Test.value, x)))
180             for x in test_files
181         )
182         return train_size + val_size + test_size >= required
183 
184     def have_samples(self, required: int) -> bool:
185         """Do we have enough samples."""
186         return len(self.storage.samples_dir()) > required
187 
188     def train(
189         self,
190         model: torch.nn.Module,
191         criterion,
192         optimizer,
```

```
193         scheduler,
194         notifier: callable = print,
195         **kwargs
196     ) -> Tuple[
197         torch.nn.Module,
198         Union[
199             Dict[Phase, dict],
200             Dict[Any, Dict[str, Union[Union[float, int, int,
                ↪  float, bool, bool], Any]]],
201         ],
202         float,
203     ]:
204         """Model training."""
205         num_epochs = kwargs.get("num_epochs",
             ↪  self._num_epochs)
206         batch_size: int = kwargs.get("batch_size",
             ↪  self._batch_size)
207         num_workers: int = kwargs.get("num_workers",
             ↪  self._num_workers)
208         random_seed: int = kwargs.get("random_seed",
             ↪  self._random_seed)
209         torch.manual_seed(random_seed)
210         phases: List[Phase] = [Phase.Train, Phase.Val]
211         data_transforms: Dict[Phase,
             ↪  torchvision.transforms.Compose] = {
212             Phase.Train: self._transforms[Phase.Train],
213             Phase.Val: self._transforms[Phase.Val],
214         }
215         image_dataset = {
216             x: torchvision.datasets.ImageFolder(
217                 os.path.join(self.root_dir, x),
                     ↪  data_transforms[x]
218             )
219             for x in phases
```

```
220         }
221         dataloaders = {
222             x: DataLoader(
223                 image_dataset[x],
224                 shuffle=x is Phase.Train,
225                 batch_size=batch_size,
226                 num_workers=num_workers,
227             )
228             for x in phases
229         }
230         dataset_sizes = {x: len(image_dataset[x]) for x in
    ↪  phases}
231         best_model_wts = copy.deepcopy(model.state_dict())
232         min_loss: float = 1.0
233         last_epoch = {Phase.Train: {}, Phase.Val: {}}
234         since = time.time()
235         for epoch in range(self._num_epochs):
236             self._notifier(
237                 "Epoch {}/{}".format(epoch + 1,
    ↪  self._num_epochs),
238                 state={"epoch": epoch + 1, "total":
    ↪  self._num_epochs},
239             )
240             self._notifier("-" * 10)
241             (
242                 model,
243                 optimizer,
244                 scheduler,
245                 best_model_wts,
246                 min_loss,
247                 epoch_stats,
248             ) = self._train_epoch(
249                 model,
250                 min_loss,
```

```
                best_model_wts,
                phases=phases,
                dataloaders=dataloaders,
                dataset_sizes=dataset_sizes,
                scheduler=scheduler,
                optimizer=optimizer,
                criterion=criterion,
            )
            if epoch == num_epochs - 1:
                last_epoch = epoch_stats
        time_elapsed = time.time() - since
        self._notifier(
            "Training complete in {:.0f}m {:.0f}s".format(
                time_elapsed // 60, time_elapsed % 60
            ),
            "Min val Loss: {:4f}".format(min_loss),
            duration=time_elapsed,
            min_loss=min_loss,
        )

        # load best model weights
        model.load_state_dict(best_model_wts)
        return model, last_epoch, min_loss

    def test(self, model: torch.nn.Module, **kwargs) -> dict:
        """Model testing."""
        batch_size: int = kwargs.get("batch_size",
         ↪  self._batch_size)
        num_workers: int = kwargs.get("num_workers",
         ↪  self._num_workers)
        test_folder: str = Phase.Test.value
        model.eval()
        true_pos: int = 0
        true_neg: int = 0
```

```
283             false_pos: int = 0
284             false_neg: int = 0
285             test_loss: float = 0.0
286             correct: float = 0
287             test_transform = self._transforms[Phase.Test]
288             test_dataset = torchvision.datasets.ImageFolder(
289                 os.path.join(self.root_dir, test_folder),
                     ↪ test_transform
290             )
291             test_loader: DataLoader = DataLoader(
292                 test_dataset, batch_size=batch_size,
                     ↪ shuffle=True, num_workers=num_workers
293             )
294             total_length: int = len(test_loader.dataset)
295             with torch.no_grad():
296                 for i, (data, target) in enumerate(test_loader):
297                     data, target = data.to(self.device),
                         ↪ target.to(self.device)
298                     outputs = model(data)
299                     _, predictions = torch.max(outputs, 1)
300                     test_loss += f.cross_entropy(outputs,
                         ↪ target).item() * data.size(0)
301                     correct +=
                         ↪ predictions.eq(target.view_as(predictions)).sum()...
302                     (true_pos, true_neg, false_pos, false_neg) =
                         ↪ self._update_test_stats(
303                         data,
304                         target,
305                         predictions,
306                         true_pos=true_pos,
307                         true_neg=true_neg,
308                         false_pos=false_pos,
309                         false_neg=false_neg,
310                     )
```

```
311
312         test_loss /= len(test_loader.dataset)
313         return {
314             "test_loss": test_loss,
315             "test_acc": correct / total_length,
316             "test_total": total_length,
317             "test_correct": correct,
318             "test_true_pos": true_pos,
319             "test_true_neg": true_neg,
320             "test_false_pos": false_pos,
321             "test_false_neg": false_neg,
322         }
323
324     def __sample(self, model, output_handler, **kwargs):
325         batch_size: int = kwargs.get("batch_size",
              ↪ self._batch_size)
326         num_workers: int = kwargs.get("num_workers",
              ↪ self._num_workers)
327         samples_dir = self.storage.samples_dir(True)
328         model.eval()
329         samples_dataset = ImageBinarySamplesDataset(
330             samples_dir, self._transforms[Phase.Sample]
331         )
332         samples_loader: DataLoader = DataLoader(
333             samples_dataset,
334             batch_size=batch_size,
335             shuffle=True,
336             num_workers=num_workers,
337         )
338         device = self._device
339         with torch.no_grad():
340             for i, (image_data, image_paths) in
                  ↪ enumerate(samples_loader):
341                 data_size = image_data.size(0)
```

```
342                 image_data = image_data.to(device)
343                 outputs = model(image_data)
344                 output_handler(outputs, image_paths,
                     ↪  data_size)
345
346     def margin_confidence_sampling(
347         self, model: torch.nn.Module, count, **kwargs
348     ) -> List[Tuple[str, torch.Tensor]]:
349         """Margin Confidence sampling."""
350         img_margins = []
351
352         def handler(outputs, img_batch, data_size):
353             """Handle samples model output."""
354             for j in range(data_size):
355                 probabilities = torch.softmax(outputs[j], 0)
356                 margin = max(probabilities) -
                     ↪  min(probabilities)
357                 img_margins.append((img_batch[j], margin))
358
359         self.__sample(model, handler)
360         img_margins.sort(key=lambda x: x[1])
361         return img_margins[:count]
362
363     def least_confidence_sampling(
364         self, model: torch.nn.Module, count: int, **kwargs
365     ) -> List[Tuple[str, torch.Tensor]]:
366         """Least confidence sampling."""
367         confidences = []
368
369         def handler(outputs, img_batch, data_size):
370             """Handle samples model output.
371
372             :param outputs:
373             :param img_batch:
```

```
374             :param data_size:
375             """
376             for j in range(data_size):
377                 _img = img_batch[j]
378                 probabilities = torch.softmax(outputs[j], 0)
379                 confidence = torch.max(probabilities)
380                 confidences.append((_img, confidence))
381 
382         self.__sample(model, handler, **kwargs)
383         confidences.sort(key=lambda x: x[1])
384         return confidences[:count]
385 
386     def entropy_sampling(
387         self, model: torch.nn.Module, count, **kwargs
388     ) -> List[Tuple[str, torch.Tensor]]:
389         """Max entropy sampling."""
390         entropies = []
391 
392         def handler(outputs, img_batch, data_size):
393             """Handle samples model output.
394 
395             :param outputs:
396             :param img_batch:
397             :param data_size:
398             """
399             for j in range(data_size):
400                 _img = img_batch[j]
401                 probabilities = torch.softmax(outputs[j], 0)
402                 entropy = 0
403                 for prob in probabilities:
404                     entropy += -prob * math.log(prob, 2)
405                 entropies.append((_img, entropy))
406 
407         self.__sample(model, handler, **kwargs)
```

```
408             entropies.sort(key=lambda x: x[1], reverse=True)
409             return entropies[:count]
410
411         def random_sampling(self, count: int, **kwargs):
412             """Handle samples model output.
413
414             :param count: How many files
415             :param kwargs: Optional random seed
416             """
417             random_seed = kwargs.get("random_seed",
                ↪  self._random_seed)
418             random.seed(random_seed)
419             samples_dir = self.storage.samples_dir(True)
420             sample_files = self.storage.samples_dir()
421             random.shuffle(sample_files)
422             return [(os.path.join(samples_dir, x), None) for x in
                ↪  sample_files[:count]]
423
424         def get_samples(
425             self,
426             count: int,
427             strategy: QueryStrategy,
428             model: torch.nn.Module = None,
429             **kwargs
430         ):
431             """Get samples from the samples pool folder.
432
433             @:param count: how many samples
434             @:param strategy: how to get these samples
435             """
436             if strategy == QueryStrategy.Random:
437                 return self.random_sampling(count)
438             if model is None:
439                 raise ValueError("Model not provided")
```

```
440             samples = []
441             if strategy == QueryStrategy.Margin:
442                 samples = self.margin_confidence_sampling(model,
                    ↪  count, **kwargs)
443             if strategy == QueryStrategy.LeastConfidence:
444                 samples = self.least_confidence_sampling(model,
                    ↪  count, **kwargs)
445             if strategy == QueryStrategy.Entropy:
446                 samples = self.entropy_sampling(model, count,
                    ↪  **kwargs)
447             return samples
```

Αρχείο 12: server/tasks/image/binary.py

Αρχείο υλοποίησης των μεθόδων που ορίζονται στο προηγούμενο αρχείο για τν περίπτωση προβλημάτων δυαδικής κατηγοριοποίησης.

```
1   import os
2   import shutil
3   import uuid
4   import random
5   import torchvision
6   import torch.nn as nn
7   import torch.optim as optim
8   from typing import List, Dict, Optional, Tuple, Union
9
10  from sqlalchemy.orm import Session
11  from torch.optim import lr_scheduler
12  from server import SHARED_DIR, STATIC_ROOT
13  from server.app.models import Task, Resource
14  from active_transfer import (
15      Task as ATTask,
16      QueryStrategy,
17      ActiveTransfer,
```

```
    PhaseSize,
    weighted_dir_items,
    Phase,
)

MODEL_NAME = "model.pt"
DATA_DIR_NAME = ".data"


def get_dataset_transform(is_train: bool = False):
    """Get dataset transformation.

    :param is_train: Whether is a train dataset or not
    :return:The transform function
    """
    if is_train:
        return torchvision.transforms.Compose(
            [

                ↪  torchvision.transforms.RandomResizedCrop(224),

                ↪  torchvision.transforms.RandomHorizontalFlip(),
                torchvision.transforms.ToTensor(),
                torchvision.transforms.Normalize(
                    [0.485, 0.456, 0.406], [0.229, 0.224,
                     ↪  0.225]
                ),
            ]
        )
    return torchvision.transforms.Compose(
        [
            torchvision.transforms.Resize(256),
            torchvision.transforms.CenterCrop(224),
            torchvision.transforms.ToTensor(),
```

```
            torchvision.transforms.Normalize(
                [0.485, 0.456, 0.406], [0.229, 0.224, 0.225]
            ),
        ]
    )


def _labeled_sizes(task_root, class_names):
    labeled_files = [os.listdir(os.path.join(task_root, x)) for x in class_names]
    return [len(x) for x in labeled_files]


def _move_files(samples: dict, dst_root: str, src_root: str):
    for class_name, item_names in samples.items():
        src_dir = os.path.join(src_root, class_name)
        dst_dir = os.path.join(dst_root, class_name)
        if (
            os.path.exists(src_dir)
            and os.path.exists(dst_dir)
            and os.path.isdir(src_dir)
            and os.path.isdir(dst_dir)
        ):
            for _file in item_names:
                src = os.path.join(src_dir, _file)
                _extension = _file.split(".")[-1]
                dst = os.path.join(
                    dst_dir,
                    f"{len(os.listdir(dst_dir))+1:06d}.{_extension}"
                )
                if os.path.exists(src):
                    shutil.move(src, dst)
```

```
81  def _check_dirs(task: Task, at: ActiveTransfer, class_names:
     ↪  List[str]):
82      should_train = False
83      task_root = os.path.realpath(os.path.join(SHARED_DIR,
         ↪  task.id))
84      val_path = at.val_dir(True)
85      test_path = at.test_dir(True)
86      train_path = at.train_dir(True)
87      val_current = [os.listdir(os.path.join(val_path, x)) for
         ↪  x in class_names]
88      test_current = [os.listdir(os.path.join(test_path, x))
         ↪  for x in class_names]
89      train_current = [os.listdir(os.path.join(train_path, x))
         ↪  for x in class_names]
90      val_sizes = [len(x) for x in val_current]
91      test_sizes = [len(x) for x in test_current]
92      train_sizes = [len(x) for x in train_current]
93      labeled_sizes = _labeled_sizes(task_root, class_names)
94      labeled_min = min(labeled_sizes)
95      val_total = sum(val_sizes)
96      train_total = sum(train_sizes)
97      test_total = sum(test_sizes)
98      if val_total < task.val_count and labeled_min >
         ↪  val_total:
99          samples = weighted_dir_items(
100             task_root, count=task.val_count - val_total,
                 ↪  names=class_names
101         )
102         _move_files(samples, val_path, task_root)
103         labeled_sizes = _labeled_sizes(task_root,
             ↪  class_names)
104         labeled_min = min(labeled_sizes)
105     if test_total < task.test_count and labeled_min >
         ↪  test_total:
```

```
106             samples = weighted_dir_items(
107                 task_root, count=task.test_count - test_total,
                        names=class_names
108             )
109             _move_files(samples, test_path, task_root)
110             labeled_sizes = _labeled_sizes(task_root,
                    class_names)
111             labeled_min = min(labeled_sizes)
112         if train_total < task.append_count and labeled_min >
                train_total:
113             # initial train
114             samples = weighted_dir_items(
115                 task_root, count=task.append_count - train_total,
                        names=class_names
116             )
117             _move_files(samples, train_path, task_root)
118             should_train = True
119         return should_train
120
121
122 def _prepare(at: ActiveTransfer, task: Task) ->
        Tuple[ActiveTransfer, any, any, any]:
123     # TODO: transfer: include fine tuning (not only fixed
            extractor)
124     for param in at.model.parameters():
125         param.requires_grad = False
126     num_features = at.model.fc.in_features
127     at.model.fc = nn.Linear(num_features, 2)
128     at.model = at.model.to(at.device)
129     criterion = nn.CrossEntropyLoss()
130     lr = task.optimizer_lr
131     momentum = task.optimizer_momentum
132     params = at.model.fc.parameters()
133     optimizer = optim.SGD(params, lr=lr, momentum=momentum)
```

```
    step_size = task.scheduler_step
    scheduler = lr_scheduler.StepLR(optimizer,
     ↪  step_size=step_size)
    return at, criterion, optimizer, scheduler


def _move_samples_to(at: ActiveTransfer, labels: Dict[str,
 ↪  bool], root_dir: str):
    class_names: List[str] = at.class_names
    class_files = [len(os.listdir(os.path.join(root_dir, x)))
     ↪  + 1 for x in class_names]
    not_moved = {}
    limit_files = root_dir != at.train_dir(True)
    upper_limit = -1
    if limit_files:
        upper_limit = (
            at.scenario.test_size
            if root_dir == at.test_dir(True)
            else at.scenario.val_size
        )
    for image, class_value in labels.items():
        image_name = image.split("/")[-1]
        image_extension = image_name.split(".")[-1]
        if class_value is bool:
            class_value = class_names[0] if class_value else
             ↪  class_names[1]
        class_name = class_value
        _array_index = 0 if class_value == class_names[0]
         ↪  else 1
        destination_name = class_files[_array_index]
        class_files[_array_index] += 1
        destination_dir = os.path.join(root_dir, class_name)
        if not limit_files or (
            limit_files and len(os.listdir(destination_dir))
             ↪  < upper_limit
```

```
        ):
            src = os.path.join(SHARED_DIR, image)
            destination_name =
             ↪ f"{destination_name:06d}.{image_extension}"
            dst = os.path.join(destination_dir,
             ↪ destination_name)
            shutil.move(src, dst)
        else:
            not_moved[image] = class_value
    return not_moved


def _samples_destination(task: Task, at: ActiveTransfer):
    val_dir = at.val_dir(True)
    val_files = sum(len(os.listdir(os.path.join(val_dir, x)))
     ↪ for x in at.class_names)
    if val_files < task.val_count:
        return val_dir
    else:
        test_dir = at.test_dir(True)
        test_files = sum(
            len(os.listdir(os.path.join(test_dir, x))) for x
             ↪ in at.class_names
        )
        if test_files < task.test_count:
            return test_dir
    return at.train_dir(True)


def _new_samples(
    task: Task,
    at: ActiveTransfer,
    db: Session,
    commits: Resource,
```

```
193     mappings: Dict[str, Union[bool, str]],
194 ):
195     # TODO: also check if we should finish the task (compare
        ↪ accuracies in task.results?)
196     root_dst_dir = _samples_destination(task, at)
197     # val is full?
198     not_moved = _move_samples_to(at, mappings, root_dst_dir)
199     if len(not_moved) > 0:
200         root_dst_dir = _samples_destination(task, at)
201         # test is full ?
202         not_moved = _move_samples_to(at, mappings,
            ↪ root_dst_dir)
203     if len(not_moved) > 0:
204         root_dst_dir = _samples_destination(task, at)
205         # do not check for train
206         _move_samples_to(at, mappings, root_dst_dir)
207     db.delete(commits)
208     db.commit()
209     task.resources_id = str(uuid.uuid4())
210     db.commit()
211     db.refresh(task)
212
213
214 def _check_task_resources(task: Task, db: Session, at:
    ↪ ActiveTransfer):
215     commits: Resource =
        ↪ db.query(Resource).filter_by(root_path=str(task.id)).first()
216     if not commits:
217         return False
218     batch = task.append_count
219     values = commits.values
220     if len(values) < batch:
221         return False
222     current_commits = 0
```

```
    min_commits = task.min_peers_count
    mappings = {}
    accuracy = task.peer_accuracy
    for item_path, answers in values.items():
        item_answers = {x: 0 for x in task.class_names}
        for answer in answers:
            if answer is bool:
                answer = task.class_names[0] if answer else
                 ↪  task.class_names[1]
            item_answers[answer] += 1
        if len(answers) >= min_commits:
            winner = max(item_answers.values())
            total = sum(item_answers.values())
            item_accuracy = winner / total
            if item_accuracy >= accuracy:
                mappings[item_path] = [
                    x for x, y in item_answers.items() if y
                     ↪  == winner
                ][0]
                current_commits += 1
    samples_ready = current_commits >= min_commits
    if samples_ready:
        _new_samples(task, at, db, commits, mappings)
    return samples_ready


def enough_public_dir_items(task: Task) -> bool:
    task_public_dir = os.path.join(STATIC_ROOT, str(task.id))
    if not os.path.exists(task_public_dir):
        os.makedirs(task_public_dir, exist_ok=True)
        os.chmod(task_public_dir, 0o777)
        return False
    return len(os.listdir(task_public_dir)) >=
     ↪  task.append_count
```

```
254
255
256 def enough_sample_items(at: ActiveTransfer, task: Task) ->
     ↪ bool:
257     _samples_dir = at.samples_dir(True)
258     if not os.path.exists(_samples_dir):
259         os.makedirs(_samples_dir, exist_ok=True)
260         os.chmod(_samples_dir, 0o777)
261         return False
262     return len(os.listdir(_samples_dir)) >= task.append_count
263
264
265 def get_at_instance(
266     task: Task, class_names: List[str], notifier: callable
267 ) -> ActiveTransfer:
268     task_root = os.path.realpath(os.path.join(SHARED_DIR,
         ↪ str(task.id)))
269     task_data_root = os.path.realpath(os.path.join(task_root,
         ↪ DATA_DIR_NAME))
270     if not os.path.exists(task_data_root):
271         os.makedirs(task_data_root)
272         os.chmod(task_data_root, 0o777)
273     model = torchvision.models.resnet50(pretrained=True)
274     model_path = os.path.join(task_root, MODEL_NAME)
275     sizes = PhaseSize(
276         {"train": task.append_count, "val": task.val_count,
             ↪ "test": task.test_count}
277     )
278     non_train_transform = get_dataset_transform()
279     train_transform = get_dataset_transform(is_train=True)
280     transforms: Dict[Phase, torchvision.transforms.Compose] =
         ↪ {
281         Phase.Train: train_transform,
282         Phase.Val: non_train_transform,
```

```
283             Phase.Sample: non_train_transform,
284             Phase.Test: non_train_transform,
285         }
286         at_task = ATTask(
287             phase_sizes=sizes,
288             append_size=task.append_count,
289             category=task.category,
290             subcategory=task.subcategory,
291             peers=task.min_peers_count,
292 
                ↪  strategy=QueryStrategy.from_string(task.strategy.name.lowe
293             peer_accuracy=task.peer_accuracy,
294         )
295         return ActiveTransfer(
296             task_data_root,
297             model=model,
298             model_path=model_path,
299             transforms=transforms,
300             class_names=class_names,
301             scenario=at_task,
302             num_epochs=task.epochs,
303             batch_size=16,
304             num_workers=1,
305             notifier=notifier,
306         )
307 
308 
309 def _add_samples(task: Task, at: ActiveTransfer, db:
    ↪  Session):
310     samples_dir = at.samples_dir(True)
311     dst_root = os.path.join(STATIC_ROOT, str(task.id))
312     os.makedirs(dst_root, exist_ok=True)
313     os.chmod(dst_root, 0o777)
314     current_files = os.listdir(dst_root)
```

```
    if len(current_files) < task.append_count:
        all_files = os.listdir(samples_dir)
        random.shuffle(all_files)
        for file_name in all_files[: task.append_count]:
            src = os.path.join(samples_dir, file_name)
            dst = os.path.join(dst_root, file_name)
            shutil.move(src, dst)
    else:
        _check_task_resources(task, db, at)
    return at, None, None, None, False, False


def config(
    task: Task, notifier: callable, db: Session
) -> Tuple[
    Optional[ActiveTransfer], Optional[any], Optional[any],
     ↪  Optional[any], bool, bool
]:
    """
    Checks:
        -  initial folders: there are not enough images in
 ↪  the test/val/folders
            - check the unlabeled pool folder and if we have
 ↪  enough samples
                move them to the public dir and ask for
 ↪  labeling
        -  initial train: there are enough images in the
 ↪  train/val/test folders
            and we do not have a trained model yet
            - we should start training (first time)
        -  retrain: there exists a trained model and we have
 ↪  enough new labeled samples
            - we add new samples to the train folder and
 ↪  should retrain the model
```

```
342     :param task: The stored task
343     :param notifier: The function to call on updates
344     :param db: The db session
345     :return: an ActiveTransfer instance, the loss criterion,
        ↪ the optimizer,
346     the rl_scheduler, if we are ready to train and if we are
        ↪ ready to get new samples
347     """
348     should_train = False
349     class_names = task.class_names
350     if len(class_names) == 2:
351         task_root = os.path.realpath(os.path.join(SHARED_DIR,
            ↪ str(task.id)))
352         for x in class_names:
353             os.makedirs(os.path.join(task_root, x),
                ↪ exist_ok=True)
354         at = get_at_instance(task, class_names, notifier)
355         if not at.can_train or not at.can_test:
356             should_train = _check_dirs(task, at, class_names)
357             if not should_train:
358                 should_train = at.can_train and not
                    ↪ os.path.exists(
359                     os.path.join(task_root, MODEL_NAME)
360                 )
361             if not should_train and not at.can_train and
                ↪ at.have_samples:
362                 return _add_samples(task, at, db)
363             if should_train:
364                 at, criterion, optimizer, scheduler =
                    ↪ _prepare(at, task)
365                 return at, criterion, optimizer, scheduler,
                    ↪ True, False
366         if not should_train and at.can_train:
367             should_train = _check_task_resources(task, db,
                ↪ at)
```

```
            if should_train and at.can_train:
                at, criterion, optimizer, scheduler =
                ↪ _prepare(at, task)
                return at, criterion, optimizer, scheduler,
                ↪ True, False
        if not should_train:
            _enough_sample_items = enough_sample_items(at,
            ↪ task)
            _enough_public_dir_items =
            ↪ enough_public_dir_items(task)
            should_sample = _enough_sample_items and not
            ↪ _enough_public_dir_items
            if should_sample:
                at, criterion, optimizer, scheduler =
                ↪ _prepare(at, task)
                return at, criterion, optimizer, scheduler,
                ↪ False, True
    return None, None, None, None, False, False
```

Αρχείο 13: mobile/package.json

Αρχείο με τις ρυθμίσεις και τις προαπαιτούμενες βιβλιοθήκες για την εφαρμογή για κινητές συσκευές.

```
{
  "name": "activecrowds",
  "version": "0.0.1",
  "private": true,
  "scripts": {
    "android": "react-native run-android",
    "ios": "react-native run-ios",
    "start": "react-native start",
    "test": "jest"
  },
```

```
  "dependencies": {
    "@react-native-community/async-storage": "^1.7.1",
    "@react-native-community/masked-view": "^0.1.6",
    "@react-native-community/netinfo": "^5.5.0",
    "axios": "^0.19.2",
    "react": "16.12.0",
    "react-native": "0.61.5",
    "react-native-device-info": "^5.5.3",
    "react-native-elements": "^1.2.7",
    "react-native-fast-image": "^7.0.2",
    "react-native-gesture-handler": "^1.6.0",
    "react-native-image-progress": "^1.1.1",
    "react-native-reanimated": "^1.7.0",
    "react-native-screens": "^2.0.0-beta.4",
    "react-native-vector-icons": "^6.6.0",
    "react-navigation": "^4.1.1",
    "react-navigation-stack": "^2.1.1",
    "rn-fetch-blob": "^0.12.0",
    "uuid": "^3.4.0"
  },
  "devDependencies": {
    "@babel/core": "^7.8.4",
    "@babel/runtime": "^7.8.4",
    "@react-native-community/eslint-config": "^0.0.7",
    "@types/jest": "^25.1.2",
    "@types/react": "16.9.19",
    "@types/react-native": "^0.61.15",
    "@types/react-test-renderer": "16.9.2",
    "@types/uuid": "^3.4.7",
    "babel-jest": "^25.1.0",
    "jest": "^25.1.0",
    "metro-react-native-babel-preset": "^0.58.0",
    "react-native-safe-area-context": "^0.7.3",
    "react-test-renderer": "16.12.0",
```

```
    "typescript": "^3.7.5"
  },
  "jest": {
    "preset": "react-native",
    "moduleFileExtensions": [
      "ts",
      "tsx",
      "js",
      "jsx",
      "json",
      "node"
    ]
  }
}
```

Αρχείο 14: mobile/src/Models.ts

Αρχείο ορισμού των μοντέλων που χρησιμοποιούνται στην εφαρμογή για κινητές συσκευές.

```
import { CategoryType, TaskType, TaskStatus,
 ↪ SubcategoryType, TaskCategoryType, TaskSubcategoryType
 ↪ } from './types';

export class Task implements TaskType {
    id: string;
    resources_id: string;
    title: string;
    cover: string;
    description: string;
    class_names: string[];
    status: TaskStatus;
    category: CategoryType;
    subcategory: SubcategoryType;
    constructor(id: string, resources_id: string, title:
     ↪ string, status: TaskStatus, cover: string,
     ↪ description: string, category: CategoryType,
     ↪ subcategory: SubcategoryType, class_names:
     ↪ string[]) {
```

```
14          this.id = id;
15          this.resources_id = resources_id;
16          this.title = title;
17          this.cover = cover;
18          this.description = description;
19          this.status = status;
20          this.category = category;
21          this.subcategory = subcategory;
22          this.class_names = class_names;
23      }
24
25      toJSON(): TaskType {
26          return Object.assign({}, this);
27      }
28
29      static fromJSON(json: TaskType) : Task {
30          let task = Object.create(Task.prototype);
31          return Object.assign(task, json);
32      }
33  }
34
35  export class SubCategory implements TaskSubcategoryType {
36      name: SubcategoryType;
37      description: string;
38      cover: string;
39
40      constructor(name: SubcategoryType, description: string,
        ↪ cover: string) {
41          this.name = name;
42          this.description = description;
43          this.cover = cover;
44      }
45
46      toJSON(): TaskSubcategoryType {
```

```
47          return Object.assign({}, this);
48      }
49
50      static fromJSON(json: TaskSubcategoryType) :
        ↪ SubCategory {
51          let subcategory =
            ↪ Object.create(SubCategory.prototype);
52          return Object.assign(subcategory, json);
53      }
54  }
55
56  export class Category implements TaskCategoryType {
57      name: CategoryType;
58      description: string;
59      cover: string;
60      subcategories: TaskSubcategoryType[];
61      constructor(name: CategoryType, description: string,
        ↪ cover: string, subcategories:
        ↪ TaskSubcategoryType[]) {
62          this.name = name;
63          this.description = description;
64          this.cover = cover;
65          this.subcategories = subcategories;
66      }
67
68      toJSON(): TaskCategoryType {
69          return Object.assign({}, this);
70      }
71
72      static fromJSON(json: TaskCategoryType): Category {
73          let category = Object.create(Category.prototype);
74          return Object.assign(category, json, {
75              subcategories: json.subcategories
76                  .map((_subcategory: TaskSubcategoryType) =>
                    ↪ SubCategory.fromJSON(_subcategory)),
```

```
77              });
78          }
79      }
```