ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ
Σχολή Μηχανικών
Τμήμα Μηχανικών Πληροφορικής και Υπολογιστών

Μοντέλα Βαθιάς Μάθησης για την πρόβλεψη χρονοσειρών με χρήση της βιβλιοθήκης Tensorflow

Διπλωματική Εργασία

του

ΣΑΓΚΡΙΩΤΗ ΠΑΝΑΓΙΩΤΗ

Επιβλέπων Καθηγητής

Πάρις Μαστοροκώστας

Καθηγητής

Αιγάλεω

Οκτώβριος 2023

ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ

Σχολή Μηχανικών
Τμήμα Μηχανικών Πληροφορικής και Υπολογιστών

Μοντέλα Βαθιάς Μάθησης για την πρόβλεψη χρονοσειρών με χρήση της βιβλιοθήκης Tensorflow

Ονοματεπώνυμο *Παναγιώτης Σαγκριώτης*

Αριθμός Μητρώου 71345100

Επιβλέπων Καθηγητής Πάρις Μαστοροκώστας
Καθηγητής

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 11η Οκτωβρίου 2023		
Ioannis Amorginos Digitally signed by Ioannis Amorginos Date: 2023.10.11 18:09:35 +03'00' Ιωάννης Αμοργίνος *Λέκτορας Εφαρμογών*	Paris Mastorokostas Digitally signed by Paris Mastorokostas Date: 2023.10.11 09:44:10 +03'00' Πάρις Μαστοροκώστας *Καθηγητής*	Panagiota Tselenti Digitally signed by Panagiota Tselenti Date: 2023.10.11 10:55:38 +03'00' Παναγιώτα Τσελέντη *ΕΔΙΠ*

Αιγάλεω

Οκτώβριος 2023

UNIVERSITY OF WEST ATTICA

School of Engineering
Department of Informatics and Computer Engineering

Deep Learning models for time-series forecasting using Tensorflow library.

Full Name Panagiotis Sagriotis

Identification Number 71345100

Supervisor Paris Mastorocostas
Professor

Egaleo, October 2023

ΔΗΛΩΣΗ ΣΥΓΓΡΑΦΕΑ ΔΙΠΛΩΜΑΤΙΚΗΣ ΕΡΓΑΣΙΑΣ

Ο κάτωθι υπογεγραμμένος **Σαγκριώτης Παναγιώτης** του Ζαννή, με αριθμό μητρώου **71345100** φοιτητής του Πανεπιστημίου Δυτικής Αττικής της Σχολής Μηχανικών του **Τμήματος Μηχανικών Πληροφορικής και Υπολογιστών**, δηλώνω υπεύθυνα ότι:

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

Ο Δηλών,

Παναγιώτης Σαγκριώτης

Περίληψη

Σκοπός της εργασίας είναι αρχικά η εξοικείωση με τη βιβλιοθήκη Tensorflow και τους αλγορίθμους Βαθιάς Μάθησης που περιλαμβάνει. Ειδικότερα έγινε μελέτη των μοντέλων RNN και LSTM, και η εφαρμογή τους στο πρόβλημα της βραχυπρόθεσμης πρόβλεψης ηλεκτρικού φορτίου για το ελληνικό διασυνδεδεμένο σύστημα.

Τα αποτελέσματα είναι αρκετά ενθαρρυντικά, καθώς επιτεύχθηκαν αξιόλογες προβλέψεις.

Λέξεις Κλειδιά

Τεχνητή Νοημοσύνη, Μηχανική Μάθηση, Βαθιά Μάθηση, RNN, LSTM, αναδρομικά νευρωνικά δίκτυα, χρονοσειρές, πρόβλεψη χρονοσειρών, πρόβλημα πρόβλεψης ηλεκτρικού φορτίου.

Abstract

The purpose of this thesis is initially to get familiar with the Tensorflow library and the Deep Learning algorithms it includes. Specifically, RNN and LSTM models were studied and applied to the problem of short-term electric load forecasting for the Greek interconnected power system.

The results are encouraging, managing to make adequate predictions.

Key words

Artificial Intelligence, Machine Learning, Deep Learning, RNN, LSTM, recurrent neural networks, time series, time series forecasting, power load forecasting problem.

Ευχαριστίες

Ευχαριστώ τον επιβλέποντα καθηγητή μου, κύριο Πάρι Μαστοροκώστα, για τις γνώσεις που μου μετέδωσε, τη βοήθεια και την υποστήριξη του κατά τη διάρκεια των σπουδών μου αλλά και σε όλη την πορεία της εργασίας μου.

Ευχαριστώ τον υποψήφιο διδάκτορα Γ. Κανδηλογιαννάκη για τη βοήθεια του και τα σύνολα δεδομένων που μου παρείχε στην εργασία μου.

Ευχαριστώ την οικογένεια μου για στήριξη της σε όλη τη διάρκεια των σπουδών μου.

Περιεχόμενα

Κατάλογος Εικόνων

Κατάλογος Πινάκων

1. Βαθιά Μάθηση (Deep Learning)

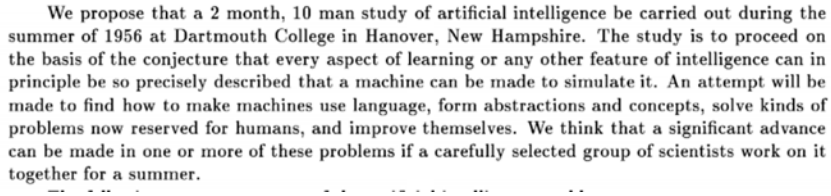
Τα τελευταία χρόνια ο όρος “AI” (Artificial Intellignce - Τεχνητή Νοημοσύνη) εμφανίζεται όλο και συχνότερα σε συζητήσεις, άρθρα, ειδήσεις, κοινωνικά δίκτυα. Οι προσεγγίσεις Machine Learning και Deep Learning χρησιμοποιούνται τακτικά από μεγάλες και μικρές εταιρείες πληροφορικής στην παραγωγή λογισμικού για επίλυση προβλημάτων, με πολύ σημαντικά επιτεύγματα μέχρι στιγμής. Για την κατανόηση του όρου Βαθιά Μάθηση θα πρέπει να αναλυθεί η σχέση του με τις προαναφερθείσες προσεγγίσεις.

Τεχνητή Νοημοσύνη:

Η τεχνητή νοημοσύνη μπορεί να ορισθεί ως η προσπάθεια να αυτοματοποιηθούν διανοητικές εργασίες που συνήθως εκτελούν οι άνθρωποι με νοημοσύνη, σκέψη και λογική.

Οι εργασίες με Τεχνητή Νοημοσύνη μπορεί να περιλαμβάνουν αναγνώριση προτύπων σε δεδομένα, κατανόηση και παραγωγή φυσικής γλώσσας, αναγνώριση εικόνων, λήψη αποφάσεων βασισμένων σε πληροφορίες και πολλά άλλα. Ο σκοπός της είναι η δημιουργία υπολογιστικών συστημάτων που μπορούν να εκτελούν αυτές τις εργασίες με παρόμοια ή ακόμη και υψηλότερη απόδοση από τους ανθρώπους.

Ο όρος Τεχνητή Νοημοσύνη εμφανίστηκε τη δεκαετία του 1950 με το συνολικό επιστημονικό έργο του Alan Turing να ξεχωρίζει θέτοντας το ερώτημα κατά πόσο “οι μηχανές μπορούν να σκεφτούν”. Εν τέλει το 1956 καθιερώθηκε επίσημα σαν ερευνητικό πεδίο όταν ο John McCarthy οργάνωσε ένα καλοκαιρινό σεμινάριο με στόχο να διερευνηθεί αυτή η προσέγγιση.



We propose that a 2 month, 10 man study of artificial intelligence be carried out during the summer of 1956 at Dartmouth College in Hanover, New Hampshire. The study is to proceed on the basis of the conjecture that every aspect of learning or any other feature of intelligence can in principle be so precisely described that a machine can be made to simulate it. An attempt will be made to find how to make machines use language, form abstractions and concepts, solve kinds of problems now reserved for humans, and improve themselves. We think that a significant advance can be made in one or more of these problems if a carefully selected group of scientists work on it together for a summer.

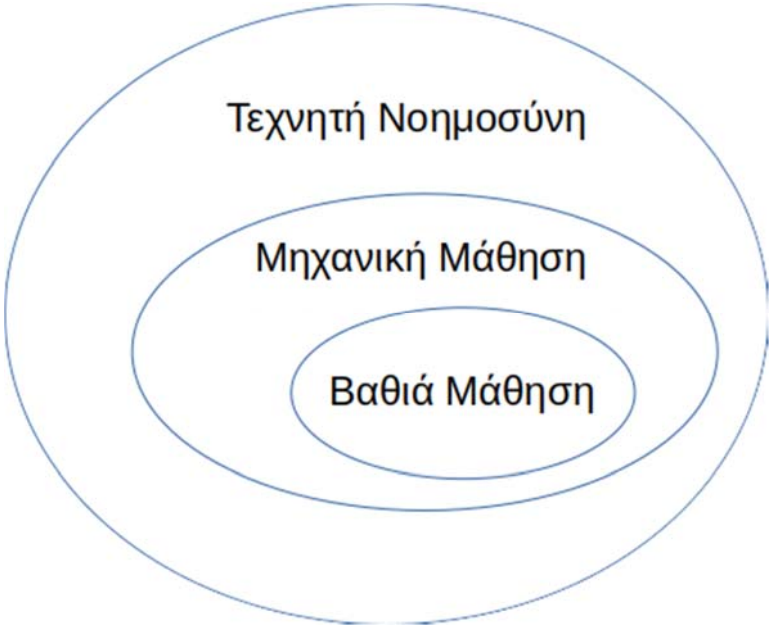
Εικόνα 1.1: Σεμινάριο John McCarthy για μελέτη ΤΝ

PROPOSAL FOR RESEARCH BY JOHN MCCARTHY

During next year and during the Summer Research Project on Artificial Intelligence, I propose to study the relation of language to intelligence. It seems clear that the direct application of trial and error methods to the relation between sensory data and motor activity will not lead to any very complicated behavior. Rather it is necessary for the trial and error methods to be applied at a higher level of abstraction. The human mind apparently uses language as its means of handling complicated phenomena. The trial and error processes at a higher level frequently take the form of formulating conjectures and testing them. The English language has a number of properties which every formal language described so far lacks.

Εικόνα 1.2: Πρόταση John McCarthy [Δ1]

Η τεχνητή νοημοσύνη (ΤΝ) είναι ένα ευρύ επιστημονικό πεδίο που περιλαμβάνει μηχανική μάθηση, βαθιά μάθηση και άλλες προσεγγίσεις που μπορεί να μην εμπλέκουν καμία μορφή μάθησης.



Εικόνα 1.3: Σχέση Προσεγγίσεων Τεχνητής Νοημοσύνης

Μέχρι το 1980, στο μεγαλύτερο μέρος της βιβλιογραφίας για την τεχνητή νοημοσύνη δεν αναφερόταν καθόλου ο όρος "μάθηση". Για παράδειγμα, οι πρώιμοι προγραμματισμοί για σκάκι βασίζονταν μόνο σε ρητούς κανόνες που είχαν δημιουργηθεί από τους προγραμματιστές. Η δημοφιλέστερη πεποίθηση ήταν ότι η τεχνητή νοημοσύνη σε ανθρώπινο επίπεδο θα μπορούσε να επιτευχθεί με τη χειροκίνητη δημιουργία ενός μεγάλου συνόλου σαφών κανόνων για τον χειρισμό της πληροφορίας που θα αποθηκευόταν σε ειδικές βάσεις δεδομένων *(symbolic AI)*. Ωστόσο, αυτή η προσέγγιση δεν καλύπτει πιο πολύπλοκα προβλήματα. Η μηχανική μάθηση αναδείχθηκε για να επιλύσει αυτό το πρόβλημα, επιτρέποντας στους υπολογιστές να μαθαίνουν από δεδομένα.

Μηχανική Μάθηση (Machine Learning) :

Η Μηχανική Μάθηση (Machine Learning) είναι μια υποκατηγορία της τεχνητής νοημοσύνης, που ασχολείται με τον σχεδιασμό και την ανάπτυξη αλγορίθμων που επιτρέπουν σε ένα υπολογιστικό σύστημα να "μαθαίνει" από δεδομένα.

Η μηχανική μάθηση ακολουθεί διαφορετική προσέγγιση από τον κλασσικό προγραμματισμό. Η είσοδος ενός τέτοιου συστήματος είναι δεδομένα με τις αντίστοιχες απαντήσεις, και έξοδος είναι οι κανόνες που οδηγούν στο αποτέλεσμα. Έχει την έννοια της “εκπαίδευσης” και όχι του ρητού προγραμματισμού. Στο σύστημα παρουσιάζονται πολλά παραδείγματα που σχετίζονται με μια εργασία, αυτό εντοπίζει τη δομή σε αυτά και δημιουργεί κανόνες για την αυτοματοποίηση της εργασίας.

Εικόνα 1.4: Κλασσικός προγραμματισμός και Μηχανική Μάθηση

Εκτός από τα δείγματα εισόδου και τα αναμενόμενα αποτελέσματα για αυτή την προσέγγιση είναι απαραίτητη η χρήση ενός μέτρου για την αξιολόγηση της απόδοσης του αλγορίθμου. Η χρήση ενός τρόπου υπολογισμού της απόκλισης των εξόδων του μοντέλου από τις αναμενόμενες και η χρήση του αποτελέσματος ως σήμα ανατροφοδότησης είναι που εισάγουν την έννοια “μάθηση”.

Υπάρχουν τρεις βασικές τεχνικές μηχανικής μάθησης ανάλογα με το σήμα ανατροφοδότησης:

Επιβλεπόμενη μάθηση (Supervised Learning):

Ο αλγόριθμος μαθαίνει από ένα σύνολο δεδομένων με ετικέτες, πράγμα που σημαίνει ότι κάθε είσοδος συνδέεται με τη σωστή έξοδο ή στόχο. Ο στόχος της μάθησης με επίβλεψη είναι μάθει ο αλγόριθμος έναν γενικό κανόνα για την αντιστοίχιση των εισόδων στις εξόδους, ώστε να μπορεί να προβλέπει ή να ταξινομεί με ακρίβεια νέα, μη προβλεπόμενα δεδομένα. Κατά τη διάρκεια της φάσης εκπαίδευσης, ο αλγόριθμος προσαρμόζει τις παραμέτρους του βάσει των αποκλίσεων μεταξύ των προβλέψεών του και των πραγματικών ετικετών στα

εκπαιδευτικά δεδομένα. Αυτή η διαδικασία συνεχίζεται επαναληπτικά μέχρι οι προβλέψεις του αλγορίθμου να συμφωνούν στενά με τις πραγματικές .

Κοινές εφαρμογές περιλαμβάνουν την ταξινόμηση εικόνων, τον εντοπισμό ανεπιθύμητων email, την ιατρική διάγνωση, τη μετάφραση γλωσσών κ.α.

Μη επιβλεπόμενη μάθηση (Unsupervised Learning):

Ο αλγόριθμος μαθαίνει μοτίβα και σχέσεις από μη επισημασμένα δεδομένα χωρίς προκαθορισμένες εξόδους. Ο κύριος στόχος της μη επιβλεπόμενης μάθησης είναι να ανακαλύψει ενσωματωμένες δομές ή ομαδοποιήσεις (κοινά χαρακτηριστικά) μέσα στα δεδομένα που ενδεχομένως δεν είναι εμφανή μέσω του χειροκίνητου ελέγχου. Εφόσον τα παραδείγματα τα οποία χρησιμοποιούνται δεν είναι χαρακτηρισμένα, δεν υπάρχει σφάλμα ή σήμα ανταμοιβής για να αξιολογηθούν οι πιθανές λύσεις.

Η μη επιβλεπόμενη μάθηση χρησιμοποιείται συχνά για εργασίες όπως η κατηγοριοποίηση πελατών, η ανίχνευση ανωμαλιών και η συμπίεση δεδομένων.

Ενισχυτική μάθηση (Reinforcement Learning):

Ο αλγόριθμος μάθησης, προσπαθεί να μάθει μέσα από την άμεση αλληλεπίδραση με το περιβάλλον. Εκτελεί δράσεις στο περιβάλλον και λαμβάνει ανατροφοδότηση σε μορφή ανταμοιβών ή ποινών για κάθε ενέργεια που πραγματοποιεί, όπως τα ανάλογα πρότυπα μάθησης ανταμοιβής-τιμωρίας των έμβιων όντων.

Ο στόχος της ενισχυτικής μάθησης είναι να εκπαιδευτεί ο αλγόριθμος να λαμβάνει αποφάσεις που θα του επιτρέψουν να μεγιστοποιεί την αναμενόμενη μακροπρόθεσμη ανταμοιβή. Κατά τη διάρκεια της εκπαίδευσης, μαθαίνει ποιες ενέργειες οδηγούν σε υψηλότερες ανταμοιβές και ποιες σε χαμηλότερες, αναπτύσσοντας έτσι μια πολιτική ή στρατηγική για να μεγιστοποιήσει την συνολική ανταμοιβή.[Δ2-Δ7]

Ο στόχος στη μηχανική μάθηση γενικά είναι η δημιουργία ενός μοντέλου που θα μάθει χρήσιμους μετασχηματισμούς των δεδομένων εισόδου ώστε να πλησιάζει περισσότερο την αναμενόμενη έξοδο.

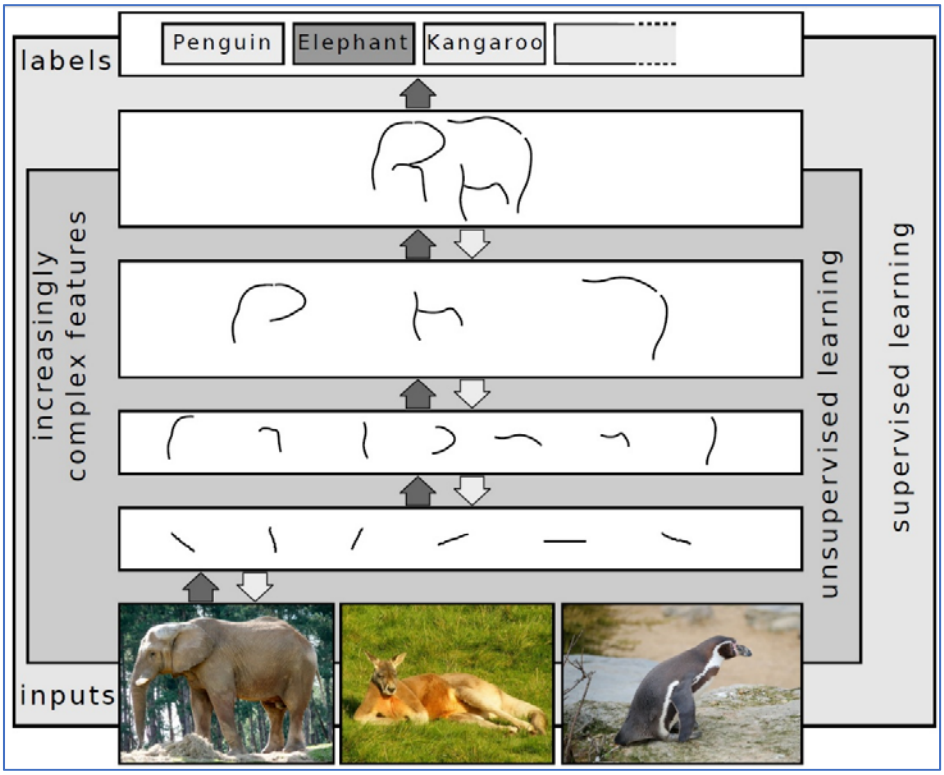
Ένα απλοποιημένο παράδειγμα είναι η εργασία ταξινόμησης εικόνων με ζώα. Η αρχική αναπαράσταση των δεδομένων θα είναι η φωτογραφία σε Red – Green – Blue (RGB). Η εύρεση των χρήσιμων χαρακτηριστικών (*feature engineering*) όπως ακμές, σχήματα, πληροφορίες χρώματος, γίνεται χειροκίνητα. Τα δεδομένα εισόδου θα είναι τα χαρακτηριστικά και οι ετικέτες. Το μοντέλο θα ανακαλύψει έναν χρήσιμο μετασχηματισμό που θα παράγει την επιθυμητή αναπαράσταση δηλαδή την αντιστοίχιση του ζώου στην ετικέτα.

Επομένως η Μηχανική Μάθηση μπορούμε να πούμε ότι είναι η αναζήτηση για χρήσιμες αναπαραστάσεις και κανόνες πάνω σε ορισμένα δεδομένα εισόδου, χρησιμοποιώντας σαν οδηγό ένα σήμα ανατροφοδότησης. Αυτοί οι μετασχηματισμοί μπορεί να γίνονται χρησιμοποιώντας μεθόδους όπως Δέντρα Απόφασης (*Decision Trees)*, Μηχανές Διανυσμάτων Υποστήριξης (*Support Vector Machines*), Κ-Κοντινότεροι Γείτονες (*K-Nearest Neighbors*), Νευρωνικά Δίκτυα *(Neural Networks)* κ.α. Οι απλές μέθοδοι Μηχανικής Μάθησης περιλαμβάνουν λίγα επίπεδα μετασχηματισμών, ένα ή δύο, για αυτό λέγεται και Επιφανειακή Μάθηση (*Shallow Learning*)

Βαθιά Μάθηση (Deep Learning)

Η Βαθιά Μάθηση είναι μια υποκατηγορία της Μηχανικής Μάθησης που βασίζεται σε Τεχνητά Νευρωνικά Δίκτυα (*Artifial Neural Networks*)– εν συντομία Νευρωνικά Δίκτυα. Οι αλγόριθμοι μαθαίνουν να αναπαριστούν ιεραρχικά πρότυπα (μοτίβα) και χαρακτηριστικά από ακατέργαστα δεδομένα. Ο όρος "βαθιά" υποδηλώνει το βάθος του δικτύου, καθώς περιλαμβάνει πολλά επίπεδα (*layers*) συνδεδεμένων νευρώνων, με κάθε ένα να συνεισφέρει στην εξαγωγή αυξανόμενης πολυπλοκότητας χαρακτηριστικών από τα δεδομένα εισόδου [Β1, 1-18].

Για παράδειγμα, στην επεξεργασία εικόνας, τα χαμηλότερα επίπεδα μπορεί να ανιχνεύουν ακμές, ενώ τα υψηλότερα επίπεδα μπορεί να ανιχνεύουν χαρακτηριστικά που είναι κατανοητά κι από έναν άνθρωπο, όπως ψηφία, γράμματα ή πρόσωπα. [Δ8]

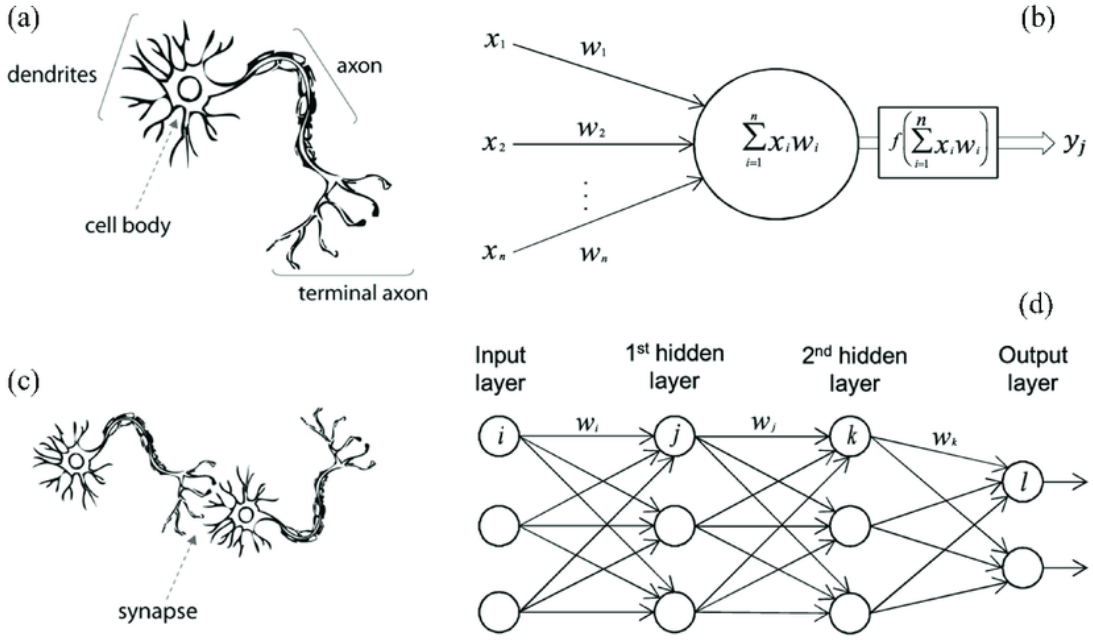


Εικόνα 1.5: Αναπαράσταση επιπέδων Βαθιάς Μάθησης για ταξινόμηση εικόνων

2. Νευρωνικά Δίκτυα (Neural Networks)

Τα Νευρωνικά Δίκτυα είναι εμπνευσμένα από τη δομή και τη λειτουργία των συνδεδεμένων νευρώνων του ανθρώπινου εγκεφάλου. Είναι ένα σύνολο από τεχνητούς νευρώνες (κόμβους) που συνδέονται μεταξύ τους και λειτουργούν ως ένα ενιαίο σύστημα για την επεξεργασία πληροφοριών.

Ένα Νευρωνικό Δίκτυο μπορεί να αποτελείται από διάφορα επίπεδα *(layers)*, καθένα από τα οποία περιέχει έναν ή περισσότερους νευρώνες. Κάθε νευρώνας λαμβάνει εισόδους και για να τις επεξεργαστεί, κάθε μία πολλαπλασιάζεται με έναν συντελεστή που ονομάζεται "βάρος." Έπειτα, τα αποτελέσματα αυτών των πολλαπλασιασμών συγκεντρώνονται και υποβάλλονται σε μια συνάρτηση ενεργοποίησης, η οποία καθορίζει το επίπεδο δραστηριότητας του νευρώνα. Οι έξοδοι αυτών των νευρώνων στέλνονται στους νευρώνες του επόμενου επιπέδου ως είσοδοι, και η διαδικασία συνεχίζεται μέχρι το τελευταίο επίπεδο.



Εικόνα 2.1: Βιολογικοί και Τεχνητοί Νευρώνες [Δ9]
(a) Ένας βιολογικός νευρώνας,(b) ένας τεχνητός νευρώνας,
(c) σύνδεση βιολογικών νευρώνων, (d) σύνδεση τεχνητών νευρώνων

Ένα επίπεδο απλού νευρωνικού δικτύου κάνει τον μετασχηματισμό:

output = activation_function(dot(W, *input*) + b) [B1, 63]

activation_function = συνάρτηση ενεργοποίησης

W = πίνακας βαρών

dot = πράξη εσωτερικού γινομένου

b = σταθερά πόλωσης (bias) που βοηθάει την εκπαίδευση και τη γενίκευση καθώς επιτρέπει στο μοντέλο να προσαρμόζει την έξοδο κι όταν η είσοδος είναι μηδέν.

Η έννοια των ANNs είναι η βάση για την ανάπτυξη πιο πολύπλοκων αρχιτεκτονικών, όπως τα βαθιά νευρωνικά δίκτυα (DNNs) και τα αναδρομικά νευρωνικά δίκτυα (RNNs) που χρησιμοποιούνται ευρέως στον τομέα της βαθιάς μάθησης.

Βάρη (Weights)

Ο όρος "βάρη" αναφέρεται στους συντελεστές που ρυθμίζουν την επίδραση της κάθε εισόδου στον νευρώνα και κατευθύνουν τη διαδικασία εκμάθησης.

Τα βάρη είναι οι παράμετροι που προσαρμόζονται κατά τη διάρκεια της εκπαίδευσης του μοντέλου, ώστε το δίκτυο να μπορεί να αναγνωρίζει συγκεκριμένα μοτίβα και χαρακτηριστικά στα δεδομένα εισόδου. Επομένως καθορίζουν τη σημαντικότητα της κάθε εισόδου για τον συγκεκριμένο νευρώνα.

Συνάρτηση ενεργοποίησης (Activation Funtion)

Η συνάρτηση ενεργοποίησης επιλέγει πώς θα ανταποκριθεί ο νευρώνας σε μια συνδυασμένη είσοδο από άλλους νευρώνες ή από τα δεδομένα εισόδου. Αν ο νευρώνας ενεργοποιηθεί, σημαίνει ότι θα παράγει μια έξοδο και αντίστοιχα αν παραμείνει ανενεργός, δεν παράγει έξοδο. Η επιλογή της συνάρτησης ενεργοποίησης επηρεάζει τον τρόπο με τον οποίο ο νευρώνας αντιδρά στις εισόδους και επηρεάζει τον τρόπο λειτουργίας του Νευρωνικού Δικτύου στο σύνολό του.[Δ10]

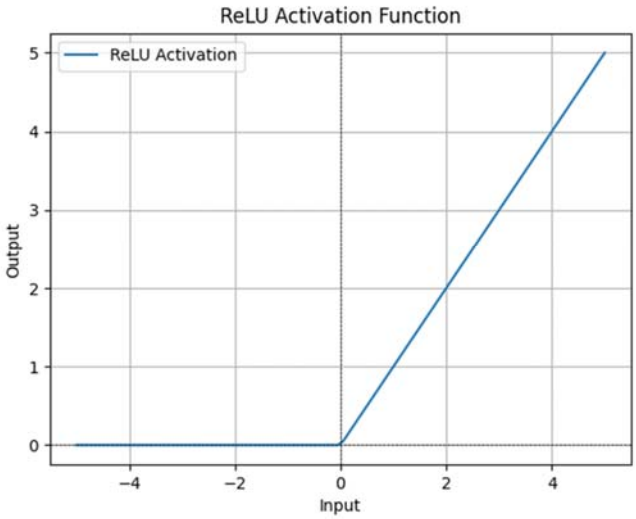
Παραδείγματα συναρτήσεων ενεργοποίησης:

ReLu (Rectified Linear Unit) - Συνάρτηση Ράμπας:

$$ReLU(x) = \max(0,\ x)$$

Όπου x είναι το συνολικό άθροισμα των εισόδων που έχουν πολλαπλασιαστεί με τα βάρη.

Η συνάρτηση ράμπας επιστρέφει το x αν το x είναι θετικό, και επιστρέφει το 0 αν το x είναι αρνητικό. Αυτό σημαίνει ότι ο νευρώνας ενεργοποιείται όταν το συνολικό άθροισμα των εισόδων είναι θετικό, αλλά παραμένει ανενεργός όταν το συνολικό άθροισμα είναι αρνητικό.

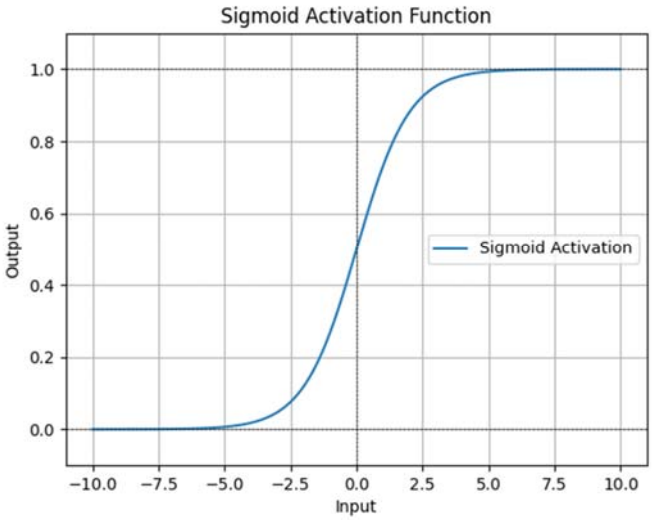


Εικόνα 2.2: Συνάρτηση ενεργοποίησης Relu

Sigmoid - Σιγμοειδής Συναρτηση:

$$sigmoid(x) = \frac{1}{1 + e^{-x}}$$

Η σιγμοειδής συνάρτηση περιορίζει το x στο ανοιχτό διάστημα (0, 1). Προσεγγίζει το 0 όταν οι τιμές είναι μεγάλες αρνητικές και το 1 όταν οι τιμές είναι μεγάλες θετικές. Χρησιμοποιείται συνήθως ως συνάρτηση ενεργοποίησης της εξόδου του μοντέλου σε προβλήματα δυαδικής ταξινόμησης.

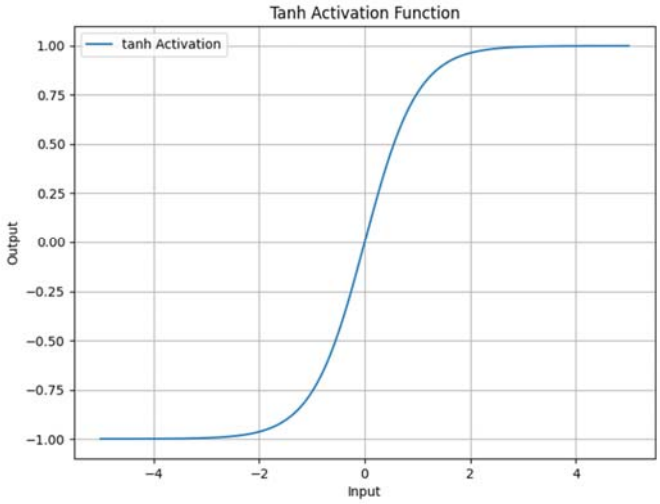


Εικόνα 2.4: Συνάρτηση Ενεργοποίησης Sigmoid

Tanh (Hyperbolic Tangent) - Υπερβολική Εφαπτομένη:

$$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

Η Υπερβολική Εφαπτομένη συνάρτηση περιορίζει το x στο ανοιχτό διάστημα (-1, 1). Προσεγγίζει το -1 όταν οι τιμές είναι μεγάλες αρνητικές και το 1 όταν οι τιμές είναι μεγάλες θετικές.



Εικόνα 2.5: Συνάρτηση Ενεργοποίησης Tanh

Η επιλογή συνάρτησης ενεργοποίησης εξαρτάται από το σύνολο δεδομένων και την εκάστοτε εργασία

Εκπαίδευση και αξιολόγηση μοντέλων νευρωνικών δικτυών

Τα ANN χρησιμοποιούνται σε ποικίλες εφαρμογές Βαθιάς Μάθησης. Κατά τη διάρκεια της εκπαίδευσης, τα βάρη και οι παράμετροι των νευρώνων προσαρμόζονται έτσι ώστε το μοντέλο να μπορεί να μάθει και να αναπαραστήσει πολύπλοκα μοτίβα και συσχετίσεις στα δεδομένα εισόδου.

Η διαδικασία εκπαίδευσης ενός νευρωνικού δικτύου μπορεί να αναλυθεί από τα εξής απλοποιημένα βήματα:

1. Αρχικοποίηση (Initialization):

Αρχικοποίηση των βαρών του νευρωνικού δικτύου. Οι τιμές αρχικοποίησης είναι τυχαίες εκτός κι αν ορισθεί διαφορετικά.

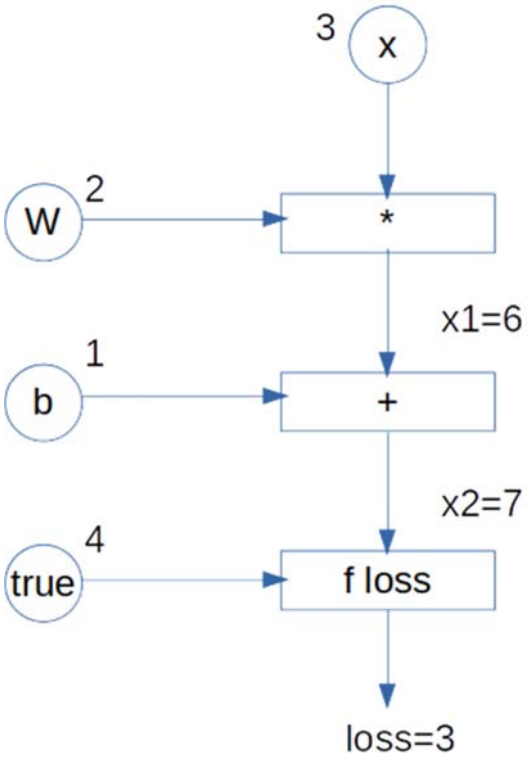
2. Προώθηση στο δίκτυο - Ευθεία Τροφοδότηση (Forward Pass):

Τροφοδότηση μιας παρτίδας (batch) δεδομένων στο νευρωνικό δίκτυο ώστε να παραχθεί μια έξοδος. Οι είσοδοι "προωθούνται" μέσα από τα διάφορα επίπεδα του νευρωνικού δικτύου, υπολογίζονται συνδυασμοί εισόδων και βαρών, προστίθεται πόλωση (bias) και εφαρμόζονται συναρτήσεις ενεργοποίησης, προκειμένου να παραχθούν οι τελικές έξοδοι του δικτύου.

3. Υπολογισμός σφάλματος/απώλειας (loss calculation):

Υπολογισμός απώλειας αναφέρεται στη διαδικασία του υπολογισμού του σφάλματος μεταξύ των πραγματικών εξόδων του μοντέλου και των επιθυμητών εξόδων (ground truth - target outputs) για μια συγκεκριμένη παρτίδα των δεδομένων εκπαίδευσης.

Ο υπολογισμός αυτός συνήθως γίνεται με τη χρήση κάποιας συνάρτησης απώλειας (loss function) που αξιολογεί το μέγεθος του σφάλματος. Αυτή η συνάρτηση μετράει πόσο διαφορετικά είναι τα προβλεπόμενα από τα πραγματικά αποτελέσματα και παράγει ένα μέγεθος που αντιπροσωπεύει την απόδοση του μοντέλου. Ο στόχος κατά την εκπαίδευση είναι να ελαχιστοποιηθεί αυτή η απώλεια, δηλαδή να μειωθεί το σφάλμα των προβλέψεων.

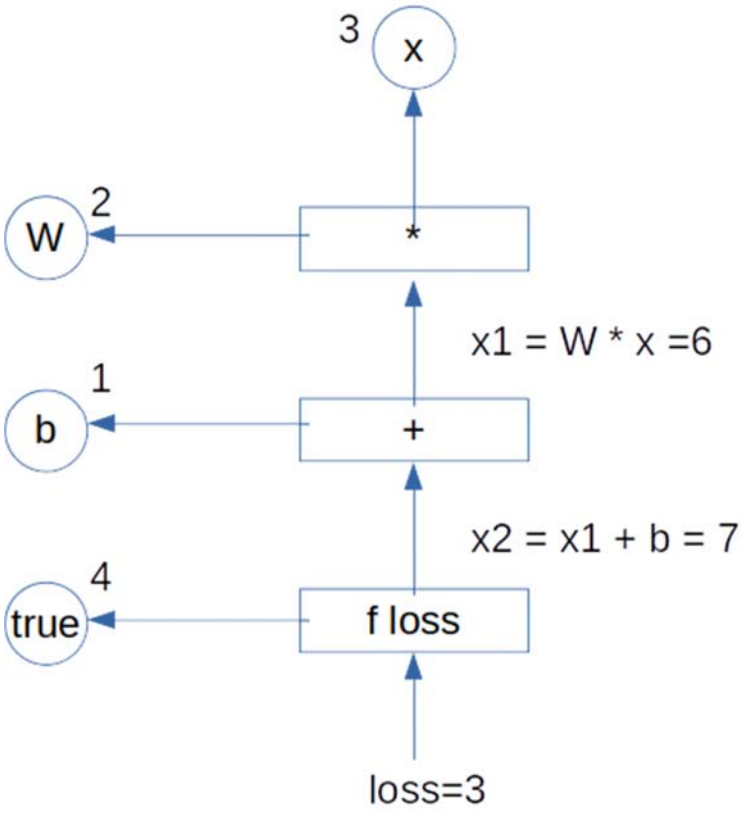


Εικόνα 2.6: Διαδικασία εκπαίδευσης - Loss calculation

Για $f_{loss} = abs\,(true\ -\ predicted)$

4. <u>Ανάστροφη Διάδοση - Οπισθοδρόμηση (Backpropagation):</u>

Το δίκτυο αντιστρέφει την διαδικασία προώθησης προς τα εμπρός για να υπολογίσει το πώς πρέπει να ενημερώσει τα βάρη των νευρώνων, προκειμένου να μειώσει το σφάλμα. Αυτό επιτυγχάνεται υπολογίζοντας την παράγωγο της συνάρτησης απώλειας σε σχέση με τα βάρη και τις πολώσεις του μοντέλου, με τη βοήθεια του κανόνα της αλυσίδας, ώστε να αντιληφθεί πώς συμβάλει η κάθε τιμή τους στο συνολικό σφάλμα.



Εικόνα 2.7: Διαδικασία εκπαίδευσης - Backpropagation

$$\frac{\partial floss}{\partial W} = \frac{\partial floss}{\partial x2} \cdot \frac{\partial x2}{\partial x1} \cdot \frac{\partial x1}{\partial W} = 1 \cdot 1 \cdot 3 = 3$$

$$\frac{\partial floss}{\partial b} = \frac{\partial floss}{\partial x2} \cdot \frac{\partial x2}{\partial b} = 1 \cdot 1 = 1$$

5. Καταβολή της κλίσης (Gradient Descent):

Ενημέρωση των βαρών και των πολώσεων στην αντίθετη κατεύθυνση από αυτή των παραγώγων ώστε να ελαχιστοποιηθεί το σφάλμα με κάποιον αλγόριθμο βελτιστοποίησης (optimizer). Το μέγεθος την ενημέρωσης εξαρτάται από τον ρυθμό εκπαίδευσης (learning rate) . Η διαδικασία αυτή ονομάζεται Gradient Descent και σκοπός είναι η εύρεση ενός καλύτερου συνόλου τιμών που παράγουν μικρότερο σφάλμα. Για το συγκεκριμένο παράδειγμα:

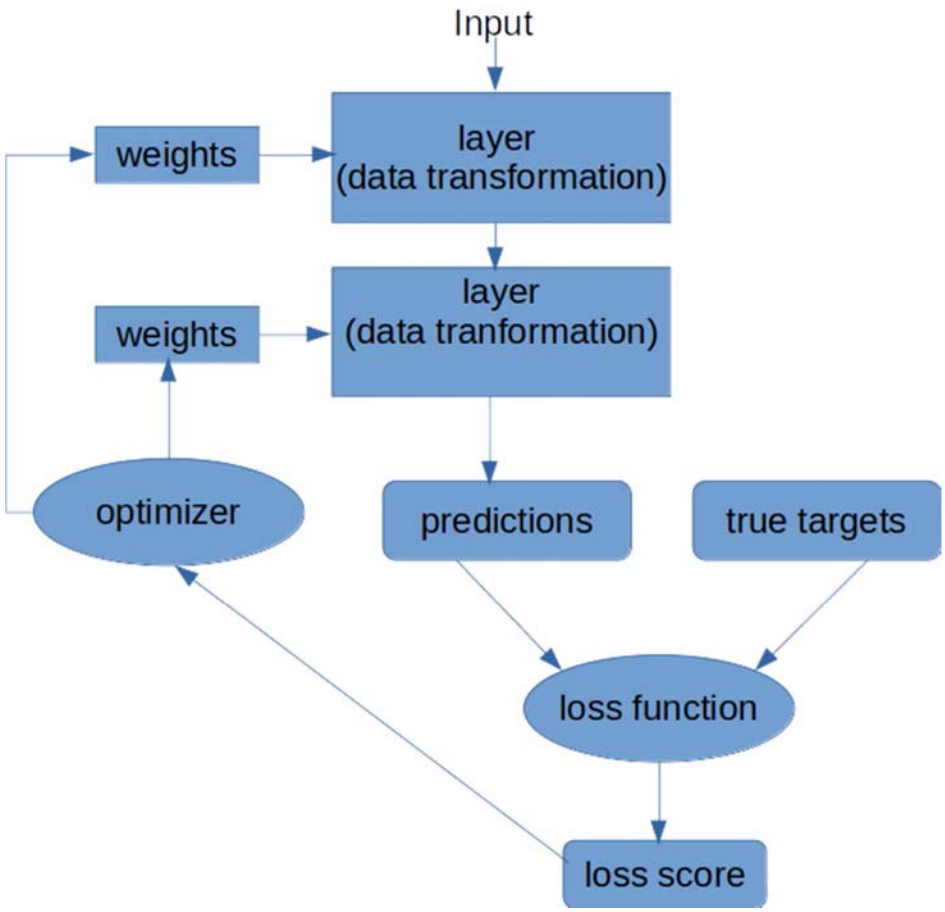
$$W_{n+1} = W_n - (learning\ rate) \cdot \frac{\partial floss}{\partial W} = 2 - 0.01 \cdot 3 = 1.97$$

$$b_{n+1} = b_n - (learning\ rate) \cdot \frac{\partial floss}{\partial b} = 1 - 0.01 \cdot 1 = 0.99$$

Όπου Wn+1, bn+1 είναι οι ενημερωμένες τιμές και Wn, bn οι παρούσες.

6. Επανάληψη:

Επανάληψη όλων των βημάτων - πολλαπλές εποχές (epochs) πάνω στο σύνολο δεδομένων εκπαίδευσης. Αυτό επιτρέπει στο μοντέλο να προσαρμόσει τις παραμέτρους του και να μάθει από τα δεδομένα



Εικόνα 2.8: Διαδικασία εκπαίδευσης συνολικά

Αφού ολοκληρωθεί η εκπαίδευση θα πρέπει να γίνει αξιολόγηση της απόδοσης. Ένα μέρος από το σύνολο δεδομένων (10%-20%) έχει διαχωριστεί από τη διαδικασία της εκπαίδευσης ώστε να είναι εντελώς καινούριο για το μοντέλο. Πάνω σε αυτό το σύνολο δοκιμής γίνονται οι προβλέψεις κι έπειτα βάση κάποιου μέτρου αξιολόγησης (π.χ RMSE) αξιολογείται η ικανότητα γενίκευσης του μοντέλου (generalization). [B1, 48-66]

Πρόβλημα Υπερπροσαρμογής στα δεδομένα εκπαίδευσης (Overfitting)

Το “overfitting” είναι ένα πρόβλημα που μπορεί να προκύψει κατά τη διάρκεια της εκπαίδευσης. Είναι μια κατάσταση στη μηχανική και βαθιά μάθηση όπου ένα μοντέλο μαθαίνει τα δεδομένα εκπαίδευσης τόσο καλά ώστε να προσαρμόζεται υπερβολικά σε αυτά, και ως αποτέλεσμα αποτυγχάνει να γενικεύσει σωστά σε νέα, απροσδόκητα δεδομένα. Αυτό συμβαίνει όταν το μοντέλο αποκτά πολύ μεγάλη χωρητικότητα και εκπαιδεύεται για πολλές εποχές οπότε μαθαίνει το θόρυβο στα δεδομένα εκπαίδευσης, αντί να μάθει τα γενικά πρότυπα.

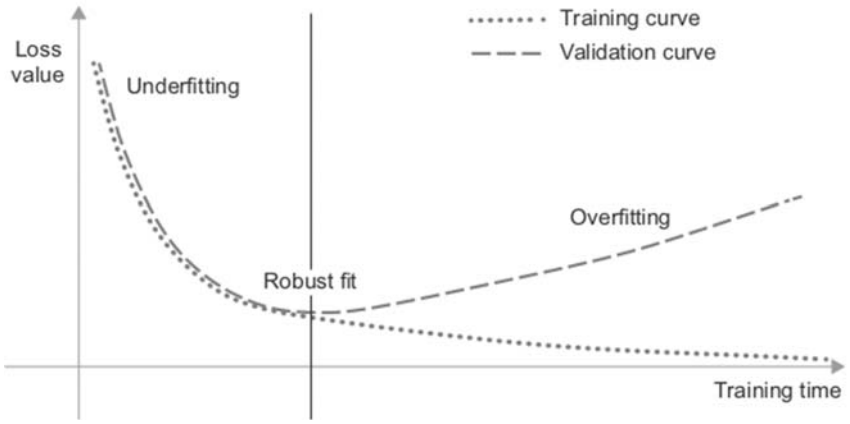
Κάποιες από τις κοινές τεχνικές για την αντιμετώπιση του overfitting στο TensorFlow/Keras περιλαμβάνουν:

- Διαίρεση των δεδομένων: Διαίρεσης των δεδομένων σε σύνολο εκπαίδευσης (training), επικύρωσης (validation) και δοκιμής (testing). Αυτό επιτρέπει τον έλεγχο του μοντέλου σε διαφορετικά σύνολα δεδομένων και βοηθά στην αξιολόγηση της γενίκευσής του.

- Περιορισμός της χωρητικότητας του μοντέλου: Μείωση του αριθμού των επιπέδων και/ή των νευρώνων του μοντέλου. Μικρότερα μοντέλα έχουν λιγότερη δυνατότητα να μάθουν τον θόρυβο και πετυχαίνουν καλύτερη γενίκευση.

- Προσθήκη Dropout: Το dropout είναι μια τεχνική όπου τυχαία επιλεγμένοι νευρώνες απενεργοποιούνται κατά την εκπαίδευση βάση κάποιας πιθανότητας.

- Early Stopping: Χρήση του μηχανισμού "early stopping" για να διακοπεί η εκπαίδευση όταν η απόδοση στα δεδομένα επικύρωσης σταματήσει να βελτιώνεται.[Β1, 143-145], [Δ11]



Εικόνα 2.9: Αναπαράσταση Overfitting

Πρόβλημα Εξαφάνισης Παραγώγου(Gradient Vanishing Problem)

Το "gradient vanishing" είναι ένα πρόβλημα που μπορεί να προκύψει κατά την εκπαίδευση νευρωνικών δικτύων, ιδιαίτερα σε βαθιά δίκτυα. Οι παράγωγοι των συναρτήσεων σφάλματος μπορεί να μειώνονται σημαντικά καθώς προχωρούμε προς τα πίσω στο δίκτυο κατά τη διαδικασία της αντίστροφης διάδοσης (backpropagation), και συνεπώς τα βάρη και παράμετροι στα ανώτερα επίπεδα του δικτύου δεν ενημερώνονται αποτελεσματικά. Αυτό το φαινόμενο είναι ιδιαίτερα εμφανές όταν χρησιμοποιούνται συναρτήσεις ενεργοποίησης με μη πεπερασμένες παραγώγους, όπως η σιγμοειδής συνάρτηση (sigmoid) ή η υπερβολική

εφαπτομένη συνάρτηση (tanh). Σε αυτές τις περιπτώσεις, οι παράγωγοι μπορεί να προσεγγίζουν το μηδέν για μεγάλες ή μικρές εισόδους, μειώνοντας σημαντικά τη δυνατότητα των βαρών να αλλάξουν[Δ12].

Όταν εμφανίζεται αυτό το πρόβλημα, τα επίπεδα του δικτύου που είναι μακριά από την έξοδο δεν μαθαίνουν αποτελεσματικά, και το δίκτυο γενικά δυσκολεύεται να εκπαιδευτεί για πολύ βαθιά αρχιτεκτονική. Για την αντιμετώπιση αυτού του προβλήματος, έχουν προταθεί πολλές τεχνικές, όπως η χρήση άλλων συναρτήσεων ενεργοποίησης (όπως η Rectified Linear Unit - ReLU) και η χρήση ειδικών αρχιτεκτονικών όπως τα αναδρομικά νευρωνικά δίκτυα LSTM και GRU (Gated Recurrent Units).

3. Αναδρομικά Νευρωνικά Δίκτυα (Recurrent Neural Networks)

Τα Αναδρομικά Νευρωνικά Δίκτυα (Recurrent Neural Networks - RNNs) είναι ένα είδος τεχνητών νευρωνικών δικτύων που έχουν σχεδιαστεί ειδικά για την επεξεργασία ακολουθιών δεδομένων. Σε αντίθεση με άλλα νευρωνικά δίκτυα, τα RNN έχουν τη δυνατότητα να αντιλαμβάνονται τη χρονική δομή και τις συσχετίσεις μεταξύ των δεδομένων σε ακολουθίες όπως προτάσεις, μουσική, χρονοσειρές και άλλα.

Η κύρια ιδέα που καθοδηγεί τα RNN είναι η χρήση μιας εσωτερικής κρυφής κατάστασης (hidden state) που αναπαριστά τη μνήμη του δικτύου για τις προηγούμενες εισόδους που έχουν επεξεργαστεί. Αυτή η κατάσταση παραμένει αναλλοίωτη καθ' όλη τη διάρκεια της επεξεργασίας της ακολουθίας, ενώ την ίδια στιγμή ενημερώνεται και προσαρμόζεται στην κάθε νέα είσοδο. Είναι εμπνευσμένα από την ανθρώπινη μνήμη όπως κατά την ανάγνωση της παρούσας παραγράφου ο ανθρώπινος νους συγκρατεί τις πληροφορίες των προηγούμενων προτάσεων για την κατανόηση της.

Ένα RNN μπορεί να θεωρηθεί ως δίκτυο με επαναλαμβανόμενη δομή, καθώς η κρυφή κατάσταση που διατηρείται καθ' όλη τη διαδικασία επεξεργασίας ανανεώνεται σε κάθε χρονικό βήμα. Αυτή η επαναλαμβανόμενη δομή επιτρέπει στο δίκτυο να επηρεάζεται από την προηγούμενη κατάσταση και να διατηρεί πληροφορίες για προηγούμενες εισόδους, επιτρέποντας του να αναγνωρίζει μοτίβα και συσχετίσεις που εκτείνονται σε διάφορα χρονικά βήματα.

Τα RNNs χρησιμοποιούνται σε πολλές εφαρμογές, συμπεριλαμβανομένων:

- Μηχανική μετάφραση: Τα RNNs μπορούν να εκπαιδευτούν να μεταφράζουν από μία γλώσσα σε μία άλλη, αντιστοιχίζοντας τις λέξεις και τις φράσεις σε αντίστοιχες μεταφράσεις.
- Αναγνώριση φωνής: Τα RNNs μπορούν να αναλύσουν και να επεξεργαστούν ομιλία και φωνητικά σήματα για την αναγνώριση και τη μετατροπή τους σε κείμενο.
- Ανάλυση απόψεων: Τα RNNs μπορούν να κατανοήσουν και να αναλύσουν τις απόψεις και τα συναισθήματα που εκφράζονται σε κείμενα
- Πρόβλεψη χρονοσειρών: Τα RNNs μπορούν να εκπαιδευτούν να προβλέπουν τις επόμενες τιμές σε μια χρονοσειρά.

Ένα RNN μπορεί να είναι απλό (SimpleRNN), χρησιμοποιώντας μια βασική αναδρομική μονάδα, ή μπορεί να έχει πιο σύνθετες αρχιτεκτονικές όπως τα Long Short-Term Memory (LSTM) και τα Gated Recurrent Units (GRU). Αυτές οι προηγμένες μονάδες είναι σχεδιασμένες για να αντιμετωπίσουν το πρόβλημα της εξαφάνισης της εξαφάνισης παραγώγου (vanishing gradients) και να διατηρήσουν μακροπρόθεσμες εξαρτήσεις στη μνήμη του δικτύου.

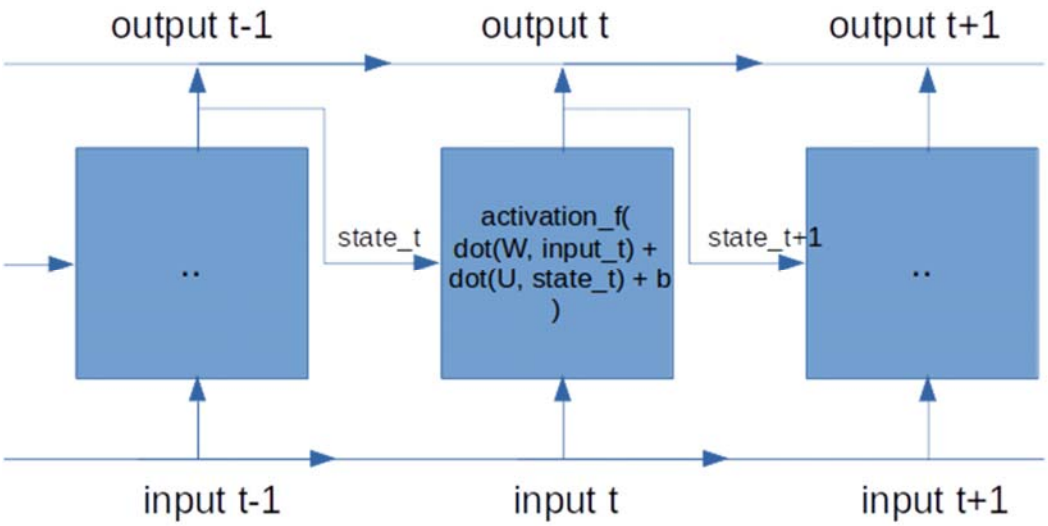
SimpleRNN

Ο ψευδοκώδικας ενός απλού Αναδρομικού Νευρωνικού Δικτύου (RNN) μπορεί να αναλυθεί ως εξής:

```
state_t = 0
for input_t in input_sequence:
    output_t = activation_function(dot(W, input_t) + dot(U, state_t) + b)
    state_t = output_t
```

[B1, 294-296]

Εδώ υπάρχουν δύο πίνακες βαρών W, U αντί για έναν όπως στα απλά ANN.



Εικόνα 3.1: Αναπαράσταση SimpleRNN

LSTM (Long Short-Term Memory)

Τα Long Short-Term Memory είναι ένα είδος Αναδρομικού Νευρωνικού Δικτύου που αναπτύχθηκε για να αντιμετωπίσει το πρόβλημα της εξαφάνισης της παραγώγου (vanishing gradient problem) και να διατηρήσει βραχυπρόθεσμες εξαρτήσεις (Short-Term) στη μνήμη του, καθ' όλο το μήκος πολύ μεγάλων ακολουθιών εισόδου (long).

Τα LSTM έχουν ευρεία χρήση σε προβλήματα που απαιτούν μνήμη για βραχυπρόθεσμες εξαρτήσεις, όπως η ανάλυση χρονοσειρών.

Η θεωρία των LSTM βασίζεται σε μια ειδική αρχιτεκτονική με πύλες (gates) που επιτρέπουν στο δίκτυο να ελέγχει την πληροφορία που περνά από την κατάστασή του και να αποφασίζει ποια πληροφορία να αποθηκεύσει και ποια να αγνοήσει. Η κεντρική ιδέα είναι η χρήση ειδικών μονάδων μνήμης, γνωστών ως "κύτταρα μνήμης" (memory cells).

Ένα κύτταρο μνήμης LSTM αποτελείται από τέσσερις βασικές μονάδες:

- Πύλη απόρριψης (Forget gate): Αποφασίζει ποια πληροφορία θα αγνοηθεί από την προηγούμενη κατάσταση κυττάρου.
- Πύλη εισόδου (Input gate): Αποφασίζει ποια πληροφορία απ' την είσοδο πρέπει θα αποθηκευτεί στο κύτταρο μνήμης.
- Πύλη εξόδου (Output gate): Αποφασίζει ποια πληροφορία πρέπει θα εξαχθεί από το κύτταρο μνήμης.
- Κατάσταση κύτταρου (Cell state): Το κύτταρο μνήμης διατηρεί μια κατάσταση που περνά από το ένα χρονικό βήμα στο επόμενο. Το αποτέλεσμα είναι συνάρτηση των παραπάνω

Οι μονάδες αυτές ελέγχονται από συναρτήσεις ενεργοποίησης που παίρνουν ως είσοδο την τρέχουσα είσοδο και την προηγούμενη κατάσταση. Οι συναρτήσεις αυτές παράγουν τις τιμές των πυλών που ελέγχουν την ροή της πληροφορίας μέσα στο κύτταρο μνήμης.

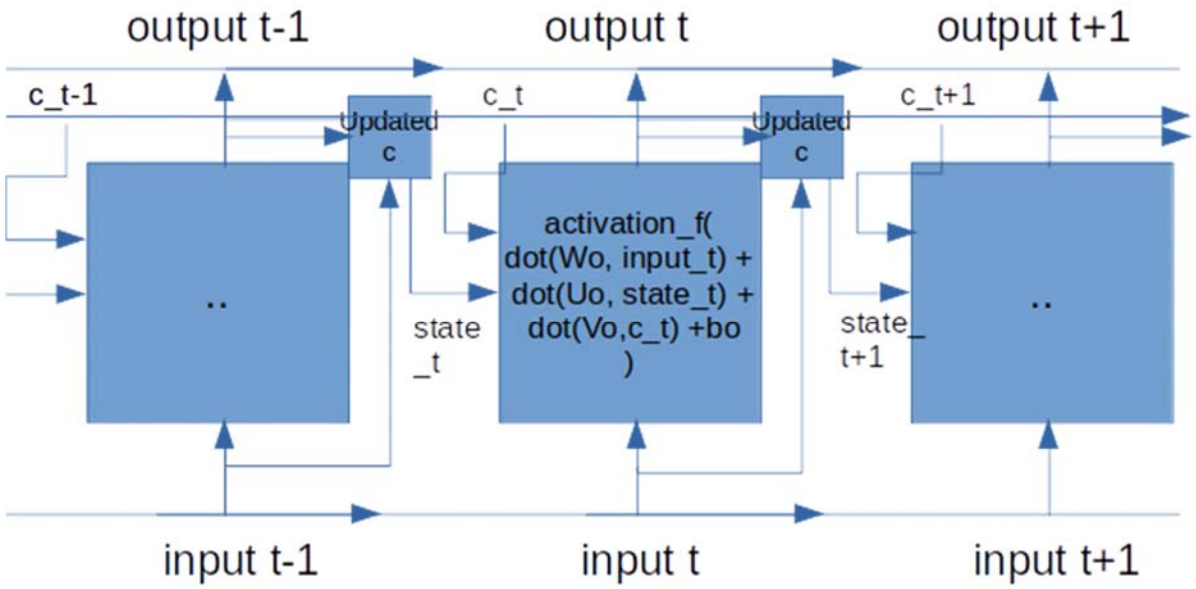
Ο ψευδοκώδικας για τον υπολογισμό της εξόδου μπορει να αναλυθεί ως :

```
output_t = activation(dot(state_t, Uo) + dot(input_t, Wo) + dot(c_t, Vo) + bo)
```

'Οπου c_t είναι η κατάσταση του κυττάρου μνήμης και είναι συνάρτηση των αποτελεσμάτων των πυλών εισόδου, απόρριψης και εξόδου i_t, f_t, k_t αντίστοιχα.

```
i_t = activation(dot(state_t, Ui) + dot(input_t, Wi) + bi)

f_t = activation(dot(state_t, Uf) + dot(input_t, Wf) + bf)

k_t = activation(dot(state_t, Uk) + dot(input_t, Wk) + bk)
```

```
c_t+1 = i_t * k_t + c_t * f_t
```



Εικόνα 3.1: Αναπαράσταση LSTM

GRU (Gated Recurrent Units)

Τα GRU (Gated Recurrent Units) είναι ένα είδος Αναδρομικού Νευρωνικού Δικτύου, όπως και τα LSTM όπου σχεδιάστηκαν για την αντιμετώπιση του προβλήματος της εξαφάνισης παραγώγων. Είναι βασισμένα στην ίδια ιδέα και μοιράζονται τα περισσότερα χαρακτηριστικά άλλα έχουν πιο απλή αρχιτεκτονική και λιγότερες παραμέτρους.

Ένα κύτταρο μνήμης GRU αποτελείται από δύο βασικές μονάδες:

- Πύλη Επαναφοράς (Reset gate): Αποφασίζει ποια πληροφορία θα αγνοηθεί από την προηγούμενη κρυφή κατάσταση.
- Πύλη Ενημέρωσης (Update gate): Αποφασίζει ποια πληροφορία θα συντηρηθεί από την προηγούμενη κρυφή κατάσταση.

Σε αντίθεση με τα LSTM που διατηρούν την κυτταρική κατάσταση χωριστά από την κρυφή κατάσταση, τα GRU χρησιμοποιούν μια μοναδική παράμετρο κατάστασης.

Λόγω της πιο απλοποιημένης αρχιτεκτονικής τους έχουν μικρότερο υπολογιστικό κόστος κατά την εκπαίδευση άρα επιτυγχάνουν και καλύτερο χρόνο και σε κάποιες περιπτώσεις καλύτερη απόδοση. [B1, 299], [Δ13-Δ15]

4. Χρονοσειρές (Timeseries)

Με τον όρο χρονοσειρά (time series) νοείται μια χρονικά ιεραρχημένη ακολουθία παρατηρήσεων που καταγράφονται σε ίσα τακτά χρονικά διαστήματα. Μια χρονοσειρά θα μπορούσε να θεωρηθεί ότι εκφράζει τη διαχρονική εξέλιξη μιας μεταβλητής κατά τη διάρκεια ίσων διαδοχικών χρονικών περιόδων [Δ16]. Παραδείγματα χρονοσειρών συναντάμε στην καθημερινότητα, σε όλους τους επιστημονικούς τομείς. Μερικά είναι η θερμοκρασία, το ωριαίο φορτίου του ηλεκτρικού δικτύου, η επισκεψιμότητα μιας ιστοσελίδας, η τιμή μιας μετοχής, οι πωλήσεις ενός προϊόντος.

Η ανάλυση και η επεξεργασία τέτοιων δεδομένων απαιτεί την κατανόηση της δυναμικής του συστήματος που διερευνάται-την περιοδικότητα, την τάση, τις ακανόνιστες διακυμάνσεις του.

- **Εποχικότητα**: Τα επαναλαμβανόμενα πρότυπα (μοτίβα) που εμφανίζονται σε τακτικά διαστήματα, όπως καθημερινά, εβδομαδιαία, μηνιαία ή ετήσια. Πρόκειται για τακτικές και προβλέψιμες αλλαγές που παρατηρούνται στη χρονοσειρά. Αυτές οι αλλαγές συνήθως προκαλούνται από φυσικά ή κοινωνικά γεγονότα που επαναλαμβάνονται περίπου την ίδια περίοδο.
  Η καταγραφή της θερμοκρασίας είναι ένα αντιπροσωπευτικό παράδειγμα δεδομένων με εποχικότητα.

- **Τάση**: Η κατεύθυνση ή πορεία των δεδομένων κατά τη διάρκεια ενός εκτεταμένου χρονικού διαστήματος. Περιγράφει τη μακροπρόθεσμη αύξηση, μείωση ή σταθερότητα των τιμών και παρατηρήσεων σε μια χρονοσειρά, ανεξάρτητα από τις πρόσκαιρες διακυμάνσεις που προκαλούνται από την εποχικότητα. Το πρότυπο της τάσης στα δεδομένα μπορεί να επηρεάζεται από οικονομικούς ή δημογραφικούς παράγοντες. Η τάση της θερμοκρασίας είναι ανοδική ανά τα έτη λόγω της κλιματικής αλλαγής.

- **Ακανόνιστες διακυμάνσεις**: Οι διακυμάνσεις που αντιπροσωπεύουν το "θόρυβο" που υπάρχει στα δεδομένα και δεν μπορούν να εξηγηθούν από παράγοντες όπως η εποχικότητα και η τάση. Αυτές οι τυχαίες διακυμάνσεις οφείλονται σε μη προβλέψιμους παράγοντες, τυχαία και εκτός ελέγχου γεγονότα.

Μερικές εφαρμογές της ανάλυσης χρονοσειρών μπορεί να είναι :

- **Ταξινόμηση**: Αντιστοίχιση μιας χρονοσειράς σε μία ή πολλές προκαθορισμένες κατηγορίες. Για παράδειγμα, δεδομένης της επισκεψημότητας μιας σελίδας, ταξινόμηση του επισκέπτη ως άνθρωπο ή bot.

- **Ανίχνευση συμβάντος**: Αναγνώριση της εμφάνισης ενός συγκεκριμένου αναμενόμενου γεγονότος εντός μιας συνεχούς ροής δεδομένων. Για παράδειγμα η "ανίχνευση λέξεων-

κλειδιών, " όπου ένα μοντέλο παρακολουθεί μια ηχητική ροή και ανιχνεύει φωνητικές εκφράσεις όπως "Ok Google" ή "Hey Alexa."

- **Ανίχνευση ανωμαλιών:** Ανίχνευση ασυνήθιστων συμβάντων εντός μιας συνεχούς ροής δεδομένων. Για παράδειγμα, στην καταγραφή κίνησης των αυτοκινήτων σε ένα οδικό δίκτυο, ανωμαλίες μπορεί να είναι κινήσεις που είναι εκτός του συνηθισμένου προτύπου, όπως πρόσκρουση ή κίνηση σε αντίθετη κατεύθυνση.

- **Πρόβλεψη χρονοσειρών**: Η πιο σημαντική και συχνή εφαρμογή. Πρόβλεψη του επόμενου βήματος στη χρονοσειρά. Για παράδειγμα η πρόβλεψη του καιρού τις επόμενες ημέρες ή πρόβλεψη του ηλεκτρικού φορτίου την επόμενη ώρα/μέρα για την αποτελεσματική αντιμετώπιση της ζήτησης.

Λόγω της φύσης τους η πρόβλεψη χρονοσειρών μπορεί να είναι απαιτητική καθώς τα προβλήματα χρονοσειρών προσθέτουν την πολυπλοκότητα της χρονικής διάταξης ή της χρονικής εξάρτησης μεταξύ των παρατηρήσεων. Επομένως απαιτείται εξειδικευμένη διαχείριση των δεδομένων κατά την εφαρμογή και αξιολόγηση των μοντέλων πρόβλεψης. Οι πρόσθετες πληροφορίες όπως τάση και εποχικότητα που μπορούν να αξιοποιηθούν για τη βελτίωση της ακρίβειας ενός μοντέλου.

Η Βαθιά Μάθηση (Deep Learning) έχει σημαντικά πλεονεκτήματα έναντι των παραδοσιακών προσεγγίσεων μηχανικής μάθησης και στατιστικής καθώς δεν απαιτεί εξειδίκευση στον τομέα της στατιστικής, υποστηρίζει ελλιπή/μη διαθέσιμα δεδομένα, πολλαπλές μεταβλητές ως είσοδο και περίπλοκες μη γραμμικές σχέσεις.[B1, 280-281], [B2]

5. Βραχυπρόθεσμη πρόβλεψη ηλεκτρικού φορτίου

Η βραχυπρόθεσμη πρόβλεψη του ηλεκτρικού φορτίου επικεντρώνεται στην πρόβλεψη των μοτίβων και των διακυμάνσεων της ζήτησης σε ένα σχετικά σύντομο χρονικό ορίζοντα (έως 24 ώρες). Στόχος είναι να αντιμετωπιστεί η ζήτηση ηλεκτρικής ενέργειας, συνήθως για ένα χρονικό ορίζοντα που κυμαίνεται από λίγα λεπτά έως αρκετές ημέρες στο μέλλον με ακριβείς προβλέψεις προκειμένου να υποστηριχθεί η λειτουργία του δικτύου .

Ένα μοντέλο που πραγματοποιεί ακριβείς προβλέψεις μπορεί να χρησιμοποιηθεί σε διάφορες περιπτώσεις

Ορθή λειτουργία του Δικτύου: Οι προβλέψεις είναι κρίσιμες για τα συστήματα του δικτύου ώστε να διατηρήσουν την ισορροπία μεταξύ προσφοράς και ζήτησης σε πραγματικό χρόνο. Οι ακριβείς προβλέψεις βοηθούν στον προγραμματισμό της παραγωγής ενέργειας, τη διαχείριση της σταθερότητας του δικτύου και την αποφυγή αποκλεισμών ή υπερφορτώσεων.[Δ17]

Ενσωμάτωση Ανανεώσιμων Πηγών Ενέργειας: Οι προβλέψεις είναι απαραίτητες κατά την ενσωμάτωση αστάθμητων ανανεώσιμων πηγών ενέργειας, όπως οι αιολικοί και ηλιακοί πόροι, στο δίκτυο. Τα συστήματα του δικτύου πρέπει να προβλέπουν ακριβώς τη ζήτηση για να διασφαλίσουν συνολικά μια σταθερή προσφορά ενέργειας.

Αγορά Ενέργειας: Οι έμποροι στην αγορά ενέργειας βασίζονται στις προβλέψεις της ζήτησης για να λαμβάνουν ενημερωμένες αποφάσεις σχετικά με την αγορά και πώληση ηλεκτρικής ενέργειας στις αγορές σε πραγματικό χρόνο.

Παράγοντες που επηρεάζουν το φορτίο:

Καιρικές συνθήκες: Οι καιρικές συνθήκες όπως είναι λογικό επηρεάζουν άμεσα τη ζήτηση ηλεκτρικής ενέργειας. Τους κρύους και ζεστούς μήνες η ζήτηση είναι πολύ αυξημένη αφού χρειάζονται επιπλέον τεχνητά μέσα (κλιματισμός, σόμπες κλπ) για να διατηρηθεί μια ικανοποιητική θερμοκρασία σε εσωτερικούς χώρους. Αντίθετα τους μήνες της άνοιξης και του φθινοπώρου που η θερμοκρασία είναι πιο ουδέτερη η ζήτηση περιορίζεται.

Οικονομικές συνθήκες: Η ζήτηση ηλεκτρικής ενέργειας είναι ανάλογη και με την οικονομία ειδικά όταν υπάρχει σημαντική εγχώρια βιομηχανική δραστηριότητα. Σε περιόδους οικονομικής ύφεσης η ζήτηση αυξάνεται και αντίστροφα.

Δημογραφικές συνθήκες: Η αύξηση του πληθυσμού σε μια περιοχή αυξάνει τη ζήτηση για ηλεκτρική ενέργεια, καθώς υπάρχουν περισσότεροι καταναλωτές.

Τεχνολογικές εξελίξεις: Οι τεχνολογικές εξελίξεις έχουν επίσης σημασία στην ζήτηση ηλεκτρικής ενέργειας ειδικότερα πλέον που τα ηλεκτρικά αυτοκίνητα όλο και πληθαίνουν.

6. Εργαλεία και βιβλιοθήκες

Tensorflow

Το Tensorflow είναι ένα πλαίσιο βιβλιοθηκών ανοιχτού κώδικα βασισμένο σε γλώσσα Python που δημιουργήθηκε από την Google και ενδείκνυται για την ανάπτυξη εφαρμογών Μηχανικής Μάθησης. Εκτός της Python είναι διαθέσιμο για ανάπτυξη σε C++, Javascript, Java. Εκτελείται σε CPU, GPU αλλά και σε πιο εξειδικευμένο υλικό, TPU (Tensor Process Unit). Υποστηρίζεται από Λειτουργικά Συστήματα Linux, Windows, macOS, Android και iOS.

Η ευέλικτη αρχιτεκτονική του βασίζεται σε υπολογιστικά γραφήματα. Αυτό σημαίνει ότι αντί να εκτελεί αμέσως τους υπολογισμούς που δίνονται, το TensorFlow δημιουργεί ένα γράφημα που απεικονίζει τις λειτουργίες και τις διαδικασίες που πρέπει να εκτελεστούν. Αυτός το γράφημα αντιπροσωπεύει τη σειρά των υπολογιστικών βημάτων που απαιτούνται για να φτάσουμε στο επιθυμητό αποτέλεσμα.

Μερικά σημαντικά χαρακτηριστικά του Tensorflow είναι:

- Συναρτήσεις Σφάλματος (Loss Functions): Περιέχει μια συλλογή συναρτήσεων για τον υπολογισμό του σφάλματος μεταξύ των προβλέψεων και των πραγματικών τιμών όπως την Mean Squared Error (MSE).

- Μέτρα (Metrics): Περιέχει μια συλλογή συναρτήσεων για την αξιολόγησης της απόδοσης του μοντέλου σε καινούρια δεδομένα όπως Accuracy, Precision, Root Mean Squared Error (RMSE), Mean Absolute Error (MAE) κ.α.

- Αλγόριθμοι Βελτιστοποίησης (Opmizers): Περιέχει μια συλλογή αλγορίθμων για την αποτελεσματική ενημέρωση των βαρών οι οποίοι είναι κυρίως βασισμένοι στη λογική του Gradient Descent όπως ο SGD, Adam κ.α.

Ιστοσελίδα: https://www.tensorflow.org/

Keras

Το Keras είναι ένα πλαίσιο βιβλιοθηκών ανοιχτού κώδικα που προσφέρει εργαλεία υψηλού επιπέδου για τη δημιουργία και την εκπαίδευση νευρωνικών δικτύων οποιουδήποτε τύπου. Αρχικά δημιουργήθηκε από τον François Chollet και αργότερα προσαρτήθηκε στο Tensorflow.

Προσφέρει ένα πολύ φιλικό περιβάλλον ανάπτυξης και είναι ιδανικό για αρχάριους έως πολύ έμπειρους προγραμματιστές αφού έχει τη δυνατότητα δημιουργίας εξειδικευμένων και προσαρμοσμένων νευρωνικών δικτύων εκτός των διαδεδομένων.

Ιστοσελίδα: https://keras.io/

Keras Tuner

Το Keras Tuner είναι μια βιβλιοθήκη του Keras που χρησιμοποιείται για την βελτιστοποίηση των υπερπαραμέτρων του εκάστοτε μοντέλου. Χρησιμοποιεί αλγόριθμους αναζήτησης για να δοκιμάσει διαφορετικούς συνδυασμούς από layers, units (neurons), learning rates, batch sizes, activation funtions.

Ο αλγόριθμος αναζήτησης μπορεί να είναι GridSearch, RandomSearch, BayesianOptimazation

Ιστοσελίδα: https://keras.io/keras_tuner/

Google Colab

Το Google Colab είναι μια δωρεάν υπηρεσία για την ανάπτυξη και εκτέλεση κώδικα που δεν απαιτεί καμία εγκατάσταση τοπικά αφού βρίσκεται εξολοκλήρου σε cloud. Είναι μια ιστοσελίδα που παρέχει απευθείας τη δυνατότητα δημιουργίας μοντέλων με Keras και την εκτέλεση τους δωρεάν σε CPU, GPU και TPU της Google. Οι περισσότερες βιβλιοθήκες είναι ήδη προ εγκατεστημένες και χρειάζεται μόνο η προσάρτηση τους στον κώδικα από το χρήστη.

Ιστοσελίδα: https://colab.research.google.com

Numpy

Το NumPy (Numerical Python) είναι μια ανοιχτού κώδικα βιβλιοθήκη προγραμματισμού σε Python που παρέχει υψηλής απόδοσης υποστήριξη για πίνακες πολλών διαστάσεων, καθώς και μια σειρά από συναρτήσεις και εργαλεία για αριθμητικούς υπολογισμούς. Η βιβλιοθήκη αυτή είναι εξαιρετικά δημοφιλής σε πολλούς επιστημονικούς κλάδους, συμπεριλαμβανομένης της μηχανικής μάθησης, καθώς επιτρέπει την εκτέλεση γρήγορων και αποτελεσματικών υπολογισμών.

Το NumPy είναι το θεμέλιο για πολλές άλλες επιστημονικές και τεχνικές βιβλιοθήκες σε Python, όπως οι Pandas και Matplotlib.

Ιστοσελίδα: https://numpy.org/

Pandas

Το Pandas είναι μια ανοιχτού κώδικα βιβλιοθήκη προγραμματισμού σε Python που χρησιμοποιείται κυρίως για την ανάλυση και τον χειρισμό δεδομένων. Η βιβλιοθήκη αυτή παρέχει εργαλεία και δομές δεδομένων που επιτρέπουν την εύκολη εισαγωγή, ανάλυση και επεξεργασία δεδομένων από αρχεία CSV, Excel κλπ. Επιπλέον έχει τη δυνατότητα πολύ φιλικής προς το χρήστη οπτικοποίησης των δεδομένων.

Ιστοσελίδα: https://pandas.pydata.org/

MatPlotlib

Το Matplotlib είναι μια ανοιχτού κώδικα βιβλιοθήκη προγραμματισμού σε Python που χρησιμοποιείται για τη δημιουργία γραφικών αναπαραστάσεων, όπως γραφήματα, διαγράμματα, πίνακες και άλλα. Η βιβλιοθήκη Matplotlib παρέχει ένα ευέλικτο περιβάλλον για την οπτικοποίηση δεδομένων και τη δημιουργία γραφικών παραστάσεων με διάφορες επιλογές προσαρμογής και στυλ καθώς και την αποθήκευση αυτών σε διάφορους τύπους αρχείων εικόνας.

Ιστοσελίδα: https://matplotlib.org/

7. Δεδομένα

Τα δεδομένα που χρησιμοποιήθηκαν για την βραχυπρόθεσμη πρόβλεψη ηλεκτρικού φορτίου είναι από την επίσημη ιστοσελίδα του Ανεξάρτητου Διαχειριστή Μεταφοράς Ηλεκτρικής Ενέργειας (ΑΔΜΗΕ) και αφορούν τη χρονική περίοδο 2013-2016.

Τα έτη 2013 - 2015 χρησιμοποιήθηκαν ως σύνολο εκπαίδευσης και το έτος 2016 ως σύνολο δοκιμής

Η συλλογή και προεπεξεργασία των δεδομένων έγινε από τον υποψήφιο Διδάκτορα Γ. Κανδηλογιαννάκη. Πραγματοποιήθηκε εκτενής προεπεξεργασία και εμπλουτισμός των συνόλων δεδομένων με συνολικά 15 τιμές καθυστέρησης ανά ώρα, δημιουργώντας μια συλλογή δυνητικών εισόδων συστήματος. Συμπεριλαμβάνεται επιπλέον η 16η στήλη που έχει την τιμή του πραγματικού φορτίου:

Power Load															
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
d-1 / h-2	d-1 / h-1	d-1 / h	d-1 / h+1	d-1 / h+2	d-2 / h-2	d-2 / h-1	d-2 / h	d-2 / h+1	d-2 / h+2	d-7 / h-2	d-7 / h-1	d-7 / h	d-7 / h+1	d-7 / h+2	d / h

Πίνακας 7.1: Τιμές καθυστέρησης δεδομένων

όπου **h**: η εκάστοτε ώρα και **d**: η ημέρα που ανήκει η **h**.

Επομένως η αρχική συλλογή δεδομένων εκπαίδευσης με 1095 στιγμιότυπα (3 x 365, όπου η κάθε γραμμή δεδομένων αντιστοιχεί σε μία ημέρα) μετετράπη σε μία συλλογή 26088 στιγμιότυπων (δηλ. 26280 - 192) όπου η κάθε γραμμή δεδομένων αντιστοιχεί σε μία ώρα και η συλλογή δεδομένων δοκιμής από 366 σε 8592 στιγμιότυπα (δηλ. 8784 - 192) αντιστοίχως.

Οι πρώτες 192 ώρες (8 ημέρες) εξαιρούνται από τα παραπάνω σύνολα διότι χρησιμοποιούνται ως τιμές καθυστέρησης στην αρχή του εκάστοτε συνόλου.

Για τις ώρες που δεν έχει καταγραφεί τιμή, το πεδίο έχει συμπληρωθεί με τη μέση τιμή της προηγούμενης και επόμενης ημέρας την ίδια ώρα.

Παρακάτω παρουσιάζεται η δομή των δεδομένων:

```python
import numpy as np
import matplotlib as plt
import pandas as pd

train_data = "ADMHE_System_Load_2013-01-01_2015-12-31_training_15in_1out_2013-2015trn_2016tst.dat"
test_data = "ADMHE_System_Load_2016-01-01_2016-12-31_testing_15in_1out_2013-2015trn_2016tst.dat"

train = np.loadtxt(train_data)
test = np.loadtxt(test_data)
pd.DataFrame(train)
# pd.DataFrame(test)
```

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	6615.0	6141.0	5525.0	5040.0	5011.0	6356.0	6033.0	5444.0	4846.0	4834.0	5718.0	5509.0	5029.0	4593.0	4598.0	5925.0
1	6141.0	5525.0	5040.0	5011.0	4817.0	6033.0	5444.0	4846.0	4834.0	4658.0	5509.0	5029.0	4593.0	4598.0	4374.0	5427.0
2	5525.0	5040.0	5011.0	4817.0	4685.0	5444.0	4846.0	4834.0	4658.0	4510.0	5029.0	4593.0	4598.0	4374.0	4246.0	5387.0
3	5040.0	5011.0	4817.0	4685.0	4784.0	4846.0	4834.0	4658.0	4510.0	4554.0	4593.0	4598.0	4374.0	4246.0	4278.0	5207.0
4	5011.0	4817.0	4685.0	4784.0	5321.0	4834.0	4658.0	4510.0	4554.0	4988.0	4598.0	4374.0	4246.0	4278.0	4591.0	5146.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
26083	7571.0	7888.0	7889.0	7711.0	7291.0	6611.0	7184.0	7242.0	7145.0	6710.0	6996.0	7592.0	7669.0	7492.0	6950.0	8937.0
26084	7888.0	7889.0	7711.0	7291.0	6690.0	7184.0	7242.0	7145.0	6710.0	6149.0	7592.0	7669.0	7492.0	6950.0	6239.0	8527.0
26085	7889.0	7711.0	7291.0	6690.0	6179.0	7242.0	7145.0	6710.0	6149.0	5701.0	7669.0	7492.0	6950.0	6239.0	5782.0	7695.0
26086	7711.0	7291.0	6690.0	6179.0	5501.0	7145.0	6710.0	6149.0	5701.0	5089.0	7492.0	6950.0	6239.0	5782.0	5263.0	6712.0
26087	7291.0	6690.0	6179.0	5501.0	4959.0	6710.0	6149.0	5701.0	5089.0	4605.0	6950.0	6239.0	5782.0	5263.0	4833.0	6211.0

26088 rows × 16 columns

Εικόνα 7.1: Δεδομένα εκπαίδευσης - Μορφή

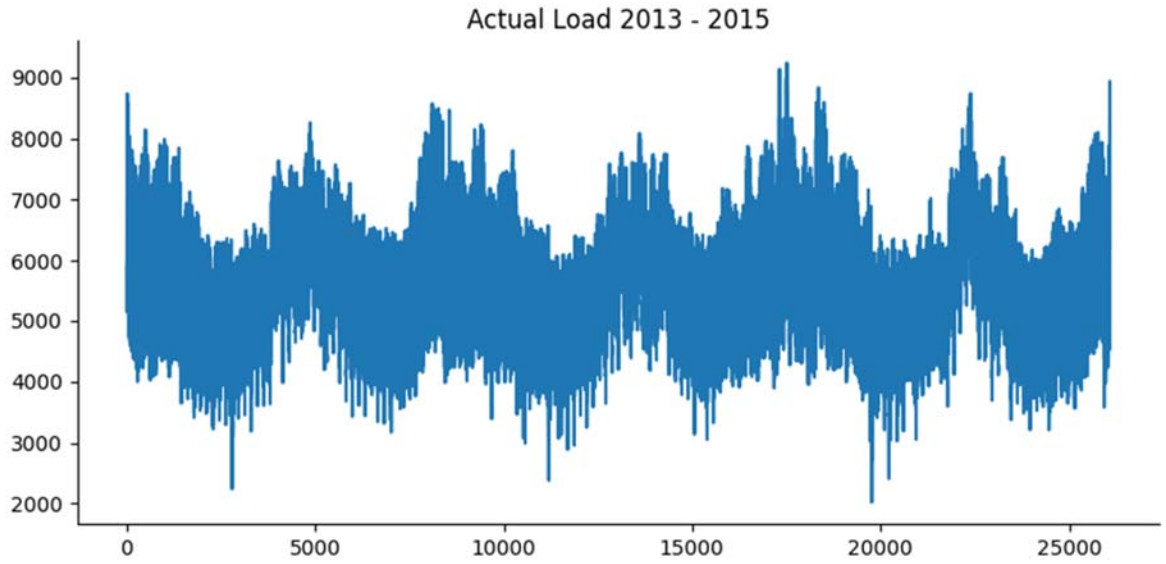
```python
import numpy as np
import matplotlib as plt
import pandas as pd

train_data = "ADMHE_System_Load_2013-01-01_2015-12-31_training_15in_1out_2013-2015trn_2016tst.dat"
test_data = "ADMHE_System_Load_2016-01-01_2016-12-31_testing_15in_1out_2013-2015trn_2016tst.dat"

train = np.loadtxt(train_data)
test = np.loadtxt(test_data)
# pd.DataFrame(train)
pd.DataFrame(test)
```

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	6077.0	5629.0	5084.0	4611.0	4582.0	5540.0	5216.0	4764.0	4352.0	4317.0	6333.0	6057.0	5585.0	5120.0	5043.0	5172.0
1	5629.0	5084.0	4611.0	4582.0	4405.0	5216.0	4764.0	4352.0	4317.0	4119.0	6057.0	5585.0	5120.0	5043.0	4799.0	4756.0
2	5084.0	4611.0	4582.0	4405.0	4300.0	4764.0	4352.0	4317.0	4119.0	4026.0	5585.0	5120.0	5043.0	4799.0	4626.0	4680.0
3	4611.0	4582.0	4405.0	4300.0	4366.0	4352.0	4317.0	4119.0	4026.0	4127.0	5120.0	5043.0	4799.0	4626.0	4640.0	4482.0
4	4582.0	4405.0	4300.0	4366.0	4788.0	4317.0	4119.0	4026.0	4127.0	4498.0	5043.0	4799.0	4626.0	4640.0	4790.0	4334.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
8587	7819.0	8299.0	8302.0	7941.0	7269.0	7715.0	8061.0	8040.0	7837.0	7332.0	7159.0	7837.0	7901.0	7584.0	6957.0	8971.0
8588	8299.0	8302.0	7941.0	7269.0	6776.0	8061.0	8040.0	7837.0	7332.0	6763.0	7837.0	7901.0	7584.0	6957.0	6348.0	8563.0
8589	8302.0	7941.0	7269.0	6776.0	6395.0	8040.0	7837.0	7332.0	6763.0	6261.0	7901.0	7584.0	6957.0	6348.0	5947.0	7705.0
8590	7941.0	7269.0	6776.0	6395.0	5779.0	7837.0	7332.0	6763.0	6261.0	5646.0	7584.0	6957.0	6348.0	5947.0	5414.0	6759.0
8591	7269.0	6776.0	6395.0	5779.0	5232.0	7332.0	6763.0	6261.0	5646.0	5072.0	6957.0	6348.0	5947.0	5414.0	4980.0	6320.0

8592 rows × 16 columns

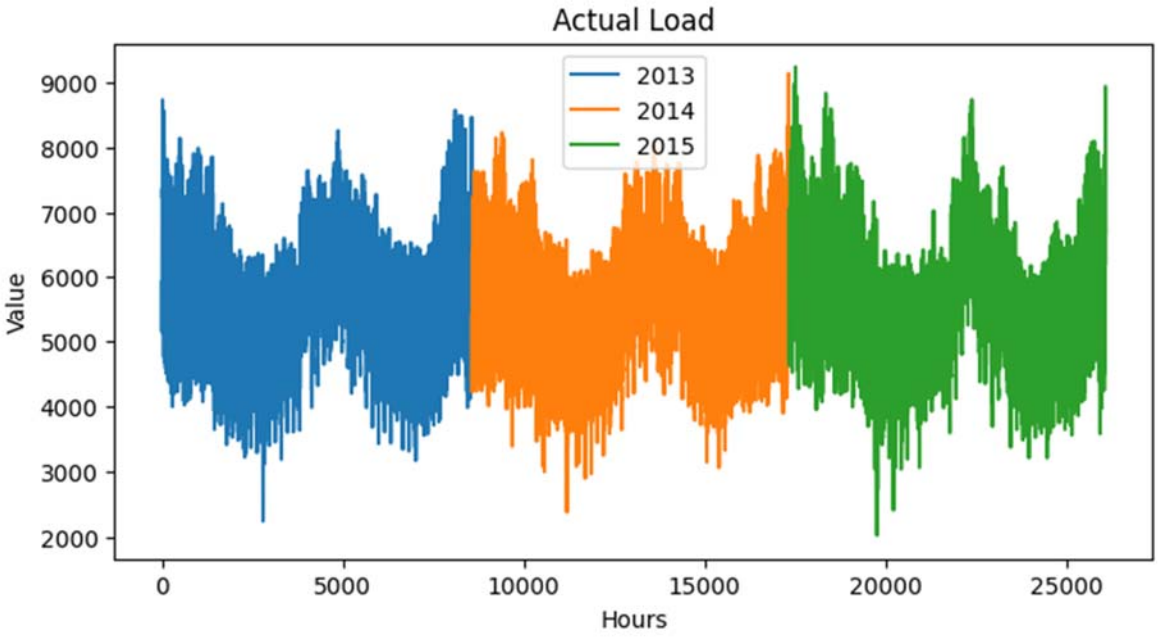
Εικόνα 7.2: Δεδομένα δοκιμής - Μορφή

Επομένως σχεδιάζοντας την τελευταία στήλη μπορούμε να εξάγουμε διάφορα συμπεράσματα:



Εικόνα 7.3: Δεδομένα εκπαίδευσης - Γραφική αναπαράσταση

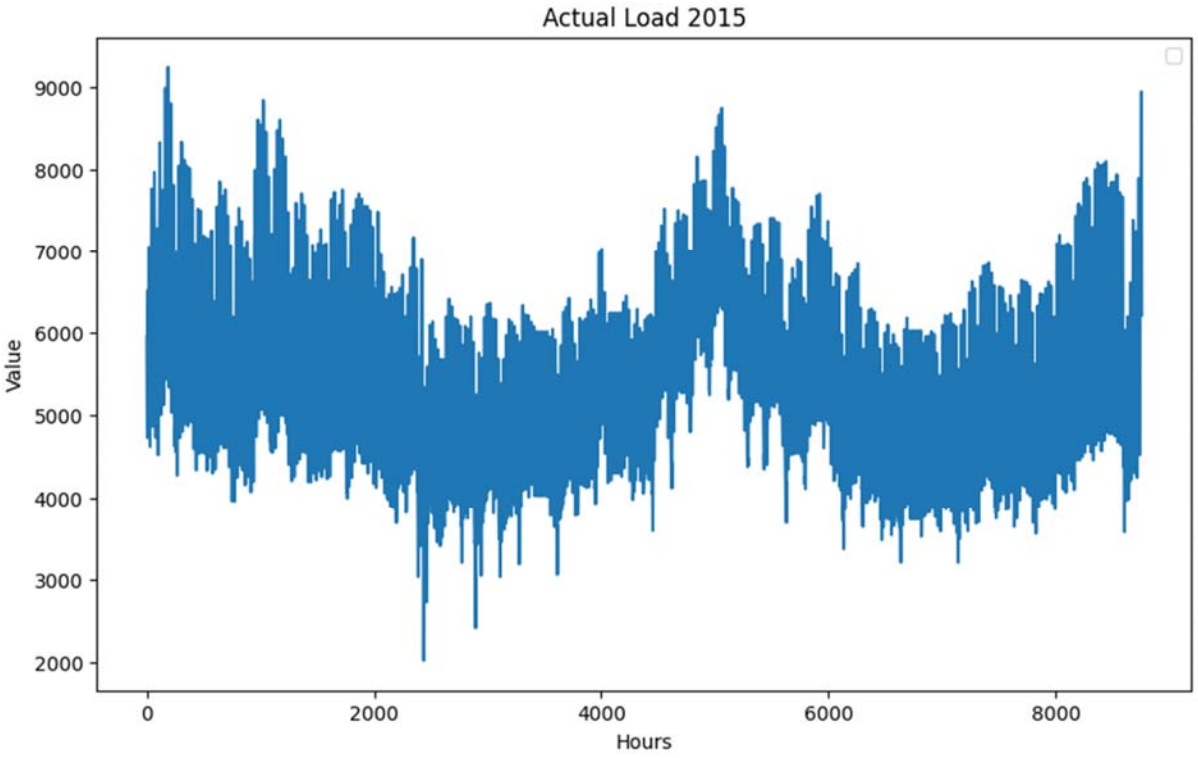
Με μια πρώτη ματιά η περιοδικότητα στα δεδομένα είναι εμφανής:



Εικόνα 7.4: Δεδομένα εκπαίδευσης - Γραφική αναπαράσταση ανά έτος

Το κάθε έτος έχει 8.760 ώρες (365 x 24), και είναι εμφανές το επαναλαμβανόμενο μοτίβο ανά αυτή την περίοδο.

Εστιάζοντας στο τελευταίο έτος μπορούμε να εξάγουμε συμπεράσματα για τις εποχές:



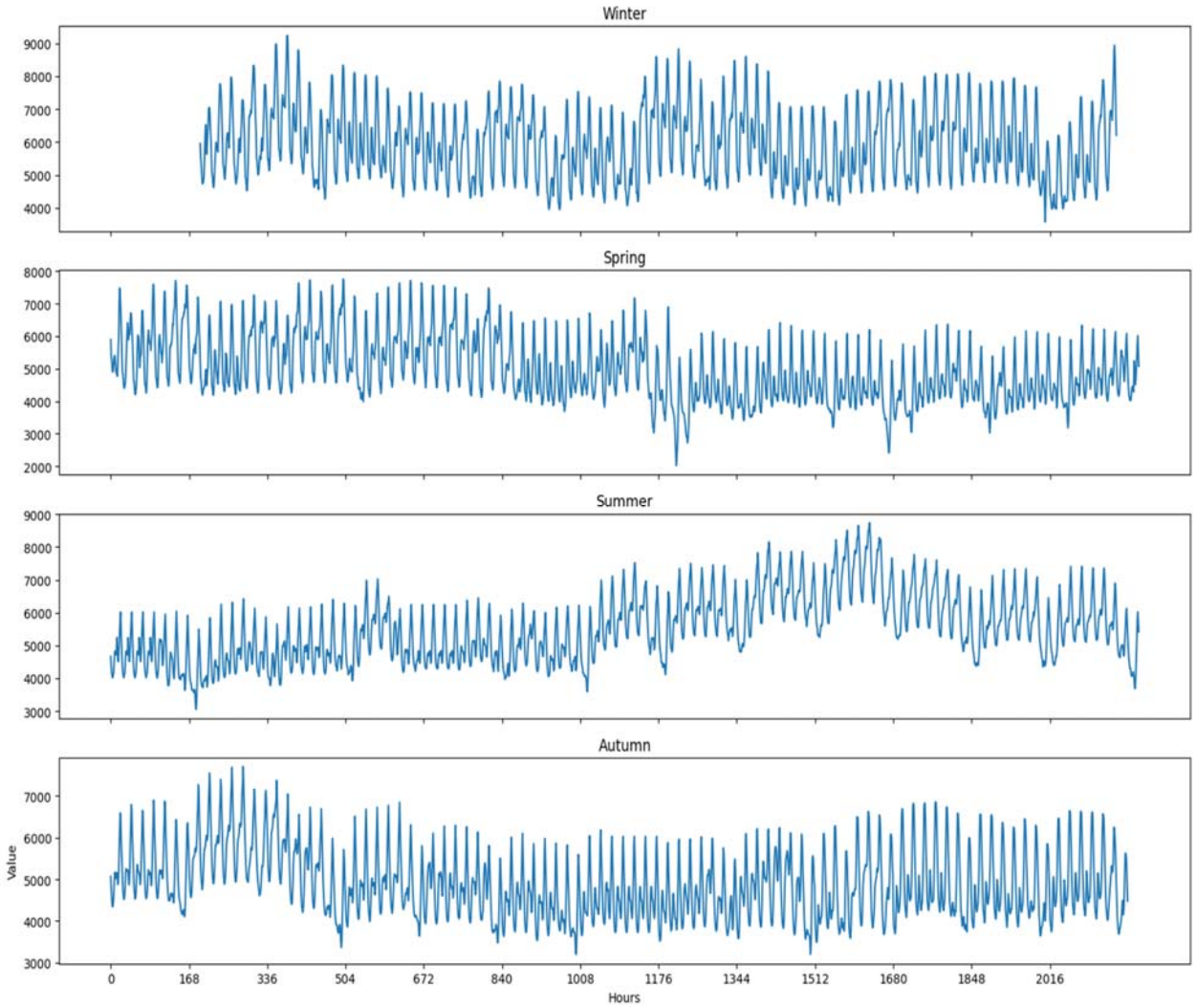
Εικόνα 7.5: Δεδομένα εκπαίδευσης - Γραφική αναπαράσταση έτους 2015

Οι πρώτες περίπου 1500 ώρες αντιστοιχούν στους χειμερινούς μήνες Ιανουάριο και Φεβρουάριο όπου η ζήτηση είναι αυξημένη.

Η ζήτηση μειώνεται αισθητά από τις 2000 έως τις 4000 ώρες όπου πρόκειται για την ανοιξιάτικη περίοδο και τις αρχές του καλοκαιριού.

Εν συνεχεία η ζήτηση ανεβαίνει ξανά, και φτάνει το μέγιστο στις 5000 ώρες περίπου, όπου πρόκειται για τα μέσα του καλοκαιριού, Ιούλιο - Αύγουστο. Η μέγιστη τιμή του καλοκαιριού είναι αισθητά μικρότερη από αυτή του χειμώνα κατά 500 μονάδες (MW)

Τέλος η ζήτηση μειώνεται ομαλά για το φθινόπωρο και ανεβαίνει ξανά για το χειμώνα φτάνοντας τις μέγιστες τιμές.

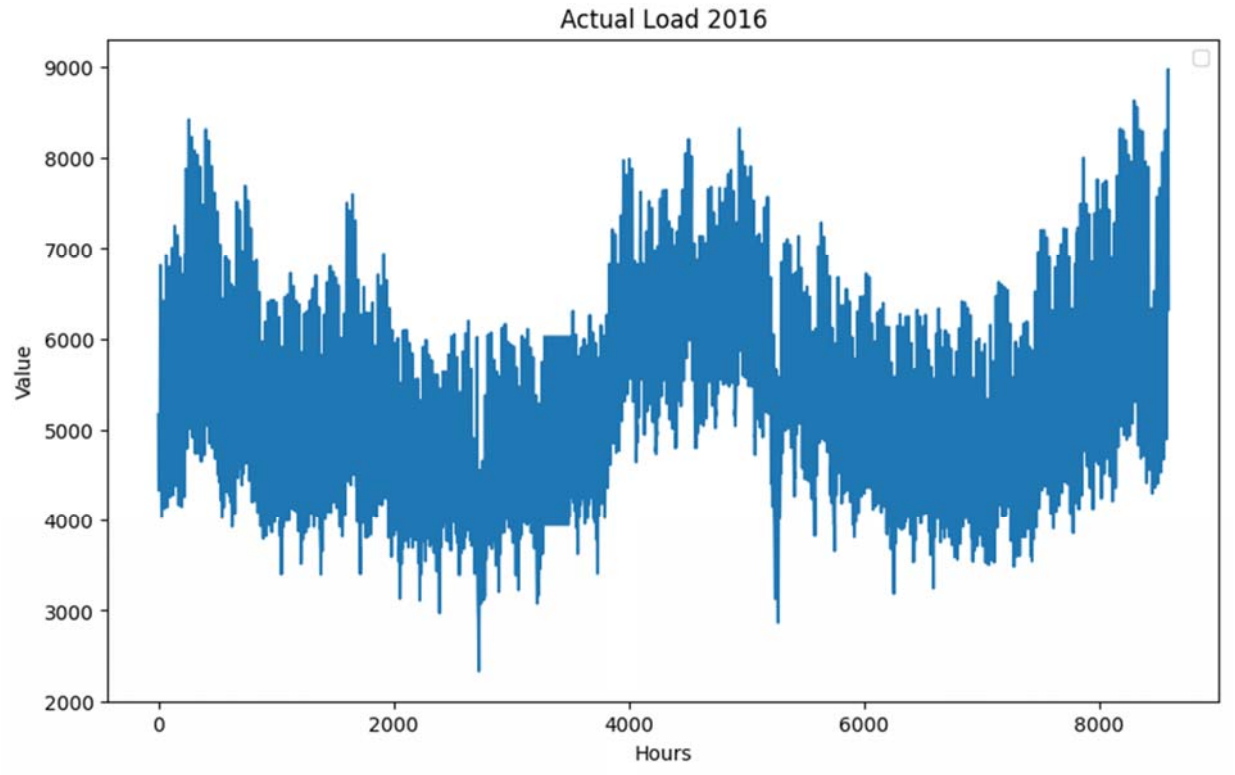


Εικόνα 7.6: Δεδομένα εκπαίδευσης - Γραφική αναπαράσταση εποχών 2015

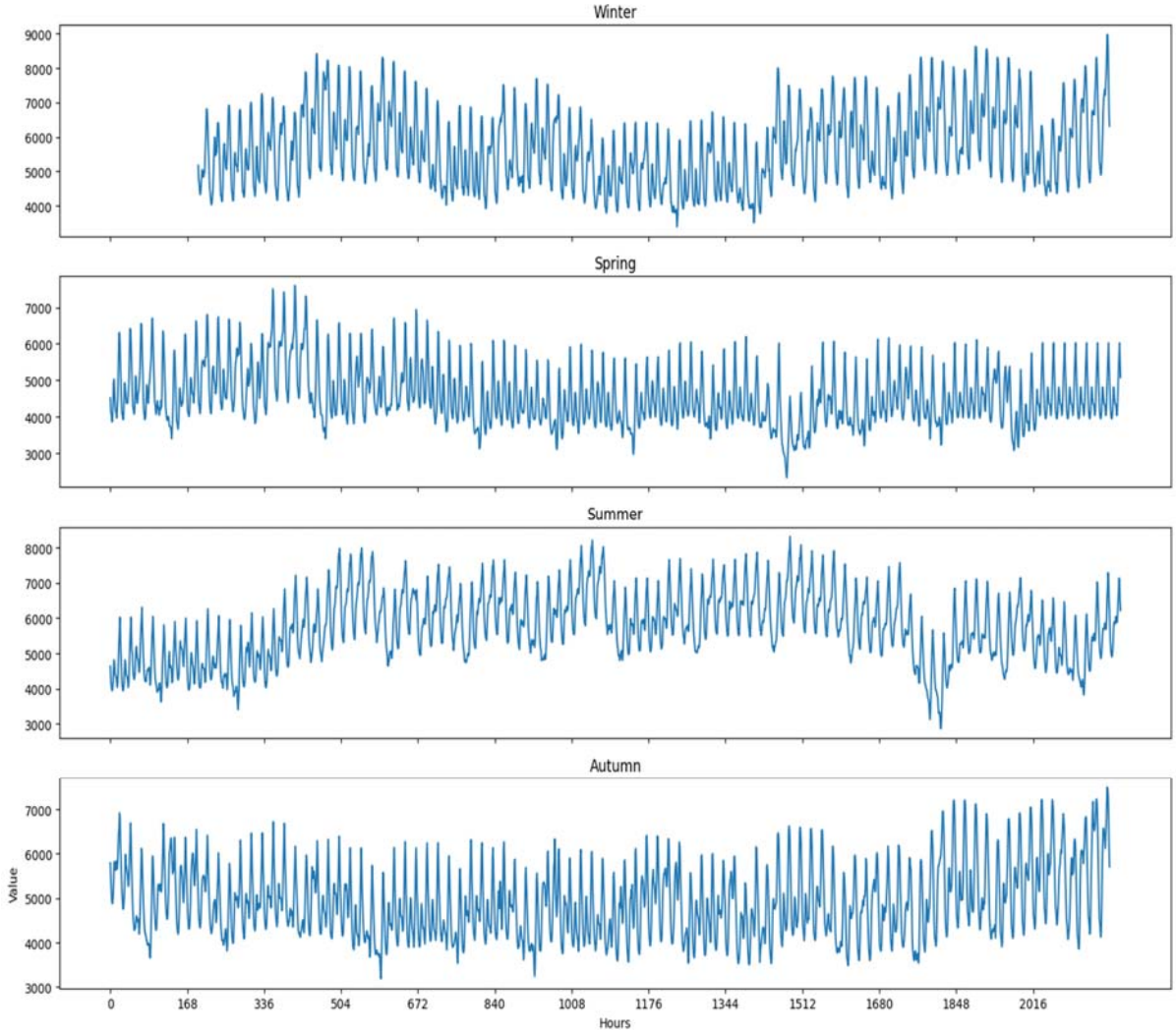
	WINTER	**SPRING**	**SUMMER**	**AUTUMN**
MAX	9237.0	7753.0	8737.0	7700.0
MIN	3576.0	2021.0	3056.0	3203.0

Πίνακας 7.2: Μέγιστα και ελάχιστα συνόλου εκπαίδευσης, έτος 2015

Το ίδιο πρότυπο εμφανίζεται και στα δεδομένα δοκιμής για το έτος 2016:



Εικόνα 7.7: Δεδομένα δοκιμής - Γραφική αναπαράσταση

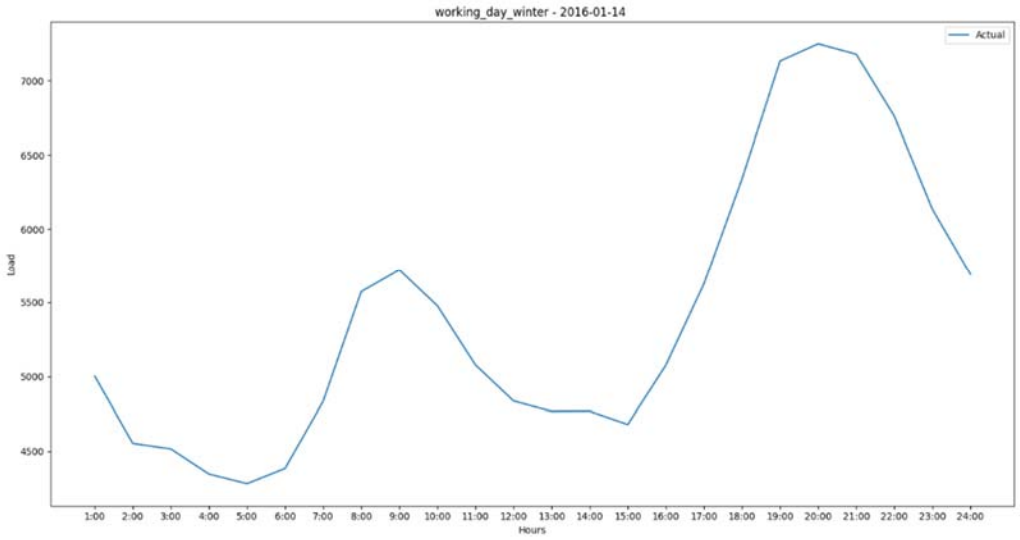


Εικόνα 7.8: Δεδομένα δοκιμής - Γραφική αναπαράσταση εποχών

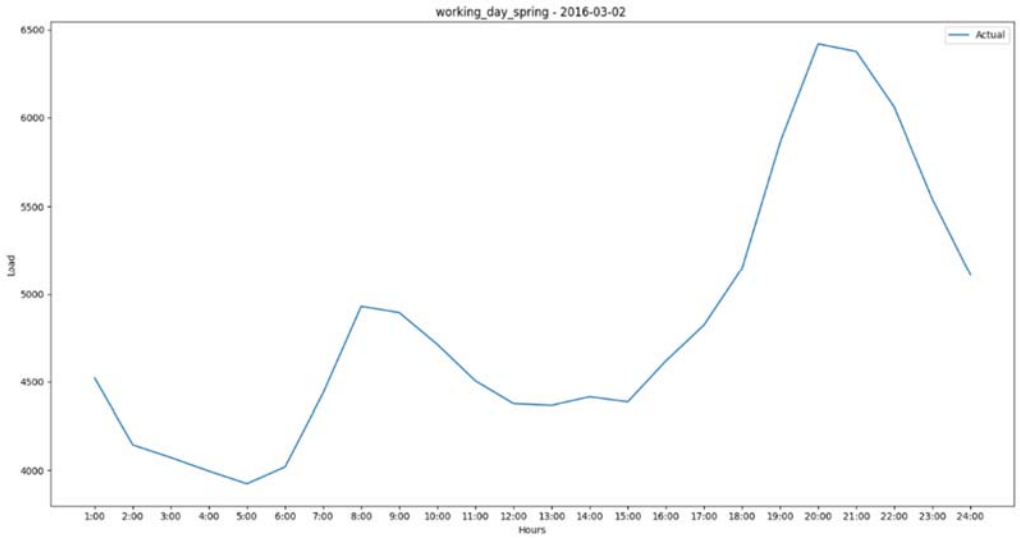
	WINTER	SPRING	SUMMER	AUTUMN
MAX	8971.0	7598.0	8316.0	7494.0
MIN	3399.0	2331.0	2867.0	3186.0

Πίνακας 7.3: Μέγιστα και ελάχιστα συνόλου δοκιμής

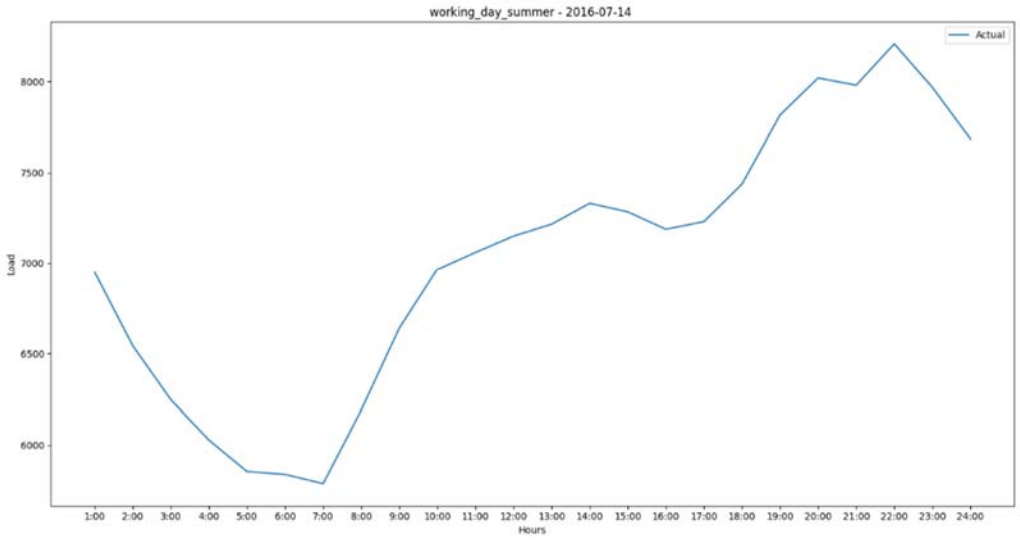
Στη συνέχεια σχεδιάζοντας τυχαίες ημερομηνίες από κάθε εποχή του έτους δοκιμής μπορούμε να εξάγουμε συμπεράσματα για τα ημερήσια πρότυπα που θα πρέπει να αντιληφθεί το εκάστοτε μοντέλο και για τις προκλήσεις που θα παρουσιαστούν:



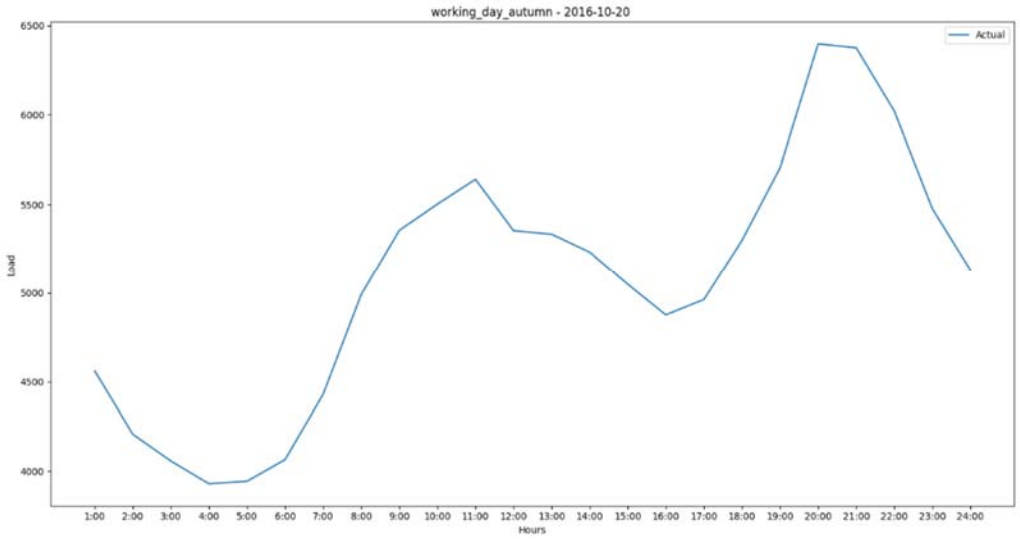
Εικόνα 7.9: Δεδομένα δοκιμής – χειμερινή Ημέρα



Εικόνα 7.10: Δεδομένα δοκιμής – εαρινή Ημέρα



Εικόνα 7.11: Δεδομένα δοκιμής – θερινή Ημέρα



Εικόνα 7.12: Δεδομένα δοκιμής – χειμερινή Ημέρα

Είναι προφανές ότι τις καθημερινές, ανεξάρτητα εποχής, οι απογευματινές έως βραδινές ώρες παρουσιάζουν την μεγαλύτερη κατανάλωση. Αυτό συμβαίνει διότι αυτές τις ώρες είναι πιθανότερο να βρίσκεται κάποιος σπίτι και άρα να χρησιμοποιεί ηλεκτρικές συσκευές με υψηλή κατανάλωση όπως κλιματιστικό, πλυντήριο, κουζίνα κλπ.

8. Αρχιτεκτονικές

Όλα τα μοντέλα που δημιουργήθηκαν έχουν στόχο τη Βραχυπρόθεσμη Πρόβλεψη Ηλεκτρικού Φορτίου της επόμενης ώρας. Αρχικά δοκιμάστηκε η απλή αρχιτεκτονική πλήρως συνδεδεμένων νευρωνικών δικτύων Dense και στη συνέχεια πιο περίπλοκα αναδρομικά νευρωνικά δίκτυα SimpleRNN, LSTM, GRU. Για την παρακολούθηση της εκπαίδευσης και την αξιολόγηση των αποτελεσμάτων χρησιμοποιήθηκαν τα εξής:

Εκπαίδευση:

Για τον υπολογισμό του σφάλματος των μοντέλων κατά την εκπαίδευση χρησιμοποιήθηκε ως συνάρτηση απώλειας το Μέσο Τετραγωνικό Σφάλμα (Mean Squared Error – MSE) που εκφράζεται από τη σχέση:

$$MSE = \frac{1}{N}\sum_{h=1}^{N}(actual_h - predicted_h)^2$$

N: Το πλήθος των δεδομένων

actual: η πραγματική τιμή

predicted: η τιμή πρόβλεψης

Το Μέσο Τετραγωνικό Σφάλμα εμπεριέχεται στη βιβλιοθήκη του Tensorflow και η χρήση του είναι πολύ διαδεδομένη για εργασίες πρόβλεψης μεγεθών.

Επικύρωση:

Για την αξιολόγηση της απόδοσης κατά την εκπαίδευση χρησιμοποιήθηκε το 10% των δεδομένων εκπαίδευσης ως σύνολο επικύρωσης (validation set). Χρησιμοποιήθηκαν δύο μέτρα: η Ρίζα Μέσου Τετραγωνικού Σφάλματος (Root Mean Squared Error – RMSE) και το Μέσο Απόλυτο Ποσοστιαίο Σφάλμα (Mean Absolute Percentage Error – MAPE).

$$RMSE = \sqrt{\frac{1}{N}\sum_{h=1}^{N}(actual_h - predicted_h)^2}$$

και

$$MAPE = \frac{1}{N}\sum_{h=1}^{N}\frac{|actual_h - predicted_h|}{actual_h}\cdot 100$$

Τα δύο παραπάνω μέτρα εμπεριέχονται ήδη στη βιβλιοθήκη Tensorflow. Το αποτέλεσμα του RMSE είναι στην ίδια τάξη μεγέθους με τα δεδομένα και συνδυαστικά με το MAPE προσφέρει μια ευκρινή εικόνα για το μέγεθος των σφαλμάτων και τον προσανατολισμό της εκπαίδευσης.

Αξιολόγηση προβλέψεων

Για τον υπολογισμό της απόδοσης των προβλέψεων των μοντέλων στο σύνολο δοκιμής χρησιμοποιήθηκαν δύο μέτρα, η Ρίζα Μέσου Τετραγωνικού Σφάλματος (Root Mean Squared Error – RMSE) και το Μέσο Ποσοστιαίο Σφάλμα σε σχέση με την ημερήσια μέγιστη τιμή (Average Percentage Error – APE) όπου εκφράζεται από τη σχέση:

$$APE = \frac{1}{N}\sum_{d=1}^{N}\left[\frac{1}{24}\cdot\sum_{h=1}^{24}\frac{|actual_{d,h} - predicted_{d,h}|}{actual_{\max d}}\right]\cdot 100\%$$

N: Το πλήθος των ημερών στο εκάστοτε σύνολο δεδομένων

h: η ώρα

d: η ημέρα στην οποία ανήκει η h

$actual_{maxd:}$ η μέγιστη πραγματική τιμή για την ημέρα d

Το Μέσο Ποσοστιαίο Σφάλμα σε σχέση με την ημερήσια μέγιστη τιμή δε συμπεριλαμβάνεται στις έτοιμες βιβλιοθήκες όποτε δημιουργήθηκε συνάρτηση. Είναι διαφορετικό μέτρο από το MAPE και είναι κατάλληλο για τη διερεύνηση των βραχυπρόθεσμων προβλέψεων σε χρονικό διάστημα ημέρας.

Επιπλέον υπολογίστηκαν συμπληρωματικά και οι ώρες για όλο το έτος δοκιμής για τις οποίες η απόλυτη τιμή της διαφοράς πρόβλεψης από την πραγματική ήταν πάνω από 100, 200, 400, και 500 MW (forecast error duration curve).

9. Προσεγγίσεις

Για να προβλεφθεί η τελευταία στήλη των δεδομένων (επόμενη ώρα) για όλο το έτος δοκιμής χρησιμοποιήθηκαν δύο προσεγγίσεις κατά την εκπαίδευση.

Προσέγγιση 1: Συνδυασμοί τιμών καθυστέρησης

Χρησιμοποιήθηκαν διαφορετικοί συνδυασμοί από τις 15 τιμές καθυστέρησης ως είσοδος για την πρόβλεψη της τελευταίας στήλης.

Ονομασία συνδυασμού	Στήλες συνδυασμού	Περιγραφή συνδυασμού
one_day	(0, 1, 2, 3, 4)	Στήλες μια ημέρα πριν
two_days	(5, 6, 7, 8, 9)	Στήλες δύο ημέρες πριν
seven_days	(10, 11, 12, 13, 14)	Στήλες επτά ημέρες πριν
h_all	(2, 7, 12)	Στήλες ίδια ώρα, όλες οι ημέρες πριν
h+1_all	(3, 8, 13)	Στήλες μια ώρα μετά, όλες οι ημέρες πριν
h+2_all	(4, 9, 14)	Στήλες δύο ώρες μετά, όλες οι ημέρες πριν
h-1_all	(1, 6, 11)	Στήλες μια ώρα πριν, όλες οι ημέρες πριν
h-2_all	(0, 5, 10)	Στήλες δύο ώρες πριν, όλες οι ημέρες πριν
h_h+1_one_day	(2, 3)	Στήλες ίδια ώρα και μία ώρα μετά, μια ημέρα πριν
h_h+1_one_two_days	(2, 3, 7, 8)	Στήλες ίδια ώρα και μία ώρα μετά, μια και δυο ημέρες πριν
h_h+1_all	(2, 3, 7, 8, 12, 13)	Στήλες ίδια ώρα και μία ώρα μετά, όλες οι ημέρες πριν
h_h+1_h+2_one_day	(2, 3, 4)	Στήλες ίδια ώρα, μία και δύο ώρες μετά, μια ημέρα πριν
h_h+1_h+2_one_two_days	(2, 3, 4, 7, 8, 9)	Στήλες ίδια ώρα, μία και δύο ώρες μετά, μια και δύο ημέρες πριν
h_h+1_h+2_all	(2, 3, 4, 7, 8, 9, 12, 13, 14)	Στήλες ίδια ώρα, μία και δύο ώρες μετά, όλες οι ημέρες πριν
h-1_h_one_day	(1, 2)	Στήλες μια ώρα πριν και ίδια ώρα, μια ημέρα πριν
h-1_h_one_two_days	(1, 2, 6, 7)	Στήλες μια ώρα πριν και ίδια

		ώρα, μια και δύο ημέρες πριν
h-1_h_all	(1, 2, 6, 7, 11, 12)	Στήλες μια ώρα πριν και ίδια ώρα, όλες οι ημέρες πριν
h-2_h-1_h_one_day	(0, 1, 2)	Στήλες δύο ώρες πριν και μια ώρα πριν και ίδια ώρα, μια ημέρα πριν
h-2_h-1_h_two_days	(5, 6, 7)	Στήλες δύο ώρες πριν και μια ώρα πριν και ίδια ώρα, δύο ημέρες πριν
h-2_h-1_h_seven_days	(10, 11, 12)	Στήλες δύο ώρες πριν και μια ώρα πριν και ίδια ώρα, επτά ημέρες πριν
h-2_h-1_h_one_two_days	(0, 1, 2, 5, 6, 7)	Στήλες δύο ώρες πριν και μια ώρα πριν και ίδια ώρα, μία και δύο ημέρες πριν
h-2_h-1_h_all	(0, 1, 2, 5, 6, 7, 10, 11, 12)	Στήλες δύο ώρες πριν και μια ώρα πριν και ίδια ώρα, όλες οι ημέρες πριν
h-1_h_h+1_one_day	(1, 2, 3)	Στήλες μία ώρα πριν, ίδια ώρα και μία ώρα μετά, μια ημέρα πριν
h-1_h_h+1_one_two_days	(1, 2, 3, 6, 7, 8)	Στήλες μία ώρα πριν, ίδια ώρα και μία ώρα μετά, μια και δύο ημέρες πριν
all	(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14)	Όλες οι στήλες
doc1	(5, 6, 7, 8)	Στήλες δύο ώρες πριν, μία ώρα πριν, ίδια ώρα και μια ώρα μετά, δύο μέρες πριν
doc2	(5, 6, 8)	Στήλες δύο ώρες πριν, μια ώρα πριν και μια ώρα μετά, δύο μέρες πριν
doc3	(5, 8)	Στήλες δύο ώρες πριν και μια ώρα μετά, δύο μέρες πριν
doc4	(2, 12)	Στήλες ίδια ώρα, μία και επτά ημέρες πριν
doc5	(1, 2, 12)	Στήλες μια ώρα πριν και ίδια ώρα, μια ημέρα πριν, και ίδια ώρα, επτά ημέρες πριν
doc6	(2,)	Στήλη ίδια ώρα, μια ημέρα πριν

Πίνακας 9.1: Συνδυασμοί τιμών καθυστέρησης

Οι παραπάνω συνδυασμοί δοκιμάστηκαν σε όλα τα μοντέλα αρχικά με τις ίδιες υπερπαραμέτρους για πιο δίκαιη σύγκριση.

units	activation	optimizer	learning rate	batch size
50	relu	adam	0.001	24

Πίνακας 9.2: Υπερπαράμετροι για αρχικά μοντέλα

Στη συνέχεια ο καλύτερος συνδυασμός για το κάθε μοντέλο χρησιμοποιήθηκε για τη βελτιστοποίηση των υπερπαραμέτρων με τη χρήση της βιβλιοθήκης keras tuner. Οι υπερπαραμετροι που δοκιμάστηκαν παρουσιάζονται στον παρακάτω πίνακα:

layers	units	activation	optimizer	learning rate	batch size
1, 2, 3	25, 50, 100, 200	relu	adam	1e-2, 1e-3, 1e-4, 1e-5	6, 12, 24, 48, 168

Πίνακας 9.3: Υπερπαράμετροι αναζήτησης βάση του καλύτερου συνδυασμού

Τέλος, από τα αποτελέσματα του keras tuner χρησιμοποιήθηκαν οι καλύτερες υπερπαράμετροι για την τελική εκπαίδευση των μοντέλων πάνω στον καλύτερο συνδυασμό εισόδων. Τα αποτελέσματα παρουσιάζονται στους Πίνακες 6 και 7 αντίστοιχα .

Προσέγγιση 2: Επικαλυπτόμενα παράθυρα 24 ωρών (sliding windows):

Για την πρόβλεψη της επόμενης ώρα δοκιμάστηκε επιπλέον η τεχνική επικαλυπτόμενων παραθύρων μεγέθους 24 ωρών. Αυτό σημαίνει ότι ως είσοδος χρησιμοποιήθηκαν υποσύνολα 24 ωρών από την τελευταία στήλη ως τιμές καθυστέρησης για την πρόβλεψη της 25ης ώρας, για τις 8.592(-24) ώρες του συνόλου δοκιμής.

Κάθε δείγμα θα έχει τη μορφή:

$$X = h[i : i + 24], \; y = h[i + 24]$$

Όπου i = 0, 1, 2, ..., 8568

Εικόνα 9.1: Αναπαράσταση τεχνικής επικαλυπτόμενων παραθύρων [Δ18]

```
def create_sliding_windows(data, time_steps, stride=1):
X, y = [], []
for i in range(0, len(data) - time_steps,stride):
X.append(data[i: i + time_steps, -1])
y.append(data[i+time_steps, -1])
        return np.array(X), np.array(y)
```

	0	1	2	3	4	5	6	7	8	9	...	15	16	17	18	19	20	21	22	23	0
0	5925.0	5427.0	5387.0	5207.0	5146.0	5264.0	5812.0	6773.0	7134.0	7361.0	...	7598.0	7863.0	8133.0	8664.0	8732.0	8620.0	8054.0	7362.0	6773.0	6094.0
1	5427.0	5387.0	5207.0	5146.0	5264.0	5812.0	6773.0	7134.0	7361.0	7251.0	...	7863.0	8133.0	8664.0	8732.0	8620.0	8054.0	7362.0	6773.0	6094.0	5548.0
2	5387.0	5207.0	5146.0	5264.0	5812.0	6773.0	7134.0	7361.0	7251.0	7237.0	...	8133.0	8664.0	8732.0	8620.0	8054.0	7362.0	6773.0	6094.0	5548.0	5515.0
3	5207.0	5146.0	5264.0	5812.0	6773.0	7134.0	7361.0	7251.0	7237.0	7271.0	...	8664.0	8732.0	8620.0	8054.0	7362.0	6773.0	6094.0	5548.0	5515.0	5380.0
4	5146.0	5264.0	5812.0	6773.0	7134.0	7361.0	7251.0	7237.0	7271.0	7322.0	...	8732.0	8620.0	8054.0	7362.0	6773.0	6094.0	5548.0	5515.0	5380.0	5210.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
26059	7889.0	7711.0	7291.0	6690.0	6179.0	5501.0	4959.0	4862.0	4634.0	4520.0	...	6952.0	6959.0	6928.0	6808.0	6662.0	7133.0	7586.0	8151.0	8821.0	8937.0
26060	7711.0	7291.0	6690.0	6179.0	5501.0	4959.0	4862.0	4634.0	4520.0	4597.0	...	6959.0	6928.0	6808.0	6662.0	7133.0	7586.0	8151.0	8821.0	8937.0	8527.0
26061	7291.0	6690.0	6179.0	5501.0	4959.0	4862.0	4634.0	4520.0	4597.0	4961.0	...	6928.0	6808.0	6662.0	7133.0	7586.0	8151.0	8821.0	8937.0	8527.0	7695.0
26062	6690.0	6179.0	5501.0	4959.0	4862.0	4634.0	4520.0	4597.0	4961.0	5552.0	...	6808.0	6662.0	7133.0	7586.0	8151.0	8821.0	8937.0	8527.0	7695.0	6712.0
26063	6179.0	5501.0	4959.0	4862.0	4634.0	4520.0	4597.0	4961.0	5552.0	6193.0	...	6662.0	7133.0	7586.0	8151.0	8821.0	8937.0	8527.0	7695.0	6712.0	6211.0

26064 rows × 25 columns

Εικόνα 9.2: Δεδομένα εκπαίδευσης επικαλυπτόμενων παραθύρων

	0	1	2	3	4	5	6	7	8	9	...	15	16	17	18	19	20	21	22	23	0
0	5172.0	4756.0	4680.0	4482.0	4334.0	4366.0	4523.0	4770.0	4978.0	5051.0	...	5493.0	5836.0	6242.0	6806.0	6810.0	6603.0	6239.0	5751.0	5402.0	4920.0
1	4756.0	4680.0	4482.0	4334.0	4366.0	4523.0	4770.0	4978.0	5051.0	4940.0	...	5836.0	6242.0	6806.0	6810.0	6603.0	6239.0	5751.0	5402.0	4920.0	4505.0
2	4680.0	4482.0	4334.0	4366.0	4523.0	4770.0	4978.0	5051.0	4940.0	4845.0	...	6242.0	6806.0	6810.0	6603.0	6239.0	5751.0	5402.0	4920.0	4505.0	4425.0
3	4482.0	4334.0	4366.0	4523.0	4770.0	4978.0	5051.0	4940.0	4845.0	4850.0	...	6806.0	6810.0	6603.0	6239.0	5751.0	5402.0	4920.0	4505.0	4425.0	4216.0
4	4334.0	4366.0	4523.0	4770.0	4978.0	5051.0	4940.0	4845.0	4850.0	4978.0	...	6810.0	6603.0	6239.0	5751.0	5402.0	4920.0	4505.0	4425.0	4216.0	4093.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
8563	8302.0	7941.0	7269.0	6776.0	6395.0	5779.0	5232.0	5194.0	5005.0	4898.0	...	6975.0	7196.0	7341.0	7381.0	7219.0	7610.0	7918.0	8355.0	8948.0	8971.0
8564	7941.0	7269.0	6776.0	6395.0	5779.0	5232.0	5194.0	5005.0	4898.0	4932.0	...	7196.0	7341.0	7381.0	7219.0	7610.0	7918.0	8355.0	8948.0	8971.0	8563.0
8565	7269.0	6776.0	6395.0	5779.0	5232.0	5194.0	5005.0	4898.0	4932.0	5196.0	...	7341.0	7381.0	7219.0	7610.0	7918.0	8355.0	8948.0	8971.0	8563.0	7705.0
8566	6776.0	6395.0	5779.0	5232.0	5194.0	5005.0	4898.0	4932.0	5196.0	5621.0	...	7381.0	7219.0	7610.0	7918.0	8355.0	8948.0	8971.0	8563.0	7705.0	6759.0
8567	6395.0	5779.0	5232.0	5194.0	5005.0	4898.0	4932.0	5196.0	5621.0	6067.0	...	7219.0	7610.0	7918.0	8355.0	8948.0	8971.0	8563.0	7705.0	6759.0	6320.0

8568 rows × 25 columns

Εικόνα 9.3: Δεδομένα δοκιμής επικαλυπτόμενων παραθύρων

Οι υπερπαράμετροι που δοκιμάστηκαν με keras tuner και για αυτή την προσέγγιση είναι οι ίδιες. (Πίνακα 5)

10. Αποτελέσματα

Για κάθε μοντέλο και συνδυασμό εισόδου για την πρώτη προσέγγιση έγιναν πέντε εκτελέσεις, δεδομένης της στοχαστικής φύσης της αρχικοποίησης των βαρών, ώστε να εξαχθεί ο μέσος όρος για κάθε μέτρο. Παρακάτω παρουσιάζονται τα αποτελέσματα RMSE και APE της εκπαίδευσης και δοκιμής εκτενώς αλλά και συνοπτικά:

Model	Combination name	Combination column	Rmse train avg	Ape train avg	Rmse test avg	Ape test avg
Dense	one_day	(0, 1, 2, 3, 4)	458,446	4,699	459,572	4,933
Dense	two_days	(5, 6, 7, 8, 9)	604,438	6,476	610,757	6,844
Dense	seven_days	(10, 11, 12, 13, 14)	577,808	6,219	623,155	6,771
Dense	h_all	(2, 7, 12)	397,699	4,149	415,009	4,518
Dense	h+1_all	(3, 8, 13)	498,971	5,743	505,085	5,952
Dense	h+2_all	(4, 9, 14)	688,733	8,147	682,274	8,325
Dense	h-1_all	(1, 6, 11)	505,937	5,841	512,173	6,08
Dense	h-2_all	(0, 5, 10)	696,464	8,245	690,567	8,42
Dense	h_h+1_one_day	(2, 3)	457,818	4,659	459,589	4,908
Dense	h_h+1_one_two_days	(2, 3, 7, 8)	450,594	4,646	454,32	4,909

Dense	h_h+1_all	(2, 3, 7, 8, 12, 13)	394,508	4,098	412,935	4,469
Dense	h_h+1_h+2_one_day	(2, 3, 4)	456,917	4,652	458,685	4,9
Dense	h_h+1_h+2_one_two_days	(2, 3, 4, 7, 8, 9)	450,016	4,643	455,253	4,92
Dense	h_h+1_h+2_all	(2, 3, 4, 7, 8, 9, 12, 13, 14)	394,025	4,091	411,986	4,454
Dense	h-1_h_one_day	(1, 2)	457,969	4,642	459,927	4,901
Dense	h-1_h_one_two_days	(1, 2, 6, 7)	449,656	4,652	456,273	4,953
Dense	h-1_h_all	(1, 2, 6, 7, 11, 12)	396,67	4,143	414,714	4,513
Dense	h-2_h-1_h_one_day	(0, 1, 2)	458,708	4,652	460,674	4,912
Dense	h-2_h-1_h_two_days	(5, 6, 7)	605,324	6,434	613,01	6,836
Dense	h-2_h-1_h_seven_days	(10, 11, 12)	579,25	6,212	625,017	6,767
Dense	h-2_h-1_h_one_two_days	(0, 1, 2, 5, 6, 7)	452,623	4,723	460,652	5,029
Dense	h-2_h-1_h_all	(0, 1, 2, 5, 6, 7, 10, 11, 12)	400,423	4,199	417,985	4,561
Dense	h-1_h_h+1_one_day	(1, 2, 3)	457,621	4,687	458,702	4,92
Dense	h-1_h_h+1_one_two_days	(1, 2, 3, 6, 7, 8)	455,176	4,747	457,554	4,988

Dense	all	(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14)	397,602	4,146	414,814	4,494
Dense	doc1	(5, 6, 7, 8)	602,043	6,428	608,903	6,808
Dense	doc2	(5, 6, 8)	606,997	6,53	612,704	6,886
Dense	doc3	(5, 8)	617,448	6,728	621,943	7,056
Dense	doc4	(2, 12)	397,847	4,153	413,589	4,507
Dense	doc5	(1, 2, 12)	398,808	4,165	414,96	4,521
Dense	doc6	(2,)	457,731	4,635	459,773	4,895
LSTM	one_day	(0, 1, 2, 3, 4)	469,457	4,968	468,963	5,17
LSTM	two_days	(5, 6, 7, 8, 9)	609,742	6,603	615,736	6,958
LSTM	seven_days	(10, 11, 12, 13, 14)	593,131	6,509	636,031	7,032
LSTM	h_all	(2, 7, 12)	406,516	4,3	425,857	4,715
LSTM	h+1_all	(3, 8, 13)	509,067	5,868	514,447	6,101
LSTM	h+2_all	(4, 9, 14)	701,943	8,333	692,068	8,494
LSTM	h-1_all	(1, 6, 11)	513,41	5,916	521,677	6,194

LSTM	h-2_all	(0, 5, 10)	698,706	8,277	693,997	8,474
LSTM	h_h+1_one_day	(2, 3)	461,288	4,762	462,412	4,992
LSTM	h_h+1_one_two_days	(2, 3, 7, 8)	455,883	4,797	462,747	5,08
LSTM	h_h+1_all	(2, 3, 7, 8, 12, 13)	427,636	4,645	444,476	4,972
LSTM	h_h+1_h+2_one_day	(2, 3, 4)	463,605	4,82	465,082	5,047
LSTM	h_h+1_h+2_one_two_days	(2, 3, 4, 7, 8, 9)	478,369	5,2	487,298	5,479
LSTM	h_h+1_h+2_all	(2, 3, 4, 7, 8, 9, 12, 13, 14)	453,635	4,956	471,262	5,277
LSTM	h-1_h_one_day	(1, 2)	478,559	5,046	478,418	5,253
LSTM	h-1_h_one_two_days	(1, 2, 6, 7)	473,415	5,092	480,078	5,374
LSTM	h-1_h_all	(1, 2, 6, 7, 11, 12)	430,581	4,658	451,379	5,048
LSTM	h-2_h-1_h_one_day	(0, 1, 2)	480,915	5,033	480,465	5,249
LSTM	h-2_h-1_h_two_days	(5, 6, 7)	625,192	6,787	630,431	7,135
LSTM	h-2_h-1_h_seven_days	(10, 11, 12)	608,523	6,753	649,53	7,277
LSTM	h-2_h-1_h_one_two_days	(0, 1, 2, 5, 6, 7)	501,323	5,552	508,205	5,819

LSTM	h-2_h-1_h_all	(0, 1, 2, 5, 6, 7, 10, 11, 12)	494,559	5,476	506,513	5,782
LSTM	h-1_h_h+1_one_day	(1, 2, 3)	460,875	4,75	461,766	4,98
LSTM	h-1_h_h+1_one_two_days	(1, 2, 3, 6, 7, 8)	463,968	4,943	470,688	5,222
LSTM	all	(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14)	750,75	8,428	757,104	8,695
LSTM	doc1	(5, 6, 7, 8)	606,595	6,535	613,292	6,911
LSTM	doc2	(5, 6, 8)	616,915	6,708	622,198	7,057
LSTM	doc3	(5, 8)	617,356	6,738	621,527	7,068
LSTM	doc4	(2, 12)	398,128	4,157	413,975	4,513
LSTM	doc5	(1, 2, 12)	408,196	4,324	422,338	4,654
LSTM	doc6	(2,)	457,176	4,625	459,126	4,886
GRU	one_day	(0, 1, 2, 3, 4)	473,451	5,032	472,736	5,222
GRU	two_days	(5, 6, 7, 8, 9)	609,513	6,604	615,297	6,95
GRU	seven_days	(10, 11, 12, 13, 14)	582,201	6,339	626,254	6,86
GRU	h_all	(2, 7, 12)	405,899	4,293	425,376	4,688

GRU	h+1_all	(3, 8, 13)	505,303	5,822	512,866	6,052
GRU	h+2_all	(4, 9, 14)	698,835	8,256	693,098	8,447
GRU	h-1_all	(1, 6, 11)	513,102	5,915	523,515	6,213
GRU	h-2_all	(0, 5, 10)	699,127	8,321	695,975	8,533
GRU	h_h+1_one_day	(2, 3)	457,639	4,662	459,391	4,909
GRU	h_h+1_one_two_days	(2, 3, 7, 8)	460,581	4,9	471,781	5,205
GRU	h_h+1_all	(2, 3, 7, 8, 12, 13)	433,32	4,703	453,82	5,068
GRU	h_h+1_h+2_one_day	(2, 3, 4)	467,336	4,878	468,632	5,103
GRU	h_h+1_h+2_one_two_days	(2, 3, 4, 7, 8, 9)	455,486	4,79	462,593	5,066
GRU	h_h+1_h+2_all	(2, 3, 4, 7, 8, 9, 12, 13, 14)	505,647	5,677	520,855	6,005
GRU	h-1_h_one_day	(1, 2)	466,422	4,822	467,411	5,046
GRU	h-1_h_one_two_days	(1, 2, 6, 7)	468,379	5,031	481,387	5,364
GRU	h-1_h_all	(1, 2, 6, 7, 11, 12)	411,272	4,376	433,019	4,785
GRU	h-2_h-1_h_one_day	(0, 1, 2)	469,081	4,867	470,411	5,093

GRU	h-2_h-1_h_two_days	(5, 6, 7)	613,604	6,584	620,057	6,963
GRU	h-2_h-1_h_seven_days	(10, 11, 12)	584,929	6,303	630,205	6,85
GRU	h-2_h-1_h_one_two_days	(0, 1, 2, 5, 6, 7)	502,113	5,592	512,822	5,898
GRU	h-2_h-1_h_all	(0, 1, 2, 5, 6, 7, 10, 11, 12)	462,679	5,106	482,913	5,469
GRU	h-1_h_h+1_one_day	(1, 2, 3)	458,307	4,7	459,421	4,934
GRU	h-1_h_h+1_one_two_days	(1, 2, 3, 6, 7, 8)	472,711	5,055	480,302	5,345
GRU	all	(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14)	433,301	4,686	449,022	5
GRU	doc1	(5, 6, 7, 8)	602,697	6,442	609,19	6,82
GRU	doc2	(5, 6, 8)	608,479	6,569	614,023	6,924
GRU	doc3	(5, 8)	617,498	6,757	621,639	7,083
GRU	doc4	(2, 12)	398,282	4,16	414,638	4,52
GRU	doc5	(1, 2, 12)	405,19	4,291	421,193	4,63
GRU	doc6	(2,)	457,289	4,627	459,281	4,888
Simple RNN	one_day	(0, 1, 2, 3, 4)	456,545	4,665	457,981	4,906

Simple RNN	two_days	(5, 6, 7, 8, 9)	601,221	6,417	608,634	6,801
Simple RNN	seven_days	(10, 11, 12, 13, 14)	578,828	6,245	623,808	6,795
Simple RNN	h_all	(2, 7, 12)	393,109	4,076	415,475	4,476
Simple RNN	h+1_all	(3, 8, 13)	497,615	5,716	506,229	5,944
Simple RNN	h+2_all	(4, 9, 14)	688,512	8,123	681,002	8,3
Simple RNN	h-1_all	(1, 6, 11)	498,946	5,697	509,588	5,975
Simple RNN	h-2_all	(0, 5, 10)	684,664	8,127	684,205	8,362
Simple RNN	h_h+1_one_day	(2, 3)	457,591	4,652	459,389	4,902
Simple RNN	h_h+1_one_two_days	(2, 3, 7, 8)	448,851	4,638	456,238	4,938
Simple RNN	h_h+1_all	(2, 3, 7, 8, 12, 13)	391,173	4,06	412,844	4,451
Simple RNN	h_h+1_h+2_one_day	(2, 3, 4)	457,415	4,668	459,042	4,917
Simple RNN	h_h+1_h+2_one_two_days	(2, 3, 4, 7, 8, 9)	450,88	4,746	460,58	5,04
Simple RNN	h_h+1_h+2_all	(2, 3, 4, 7, 8, 9, 12, 13, 14)	393,558	4,102	413,711	4,471
Simple RNN	h-1_h_one_day	(1, 2)	458,771	4,654	460,799	4,912

Simple RNN	h-1_h_one_two_days	(1, 2, 6, 7)	449,375	4,664	458,089	4,977
Simple RNN	h-1_h_all	(1, 2, 6, 7, 11, 12)	390,464	4,054	413,496	4,454
Simple RNN	h-2_h-1_h_one_day	(0, 1, 2)	460,292	4,675	462,576	4,933
Simple RNN	h-2_h-1_h_two_days	(5, 6, 7)	605,939	6,444	613,66	6,846
Simple RNN	h-2_h-1_h_seven_days	(10, 11, 12)	581,198	6,245	626,918	6,793
Simple RNN	h-2_h-1_h_one_two_days	(0, 1, 2, 5, 6, 7)	450,658	4,702	459,338	5,017
Simple RNN	h-2_h-1_h_all	(0, 1, 2, 5, 6, 7, 10, 11, 12)	394,294	4,112	416,433	4,503
Simple RNN	h-1_h_h+1_one_day	(1, 2, 3)	457,207	4,668	458,488	4,91
Simple RNN	h-1_h_h+1_one_two_days	(1, 2, 3, 6, 7, 8)	448,334	4,675	457,187	4,974
Simple RNN	all	(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14)	402,805	4,258	420,155	4,592
Simple RNN	doc1	(5, 6, 7, 8)	600,55	6,387	607,679	6,783
Simple RNN	doc2	(5, 6, 8)	605,926	6,503	611,967	6,867
Simple RNN	doc3	(5, 8)	616,588	6,721	620,906	7,05
Simple RNN	doc4	(2, 12)	398,428	4,157	414,093	4,512

Simple RNN	doc5	(1, 2, 12)	397,209	4,141	412,697	4,493
Simple RNN	doc6	(2,)	457,372	4,628	459,384	4,89

Πίνακας 10.1: Αποτελέσματα για κάθε συνδυασμό για κάθε αρχικό μοντέλο

Model	Comb_name	Comb_col	Rmse_ train_ avg	Ape_ train_ avg	Rmse_ test_ avg	Ape_ test_ avg
Dense	h_h+1_h+2_all	(2, 3, 4, 7, 8, 9, 12, 13, 14)	394,025	4,091	411,986	4,454
LSTM	doc4	(2, 12)	398,128	4,157	413,975	4,513
GRU	doc4	(2, 12)	398,282	4,16	414,638	4,52
SimpleRNN	h_h+1_all	(2, 3, 7, 8, 12, 13)	391,173	4,06	412,844	4,451

Πίνακας 10.2: Οι καλύτεροι συνδυασμοί για κάθε μοντέλο του πίνακα 10.1

Για όλα τα μοντέλα, και για τις δύο προσεγγίσεις, οι καλύτεροι υπερπαράμετροι και το σύνολο των παραμέτρων παρουσιάζονται στον παρακάτω πίνακα:

Model	Input size	layers	units	Learning rate	Batch size	Parameters
Dense	9	3	200, 200, 200	0.0001	6	82601
LSTM	2	2	100, 100	0.0001	12	121301
GRU	2	2	200, 100	0.00001	6	212501
SimpleRNN	6	2	100, 100	0.0001	48	30401
Dense_24	24	2	100, 100	0.001	24	12701
LSTM_24	24	1	200	0.001	6	161801
GRU_24	24	1	200	0.0001	6	122001
SimpleRNN_24	24	3	25, 200, 100	0.00001	6	76076

Πίνακας 10.3: Αποτελέσματα αναζήτησης υπερπαραμέτρων keras tuner

Στον παρακάτω πίνακα παρουσιάζονται τα τελικά αποτελέσματα εκπαίδευσης και δοκιμής (γραμμή **Total**) για κάθε μοντέλο και προσέγγιση. Τα μοντέλα "_24" αφορούν την προσέγγιση με 24 τιμές καθυστέρησης. Επιπλέον αναλύονται και τα επιμέρους αποτελέσματα για κάθε εποχή:

Train APE(%)	Dense_24	Dense	SimpleRNN_24	SimpleRNN	LSTM_24	LSTM	GRU_24	GRU
Spring	1.58	4.44	1.49	4.44	1.70	4.40	1.60	4.36
Summer	1.49	3.78	1.30	4.15	1.40	4.26	1.18	4.18
Autumn	1.57	3.47	1.37	3.63	1.47	3.64	1.41	3.56
Winter	1.66	4.37	1.48	4.45	1.70	4.49	1.58	4.48
Total	1.58	4.01	1.41	4.17	1.57	4.19	1.44	4.14

Πίνακας 10.4: Αποτελέσματα APE εκπαίδευσης

Test APE(%)	Dense_24	Dense	SimpleRNN_24	SimpleRNN	LSTM_24	LSTM	GRU_24	GRU
Spring	1.60	4.32	1.50	4.35	1.75	4.34	1.61	4.30
Summer	1.54	4.62	1.35	4.97	1.46	4.99	1.20	4.90
Autumn	1.66	3.87	1.44	4.11	1.53	4.16	1.51	4.06
Winter	1.66	4.78	1.47	4.75	1.68	4.72	1.62	4.77
Total	1.62	4.39	1.44	4.54	1.59	4.55	1.48	4.50

Πίνακας 10.5: Αποτελέσματα APE δοκιμής

Train RMSE	Dense_24	Dense	SimpleRNN_24	SimpleRNN	LSTM_24	LSTM	GRU_24	GRU
Spring	136.59	405.00	131.30	396.53	143.69	396.78	139.53	396.84
Summer	131.18	356.22	119.42	392.39	127.07	406.76	107.97	393.54
Autumn	129.38	312.87	116.74	328.41	122.60	331.52	123.98	323.46
Winter	160.27	455.50	146.34	464.14	163.42	469.97	154.29	465.64
Total	139.63	385.23	128.76	397.46	139.81	403.45	132.27	397.23

Πίνακας 10.6: Αποτελέσματα RMSE εκπαίδευσης

Test RMSE	Dense_24	Dense	SimpleRNN_24	SimpleRNN	LSTM_24	LSTM	GRU_24	GRU
Spring	128.84	368.40	121.63	370.63	164.47	367.81	131.27	365.84
Summer	136.95	443.34	124.96	470.74	144.90	472.49	110.26	464.11
Autumn	135.11	330.31	120.35	350.26	160.39	354.86	128.10	344.41
Winter	156.65	475.07	139.86	478.50	193.34	470.20	149.75	469.94
Total	139.29	406.54	126.58	419.96	165.91	418.66	130.02	413.48

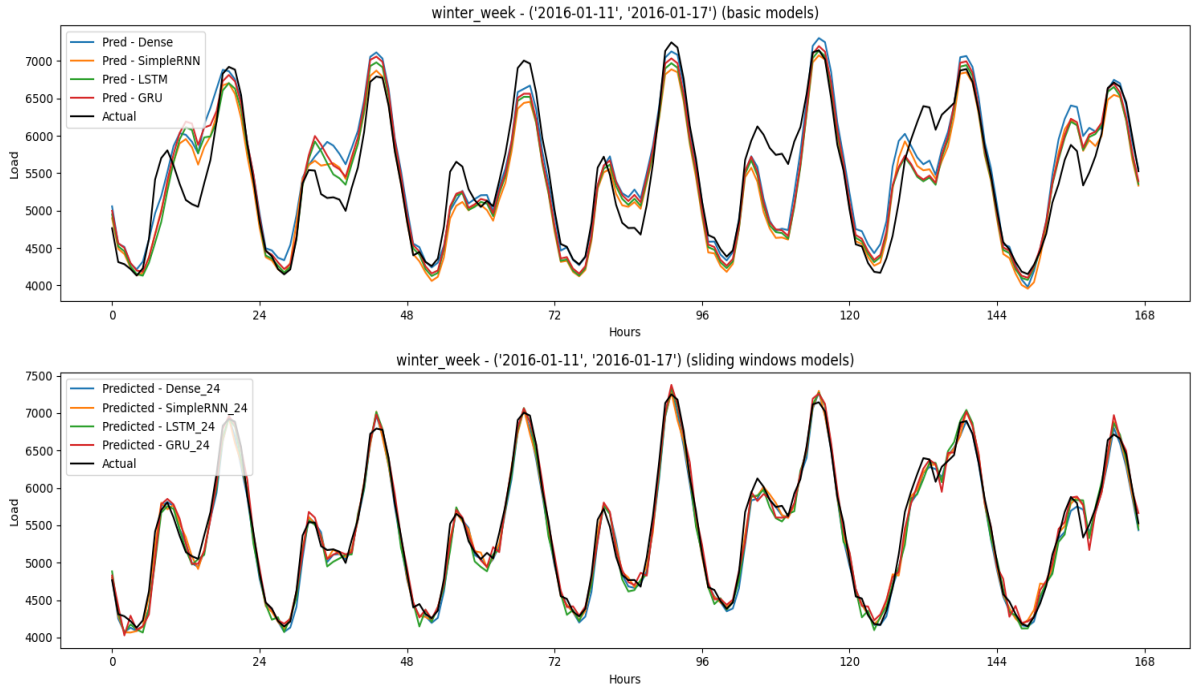
Πίνακας 10.7: Αποτελέσματα RMSE δοκιμής

Παρακάτω παρουσιάζονται το άθροισμα και το ποσοστό των ωρών του συνόλου δοκιμής όπου η απόλυτη διαφορά της πραγματικής τιμής με την προβλεπόμενη είναι πάνω από τα ποσά της πρώτης στήλης

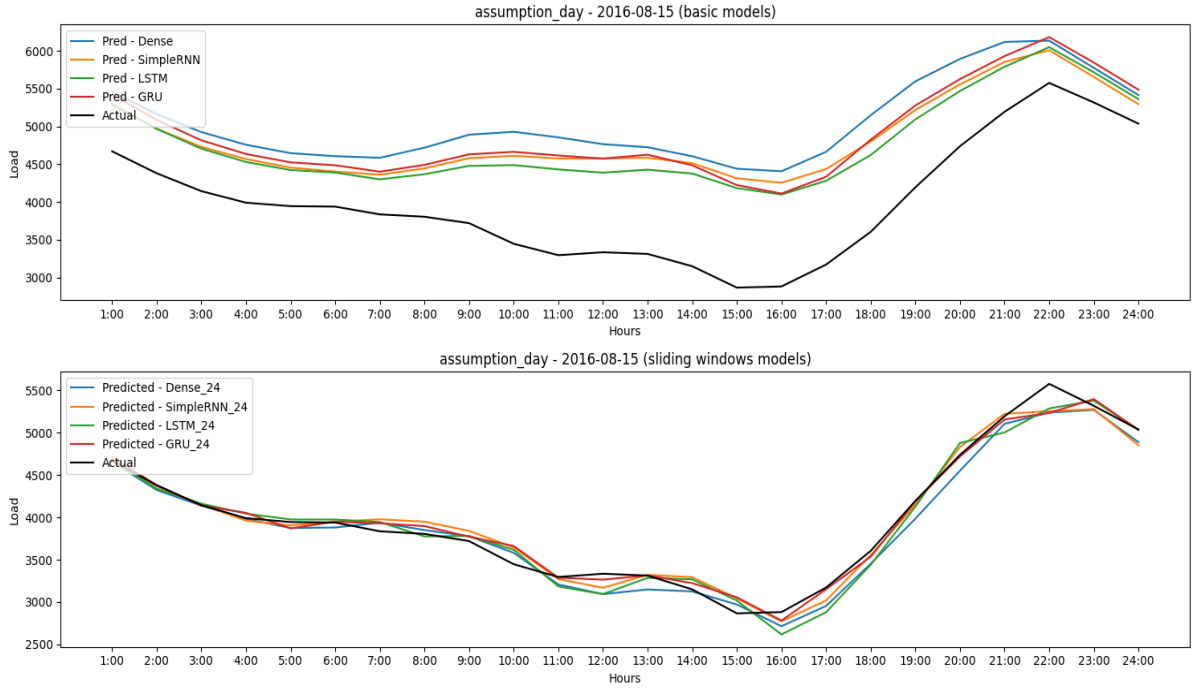
ForecastErrorDurationCurve								
Hours	Dense_24	Dense	SimpleRNN_24	SimpleRNN	LSTM_24	LSTM	GRU_24	GRU
>100MW	3737	6182	3156	6445	3680	6402	3263	6364
>200MW	1194	4339	933	4537	1215	4500	999	4444
>400MW	78	2015	61	2232	91	2265	69	2230
>500MW	22	1407	15	1575	24	1589	20	1555
Time (%)								
>100MW	43.61	72.15	36.83	75.22	42.95	74.72	38.08	74.28
>200MW	13.93	50.64	10.89	52.95	14.18	52.52	11.66	51.87
>400MW	0.91	23.52	0.71	26.05	1.06	26.44	0.81	26.03
>500MW	0.25	16.42	0.18	18.38	0.28	18.55	0.23	18.15

Πίνακας 10.8: Αποτελέσματα Forecast Error Duration Curve

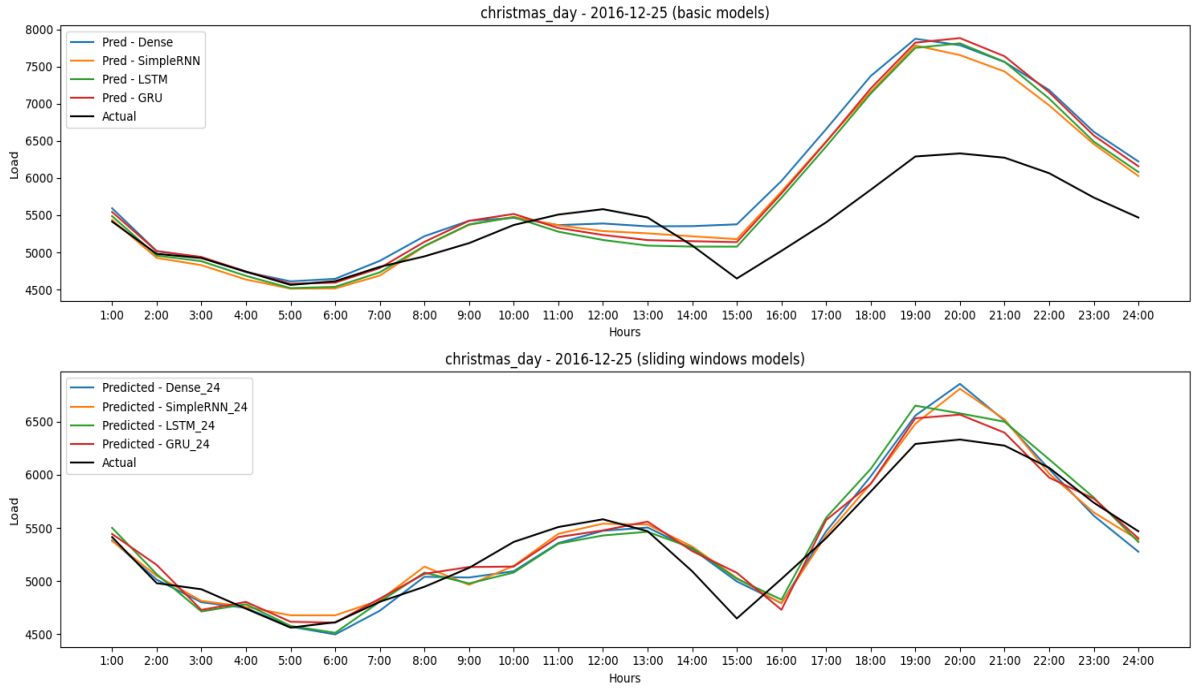
Για την καλύτερη απεικόνιση των αποτελεσμάτων σχεδιάστηκαν συγκεκριμένες ημερομηνίες από κάθε εποχή με τη βοήθεια του συνόλου μεταδεδομένων του κ. Κανδηλογιαννάκη:



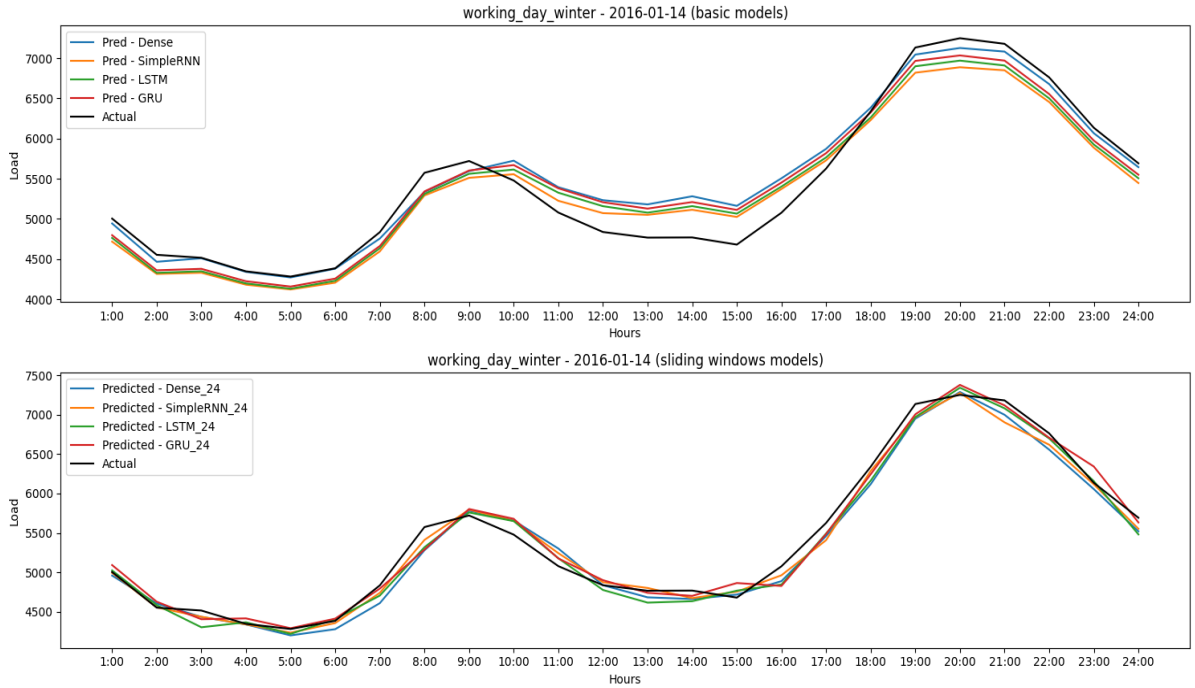
Εικόνα 10.1: Προβλέψεις μοντέλων Χειμερινής Εβδομάδας



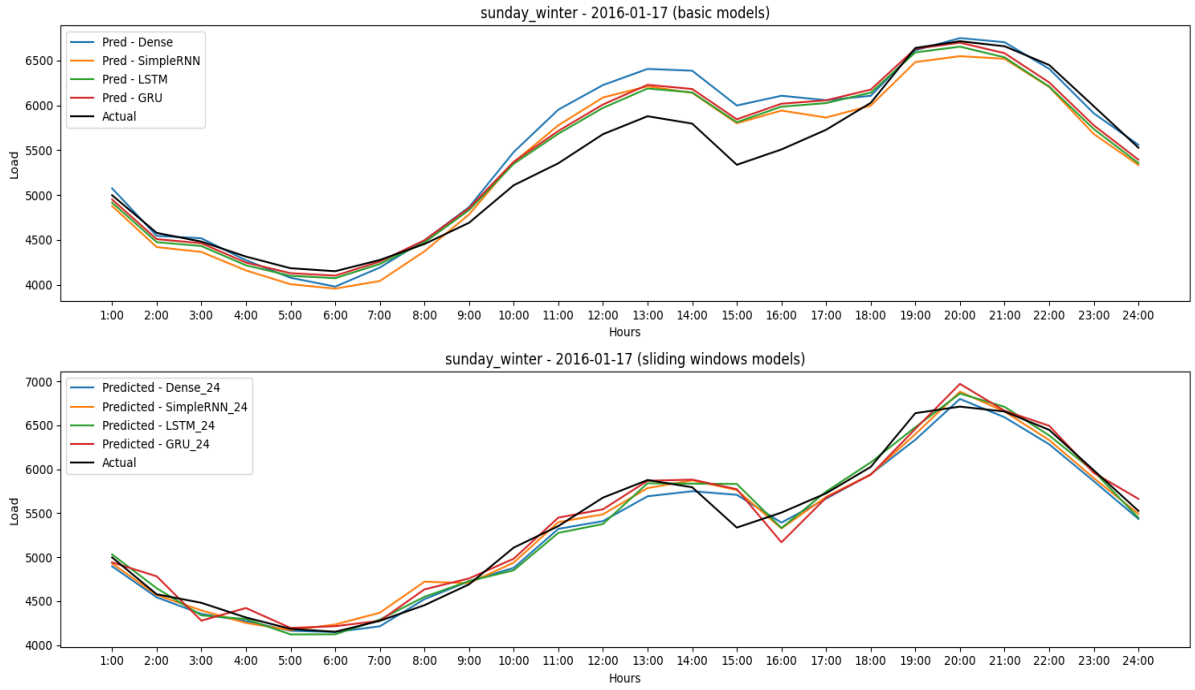
Εικόνα 10.2: Προβλέψεις μοντέλων Δεκαπενταύγουστου



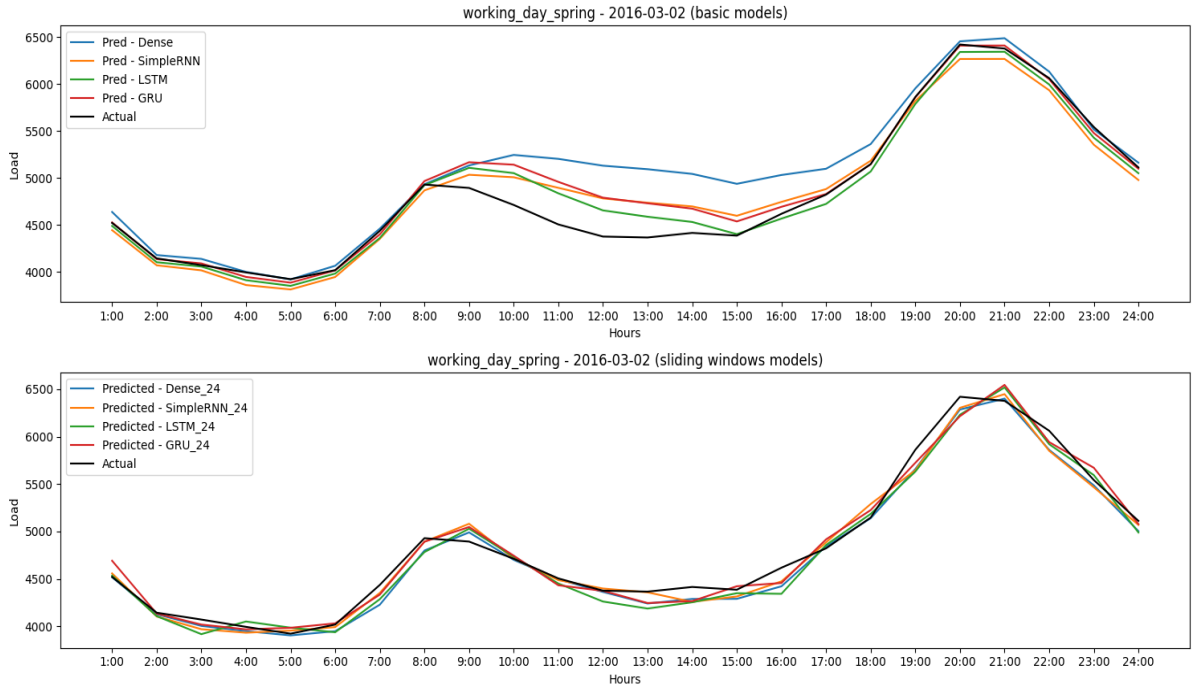
Εικόνα 10.3: Προβλέψεις μοντέλων Χριστουγέννων



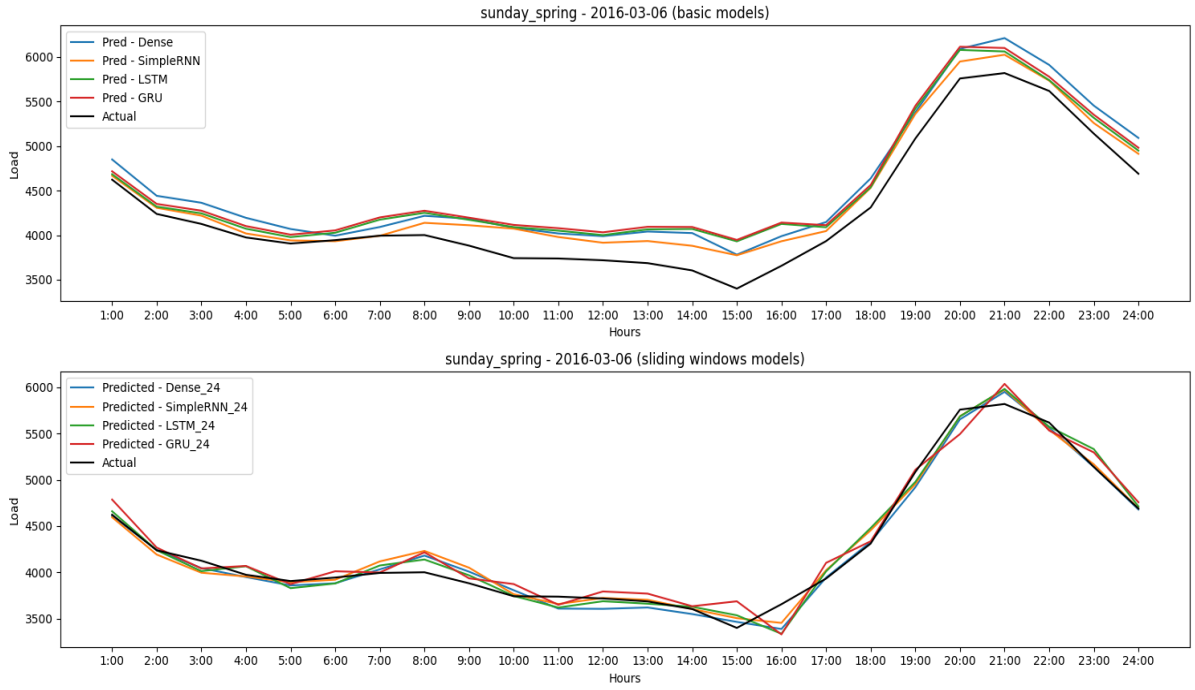
Εικόνα 10.4: Προβλέψεις μοντέλων χειμερινής εργάσιμης ημέρας



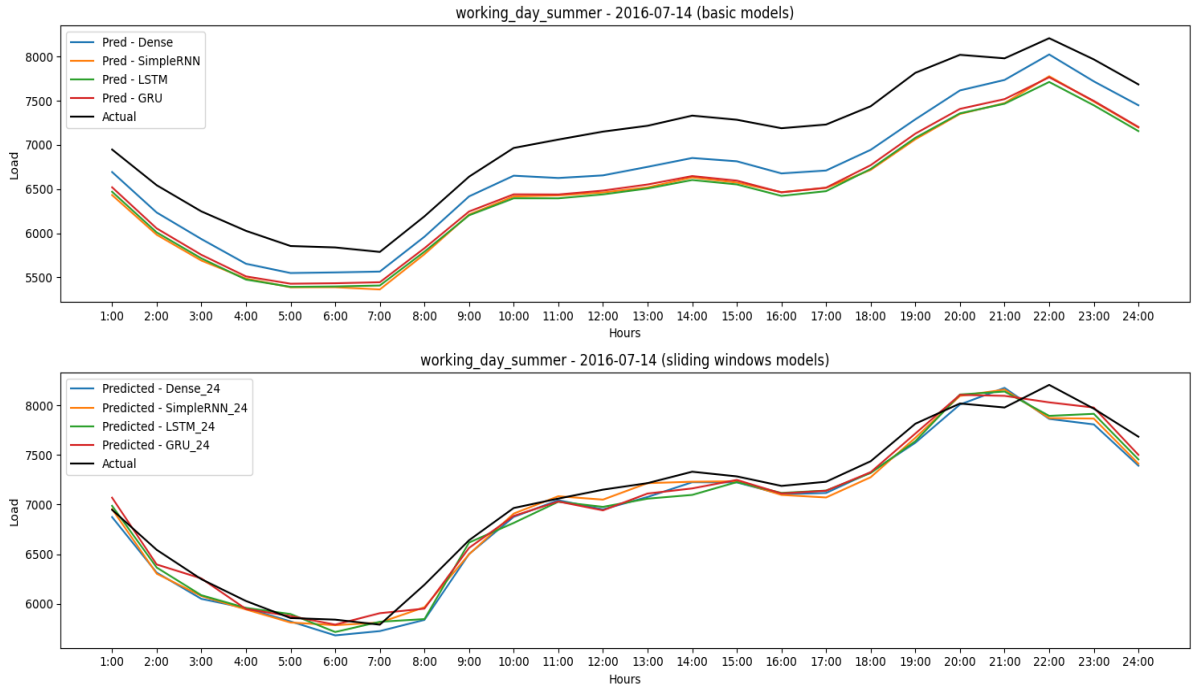
Εικόνα 10.4: Προβλέψεις μοντέλων χειμερινής Κυριακής



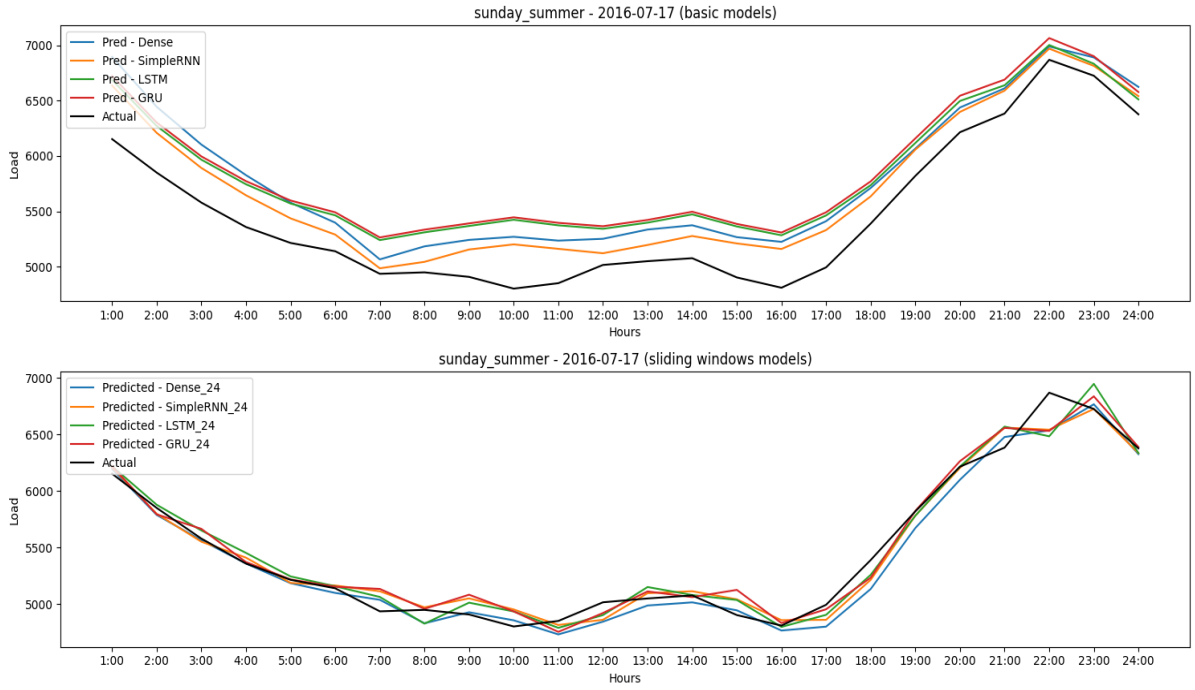
Εικόνα 10.5: Προβλέψεις μοντέλων εαρινής εργάσιμης ημέρας



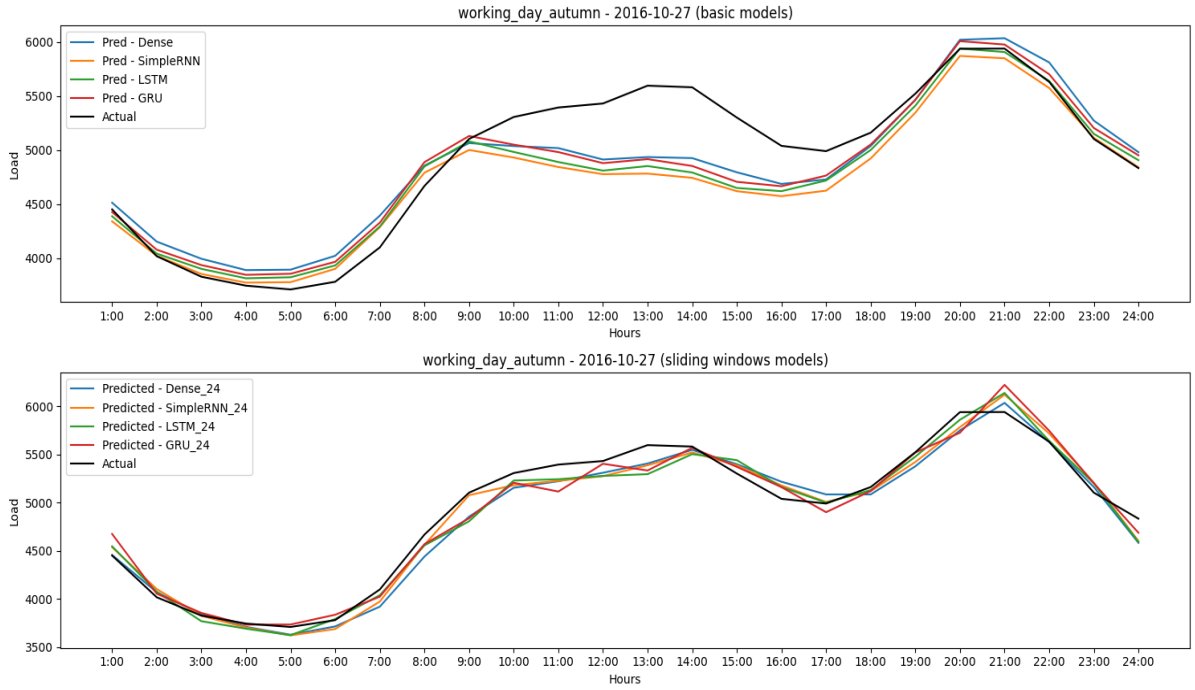
Εικόνα 10.6: Προβλέψεις μοντέλων εαρινής Κυριακής



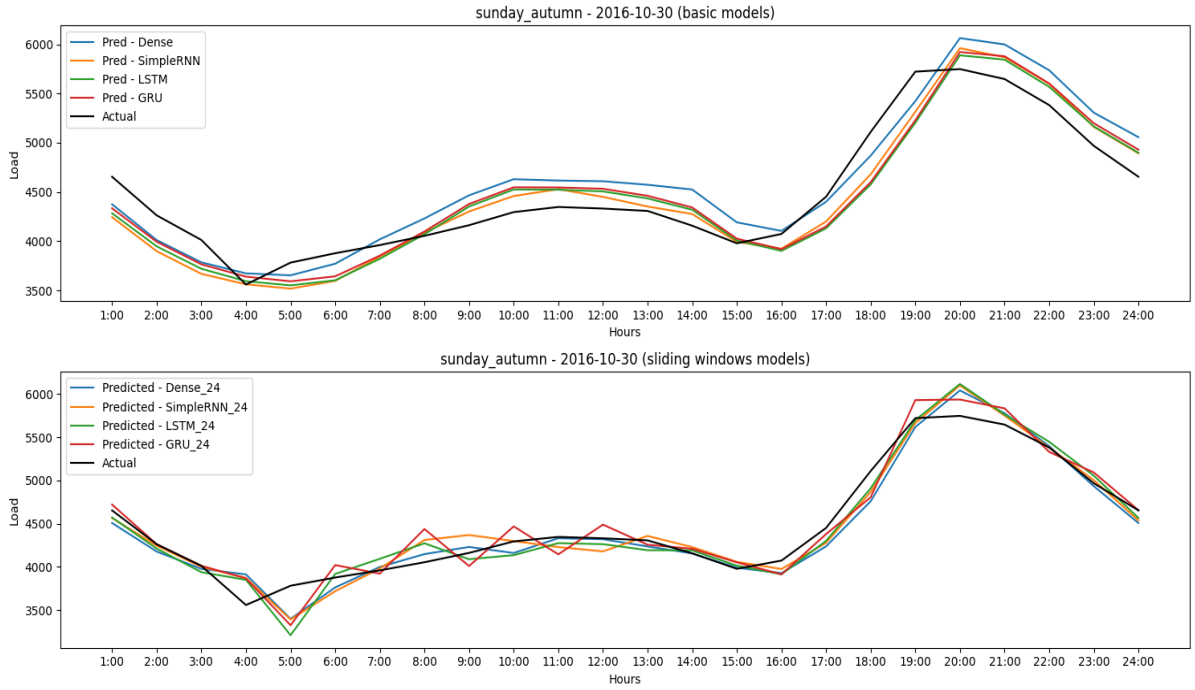
Εικόνα 10.7: Προβλέψεις μοντέλων θερινής εργάσιμης ημέρας



Εικόνα 10.8: Προβλέψεις μοντέλων θερινής Κυριακής



Εικόνα 10.9: Προβλέψεις μοντέλων φθινοπωρινής εργάσιμης ημέρας



Εικόνα 10.10: Προβλέψεις μοντέλων φθινοπωρινής Κυριακής

10. Παρατηρήσεις και συμπεράσματα

Προσέγγιση 1:

Όλα τα μοντέλα επιτυγχάνουν αποδεκτά αποτελέσματα: APE κάτω από 4.6% και RMSE κάτω από 420 MW πάνω στα νέα δεδομένα του έτους δοκιμής.

Τα RMSE και APE της δοκιμής είναι πολύ κοντά με της εκπαίδευσης και επομένως όλα τα μοντέλα επιτυγχάνουν καλή γενίκευση.

Οι συνδυασμοί με πολλές τιμές καθυστέρησης απέδωσαν χειρότερα αποτελέσματα για τα πιο περίπλοκα μοντέλα LSTM και GRU και καλύτερα στις πιο απλές αρχιτεκτονικές Dense και SimpleRNN.

Οι συνδυασμοί που απείχαν χρονικά περισσότερο από την τελευταία στήλη και όσοι αποτελούνταν κυρίως από τιμές καθυστέρησης δύο ώρες πριν ή μετά, είχαν τη χειρότερη απόδοση

Ενδιαφέρον παρουσιάζει ο καλύτερος συνδυασμός για τα μοντέλα LSTM και GRU. Αποτελούμενος μόνο από δύο τιμές καθυστέρησης της ίδιας ώρας, μια και επτά ημέρες πριν, δίνει το καλύτερο αποτέλεσμα παρά τη χρονική απόσταση. Εάν προστεθεί όμως και η ίδια ώρα δύο ημέρες πριν, το αποτέλεσμα είναι χειρότερο.

Η πιο “εύκολες” εποχές στην πρόβλεψη είναι το φθινόπωρο και η άνοιξη καθώς έχουν τα μικρότερα σφάλματα προβλέψεων. Αντίθετα ο χειμώνας και το καλοκαίρι έχουν τα χειρότερα αποτελέσματα.

Το μοντέλο Dense χρειάστηκε τις περισσότερες τιμές καθυστέρησης (εννιά) για να είναι ανταγωνιστικό προς τα RNN. Παρόλα αυτά πέτυχε το μικρότερο συνολικό σφάλμα (4.39%) για όλο το έτος δοκιμής αλλά και σε επίπεδο εποχών εκτός του χειμώνα.

Όλα τα μοντέλα αδυνατούν να κάνουν ικανοποιητική πρόβλεψη για την ημέρα των Χριστουγέννων και του Δεκαπενταύγουστου με μεγαλύτερη απόκλιση ~2000 MW.

Προσέγγιση 2:

Όλα τα μοντέλα παρουσιάζουν πολύ ικανοποιητικά αποτελέσματα: APE κάτω από **1.6%** και RMSE κάτω από **170MW**.

Τα RMSE και APE της δοκιμής είναι πολύ κοντά με της εκπαίδευσης και επομένως όλα τα μοντέλα επιτυγχάνουν καλή γενίκευση.

Το SimpleRNN παρά την απλή αρχιτεκτονική του πέτυχε με λιγότερες παραμέτρους τα μικρότερα συνολικά σφάλματα για όλο το έτος δοκιμής και τις εποχές εκτός του καλοκαιριού.

Το Dense πέτυχε αποτελέσματα προβλέψεων πολύ κοντά με το LSTM που έχει την πιο περίπλοκη αρχιτεκτονική και με κάτω από 8% των παραμέτρων του.

Ενδιαφέρον παρουσιάζει το γεγονός ότι σε αντίθεση με την Προσέγγιση 1, τα μοντέλα είχαν καλύτερη απόδοση για το καλοκαίρι και το χειμώνα και χειρότερη για τις πιο ουδέτερες διακυμάνσεις της άνοιξης και του φθινοπώρου.

Όλα τα μοντέλα εντόπισαν και πρόβλεψαν τα πρότυπα της ημέρας και ειδικά για όλη τη χειμερινή εβδομάδα που σχεδιάστηκε, την ημέρα των Χριστουγέννων αλλά και του Δεκαπενταύγουστου.

11. Επίλογος

Η βραχυπρόθεσμη πρόβλεψη ηλεκτρικού φορτίου αποτέλεσε μια ενδιαφέρουσα αφορμή για την εισαγωγή στον τομέα της Βαθιάς Μάθησης. Μελετήθηκαν και εφαρμόστηκαν σε πραγματικό πρόβλημα τα μοντέλα Dense, SimpleRNN, GRU, LSTM για διαφορετικές τιμές καθυστέρησης σε δύο προσεγγίσεις. Από τη θεωρία το αναμενόμενο αποτέλεσμα θα ήταν το LSTM και GRU να αποδώσουν καλύτερα, αλλά στην πράξη τα αποτελέσματα ήταν διαφορετικά. Τα πιο περίπλοκα Αναδρομικά Νευρωνικά Δίκτυα GRU και LSTM, για τις υπερπαραμέτρους που δοκιμάστηκαν, είχαν γενικά χειρότερη απόδοση και τους μεγαλύτερους χρόνους εκπαίδευσης. Η απλή αρχιτεκτονική των SimpleRNN και Dense επιτρέπει τη δοκιμή πολλών νευρώνων και επιπέδων κάτι που το LSTM δε μπορούσε να επιτύχει καθώς η εκπαίδευση του ήταν σχεδόν στάσιμη για τις μεγαλύτερες υπερπαραμέτρους. Επιπλέον κατάφεραν να εντοπίσουν τα πρότυπα για τη μικρή διάρκεια των 24 ωρών που δοκιμάστηκαν.

Επιτεύχθηκε μικρότερο σφάλμα πρόβλεψης από το καλύτερο μοντέλο (APE 1.44%, RMSE 126MW) με το 63% των ωρών που προβλέφθηκαν να έχουν απόλυτη διαφορά από τις πραγματικές κάτω από 100MW. Τα αποτελέσματα, αν και υποσχόμενα, αφήνουν περιθώριο πειραματισμού και βελτίωσης. Μπορούν να δοκιμαστούν επιπλέον συνδυασμοί από το σύνολο των 15 δυνητικών εισόδων, διαφορετικό μέγεθος επικαλυπτόμενων παραθύρων (6, 12, 48 ώρες) και μεγαλύτερο διάστημα πρόβλεψης από μία ώρα (επόμενες 24, 48, 168 ώρες κλπ.). Επιπλέον, μπορεί να γίνει πειραματισμός στο σύνολο των υπερπαραμέτρων με διαφορετικές τιμές και διαφορετικό μέγεθος για το σύνολο επικύρωσης (15%, 20 %).

12. Βιβλιογραφία

[B1] François Chollet - Deep Learning with Python - Manning Publications (2021)
[B2] Jason Brownlee - Deep Learning for Time Series Forecasting - Predict the Future with MLPs, CNNs and LSTMs in Python, 3-8

13. Δικτυογραφία

[Δ1] http://jmc.stanford.edu/articles/dartmouth/dartmouth.pdf

[Δ2] https://en.wikipedia.org/wiki/Artificial_intelligence

[Δ3] https://en.wikipedia.org/wiki/Machine_learning

[Δ4] https://en.wikipedia.org/wiki/Supervised_learning

[Δ5] https://en.wikipedia.org/wiki/Unsupervised_learning

[Δ6] https://en.wikipedia.org/wiki/Reinforcement_learning

[Δ7] https://en.wikipedia.org/wiki/Deep_learning

[Δ8] https://en.wikipedia.org/wiki/Deep_learning#Deep_neural_networks

[Δ9] https://www.researchgate.net/figure/A-biological-neuron-in-comparison-to-an-artificial-neural-network-a-human-neuron-b_fig2_339446790

[Δ10] https://en.wikipedia.org/wiki/Activation_function

[Δ11] https://en.wikipedia.org/wiki/Overfitting

[Δ12] https://en.wikipedia.org/wiki/Vanishing_gradient_problem

[Δ13] https://www.geeksforgeeks.org/introduction-to-recurrent-neural-network/

[Δ14] https://en.wikipedia.org/wiki/Long_short-term_memory

[Δ15] https://en.wikipedia.org/wiki/Gated_recurrent_unit

[Δ16] https://en.wikipedia.org/wiki/Time_series

[Δ17] https://en.wikipedia.org/wiki/Distribution_management_system#Distribution_Load_Forecasting_%28DLF%29

[Δ18] https://www.kaggle.com/code/cworsnup/backtesting-cross-validation-for-timeseries/notebook

14. Κώδικας

Μέτρα

```
import numpy as np

def rmse(actual, predicted):
	N = actual.shape[0]
	mse = 0
	for h in range(N):
		mse += (actual[h] - predicted[h]) ** 2
		rmse = np.sqrt((1/N) * mse)
	return rmse
	#return np.sqrt(mean_squared_error(actual, predicted))


def ape(actual, predicted):
	# Number of days in the dataset
	N = actual.shape[0]//24
	total_sum = 0

	# Reshape the predicted and actual arrays to have dimensions (number of
days, 24)
	predicted_reshaped = np.reshape(predicted, (N, 24))
	actual_reshaped = np.reshape(actual, (N, 24))

for d in range(N):
		daily_sum = 0
		# Maximum actual load for day d
		L_max_d = np.max(actual_reshaped[d, :])

for h in range(24):
			# APE for day d at hour h
			daily_peak = np.abs(predicted_reshaped[d, h] -
	actual_reshaped[d, h]) / L_max_d

			daily_sum += daily_peak
		total_sum += daily_sum

	ape_value = (1 / N) * (1 / 24) * total_sum * 100

	return ape_value
```

```
def forecast_error_duration_curve(actual, predicted, thresholds):
        # Calculate the absolute differences
        abs_diff = np.abs(actual- predicted)
        # Initialize a dictionary to store the counts and percentages
        results = {}

        # Count the occurrences and calculate percentages for each threshold
        total_samples = len(abs_diff)

        for threshold in thresholds:
                count = np.sum(abs_diff > threshold)
                percentage = (count / total_samples) * 100
                results[f'>{threshold} MW'] = {'Hours': count, 'Time %':
        percentage}

        return results
```

Δοκιμή διαφορετικών συνδυασμών για είσοδο

```
#Test Input Combinations
import tensorflow as tf
import numpy as np
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, LSTM, SimpleRNN, GRU
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.optimizers import Adam

def rmse(actual, predicted):
        return np.sqrt(mean_squared_error(actual, predicted))

def ape(actual, predicted):
        # Number of days in the dataset
        N = actual.shape[0]//24
        total_sum = 0

        # Reshape the predicted and actual arrays to have dimensions                (number of
days, 24)
        predicted_reshaped = np.reshape(predicted, (N, 24))
        actual_reshaped = np.reshape(actual, (N, 24))

for d in range(N):
                daily_sum = 0
                # Maximum actual load for day d
                L_max_d = np.max(actual_reshaped[d, :])

for h in range(24):
                        # APE for day d at hour h
                        daily_peak = np.abs(predicted_reshaped[d, h] -
        actual_reshaped[d, h]) / L_max_d

                        daily_sum += daily_peak
                total_sum += daily_sum

        ape_value = (1 / N) * (1 / 24) * total_sum * 100

        return ape_value


train= '/content/drive/MyDrive/ADMHE_Datasets/15in_1out_2013-
2015trn_2016tst/ADMHE_System_Load_2013-01-01_2015-12-31_training_15in_1out_2013-
2015trn_2016tst.dat'
test = '/content/drive/MyDrive/ADMHE_Datasets/15in_1out_2013-
2015trn_2016tst/ADMHE_System_Load_2016-01-01_2016-12-31_testing_15in_1out_2013-
2015trn_2016tst.dat'

train_data = np.loadtxt(train)
```

```
test_data = np.loadtxt(test)

input_comb = [
        #one day before, all lag values
        {"name": "one_day", "column": (0, 1, 2, 3, 4)},
        #two days before, all lag values
        {"name": "two_days", "column": (5, 6, 7, 8, 9)},
        #seven days before, all lag values
        {"name": "seven_days", "column": (10, 11, 12, 13, 14)},
        # all 'h' columns from one, two, seven days
        {"name": "h_all", "column": (2, 7, 12)},
        # all 'h+1' columns from one, two, seven days
        {"name": "h+1_all", "column": (3, 8, 13)},

        #all 'h+2' columns from one, two, seven days
        {"name": "h+2_all", "column": (4, 9, 14)},

        {"name": "h-1_all", "column": (1, 6, 11)},
        {"name": "h-2_all", "column": (0, 5, 10)},
        # h, h+1 columns from one day
        {"name": "h_h+1_one_day", "column": (2, 3)},
        # all 'h, h+1' columns from one, two days
        {"name": "h_h+1_one_two_days", "column": (2, 3, 7, 8)},
        #all 'h, h+1' columns from one, two, seven days
        {"name": "h_h+1_all", "column": (2, 3, 7, 8, 12, 13)},
        {"name": "h_h+1_h+2_one_day", "column": (2, 3, 4)},
        {"name": "h_h+1_h+2_one_two_days", "column": (2, 3, 4, 7, 8, 9)},      {"name":
"h_h+1_h+2_all", "column": (2, 3, 4, 7, 8, 9, 12, 13,       14)},

        {"name": "h-1_h_one_day", "column": (1, 2)},
        {"name": "h-1_h_one_two_days", "column": (1, 2, 6, 7)},
        {"name": "h-1_h_all", "column": (1, 2, 6, 7, 11, 12)},

        {"name": "h-2_h-1_h_one_day", "column": (0, 1, 2)},
        {"name": "h-2_h-1_h_two_days", "column": (5, 6, 7)},
        {"name": "h-2_h-1_h_seven_days", "column": (10, 11, 12)},
        {"name": "h-2_h-1_h_one_two_days", "column": (0, 1, 2, 5, 6,    7)},
        {"name": "h-2_h-1_h_all", "column": (0, 1, 2, 5, 6, 7, 10, 11,      12)},

        {"name": "h-1_h_h+1_one_day", "column": (1, 2, 3)},
        {"name": "h-1_h_h+1_one_two_days", "column": (1, 2, 3, 6, 7, 8)},

        {"name": "all", "column": (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11,        12, 13,14},

        # Tested inputs from Kandilogianakis' Documentation
        {"name": "doc1", "column": (5, 6, 7, 8)},
        {"name": "doc2", "column": (5, 6, 8)},
        {"name": "doc3", "column": (5, 8)},
        {"name": "doc4", "column": (2, 12)},
```

```
        {"name": "doc5", "column": (1, 2, 12)},
        {"name": "doc6", "column": (2, )},

]
layer_name = [Dense, LSTM, GRU, SimpleRNN]

opt = Adam(learning_rate=0.0001)

my_callbacks = [EarlyStopping(monitor='val_loss', patience=3)]

executions = 5

with open("/content/drive/MyDrive/results10.txt", "a") as file:
        file.write(
                f"Model;Shape;Comb_name;Comb_col;\

                {';'.join([f'Rmse_trn{i}' for i in
        range(executions)])};Rmse_train_avg;\
                {';'.join([f'Ape_trn{i}' for i in
        range(executions)])};Ape_train_avg;\
                {';'.join([f'Rmse_tst{i}' for i in
        range(executions)])};Rmse_test_avg;\
                {';'.join([f'Ape_tst{i}' for i in
        range(executions)])};Ape_test_avg\n"
        )

        # Loop all layers in layer_name
        for l in layer_name:

                # Loop all combinations in input_comb
                for comb in input_comb:

                        results_ape_training = []
                        results_rmse_training = []

                        results_ape_testing = []
                        results_rmse_testing = []

                        X_train = train_data[:, comb['column']]
                        y_train = train_data[:, -1]

                        X_test = test_data[:, comb['column']]
                        y_test = test_data[:, -1]

                        # Reshape the train data to (samples, timesteps,
        features) if not Dense
                        if l != Dense:

                                X_train = X_train.reshape(X_train.shape[0],
```

```
                    X_train.shape[1], 1)

                    X_test = X_test.reshape(X_test.shape[0],
        X_test.shape[1], 1)

                    input_shape = (len(comb['column']), 1)

                else :

                    input_shape = (len(comb['column']), )

                    # print (X_train.shape, y_train.shape)

                print (f"{l.__name__};{X_train.shape};   {comb['name']};
        {comb['column']}")

                file.write(f"{l.__name__};{X_train.shape};
        {comb['name']};{comb['column']};")


                # Train and test each model 5 times to get a mean                of the
testing scores

                for i in range(executions):
                    model = Sequential()
                    model.add(l(units = 50, activation = 'relu',
        input_shape = input_shape))

                    model.add(Dense(1))
                    model.compile(
                        loss='mse', optimizer=opt,
                        metrics=['mape']
                    )

                    model.fit(
                        X_train,
                        y_train,
                        epochs=500,
                        batch_size=24,
                        validation_split = 0.1,
                        callbacks = my_callbacks, verbose = 0)


                    y_train_pred = model.predict(X_train)
                    y_test_pred= model.predict(X_test)

                    train_rmse = rmse(y_train, y_train_pred)
                    test_rmse = rmse(y_test, y_test_pred)
```

```
                    train_ape = ape(y_train, y_train_pred)
                    test_ape = ape(y_test, y_test_pred)

                    print (train_rmse,train_ape,test_rmse,test_ape)

                    results_ape_training.append(train_ape)
                    results_rmse_training.append(train_rmse)

                    results_ape_testing.append(test_ape)
                    results_rmse_testing.append(test_rmse)

                total_ape_training = np.mean(results_ape_training)
                total_rmse_training = np.mean(results_rmse_training)

                total_ape_testing = np.mean(results_ape_testing)
                total_rmse_testing = np.mean(results_rmse_testing)

                file.write(";".join([f"{result:.3f}" for result in
results_rmse_training]) + f";
{total_rmse_training:.3f};")

                file.write(";".join([f"{result:.3f}" for result in
results_ape_training]) +f";
{total_ape_training:.3f};")

                file.write(";".join([f"{result:.3f}" for result in
results_rmse_testing]) + f";
{total_rmse_testing:.3f};")
                file.write(";".join([f"{result:.3f}" for result in
results_ape_testing]) +f";{total_ape_testing:.3f}\n")

        file.write("\n")
```

Keras tuner για Προσεγγιση 1

```
# Tuning for basic models
import keras_tuner
import tensorflow as tf import numpy as np
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, LSTM, SimpleRNN, GRU, InputLayer
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.optimizers import Adam

class MyHyperModel(keras_tuner.HyperModel):

        def build(self, hp):

                model = keras.Sequential()
                model.add(InputLayer(input_shape=(2, 1))) # Replace with the shape of
combination

                # Tune the number of layers.
                num_layers = hp.Int("num_layers", 1, 3)


                for i in range(num_layers):

                        # Required for stacked RNNs. False for last layer
        num_layer= 1
                        return_sequences=((num_layers > 1) and (i < num_layers
        - 1))

                        model.add( GRU(# Replace with desired model
                                units=hp.Choice(f"units_{i}",[25,50,100,200]),
                                activation='relu',
                                # comment next line for Dense Layer
                                return_sequences=return_sequences
                                )
                        )
                        # if hp.Boolean("dropout"):
                                #model.add(layers.Dropout(rate=hp.Choice("dr",
        [0.2, 0.5])))

                        model.add(layers.Dense(1))

                        learning_rate=hp.Choice("lr",[1e-5, 1e-4, 1e-3,1e-2])

                        model.compile(
                                optimizer=Adam(learning_rate=learning_rate),
                                loss="mse",
                                metrics=["mape"],
```

```
                    )

        return model


        def fit(self, hp, model, *args, **kwargs):
                return model.fit(
                        *args,

                        #Tune batch size
                        batch_size=hp.Choice("batch_size", [6, 12, 24,
                        36,48, 168]),

                        **kwargs,
                    )

train_data = np.loadtxt('/content/drive/MyDrive/ADMHE_Datasets/15in_1out_2013-
2015trn_2016tst/ADMHE_System_Load_2013-01-01_2015-12-31_training_15in_1out_2013-
2015trn_2016tst.dat')
test_data = np.loadtxt('/content/drive/MyDrive/ADMHE_Datasets/15in_1out_2013-
2015trn_2016tst/ADMHE_System_Load_2016-01-01_2016-12-31_testing_15in_1out_2013-
2015trn_2016tst.dat')


X_train = train_data[:, (2, 12)] # Replace with each model's best combination
y_train = train_data[:, -1]

X_test = test_data[:, (2, 12)] # Replace with each model's best combination
y_test = test_data[:, -1]

# Reshape to (samples, timesteps, features) for RNNs. Comment for Dense
X_train = X_train.reshape(X_train.shape[0], X_train.shape[1], 1)
X_test = X_test.reshape(X_test.shape[0], X_test.shape[1], 1)


tuner = keras_tuner.BayesianOptimization(
        MyHyperModel(),
        max_trials=100,
        executions_per_trial=3,
        # resume the previous search in the same directory.
        overwrite=False,
        objective=val_loss",
        # Set a directory to store the intermediate results.
        directory="/content/drive/MyDrive/keras_tune/GRU/Bayesian"

)
```

```
tuner.search(
        X_train,
        y_train,
        validation_split=0.1,
        epochs=200,
        callbacks=[EarlyStopping(patience = 4)],
)


tuner.results_summary()
```

Keras tuner για Προσεγγιση 2

```
# SLIDING WINDOWS
# Tuning for Sliding Windows 24


import keras_tuner
import tensorflow as tf import numpy as np
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, LSTM, SimpleRNN, GRU, InputLayer
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.optimizers import Adam

class MyHyperModel(keras_tuner.HyperModel):

        def build(self, hp):

                model = keras.Sequential()
                model.add(InputLayer(input_shape=(24, 1))) #(24,) for Dense

                # Tune the number of layers.
                num_layers = hp.Int("num_layers", 1, 3)


                for i in range(num_layers):

                        # Required for stacked RNNs. False for last layer
        num_layer= 1
                        return_sequences=((num_layers > 1) and (i < num_layers
        - 1))

                        model.add( GRU(# Replace with desired model
                                units=hp.Choice(f"units_{i}",[25,50,100,200]),
                                activation='relu',
                                # comment next line for Dense Layer
                                return_sequences=return_sequences
                                )
                        )
                        # if hp.Boolean("dropout"):
                                #model.add(layers.Dropout(rate=hp.Choice("dr",
        [0.2, 0.5])))

                        model.add(layers.Dense(1))

                        learning_rate=hp.Choice("lr",[1e-5, 1e-4, 1e-3,1e-2])

                        model.compile(
                                optimizer=Adam(learning_rate=learning_rate),
```

```
                        loss="mse",
                        metrics=["mape"],
                    )

        return model



        def fit(self, hp, model, *args, **kwargs):
            return model.fit(
                        *args,

                        #Tune batch size
                        batch_size=hp.Choice("batch_size", [6, 12, 24,
                        36,48, 168]),

                        **kwargs,
        )

        def create_sliding_windows(data, time_steps, stride=1):

            X, y = [], []

            for i in range(0, len(data) - time_steps,stride):
                X.append(data[i: i + time_steps, -1])
                y.append(data[i+time_steps, -1])

            return np.array(X), np.array(y)


train_data = np.loadtxt('/content/drive/MyDrive/ADMHE_Datasets/15in_1out_2013-
2015trn_2016tst/ADMHE_System_Load_2013-01-01_2015-12-31_training_15in_1out_2013-
2015trn_2016tst.dat')
test_data = np.loadtxt('/content/drive/MyDrive/ADMHE_Datasets/15in_1out_2013-
2015trn_2016tst/ADMHE_System_Load_2016-01-01_2016-12-31_testing_15in_1out_2013-
2015trn_2016tst.dat')

X_train, y_train = create_sliding_windows(train_data, 24)
X_test, y_test = create_sliding_windows(test_data, 24)

# Reshape to (samples, timesteps, features) for RNNs. Comment for Dense
X_train = X_train.reshape(X_train.shape[0], X_train.shape[1], 1)
X_test = X_test.reshape(X_test.shape[0], X_test.shape[1], 1)


tuner = keras_tuner.BayesianOptimization(
        MyHyperModel(),
        max_trials=100,
        executions_per_trial=3,
```

```
        # resume the previous search in the same directory.
        overwrite=False,
        objective=val_loss",
        # Set a directory to store the intermediate results.
        directory="/content/drive/MyDrive/keras_tune/GRU/Sliding_window"

)


tuner.search(
        X_train,
        y_train,
        validation_split=0.1,
        epochs=200,
        callbacks=[EarlyStopping(patience = 4)],
)


tuner.results_summary()
```

Τελικά μοντέλα

```
import numpy as np
from keras.models import Sequential
from keras.layers import Input, Dense, LSTM, SimpleRNN, GRU,
from keras.callbacks import EarlyStopping
from keras.optimizers import Adam
from keras.losses import MeanSquaredError
from keras.metrics import RootMeanSquaredError
import pandas as pd

import sys
sys.path.append("/content/drive/MyDrive")
from my_metrics import rmse, ape, forecast_error_duration_curve


# Function to create sequences of input data and target labels

def create_sliding_windows(data, time_steps, stride=1):

        X, y = [], []

        for i in range(0, len(data) - time_steps,stride):
                X.append(data[i: i + time_steps, -1])
                y.append(data[i+time_steps, -1])

        return np.array(X), np.array(y)



def write_to_excel(file, newfile, data, position, model_name):

        # Condition for sliding windows approach # The first 24 values are    missing,they are used
as lag values
        # It is required to avoid size error

        missing = len(file) - len(data)
        if missing > 0:
                # Fill with Nan
                fill = [np.nan] * missing
                data = np.concatenate((fill, data.flatten()))

        # Insert predictions to metadata excel file, on
        #'position'column.Position = 1 is second column
        file.insert(position, f"Predicted Load {model_name}", data)

        #Rename from Actual Load - TESTING (or TRAINING) column to                "Actual Load"
        file.rename(columns={file.columns[0]: 'Actual Load'},                    inplace=True)
```

```
        # Save the modified DataFrame to a new Excel file
        file.to_excel(f"{newfile}.xlsx", index=False)


train_data = np.loadtxt( '/content/drive/MyDrive/ADMHE_Datasets/15in_1out_2013-2015trn_2016tst/ADMHE_System_Load_2013-01-01_2015-12-31_training_15in_1out_2013-2015trn_2016tst.dat', delimiter=' ')


test_data = np.loadtxt( '/content/drive/MyDrive/ADMHE_Datasets/15in_1out_2013-2015trn_2016tst/ADMHE_System_Load_2016-01-01_2016-12-31_testing_15in_1out_2013-2015trn_2016tst.dat', delimiter=' ')


meta_data_train = pd.read_excel( '/content/drive/MyDrive/Actual Load - TRAINING_with_metadata.xlsx')

meta_data_test = pd.read_excel( '/content/drive/MyDrive/Actual Load - TESTING_with_metadata.xlsx')


my_callbacks = [EarlyStopping(monitor='val_loss', patience=5)]

windows_size = 24



# Initialize the hyperparameters for each model
# lag values are the columns chosen from the tested inputs
# hyperparametes are the result of keras tuner

models = {

        "Dense": {
                "layer": Dense,
                "lag_values": (2, 3, 4, 7, 8, 9, 12, 13, 14),
                "num_layers": 3,
                "units": (200, 200, 200),
                "batch_size": 6,
                "lr": 0.0001
        },

        "Dense_24": {
                "layer": Dense,
                "num_layers": 2,
                "units": (100, 100),
```

```
        "batch_size": 6,
        "lr": 0.0001
},

"LSTM": {
        "layer": LSTM,
        "lag_values": (2, 12),
        "num_layers": 2,
        "units": (100, 100),
        "batch_size": 12,
        "lr": 0.0001
},

"LSTM_24": {
        "layer": LSTM,
        "num_layers": 1,
        "units": (200, ),
        "batch_size": 6,
        "lr": 0.0001
},

"SimpleRNN": {
        "layer": SimpleRNN,
        "lag_values": (2, 3, 7, 8, 12, 13),
        "num_layers": 2,
        "units": (100, 100),

        "batch_size": 48,
        "lr": 0.0001
},

"SimpleRNN_24": {
        "layer": SimpleRNN,
        "num_layers": 3,
        "units": (25, 200 ,100),
        "batch_size": 6,
        "lr": 0.00001
},

"GRU": {
        "layer": GRU,
        "lag_values": (2, 12),
        "num_layers": 2,
        "units": (200, 100,),
        "batch_size": 6,
        "lr": 0.00001
},

"GRU_24": {
```

```
            "layer": GRU,
            "num_layers": 1,
            "units": (200, ),
            "batch_size": 6,
            "lr": 0.0001
        },
}


# Train each from the above directory and save the predictions on excels

for model_name, hyperparameters in models.items():
        layer = hyperparameters["layer"]
        num_layers = hyperparameters["num_layers"]
        units = hyperparameters["units"]
        batch_size = hyperparameters["batch_size"]
        lr = hyperparameters["lr"]


        # Prepare the data for each approach

        if model_name in ["Dense_24", "LSTM_24", "SimpleRNN_24", "GRU_24"]:

                # Create the sliding windows samples of size 24
                X_train, y_train = create_sliding_windows(train_data,                   windows_size)
                X_test, y_test = create_sliding_windows(test_data,                      windows_size)

        else:
        # Create the input / outputs samples based on each model's best inputs
                lag_values = hyperparameters["lag_values"]

                X_train = train_data[:, lag_values]
                X_test = test_data[:, lag_values]

                y_train = train_data[:, -1]
                y_test = test_data[:, -1]


        model = Sequential()

        if layer == Dense :
                # Define the input shape
                model.add(Input(shape=X_train.shape[1:]))

                # Add the required hidden layer
                for num in range(num_layers):
                        model.add(layer(
                                units = units[num],
```

```
                        activation = 'relu',
                        )
                )

else:
        # Reshape from (samples, features) to (samples, timesteps,
features) for RNN


        X_train=X_train.reshape(X_train.shape[0],
X_train.shape[1],1)

        X_test= X_test.reshape(X_test.shape[0],
X_test.shape[1], 1)

        model.add(Input(shape=X_train.shape[1:]))

        for num in range(num_layers):
                #False for one layer or last layer
                return_sequences = ((num_layers > 1) and
(num < num_layers - 1))

                print (model_name, num_layers, units[num],
return_sequences)

                model.add(layer(
                        units[num],
                        activation='relu',
                        return_sequences = return_sequences,
                        )
                )

# print (model_name, X_train.shape, y_train.shape, X_test.shape,        y_test.shape)

model.add(Dense(1))

model.compile(
        loss='mse',
        optimizer=Adam(learning_rate=lr),
        metrics=['mape', RootMeanSquaredError()]
)

model.summary()

# Train the model on the training data
model.fit(
        X_train,
        y_train,
        epochs=500,
```

```
            shuffle=True,
            validation_split=0.1,
            batch_size=batch_size,
            callbacks=my_callbacks,
            verbose=2 )

# Make predictions and calculate scores
y_train_pred = model.predict(X_train)
y_test_pred = model.predict(X_test)

train_rmse = rmse(y_train, y_train_pred)
test_rmse = rmse(y_test, y_test_pred)

train_ape = ape(y_train, y_train_pred)
test_ape = ape(y_test, y_test_pred)

print(train_ape, test_ape, train_rmse, test_rmse)

write_to_excel(
            meta_data_train,
            "/content/drive/MyDrive/train_pred_meta",
            y_train_pred,
            1,
            model_name
)

write_to_excel(
            meta_data_test,
            "/content/drive/MyDrive/test_pred_meta",
            y_test_pred,
            1,
            model_name
)
```

Γραφικές Παραστάσεις και εποχικά αποτελέσματα

```
# excel handling & Graphs


import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import sys

sys.path.append("/content/drive/MyDrive/")
from my_metrics import rmse, ape, forecast_error_duration_curve



def seasonal_scores(excel_file, model, season):

        seasonal_results = {}

        for season_id, season_name in seasons.items():

                # Filter the data for the current season
                filtered_data = excel_file[excel_file['Season'] ==                    season_id]

                # Extract the values from the filtered columns

                actual_load_season = filtered_data['Actual Load'].values
                predicted_load_season = filtered_data[
                f'Predicted Load {model}'].values


                # Calculate RMSE & APE for the current season

                rmse_score = rmse(
                        actual_load_season, predicted_load_season
                )

                ape_score = ape(actual_load_season, predicted_load_season)

                seasonal_results[season_name]={
                        "APE": ape_score,
                        "RMSE": rmse_score
                        }

                # print(
                        f"Test scores for {season_name}:
                                RMSE: {rmse_score:.3f},
```

```
                                        APE: {ape_score:.3f}"
                )

        # RMSE AND APE for whole year
        year_rmse_score = rmse(
                excel_file['Actual Load'].values,
                excel_file[f'Predicted Load {model}'].values
        )

        year_ape_score = ape(
                excel_file['Actual Load'].values,
                excel_file[f'Predicted Load {model}'].values
        )


        seasonal_results["Total"] = {
                "APE": year_ape_score,
                "RMSE": year_rmse_score
        }

        return seasonal_results



# Load the Excel files
train_file = "/content/drive/MyDrive/train_pred_meta_4.xlsx"
test_file = "/content/drive/MyDrive/test_pred_meta_4.xlsx"

train = pd.read_excel(train_file)
test = pd.read_excel(test_file)



# Define the models

models = ["Dense", "SimpleRNN", "LSTM", "GRU"]

models_24 = ["Dense_24", " SimpleRNN_24", "LSTM_24", "GRU_24"]

# Define the thresholds for forecast duration error curve

thresholds = [100, 200, 400, 500]


seasons = {
        1: "Spring",
        2: "Summer",
        3: "Autumn",
```

```
        4: "Winter"
}

dates_and_events = {

        "winter_week": ('2016-01-11', '2016-01-17'),

        "assumption_day": '2016-08-15',

        "christmas_day": '2016-12-25',

        "working_day_winter": '2016-01-14',

        "sunday_winter": '2016-01-17',

        "working_day_spring": '2016-03-02',

        "sunday_spring": '2016-03-06',

        "working_day_summer": '2016-07-14',

        "sunday_summer": '2016-07-17',

        "working_day_autumn": '2016-10-27',

        "sunday_autumn": '2016-10-30'

}

sliding_windows_size = 24


# Ignore all 24 rows with NaN values (due to sliding windows lag values #missing) so that the
model comparison is fair

test = test.drop(index=range(0, sliding_windows_size))

train = train.drop(index=range(0, sliding_windows_size))


# Column with Actual Load

actual = test['Actual Load'].values

#Print model seasonal results and forecast error duration curve

        for model in models+models_24:
```

```
                predicted = test[f'Predicted Load {model}'].values
                seasonal_train = pd.DataFrame(
                        seasonal_scores(train, model, seasons)
                )

                seasonal_test = pd.DataFrame(
                        seasonal_scores(test, model, seasons)
                )

                print (f"[{model}]\n")

                print (f"Seasonal Results\n\
                        Train:\n{seasonal_train}\n\
                        Test:\n{seasonal_test}\n")


                fedc = pd.DataFrame(
                        forecast_error_duration_curve(
                                actual, predicted, thresholds
                        )
                )

                print(f"Forecast Error Duration Curve\n{fedc}\n\n")

                # print("Seasonal Results")

                # for season, scores in seasonal_test.items():

                        # rmse_score, ape_score = scores
                        # print(f"{season}: RMSE: {rmse_score:.3f},
                                #APE: {ape_score:.3f}")

                # print("\n")


                # for MW, values in fedc.items():

                        # hours, percentage = values
                        # print(f"Error >{MW}: hours {hours},
                                percentage {percentage}")
                        # print("\n")

# Plot All the dates for basic models and _24 models

for event, date in dates_and_events.items():

        if isinstance(date, tuple):# Check if it's a date range (winter week)
```

```
                start_date, end_date = date

                event_data = test[(test['Date'] >= start_date) &                (test['Date'] <=
end_date)]

        else:

                event_data = test[test['Date'] == date]
# Create a new figure for the current event and date

        plt.figure(figsize=(15, 8))

        actual = event_data['Actual Load'].values

        # First subplot
        plt.subplot(2, 1, 1)

        # Plot all basic models along with actual values
        for model in models:

                predicted = event_data[f'Predicted Load {model}'].values
                plt.plot(predicted, label=f'Pred - {model}')

        plt.plot(
                range(len(actual)), actual, label='Actual', color='black'
        )
        plt.xlabel('Hours')
        plt.ylabel('Load')
        plt.title(f'{event} - {date} (basic models)')
        plt.legend()


        if len(actual) == 168:

                plt.xticks(range(0, 168, 24))

        else:

                ticks = range(1, 25) # Start at 1, end at 24
                labels = [f"{hour}:00" for hour in ticks]

                plt.xticks(range(24), labels)


        # Second subplot
        plt.subplot(2, 1, 2)

        # Plot models_24 along with actual values
```

```
for model in models_24:

        predicted = event_data[f'Predicted Load {model}'].values
        plt.plot(predicted, label=f'Predicted - {model}')

plt.plot(
        range(len(actual)), actual, label='Actual', color='black'
)
plt.xlabel('Hours')
plt.ylabel('Load')
plt.title(f'{event} - {date} (sliding windows models)')
plt.legend()

if len(actual) == 168:
        plt.xticks(range(0, 169, 24))

else:
        ticks = range(1, 25) # Start at 1, end at 24
        labels = [f"{hour}:00" for hour in ticks]
        plt.xticks(range(24), labels)

# Adjust layout to prevent overlapping of titles and labels
plt.tight_layout()

# Show all figures
plt.show()
```