Εθνικο Μετσοβιο Πολυτεχνειο
Σχολη Ηλεκτρολογων Μηχανικων
και Μηχανικων Υπολογιστων
Τομεας Επικοινωνιων, Ηλεκτρονικης και Συστηματων
Πληροφορικης

Μελέτη και Υλοποίηση Μεθόδων Βαθιάς Μηχανικής Μάθησης για την Πρόβλεψη Δικτυακής Κίνησης

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΚΑΛΑΪΤΖΗΣ ΑΛΕΞΑΝΔΡΟΣ

Επιβλέπων: Συμεών Παπαβασιλείου

Καθηγητής Ε.Μ.Π

Εργαστήριο Διαχείρισης και Βέλτιστου Σχεδιασμού Δικτύων Τηλεματικής

Αθήνα, Ιούλιος 2023

Εθνικο Μετσοβιο Πολυτεχνειο
Σχολη Ηλεκτρολογων Μηχανικων
και Μηχανικων Υπολογιστων
Τομεας Επικοινωνιων, Ηλεκτρονικης και Συστηματων
Πληροφορικης

Μελέτη και Υλοποίηση Μεθόδων Βαθιάς Μηχανικής Μάθησης για την Πρόβλεψη Δικτυακής Κίνησης

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΑΛΕΞΑΝΔΡΟΣ ΚΑΛΑΪΤΖΗΣ

Επιβλέπων: Συμεών Παπαβασιλείου

Καθηγητής Ε.Μ.Π

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 14η Ιουλίου 2023.

…….	…….	…….
Συμεών Παπαβασιλείου	Βασίλειος Καρυώτης	Ιωάννα Ρουσσάκη
Καθηγητής Ε.Μ.Π.	Αν. Καθηγητής Ιονίου Πανεπιστημίου	Αν. Καθηγήτρια Ε.Μ.Π.

Εργαστήριο Διαχείρισης και Βέλτιστου Σχεδιασμού Δικτύων Τηλεματικής

Αθήνα, Ιούλιος 2023

Εθνικο Μετσοβιο Πολυτεχνειο
Σχολη Ηλεκτρολογων Μηχανικων
και Μηχανικων Υπολογιστων
Τομεας Επικοινωνιων, Ηλεκτρονικης και Συστηματων
Πληροφορικης

.....
Αλέξανδρος Καλαϊτζής

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π

Περίληψη

Η γνώση της δικτυακής κίνησης έχει κομβική σημασία στις μέρες μας. Ωστόσο, παρά την σημασία της παρακολούθησης της κίνησης, οι δικτυακές συσκευές δεν έχουν την δυνατότητα της μέτρησης όλων των δεδομένων σε πραγματικό χρόνο. Για τον λόγο αυτό είναι πολύ σημαντική η πρόβλεψη της μελλοντικής κίνησης του δικτύου. Πολλές λειτουργίες διαχείρισης δικτύου, όπως ο σχεδιασμός του δικτύου και η διαμόρφωση των πολιτικων δρομολόγησης, βασίζονται στην πρόβλεψη μελλοντικής κίνησης. Η κύρια δυσκολία του προβλήματος της πρόβλεψης δικτυακής κίνησης είναι το πως θα μοντελοποιηθούν οι σχέσεις μεταξύ του συνόλου δεδομένων κίνησης, ώστε να μπορεί κανείς να προβλέψει τη μελλοντική κυκλοφορία. Ένας Πίνακας Κίνησης (Traffic Matrix) περιέχει τον όγκο της κυκλοφορίας μεταξύ όλων τον κόμβων ενός δικτύου. Η ανίχνευση και επεξεργασία των χρονικών και χωρικών εξαρτήσεων των πινάκων κίνησης είναι απαραίτητη για την λύση αυτού του προβλήματος. Στην παρούσα διπλωματική εργασία αρχικά περιγράφουμε τη δομή του Πίνακα Κίνησης και τις εφαρμογές του. Έπειτα, περιγράφουμε το πρόβλημα της Εκτίμησης Πίνακα Κίνησης (Traffic Matrix Estimation) και κάνουμε μια αναλυτική αναφορά στον τρόπο αντιμετώπισης αυτού του προβλήματος από παλαιότερες εργασίες. Η κύρια συνεισφορά της εργασίας αυτής είναι η αντιμετώπιση του προβλήματος της Πρόβλεψης Δικτυακής Κίνησης (Traffic Matrix Prediction). Αρχικά, αναφέρονται διάφορες εργασίες και οι τεχνικές που χρησιμοποιήθηκαν για την αντιμετώπιση αυτού του προβλήματος. Οι τεχνικές αυτές διαχωρίζονται σε γραμμικά μοντέλα και σε μοντέλα Νευρωνικών Δικτύων (Neural Networks). Εμείς ασχολούμαστε κυρίως με τα μοντέλα Νευρωνικών Δικτύων καθώς αυτά έχουν την δυνατότητα να ανιχνεύσουν τις χωροχρονικές εξαρτήσεις των δεδομένων. Για τον λόγο αυτό αναλύουμε το θεωρητικό υπόβαθρο των Νευρωνικών Δικτύων και γενικότερα της Μηχανικής Μάθησης (Machine Learning). Στο πρακτικό μέρος της εργασίας, με τη βοήθεια της Βαθιάς Μάθησης (Deep Learning) αναπτύσσουμε και εκπαιδεύουμε 6 μοντέλα, ώστε να επιτευχθούν οι καλύτερες δυνατές προβλέψεις δικτυακής κίνησης. Τα μοντέλα αυτά είναι το ConvLSTM, το CNN-LSTM, το LSTM, το BiLSTM, το GRU και το SimpleRNN. Έπειτα, αξιολογούμε της απόδοσης των προτεινόμενων μεθόδων, χρησιμοποιώντας δύο διαθέσιμα σύνολα δεδομένων που έχουν καταγραφεί σε δύο πραγματικά δίκτυα. Τέλος, αναλύουμε την επίδοση των μοντέλων ως προς την ακρίβεια πρόβλεψης και ως προς τον χρόνο εκπαίδευσης προκειμένου να είναι ξεκάθαρη η αποτελεσματικότητα, οι χρονικές απαιτήσεις και ο όγκος των πόρων που απαιτεί κάθε μοντέλο.

Λέξεις Κλειδιά:

Πρόβλεψη Πίνακα Κίνησης, Πίνακας Κίνησης, Πρόβλεψη Χρονοσειράς, Τεχνητή Νοημοσύνη, Μηχανική Μάθηση, Βαθιά Μάθηση, Νευρωνικά Δίκτυα, Νευρωνικά Δίκτυα με Πρόσθια Τροφοδότηση, Συνελικτικά Νευρωνικά Δίκτυα, Επαναλαμβανόμενα Νευρωνικά Δίκτυα, ConvLSTM, CNN-LSTM, LSTM, BiLSTM, GRU, SimpleRNN

Abstract

Knowledge of network traffic is crucial nowadays. Despite the importance of traffic monitoring, network devices do not have the ability to measure all data in real time. Therefore, network traffic prediction is very important. Many network management tasks, such as network planning and routing policies, are based on network traffic prediction. The main difficulty of the network traffic prediction is how to model the dependencies among traffic dataset so that we can predict network traffic. Traffic Matrix contains the volume of traffic between all nodes of a network. Detection and processing of temporal and spatial dependencies of traffic matrices is essential for the solution of this problem. In this diploma thesis we first describe the structure of the Traffic Matrix and its applications. Then, we describe the problem of Traffic Matrix Estimation and make detailed references to prior works dealing with this problem. The foremost contribution of this thesis is the treatment of the problem of Network Traffic Prediction. Firstly, we make references to works and techniques used to deal with this problem. These techniques are divided into linear models and Neural Networks models. We will mainly deal with Neural Network models because they have the ability to detect the spatiotemporal dependencies of the data. For this reason, we analyze the theoretical background of Neural Networks and Machine Learning. In the experimental part of the work, we develop and train 6 Deep Learning models, in order to achieve the best predictions of network traffic. These models are ConvLSTM, CNN-LSTM, LSTM, BiLSTM, GRU and SimpleRNN. Then, the performance of the proposed methods is evaluated using two available datasets recorded in two real networks. Finally, we analyze the performance of the models in terms of prediction accuracy and training time in order to capture the efficiency, time requirements and amount of resources required by each model.

Keywords:

Traffic Matrix Prediction, OD Traffic Matrix, Time Series Forecasting, Artificial Intelligence, Machine Learning, Deep Learning, Neural Networks, Feedforward Neural Networks, Convolutional Neural Networks, Recurrent Neural Networks, ConvLSTM, CNN-LSTM, LSTM, BiLSTM, GRU, SimpleRNN

Ευχαριστίες

Ολοκληρώνοντας τις προπτυχιακές σπουδές μου στο Εθνικό Μετσόβιο Πολυτεχνείο, θα ήθελα να εκφράσω τις ευχαριστίες μου προς τον κ. Συμεών Παπαβασιλείου για τη δυνατότητα που μου έδωσε να εκπονήσω τη διπλωματική μου εργασία σε ένα ιδιαίτερα ενδιαφέρον και σημαντικό τομέα, όπως είναι αυτός της βαθιάς μηχανικής μάθησης. Επιπλέον, θα ήθελα να ευχαριστήσω τον υποψήφιο διδάκτορα κ. Γρηγόρη Κακκάβα η συμβολή του οποίου ήταν καθοριστική και διαρκής, χωρίς την οποία δεν θα ήταν δυνατή η ολοκλήρωση της παρούσας εργασίας. Κλείνοντας, θέλω να ευχαριστήσω ιδιαιτέρως την οικογένειά μου και τους φίλους μου για την ουσιαστική και αμέριστη στήριξη που μου παρείχαν καθ'όλη την διάρκεια των σπουδών μου.

Αλέξανδρος Καλαϊτζής,
Αθήνα, Ιούλιος 2023

Περιεχόμενα

1. Εισαγωγή

Το διαδίκτυο έχει γίνει πλέον μέρος της καθημερινότητας μας. Η χρησιμότητα του σε όλους τους τομείς είναι πολλαπλή καθώς επιτρέπει την άνευ χρονικών και τοπικών περιορισμών επικοινωνία και αναζήτηση πληροφοριών. Στον τομέα των επικοινωνιών το διαδίκτυο και οι εφαρμογές του έχουν γίνει ο κυριότερος τρόπος επικοινωνίας. Η ταχεία αύξηση ροής δεδομένων από τις συσκευές δημιουργεί τεράστια κίνηση στο δίκτυο, η οποία είναι δύσκολο να μετρηθεί και να αξιολογηθεί λόγω της έλλειψης κεντρικής διαχείρισης. Η κίνηση αυτή περνάει από κινητά τηλέφωνα, υπολογιστές και δρομολογητές με διαφορετικά πρωτόκολλα και λειτουργικά συστήματα. Σε αυτό το περιβάλλον, μία από τις πιο δύσκολες εργασίες είναι η εξερεύνηση των τηλεπικοινωνιών και υπολογιστικών υποδομών και η κατανόηση των χαρακτηριστικών της. Είναι αδύνατο για κάθε κόμβο να συλλέγει να επεξεργάζεται και να μεταδίδει όλες αυτές τις πληροφορίες. Τα δεδομένα αυτά είναι πολύ σημαντικά για τους χειριστές του δικτύου οι οποίοι θέλουν να προγραμματίσουν κρίσιμες εργασίες.

Δεδομένης της αυξανόμενης σημασίας του Διαδικτύου, η γνώση της δικτυακής κίνησής έχει γίνει όλο και πιο σημαντική. Για τον λόγο αυτό οι επιστήμονες αναζητούν έναν τρόπο να μπορούν προβλέπουν τη μελλοντική κίνηση των δικτύων με ακρίβεια και ταχύτητα. Πολλές λειτουργίες διαχείρισης δικτύου, όπως ο σχεδιασμός του δικτύου και η διαμόρφωση των πολιτικων δρομολόγησης, βασίζονται στην πρόβλεψη μελλοντικής κίνησης. Η προβλεψιμότητα της κυκλοφορίας εξαρτάται από τα στατιστικά χαρακτηριστικά των παραμέτρων της και την χρονολογική σχέση των εντάσεων της κίνησης. Ένας έγκυρος και έγκαιρος Πίνακας Κίνησης είναι απαραίτητος για διαδικασίες όπως ο άμεσος προγραμματισμός και επαναδρομολόγηση της κυκλοφορίας, η μελλοντική διαχείριση της χωρητικότητας και η ανίχνευση ανωμαλιών κίνησης.

Η κύρια δυσκολία του προβλήματος της πρόβλεψης δικτυακής κίνησης είναι το πως θα μοντελοποιηθούν οι σχέσεις μεταξύ του συνόλου δεδομένων κίνησης, ώστε να μπορεί κανείς να προβλέψει τη μελλοντική κυκλοφορία. Η ανίχνευση και επεξεργασία των χρονικών και χωρικών εξαρτήσεων των πινάκων κίνησης είναι απαραίτητη για την λύση αυτού του προβλήματος. Οι αρχικές μέθοδοι που εξετάστηκαν χρησιμοποίησαν γραμμικά μοντέλα για τη μοντελοποίηση της κίνησης αλλά αδυνατούσαν να συλλάβουν τα μη γραμμικά χαρακτηριστικά της. Έτσι αναπτύχθηκαν πιο σύγχρονα μοντέλα χρονοσειρών βασισμένα στη Μηχανική Μάθηση. Με την βοήθεια της Μηχανικής Μάθησης μοντέλα μπορούν να ανιχνεύσουν τις χωροχρονικές εξαρτήσεις των δεδομένων και να επιτευχθούν πιο ακριβείς προβλέψεις δικτυακής κίνησης.

1.1 Κίνητρα - Στόχοι

Όπως αναφέραμε και στην εισαγωγή η πρόβλεψη της δικτυακής κίνησης είναι απαραίτητη για διάφορες σημαντικές διαδικασίες διαχείρισης του δικτύου. Έχουν γίνει πολλές προσπάθειες πρόβλεψης της μελλοντικής δικτυακής κίνησης με τη χρήση διαφόρων αλγορίθμων. Στη

βιβλιογραφία υπάρχουν πολλές γραμμικές προσεγγίσεις που δεν παρουσιάζουν ικανοποιητικά αποτελέσματα. Τα τελευταία χρόνια όμως, η εξέλιξη στην επιστήμη των ηλεκτρονικών υπολογιστών έχει δημιουργήσει την δυνατότητα να εξεταστεί αυτό το πρόβλημα με τη βοήθεια της Μηχανικής Μάθησης. Στα παρακάτω κεφάλαια θα αναφερθούν αρκετές εργασίες που ανέπτυξαν μοντέλα Μηχανικής Μάθησης προκειμένου να αντιμετωπίσουν αυτό το πρόβλημα. Ωστόσο, η εξέλιξη των μεθόδων δεν έχει τελειώσει και κάθε δίκτυο έχει τις ιδιαιτερότητες του, με αποτέλεσμα η κάθε μέθοδος να χρειάζεται να διαφοροποιηθεί για τις ανάγκες του εκάστοτε προβλήματος. Επίσης, έχει παρατηρηθεί ότι τα αποτελέσματα που παρουσιάζονται στην βιβλιογραφία δεν είναι συνήθως απευθείας συγκρίσιμα μεταξύ τους. Ένα άλλο πρόβλημα που προκύπτει είναι οι πόροι που απαιτεί το κάθε μοντέλο και ο χρόνος εκπαίδευσης του. Στις περισσότερες εργασίες δεν αναφέρονται καθόλου αυτά τα στοιχεία και όταν αναφέρονται είναι αδύνατη η σύγκριση τους με τα αντίστοιχα άλλων ερευνών. Μέσω της εργασίας αυτής θα προσπαθήσουμε να καλύψουμε αυτά τα κενά, ώστε να βγουν χρήσιμα συμπεράσματα.

Ο στόχος της παρούσας διπλωματικής είναι η ανάπτυξη τεχνικών που θα βοηθήσουν στο πρόβλημα της Πρόβλεψης Δικτυακής Κίνησης. Θα αναπτυχθούν διάφορες τεχνικές οι οποίες θα εφαρμοστούν και θα αξιολογηθούν σε σύνολα δεδομένων δύο δικτύων. Η σύγκριση των αποτελεσμάτων και για τα δύο δίκτυα θα μας είναι χρήσιμη στην εξαγωγή συμπερασμάτων σχετικά με την απόδοση τους. Θα αναπτυχθούν μοντέλα που θα εφαρμοστούν με διάφορες παραλλαγές με σκοπό να καταλήξουμε στην καλύτερη εκδοχή του κάθε μοντέλου. Με τον τρόπο αυτό θα γίνει άμεση σύγκριση των μοντέλων πάνω σε ίδια δεδομένα, ώστε να μπορούμε να βγάλουμε ασφαλή συμπεράσματα σχετικά με την απόδοση τους. Οι πόροι και οι δυνατότητες που έχουμε κάθε φορά επηρεάζουν τις συνθήκες του προβλήματος και πιθανόν να επηρεάζουν και την αποτελεσματικότητα των μοντέλων. Σε αυτή την εργασία θα αναλύσουμε την επίδοση των μοντέλων ως προς την ακρίβεια πρόβλεψης, αλλά και ως προς τον χρόνο εκπαίδευσης προκειμένου να είναι ξεκάθαρη η αποτελεσματικότητα, οι χρονικές απαιτήσεις και ο όγκος των πόρων που απαιτεί κάθε μοντέλο.

1.2 Οργάνωση και Διάρθρωση της Εργασίας

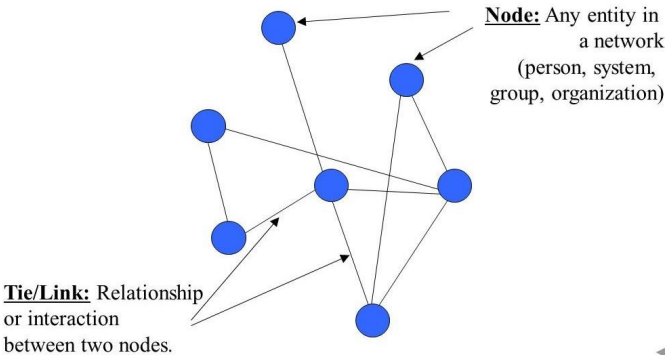
Η παρούσα διπλωματική εργασία κατανέμεται σε 5 κεφάλαια. Στο κεφάλαιο 2, αναλύεται ο Πίνακας Κίνησης (OD Traffic Matrix) και περιγράφονται τα προβλήματα της Εκτίμησης του Πίνακα Κίνησης (Traffic Matrix Estimation) (ενότητα 2.3) και της Πρόβλεψης του Πίνακα Κίνησης (Traffic Matrix Prediction) (ενότητα 2.4). Επίσης, περιγράφονται συνοπτικά παρεμφερείς εργασίες που έχουν προηγηθεί σχετικές με τα δύο προβλήματα που αναφέραμε. Στο κεφάλαιο 3, πρόκειται να περιγραφεί αναλυτικά το θεωρητικό υπόβαθρο, πάνω στο οποίο βασίστηκε η εργασία και αποτελεί βασική προϋπόθεση για την κατανόηση της. Αρχικά, γίνεται αναφορά σε βασικές έννοιες όπως η Τεχνητή Νοημοσύνη (Artificial Intelligence), η Μηχανική Μάθηση (Machine Learning) και η Βαθιά Μάθηση (Deep Learning). Στη συνέχεια,

περιγράφονται τα Νευρωνικά Δίκτυα (Neural Networks) και αναλύονται σε θεωρητικό επίπεδο οι τεχνικές Βαθιάς Μάθησης και τα είδη των νευρωνικών μοντέλων που χρησιμοποιούνται στην παρούσα εργασία. Στο κεφάλαιο 4, περιγράφεται η μεθοδολογία που επιλέχθηκε. Αρχικά, παρουσιάζονται τα μοντέλα και οι παράμετροι που χρησιμοποιήθηκαν σε καθένα από αυτά (ενότητα 4.1). Συγκεκριμένα, γίνεται ανάπτυξη και εκπαίδευση 6 μοντέλων για την επίλυση του προβλήματος της Πρόβλεψης Πίνακα Κυκλοφορίας. Έπειτα, γίνεται αναφορά στα σύνολα δεδομένων και ανάλυση των ρυθμίσεων των πειραμάτων (ενότητα 4.2). Στην τελευταία ενότητα αυτού του κεφαλαίου (ενότητα 4.3) παρουσιάζονται τα αποτελέσματα του πειράματος σε πίνακες και γραφικές. Έτσι, γίνεται συγκριτική αξιολόγηση της απόδοσης των προτεινόμενων μεθόδων, χρησιμοποιώντας δύο διαθέσιμα σύνολα δεδομένων πραγματικών πινάκων κυκλοφορίας που έχουν καταγραφεί στο δίκτυο GEANT [1] και στο δίκτυο Abilene [2]. Στο κεφάλαιο 5, εξάγονται τα συμπεράσματα της διπλωματικής και παρέχοντας ιδέες για την εξέλιξη των προτεινόμενων μοντέλων. Τέλος, πριν τις Βιβλιογραφικές Αναφορές (References) παρατίθεται ο κώδικας της εργασίας.

2. Πίνακας Κίνησης

2.1 Ορισμός

Ο πιο κοινός τρόπος για την αναπαράσταση της συγκεντρωτικής κίνησης σε ένα δίκτυο είναι η κατασκευή πινάκων κίνησης. Ο πίνακας κίνησης (Traffic Matrix) ποσοτικοποιεί την κυκλοφορία μεταξύ όλων των πιθανών ζευγών οντοτήτων προέλευσης και προορισμού [3, 4]. Οι καταχωρήσεις του πίνακα αντιπροσωπεύουν τον όγκο κίνησης που ρέει μεταξύ των κόμβων προελεύσεων και προορισμών [5]. Οι κόμβοι συχνά έχουν φυσική τοποθεσία οπότε μπορούμε να τους θεωρήσουμε χωρικές μεταβλητές. Η κίνηση ξεκινάει από έναν κόμβο και μπορεί να διασχίσει αρκετές ζεύξεις κόμβων (links) μέχρι να φτάσει στον προορισμό. Οι ζεύξεις αυτές των κόμβων συνθέτουν την τοπολογία του δικτύου, όπως βλέπουμε στην Εικόνα 2.1.1, και το μονοπάτι που θα επιλεγεί καθορίζει την δρομολόγηση (routing). Η κίνηση αποτυπώνεται σε τιμές που αντιπροσωπεύουν την μέτρηση των πακέτων των ζεύξεων σε ένα χρονικό περιθώριο που ορίζεται συνήθως μεταξύ 5-15 λεπτών [6, 7].

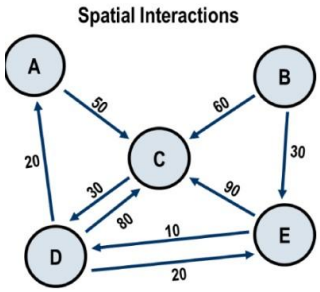


Εικόνα 2.1.1: Κόμβοι (Nodes) και Ζεύξεις (Links) ενός γράφου [8]

Σε ένα δίκτυο n κόμβων και m ζεύξεων ο τρόπος που αναπαρίσταται η κίνηση στο δίκτυο είναι ένας δισδιάστατος πίνακας μεγέθους $n \times n$, όπου το στοιχείο της γραμμής i και της στήλης j συμβολίζει την μετρημένη κίνηση σε πακέτα ή bytes από τον κόμβο i προς τον j. Ο πίνακας κίνησης χρειάζεται άλλη μία διάσταση ώστε να αποθηκεύονται πολλαπλά στιγμιότυπα δικτυακής κίνησης, έτσι καταλήγουμε σε έναν τρισδιάστατο πίνακα X με το στοιχείο $x_{i,j}(t)$ να είναι η ένταση της δικτυακής κίνησης με πηγή τον κόμβο i και προορισμό τον κόμβο j την χρονική στιγμή t [6]. Ενδεχομένως να μπορούμε να αγνοήσουμε την τρίτη διάσταση t του πίνακα και να επικεντρωθούμε στις χωρικές συσχετίσεις των κόμβων ή να παρακολουθήσουμε την πορεία μιας σύνδεσης κόμβων αγνοώντας τις άλλες δύο διαστάσεις αν η εφαρμογή το απαιτεί.

Οι πιο διαδεδομένες μορφές πίνακα κίνησης είναι ο πίνακας OD (Origin-Destination) και ο πίνακας IE (Ingress-Egress). Ο πίνακας IE συμβολίζει την ροή εισόδου και εξόδου ενός router. Αγνοεί τις εσωτερικές επικοινωνίες των διαφόρων συσκευών που ανήκουν στο ίδιο υποδίκτυο

και αποθηκεύει την κίνηση των πακέτων από και προς το κάθε υποδίκτυο. Αντίθετα, ο πίνακας OD περιέχει τις IP που δημιουργούν τα πακέτα και τις IP που προορίζονται αυτά. Συνήθως οι OD πίνακες είναι πιο μεγάλοι διότι η ροή των πακέτων μπορεί να γίνεται μεταξύ χιλιάδων IP αλλά μεταξύ εκατοντάδων routers. Η επιλογή μεταξύ των δύο πινάκων γίνεται με βάση τα χαρακτηριστικά του θέματος που θέλουμε να εξετάσουμε. Οι πίνακες OD χρησιμεύουν στην ανίχνευση μικρών αλλαγών στην ροή της κίνησης και στις ζεύξεις μεταξύ των κόμβων. Οι IE πίνακες δεν εξετάζουν την τοπολογία των υποδικτύων γι'αυτο και χρησιμεύουν σε γενικότερες παρατηρήσεις στην αλλαγή της κυκλοφορίας πακέτων μεταξύ των δικτύων. Στην παρούσα εργασία θα εμβαθύνουμε στους πίνακες OD, όπως στην Εικόνα 2.1.2, και στα συμπεράσματα που μπορούμε να βγάλουμε μέσω αυτών [6].



O/D Matrix

	A	B	C	D	E	Ti
A	0	0	50	0	0	50
B	0	0	60	0	30	90
C	0	0	0	30	0	30
D	20	0	80	0	20	120
E	0	0	90	10	0	100
Tj	20	0	280	40	50	**390**

Εικόνα 2.1.2: Κατασκευή Πίνακα OD από γράφο [9]

2.2 Εφαρμογές

Παρατηρώντας την εξέλιξη του διαδικτύου τα τελευταία χρόνια η γνώση της κίνησης του αποκτά όλο και μεγαλύτερη αξία. Για αυτόν τον λόγο έχουν αναπτυχθεί πολλές εφαρμογές που συλλέγουν και επεξεργάζονται τα δεδομένα κυκλοφορίας των δικτύων. Ο δημοφιλέστερος τρόπος αποθήκευσης και επεξεργασίας αυτών των δεδομένων είναι μέσω των Πινάκων Κίνησης μέσω των οποίων είναι εφικτό να παρατηρήσουμε τα χρονικά μεταβαλλόμενα χαρακτηριστικά κάθε ζεύγους κόμβων [4, 5]. Δυστυχώς, η κατασκευή του πίνακα κίνησης είναι πολύπλοκη και χρονοβόρα λόγω του τεράστιου αριθμού των IP σε ένα δίκτυο και της δυσκολίας μέτρησης της δικτυακής κίνησης [4]. Ωστόσο, υπάρχουν διαθέσιμοι πίνακες κίνησης στο διαδίκτυο, οπως το GEANT [1] και Abilene [2], όπου μπορούν οι επιστήμονες και τα πανεπιστήμια να χρησιμοποιήσουν για εκπαιδευτικούς και ερευνητικούς σκοπούς, προκειμένου να εκπαιδεύσουν τους αλγόριθμους τους σε αληθινά δεδομένα δικτύων [6, 10].

Χρήσιμα συμπεράσματα μπορούν να βγουν και από τεχνητούς πίνακες κίνησης, οι οποίοι συχνά χρησιμοποιούνται για να εξεταστεί αν τα μοντέλα ανταποκρίνονται σε μεγάλο σύνολο δεδομένων. Δοκιμάζονται, με αυτόν τον τρόπο, τα όρια του μοντέλου και παρατηρείται η ανταπόκριση του μοντέλου πριν αυτό δοκιμαστεί σε αληθινά σύνολα δεδομένων. Ωστόσο, το πρόβλημα κατασκευής πινάκων κίνησης είναι κλιμακούμενης δυσκολίας και η υπολογιστική

πολυπλοκότητα του εξαρτάται από τον αριθμό των παραμέτρων του μοντέλου. Για παράδειγμα, η τοπολογία του δικτύου είναι σημαντική καθώς οι τιμές των κόμβων δεν πρέπει να ξεπερνούν την χωρητικότητα των ζεύξεων που αντιστοιχούν. Λαμβάνοντας υπόψη τις παραμέτρους του εκάστοτε προβλήματος μπορεί να δημιουργηθεί τεχνητός πίνακας που να ταιριάζει αρκετά με τα ήδη υπάρχοντα σύνολα δεδομένων [6].

Τα συμπεράσματα που μπορούμε να βγάλουμε παρατηρώντας την κίνηση του δικτύου εξαρτώνται από το μέγεθος των δεδομένων που διαθέτουμε. Δεδομένα που καλύπτουν σύντομο χρονικό διάστημα δευτερόλεπτα, λεπτά ή λίγες ώρες παρουσιάζουν μεγάλες αλλαγές και είναι δύσκολο να παρατηρηθεί κάποια εξάρτηση ή κάποιο μοτίβο κίνησης πακέτων. Αντίθετα σε μεγάλα σύνολα δεδομένων που παρουσιάζουν κίνηση δικτύων σε χρονικό διάστημα ημερών, μηνών και χρόνων παρατηρούνται ισχυρές εξαρτήσεις μεταξύ κόμβων, περιοδικότητα στις στιγμές μεγάλης κυκλοφορίας μέσα στην ημέρα ή και περιοδικότητα σε περιόδους του χρόνου όπου παρατηρείται αυξημένη κίνηση [6].

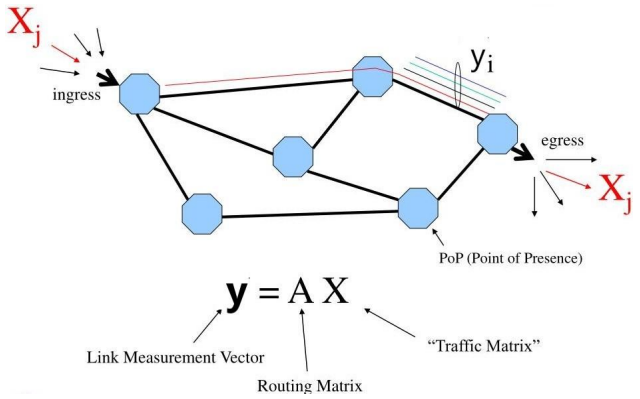
Οι πίνακες δεδομένων διευκολύνουν τους μηχανικούς και τους ερευνητές στη γενικότερη διαχείριση και προγραμματισμό του δικτύου. Μερικές χρήσιμες εφαρμογές του πίνακα κίνησης είναι η ανίχνευση ανωμαλιών κίνησης σε ένα δίκτυο και η διαχείριση της χωρητικότητας. Οι ανωμαλίες της κίνησης μπορεί να είναι κάποια διακοπή σύνδεσης κόμβων ή κάποια κατανεμημένη επίθεση άρνησης υπηρεσιών (DDoS attack). Ανεξαρτήτως της πηγής της ανωμαλίας το χρονικό διάστημα μεταξύ κάθε μέτρησης θα πρέπει να είναι μικρό, ώστε να ανιχνευθεί γρήγορα η απρόσμενη τιμή και να δρομολογηθούν οι κατάλληλες ενέργειες. Αντίθετα, στην δεύτερη εφαρμογή μας ενδιαφέρουν οι μεγάλες μετρήσεις κίνησης για ένα αρκετό χρονικό διάστημα, ώστε να εξασφαλιστεί ότι υπάρχει αρκετή χωρητικότητα για τους τρέχουσες αλλα και τις μελλοντικές ανάγκες του δικτύου [6, 10]. Ακόμα, ο πίνακας είναι απαραίτητος σε εργασίες όπως είναι η διαμόρφωση πρωτοκόλλου δρομολόγησης, η εξισορρόπηση φορτίου μεταξύ των ζεύξεων (load balancing) και η παροχή του δικτύου (network provisioning) [4, 10]. Το πρωτόκολλο δρομολόγησης εξασφαλίζει την εύρεση του ελάχιστου μονοπατιού για ροές πακέτων λαμβάνοντας υπόψη και την συμφόρηση των ζεύξεων. Συχνά, η κίνηση μοιράζεται σε διάφορα μονοπάτια προκειμένου να αποφευχθεί συμφόρηση και καθυστέρηση στην μεταφορά των πακέτων. Τέλος, με την επεξεργασία του πίνακα μπορούμε να πάρουμε αποφάσεις σχετικά με τις ενέργειες που πρέπει να ακολουθήσουμε για την διαχείριση των αποτυχιών και των σφαλμάτων που προκύπτουν [11]. Οι αποφάσεις αυτές διαφοροποιούνται αναλόγως το σφάλμα και το είδος του πακέτου. Για παράδειγμα κατα την μετάδοση ενός βίντεο είναι προτιμότερη η ανοχή στην απώλεια ενός πακέτου από την εξασφάλιση της σωστής σειράς αποδοχής των πακέτων. Σε άλλο παράδειγμα σε πιθανή αποτυχία μιας ζεύξης πρέπει να υπάρχει εναλλακτικό μονοπάτι και να υπάρχει ικανή χωρητικότητα ώστε να μην παρατηρηθεί συμφόρηση [10].

2.3 Εκτίμηση Πίνακα Κίνησης

Εκτίμηση Πίνακα Κυκλοφορίας (Traffic Matrix Estimation) είναι η διαδικασία κατά την οποία εκτιμάται η κίνηση μεταξύ κάθε ζεύγους κόμβων σε ένα δίκτυο. Η εκτίμηση αυτή προέρχεται από μετρήσεις δεδομένων σε όλες τις ζεύξεις. Στην εργασία [12] περιγράφεται αναλυτικά γιατί οι σύγχρονες εφαρμογές αδυνατούν να μετρήσουν την κίνηση των κόμβων άμεσα. Έτσι, η επιστημονική κοινότητα έχει στραφεί στην εκτίμηση του πίνακα κίνησης μέσω της χρήσης των μετρήσεων των ζεύξεων του δικτύου που είναι εύκολα διαθέσιμες. Το πρόβλημα της εκτίμησης βασίζεται σε ένα σύστημα γραμμικών εξισώσεων που περιγράφουν την σχέση μεταξύ πίνακα κίνησης, πίνακα δρομολόγησης και των μετρήσεων των ζεύξεων. Συγκεκριμένα, θεωρούμε ένα δίκτυο με n κόμβους και m ζεύξεις. Αν και προηγουμένως αναφέρθηκε ότι το $x_{i,j}$ αντιπροσωπεύει την ένταση κίνησης μεταξύ της πηγής i και του προορισμού j, εδώ είναι πιο βολικό να ταξινομήσουμε τον πίνακα σε ζεύγη κόμβων. Άρα, ο πίνακας κίνησης X είναι μεγέθους $n^2 \times 1$ και το στοιχείο x_j συμβολίζει την ένταση κίνησης του ζεύγους j [3]. Το διάνυσμα $\mathbf{y}$ είναι το $m \times 1$ διάνυσμα των μετρήσεων των ζεύξεων. Τέλος, ο πίνακας δρομολόγησης A έχει διαστάσεις $m \times n^2$ και περιγράφει την τοπολογία του δικτύου. Περιέχει όλα τα μονοπάτια μεταξύ των κόμβων προέλευσης και προορισμου [13]. Το στοιχείο $a_{i,j}$ έχει τιμή 1 όταν το ζεύγος προέλευσης - προορισμού της στήλης j διασχίζει την ζεύξη i, διαφορετικά παίρνει τιμή 0. Το σύστημα γραμμικών εξισώσεων μπορεί να διατυπωθεί ως [3, 5, 14]:

$$\mathbf{y} = AX$$

Σε αυτό το σύστημα εξισώσεων θεωρούμε δεδομένες τις μετρήσεις στις ζεύξεις ($\mathbf{y}$) γνωρίζουμε και τον πίνακα δρομολόγησης A και θέλουμε να υπολογίσουμε τον πίνακα κυκλοφορίας X. Η αντίστοιχη αναπαράσταση αυτού του προβλήματος παρουσιάζεται στην Εικόνα 2.3. Για να θεωρηθεί σωστο αυτό το σύστημα είναι απαραίτητες δύο υποθέσεις. Αρχικά, ότι ο πίνακας κίνησης είναι σταθερός κατά την διάρκεια των μετρήσεων και ότι οι μετρήσεις θεωρούνται αλάνθαστες. Στην πραγματικότητα πάντα εμφανίζονται λάθη στις μετρήσεις και για να τα λάβουμε υπόψη το σύστημα θα πρέπει να μετατραπεί σε $\mathbf{y} = Ax + z$ [6]. Στην περίπτωση που εκτελούμε K περιοδικές μετρήσεις υπολογίζουμε τα διανύσματα $y^k = (y_1^k, \dots, y_m^k)$ και $x^k = (x_1^k, \dots, x_m^k)$ όπου k μια χρονική περίοδος $k = 1, \dots, K$. Έτσι το σύστημα των εξισώσεων μετατρέπεται ως εξής: $y^k = AX^k, k = 1, \dots, K$ [3, 15].



Εικόνα 2.3: Τοπολογία του προβλήματος εκτίμησης πίνακα κίνησης [16]

Η δυσκολία σε αυτό το πρόβλημα είναι ότι το σύστημα γραμμικών εξισώσεων είναι υπο-προσδιορισμένο, διότι ο πίνακας A δεν είναι πλήρους τάξης με αποτέλεσμα να προκύπτουν πολλές λύσεις. Συγκεκριμένα, ο αριθμός των ροών κίνησης των ζευγαριών προέλευσης προορισμού (n) είναι σχεδόν πάντα μεγαλύτερος από τον αριθμό των ζεύξεων (m), $n \gg m$. Οι μέθοδοι των λύσεων που έχουν δοθεί μέχρι στιγμής ανήκουν σε δύο κατηγορίες, στις τεχνικές βελτιστοποίησης [5, 13, 14, 17] και στα στατιστικά μοντέλα [15, 18, 19, 20].

2.3.1 Τεχνικές Βελτιστοποίησης

Στην υποενότητα αυτή παρουσιάζονται συνοπτικά κάποιες αντιπροσωπευτικές εργασίες που χρησιμοποιούν τεχνικές βελτιστοποίησης. Η τεχνική βελτιστοποίησης στην εργασία [3] αντιμετωπίζει το πρόβλημα ως γραμμικό και προσπαθεί να υπολογίσει τον πίνακα X μειώνοντας τον χώρο εφικτών λύσεων με την χρήση περιορισμών. Η τιμή Y_1 είναι το άθροισμα της κίνησης που περνάει από την ζεύξη l και το μοντέλο γραμμικού προγραμματισμού περιγράφεται από την βελτιστοποίηση της συνάρτησης

$$max \sum_{j=1}^{c} w_j X_j$$

όπου w_j είναι το βάρος του ζεύγους j και $X_j \geq 0$. Η συνάρτηση έχει τον εξή περιορισμό ως προς τις ζεύξεις:

$$\sum_{j=1}^{c} \Lambda_{l,j} X_j \leq Y_l \quad l = 1, \ldots, r$$

Ως προς την ροή της κίνησης έχει περιορισμό:

$$\sum_{l=(i,j)} Y_l a_{lk} - \sum_{l=(j,i)} Y_l a_{lk} = \begin{cases} X_k & if\ \ j = src\ \ of\ \ k \\ -X_k & if\ \ i = dst\ \ of\ \ k \\ 0 & otherwise \end{cases}$$

Στο συγκεκριμένο μοντέλο οι τιμές των μετρήσεων των ζεύξεων χρησιμοποιούνται ως περιορισμοί και όχι ως στατιστικά δεδομένα. Η επιλογή του βάρους στην συνάρτηση επηρεάζει άμεσα την λύση. Παρατηρήθηκε ότι η επιλογή μοναδιαίας τιμής βάρους ($w_j = 1$) οδηγεί σε πίνακα με μεγάλες τιμές σε κοντικά ζευγάρια κόμβων και σε σχεδόν μηδενικές ροές μεταξύ απομακρυσμένων κόμβων [3]. Έπειτα από έρευνα και δοκιμές [3] βγήκε το συμπέρασμα ότι, ειδικά για μικρά δίκτυα, μια τιμή βάρους w_j ίση με το μέγεθος του μονοπατιού αποφέρει επιθυμητή λύση στο πρόβλημα.

Στην εργασία [5] δημιουργείται ο Παραλλαγμένος Αυτοκωδικοποιητής (Variational Autoencoder - VAE) που είναι ένα γεννητικό μοντέλο που δημιουργεί νέα δεδομένα παρόμοια με τα δεδομένα εκπαίδευσης. Ο VAE διερευνά παραλλαγές των δεδομένων εκπαίδευσης προς μία συγκεκριμένη επιθυμητή κατεύθυνση. Στην περίπτωση της εκτίμησης πίνακα κυκλοφορίας που εξετάζουμε εδώ, χρησιμοποιείται ο αυτοκωδικοποιητής για να συνθέσει τεχνητά παραδείγματα από τον λανθάνοντα χώρο z. Έτσι, το πρόβλημα μετατρέπεται σε ένα πρόβλημα ελαχιστοποίησης στον λανθάνοντα χώρο με d τον αυτοκωδικοποιητή, y οι μετρήσεις των ζεύξεων και $Ad(z)$ οι εκτιμώμενες μετρήσεις ζεύξεων:

$$argmin_z\left[||y - Ad(z)||_2^2\right]$$

Επίσης εξετάζεται άλλη μια μέθοδος στην οποία προστίθεται ένας όρος κανονικοποίησης:

$$argmin_z\left[||y - Ad(z)||_2^2 + c||z||_2^2\right]$$

Στόχος και στις δύο περιπτώσεις είναι να βρεθεί η καλύτερη παράμετρος z με την χρήση του βελτιστοποιητή Adam. Περιληπτικά η διαδικασία που ακολουθείται στην τεχνική αυτή είναι η εκπαίδευση του αυτοκωδικοποιητή με προηγούμενους πίνακες κυκλοφορίας και η επίλυση του προβλήματος ελαχιστοποίησης. Διαφορετικά σε περίπτωση που η διαδικασία καταλήγει σε τοπικά ελάχιστα και όχι στο ολικό επιλέγεται ταυτόχρονη βελτιστοποίηση των συναρτήσεων ελαχιστοποίησης με την εκπαίδευση του αυτοκωδικοποιητή [10]. Έτσι η διαδικασία θα καταλήξει στο βέλτιστο z^* που θα παράξει τον εκτιμώμενο πίνακα $x' = d(z^*)$.

Η εργασία [14] αναλύει δύο αλγορίθμους εκτίμησης πίνακα κίνησης χρησιμοποιώντας πληροφορίες σχετικά με την κατανομή των τιμών των εντάσεων στα ζεύγη κόμβων. Λαμβάνοντας υπόψη την κατανομή αλλά και το μετρημένο φορτίο των ζεύξεων εκτιμάται ο πίνακας κίνησης. Σε περίπτωση που υπάρχουν ελάχιστοι διαθέσιμοι πίνακες κίνησης του παρελθόντος, χρησιμοποιείται επαναληπτική μέθοδος για τον υπολογισμό της λύσης η οποία πρέπει να ικανοποιεί την εμπειρική κατανομή που προέρχεται από τους προηγούμενους πίνακες. Ο αλγόριθμος που ανέπτυξαν οι συγγραφείς της εργασίας [14] βγάζει ικανοποιητικά αποτελέσματα αρκεί να υπάρχουν αρκετά δεδομένα ή εμπειρία του ερευνητή ώστε να καθοριστεί η επιθυμητή κατανομή. Διαφορετικά, αν υπάρχουν αρκετοί προηγούμενοι πίνακες διαθέσιμοι, χρησιμοποιείται ένα μοντέλο βασισμένο στα Παραγωγικά Δίκτυα Αντιπάλων (Generative Adversarial Networks - GAN) το οποίο εκπαιδεύεται πάνω στα χαρακτηριστικά των

πινάκων και βγάζει αποτέλεσμα σχετικό με αυτούς τους πίνακες. Το μοντέλο αυτό αξιοποιεί εκτός από την κατανομή των τιμών και τις χωρικές συσχετίσεις των κόμβων. Θεωρώντας T το μοντέλο που δημιουργεί τον πίνακα κυκλοφορίας και l την μεταβλητή του, το πρόβλημα μπορεί να γραφεί ως:

$$min_l ||y - AT(l)||_2^2$$

Το μοντέλο αυτό μπορεί να χρησιμοποιηθεί όταν υπάρχει γνώση της κατανομής και να εκπαιδευτεί με παλαιότερα δεδομένα ή με δεδομένα που θα παραχθούν με βάση την κατανομή. Συμπερασματικά, η πρώτη μέθοδος βγάζει αποτελέσματα που ταιριάζουν καλύτερα στους περιορισμούς των τιμών των ζεύξεων, ενώ η δεύτερη μέθοδος συμφωνεί με την κατανομή των τιμών των κόμβων.

2.3.2 Στατιστικά Μοντέλα

Η πρώτη προσπάθεια εκτίμησης πίνακα κίνησης με στατιστικά μοντέλα έγινε από τον Y.Vardi στην εργασία [15]. Θεώρησε ότι ο πίνακας X ακολουθεί κατανομή Poisson (X_j ~ Poisson(λ_j)) και στόχος του προβλήματος είναι να βρεθούν οι παράμετροι λ_j, $j = 1, ..., n^2$. Ο συγγραφέας πρότεινε την μέθοδο της Εκτίμησης Μέγιστης Πιθανοφάνειας (Maximum Likelihood Estimation - MLE) για τον υπολογισμό των παραμέτρων του μοντέλου Poisson. Επίσης, παρουσιάζει άλλον έναν τρόπο λύσης βασισμένο στην μέθοδο των ροπών, όπου μπορεί να υπολογιστεί από εξισώσεις η μέση τιμή και η διακύμανση της κατανομής. Έτσι, το πρόβλημα μετατρέπεται σε γραμμικό αντίστροφο πρόβλημα με περιορισμούς το οποίο μπορεί να λυθεί με έναν EM αλγόριθμο (Expectation-Maximization algorithm). Πράγματι αποδεικνύεται στην εργασία ότι με την χρήση του EM αλγόριθμου υπολογίζονται οι παράμετροι Poisson και τα αποτελέσματα της μεθόδου αυτής είναι σταθερά και αρκετά φυσιολογικά. Οι δύο τεχνικές εκτίμησης εξετάζονται πάνω σε δύο είδη δικτύων, σε ένα με καθορισμένη δρομολόγηση (ντετερμινιστικό μοντέλο) και στο άλλο με τυχαία δρομολόγηση (Μαρκοβιανό μοντέλο).

Στην εργασία [18] χρησιμοποιείται μοντέλο EM όπου όπως και προηγουμένως θεωρείται ότι κάθε ζεύγος προέλευσης προορισμού ακολουθεί κανονική κατανομή. Οι παράμετροι της κανονικής κατανομής υπολογίζονται από την μέθοδο Εκτίμησης Μέγιστης Πιθανοφάνειας (Maximum Likelihood Estimation - MLE). Οι ζητούμενοι παράμετροι είναι το διάνυσμα των μέσων τιμών $\Lambda = (\lambda_1, ..., \lambda_{n^2})$ και ο πίνακας $\Sigma = \varphi\, diag(\sigma^2(\lambda_1), ..., \sigma^2(\lambda_{n^2}))$, όπου $\varphi > 0$ είναι μια παράμετρος κλίμακας και $\sigma^2(\lambda)$ είναι μια γνωστή σχέση μεταξύ μέσης τιμής και διακύμανσης της οποίας το γενικό μοντέλο είναι $\sigma^2(\lambda) = \lambda^c$, με c αριθμητική σταθερά. Άρα η δεύτερη παράμετρος μπορεί να γραφτεί στην μορφή: $\sum_j = \varphi\lambda_j^c$. Οι συγγραφείς απέδειξαν ότι η

τιμή $c = 2$ ταιριάζει στα δεδομένα του δικτύου που εξετάστηκε και χρησιμοποίησαν αυτήν την τιμή για την εργασία. Η μέθοδος εκτίμησης μέγιστης πιθανοφάνειας έχει το μέγιστο αποτέλεσμα που υπολογίζεται από την εξίσωση:

$$l(\theta|y_1, \ldots, y_k) = -\frac{K}{2} log|A\Sigma A'| - \frac{1}{2}\sum_{k=1}^{K}(y_k - A\Lambda)'(A\Sigma A')^{-1}(y_k - A\Lambda) \text{ [3]}$$

Ισχύει ότι $\theta = (\Lambda, \varphi)$. Περιληπτικά η μέθοδος που ακολουθήθηκε περιλάμβανε υπολογισμό της τιμής του θ από αλγόριθμο ΕΜ. Έπειτα, γίνεται εκτίμηση της τιμής κάθε ζευγαριού από την σχέση $X_{j,t} = E[X_{j,t}|\theta, \Upsilon]$ και τέλος εκτελείται ένας επαναληπτικός αλγόριθμος (Iterative Proportional Fitting - IPF).

Μια άλλη στατιστική μέθοδος που έχει προταθεί για την ανάλυση και την επίλυση παρόμοιων προβλημάτων η οποία περιγράφεται αναλυτικά στην εργασία [19] είναι τα μοντέλα Bayesian. Εδώ ο στόχος είναι να υπολογιστεί η κατανομή πιθανοτήτων $p(X|Y)$ όλων των ζευγαριών του X από τις μετρήσεις των ζεύξεων Y. Στην εργασία αυτή όπως και στην [15] γίνεται η υπόθεση ότι ο πίνακας X ακολουθεί κατανομή Poisson ($X_j \sim \text{Poisson}(\lambda_j)$) με $\Lambda = (\lambda_1, \ldots, \lambda_{n^2})$ το σύνολο μέσων τιμών. Η εξίσωση που περιγράφει το πρόβλημα είναι η εξής:

$$p(X|Y) = \sum P(X|\Lambda, Y) \text{ [3]}$$

και για να λυθεί πρέπει να υπολογιστούν οι εκ των υστέρων πιθανότητες $P(\Lambda|X, Y)$ και η $p(X|\Lambda, Y)$, όπου:

$$p(\Lambda|X, Y) \equiv p(\Lambda|X) = \prod_{a=1}^{n^2} p(\lambda_\alpha|X_a)$$

Η επαναληπτική διαδικασία επίλυσης του προβλήματος ξεκινάει με τον αρχικό πίνακα X^0. Υπολογίζεται η τιμή Λ^i από την πιθανότητα $p(\Lambda|X^i, Y)$ η οποία χρησιμοποιείται για να υπολογιστεί η τιμή X^{i+1} από την σχέση $p(X|\Lambda^i, Y)$. Ακολουθείται αυτή η διαδικασία μέχρι ο πίνακας X να περιέχει θετικές τιμές και να ικανοποιούνται ολοι οι περιορισμοί. Παράλληλα παρατηρήθηκε πως γράφοντας τους πίνακες $A = [A_1 A_2]$, $X = [X_1 X_2]$ αποδεικνύεται ότι $X_1 = A_1^{-1}(Y - A_2 X_2)$. Επομένως, γνωρίζοντας τον πίνακα Y και εκτιμώντας τα $n^2 - r$ ζευγάρια κόμβων του X_2 μπορούν να υπολογιστούν από την παραπάνω σχέση τα υπολειπόμενα r ζευγάρια του X_1. Οι συγγραφείς συμπεραίνουν λοιπόν, ότι μπορεί να επιλυθεί το πρόβλημα πιο εύκολα και γρήγορα χρησιμοποιώντας το υποσύνολο X_2 αντί για όλο τον πίνακα X, αρκεί η επαναληπτική διαδικασία να εφαρμοστεί στο X_2 και να υπολογιστεί το $p(X_2|\Lambda, Y)$ [3]. Μια παραλλαγή της παραπάνω μεθόδου εξετάστηκε και στην εργασία [20], όπου έγινε προσθήκη θορύβου στην εξίσωση $Y = AX + \varepsilon$, ενώ ο πίνακας όπως και ο θόρυβος ακολουθούσαν

κανονική κατανομή. Με την ίδια συλλογιστική πορεία καταλήγει σε μια σχέση που ο πίνακας X εξαρτάται μόνο από τις μέσες τιμές του, τις διακυμάνσεις του και τις τιμές του Y.

Έχουν υπάρξει προσπάθειες να συνδυαστούν οι δύο τεχνικές εκτίμησης πινάκων κυκλοφορίας. Δηλαδή, αναπτύχθηκαν μέθοδοι που συνδυάζουν στατιστικά μοντέλα με τεχνικές βελτιστοποίησης [13] [17]. Στην εργασία [13] σχεδιάζεται ένα μοντέλο Bayesian, ώστε να αποτυπωθούν οι σχέσεις των κόμβων του δικτύου και με βάση αυτό υπολογίζεται η από κοινού συνάρτηση πιθανότητας της κίνησης του δικτύου. Έπειτα, εκτιμάται η κίνηση με τη βοήθεια ενός μοντέλου βελτιστοποίησης. Αυτό γίνεται διότι δεν μπορούν να χρησιμοποιηθούν αποκλειστικά στατιστικά μοντέλα για την επίλυση του προβλήματος εκτίμησης σε ένα υπολογιστικό νέφος. Έτσι, λαμβάνονται οι προηγούμενοι πίνακες X σαν δεδομένα εκπαίδευσης ώστε να υπολογιστεί η κατανομή πιθανοτήτων και να δημιουργηθεί ο πίνακας X_0. Ο τελικός πίνακας X μπορεί να εκτιμηθεί πλέον από το εξής μοντέλο βελτιστοποίησης:

$$min||AX - Y||_2^2 + \frac{1}{\sigma^2}||X - X_0||_2^2$$

όπου σ^2 είναι ο μέσος όρος των διακυμάνσεων τον ροών των ζευγαριών. Μια διαφορετική προσέγγιση έγινε στην εργασία [17] όπου αναπτύχθηκε ένας αλγόριθμος βασισμένος στα χαρακτηριστικά των ζεύξεων και συνδυάζει στατιστικά στοιχεία με στοιχεία βελτιστοποίησης. Αρχικά, ορίζεται ένας συντελεστής l_{kz} που υποδηλώνει το ποσοστό του φορτίου της ζεύξης y_k που ανήκει στο ζεύγος x_z. Οι σχέσεις που προκύπτουν είναι οι εξής: $x_z = l_{kz}y_k$, $y_k = \sum a_{kz}x_z$ και $\sum l_{kz} = 1$. Επίσης, θεωρώντας T τα πιθανά μονοπάτια κάθε ζεύγους, κάθε τιμή x_{zt} έχει διαφορετικό βάρος ως προς την εκτίμηση του x_z. Άρα προκύπτουν άλλες δύο σχέσεις: $x_z = \sum w_{zt}x_{zt}$ και $\sum w_{zt} = 1$. Οι τιμές $l_{kz}w_{zt}$ μπορούν να προσδιοριστούν πριν την εκτίμηση του πίνακα. Ο στόχος του αλγορίθμου είναι η βελτιστοποίηση της παρακάτω εξίσωσης:

$$min||AX - Y||_2^2$$

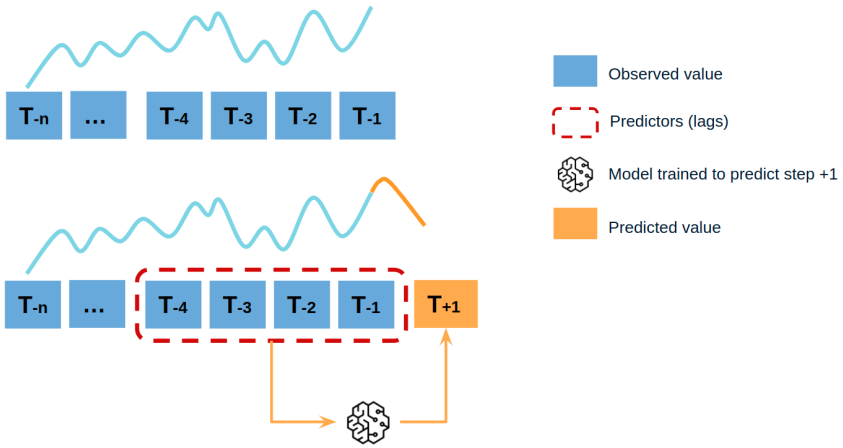
Ο αλγόριθμος υπολογίζει την τιμή κάθε ζεύγους από την σχέση $x_z = l_{kz}y_k$ και εκτιμά την τιμή του ζεύγους από την σχέση $x_z = \sum w_{zt}x_{zt}$. Έπειτα, ελέγχει το ζευγάρι κόμβων με την σχέση $AX = Y'$ και τέλος προσδιορίζει το σφάλμα $\varepsilon = Y - Y'$. Αν το σφάλμα είναι μικρότερο από ένα όριο που έχει τεθεί στην αρχή του αλγόριθμου τότε αυτός τερματίζεται, αλλιώς συνεχίζει. Συμπερασματικά, εδώ το πρόβλημα εκτίμησης του πίνακα κυκλοφορίας θεωρείται ισοδύναμο με το πρόβλημα εύρεσης ενός πίνακα X ο οποίος όταν καταχωρηθεί στο δίκτυο, θα επιφέρει έναν πίνακα Y' με μικρή απόκλιση από τον αρχικό Y. Τέλος, αξίζει να αναφερθεί ότι σε μια άλλη

έρευνα [11], αναπτύχθηκαν μέθοδοι για την επίλυση του προβλήματος της εκτίμησης και απέδειξαν ότι μετρώντας τις τιμές κίνησης ενός μικρού ποσοστού ζευγαριών κόμβων, υπάρχει σημαντική επίδραση στην ακρίβεια της εκτίμησης, παράγοντας αποτελέσματα πολύ καλύτερα από προηγούμενες μεθόδους. Επίσης, ανέφεραν την πιθανότητα αυτή η τεχνική να βελτιώνει τα αποτελέσματα και άλλων διαδεδομένων μεθόδων.

2.4 Πρόβλεψη Πίνακα Κίνησης

Πρόβλεψη Πίνακα Κυκλοφορίας (Traffic Matrix Prediction) είναι η διαδικασία κατά την οποία εκτιμάται η μελλοντική κίνηση μεταξύ κάθε ζεύγους κόμβων σε ένα δίκτυο. Η κίνηση αυτή εκτιμάται έπειτα από επεξεργασία παρελθοντικών δεδομένων του δικτύου [21, 22]. Πολλές λειτουργίες διαχείρισης δικτύου, όπως ο σχεδιασμός του δικτύου και η διαμόρφωση των πολιτικων δρομολόγησης, βασίζονται στην πρόβλεψη μελλοντικής κίνησης [23]. Ωστόσο, παρά την σημαντικότητα της πρόβλεψης της κίνησης, οι δικτυακές συσκευές δεν έχουν την δυνατότητα της παρακολούθησης σε πραγματικό χρόνο με αποτέλεσμα να μην υπάρχει έγκαιρη αντίδραση σε αλλαγές της δικτυακής κίνησης. Η προβλεψιμότητα της κυκλοφορίας εξαρτάται από τα στατιστικά χαρακτηριστικά των παραμέτρων της και την χρονολογική σχέση των εντάσεων της κίνησης [21]. Όπως αναφέραμε στο κεφάλαιο 2.1 ο πίνακας κίνησης έχει ως τρίτη διάσταση τον χρόνο ο οποίος παραλείπεται κάποιες φορές ανάλογα με το πρόβλημα. Στα προβλήματα πρόβλεψης ο χρόνος είναι απαραίτητο στοιχείο. Ένας έγκυρος και έγκαιρος πίνακας κίνησης είναι απαραίτητος για διαδικασίες όπως ο άμεσος προγραμματισμός και επαναδρομολόγηση της κυκλοφορίας, η μελλοντική διαχείριση της χωρητικότητας και η ανίχνευση ανωμαλιών κίνησης [21]. Θεωρούμε έναν πίνακα κυκλοφορίας X διαστάσεων $n \times t$ με n των αριθμών των ζευγών προέλευσης προορισμού, όπου το στοιχείο $x_{j,t}$ συμβολίζει την ένταση κίνησης του ζεύγους j την χρονική στιγμή t. Ο πίνακας αυτός ονομάζεται vectorized traffic matrix και η χρονική στιγμή t στην πραγματικότητα είναι ένα χρονικό διάστημα, συνήθως λίγων λεπτών, κατά το οποίο έχει γίνει η μέτρηση της έντασης της κίνησης στο δίκτυο. Το πρόβλημα πρόβλεψης μπορεί να οριστεί ως ένα πρόβλημα εκτίμησης του πίνακα $X_{n,t}$ μέσα από μια σειρά προηγούμενων δεδομένων κίνησης ($X_{n,t-1}$, $X_{n,t-2}$, $X_{n,t-3}$, $X_{n,t-4}$, ...). Η κύρια δυσκολία του προβλήματος αυτού είναι το πως θα μοντελοποιηθούν οι σχέσεις μεταξύ του συνόλου δεδομένων κίνησης, ώστε να μπορεί κανείς να προβλέψει το $X_{n,t}$ [23]. Οι αρχικές μέθοδοι που εξετάστηκαν χρησιμοποίησαν στατιστικά και γραμμικά μοντέλα προκειμένου να μοντελοποιήσουν την κίνηση αλλά αδυνατούσαν να συλλάβουν τα μη γραμμικά χαρακτηριστικά της. Έτσι αναπτύχθηκαν πιο σύγχρονα μοντέλα χρονοσειρών βασισμένα στα νευρωνικά δίκτυα, όπου συγκρίνοντας την απόκλιση μεταξύ της προβλεπόμενης και της πραγματικής κίνησης αποδείχθηκε ότι αυτά έχουν καλύτερη ακρίβεια πρόβλεψης. Τα μοντέλα αυτά εξετάστηκαν και αξιολογήθηκαν με βάση την ακρίβεια της πρόβλεψης τους, αλλά αξίζει να σημειωθεί και η

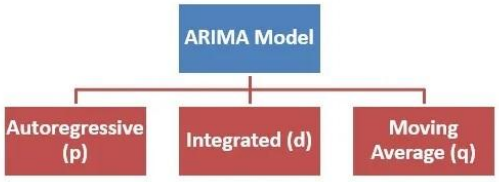
σημαντικότητα της ακρίβειας στα όρια των προβλέψεων που παίζουν καθοριστικό ρόλο σε διάφορες δικτυακές εφαρμογές [24].



Εικόνα 2.4.0: Διαδικασία πρόβλεψης χρονοσειράς [25]

2.4.1 Γραμμικά Μοντέλα

Η δημοφιλέστερη γραμμική τεχνική για την επίλυση του προβλήματος πρόβλεψης δικτυακού πίνακα κυκλοφορίας είναι το Αυτοπαλινδρομικό Μοντέλο Κινητού Μέσου Όρου (Auto-Regressive Integrated Moving Average - ARIMA) που είναι μοντέλο χρονοσειρών. Τα μοντέλα αυτά είναι στοχαστικά μαθηματικά μοντέλα με τα οποία προσπαθούμε να περιγράψουμε τη διαχρονική εξέλιξη των χρονοσειρών. Στις εργασίες [26] [27] περιγράφεται αναλυτικά το μοντέλο και τα αποτελέσματα του σε διάφορα δίκτυα. Θεωρούμε την χρονοσειρά X_t με τελεστή ολίσθησης B τέτοιο ώστε $BX_t = X_{t-1}$. Το μοντέλο ARIMA(p, d, q) είναι μια στοχαστική διαδικασία χρονοσειράς με p την παλινδρομική σειρά, d η τάξη της διαφόρισης και q τη σειρά κινούμενων μέσων όρων, όπως παρουσιάζεται και στην παρακάτω εικόνα.



Εικόνα 2.4.1: Στοιχεία μοντέλου ARIMA [28]

Το μοντέλο μπορεί να περιγραφεί από την παρακάτω σχέση:

$$\varphi_p(B)(1 - B)^d X_t = \theta_q(B)e_t$$

Όπου e_t είναι το σφάλμα, $|B| \leq 1$ και για τα πολυώνυμα Φ_p, Θ_q ισχύει:

$$\varphi_p(B) = 1 - \varphi_1 B - \varphi_2 B^2 - \ldots - \varphi_p B^p$$

$$\theta_p(B) = 1 - \theta_1 B - \theta_2 B^2 - \ldots - \theta_q B^p$$

Η διαδικασία πρόβλεψης ξεκινά με επεξεργασία των δεδομένων ώστε η χρονοσειρά X να έχει μηδενικό μέσο όρο και συνεχίζει με διαφόριση της χρονοσειράς. Έπειτα υπολογίζεται $p,\ q$ και εκτιμώνται οι παράμετροι με την μέθοδο Εκτίμησης Μέγιστης Πιθανοφάνειας. Τέλος, γίνεται πρόβλεψη των μελλοντικών τιμών βασιζόμενοι στις παραμέτρους και στο μοντέλο.

Το μοντέλο ARIMA έχει και διάφορες παραλλαγές που έχουν παρουσιαστεί λεπτομερώς [29, 30, 31], αλλά εδώ θα αναφέρουμε παρακάτω τις κυριότερες. ARMA είναι το μοντέλο που χρησιμοποιείται αν η χρονοσειρά είναι στάσιμη ως προς την μέση τιμή και την διακύμανση. Διαφορετικά μπορούμε να πούμε ότι είναι όπως το μοντέλο ARIMA με μηδενική τάξη διαφόρισης, δηλαδή δεν εκτελείται διαφόριση στην χρονοσειρά. Η σχέση μετατρέπεται στην ακόλουθη:

$$\varphi_p(B) X_t = \theta_q(B) e_t$$

Στην εργασία [32] περιγράφεται αναλυτικά το μοντέλο το οποίο χρησιμοποιείται για την εύρεση των στατιστικών ιδιοτήτων που επηρεάζουν την προβλεψιμότητα. Άλλη μια παραλλαγή είναι η εποχιακή έκδοση που ονομάζεται SARIMA (Seasonal ARIMA) που συμβολίζεται ως ARIMA $(p, d, q)(P_1,\ D_1,\ Q_1)$. Η σχέση που περιγράφει το μοντέλο είναι η εξής:

$$\varphi_p(B)\Phi_{P1}(B^{K1})(1 - B)^d(1 - B)^{D1} X_t = \theta_q(B)\Theta_{Q1}(B^{K1}) e_t$$

όπου K_1 είναι η εποχικότητα δηλαδή τα δεδομένα απέχουν K_1 περιόδους. Στο μοντέλο SARIMA εκτελείται εποχιακή διαφόριση ώστε να περιορίζονται οι διακυμάνσεις του επιπέδου. Αν η εποχιακή διαφόριση δεν αποδώσει επαρκή σταθερότητα τότε συνδυάζεται με απλή διαφόριση [30]. Τέλος, μια άλλη έκδοση του μοντέλου για μεγάλη εμβέλεια είναι το FARIMA (Fractional ARIMA). Συμβολίζεται ως FARIMA(p,d,q) με την μόνη διαφορά ότι η παράμετρος $d \in (-0.5, 0.5)$ σε αντίθεση με το ARIMA όπου το d είναι ακέραιος. Το d υπολογίζεται μέσω της παραμέτρου Hurst ($d = H - ½$) η οποία μπορεί να αποκτηθεί από κάποιες μεθόδους εκτίμησης. Γενικά ακολουθούνται τα ίδια βήματα επίλυσης με την διαδικασία του μοντέλου ARIMA με μόνη διαφορά την εκτίμηση της παραμέτρου H [26].

Μια άλλη διαδεδομένη γραμμική τεχνική είναι ο αλγόριθμος Holt-Winters [27] που ανήκει στην οικογένεια των μεθόδων εκθετικής εξομάλυνσης. Μέθοδοι εξομάλυνσης είναι τεχνικές με τις οποίες προσδιορίζονται οι μελλοντικές τιμές μιας μεταβλητής με βάση τον τρόπο εφαρμογής τους. Οι τεχνικές αυτές ονομάζονται μέθοδοι εξομάλυνσης, διότι η δημιουργία των προβλέψεων προέρχεται από την εξομάλυνση της διαχρονικής εξέλιξης των τιμών της μεταβλητής, ώστε να αναγνωριστεί καλύτερα ο τρόπος συμπεριφοράς της. Το μοντέλο αυτό είναι γνωστό για την ευχρηστία του, την ακρίβεια του, τον μικρό υπολογιστικό βαθμό δυσκολίας και την δυνατότητα

να διαμορφώσουμε εύκολα και σχετικά γρήγορα προβλέψεις για μια μεταβλητή, ειδικά σε εποχιακή χρονοσειρά. Από επαναληπτικές εξισώσεις υπολογίζεται το επίπεδο, η τάση και η εποχικότητα ενώ οι αρχικές τιμές καθορίζονται από τον μέσο όρο προηγούμενων παρατηρήσεων. Τέλος, οι παράμετροι του μοντέλου Holt-Winters υπολογίζονται από αναζήτηση καλύτερου σφάλματος εκπαίδευσης. Σύμφωνα με μετρήσεις που έγιναν πάνω σε διάφορα δίκτυα [20] το μοντέλο ARIMA έχει καλύτερες προβλέψεις από το Holt-Winters.

Στην εργασία [32] εξετάστηκαν δύο θέματα. Αρχικά, θεωρώντας περιορισμένο το σφάλμα πρόβλεψης πόσο μακριά στο μέλλον μπορεί να φτάσει η πρόβλεψη κίνησης και ποιο είναι το ελάχιστο σφάλμα πρόβλεψης σε προκαθορισμένο χρονικό διάστημα. Προκειμένου να απαντηθούν αυτά τα δύο ερωτήματα εξετάστηκε άλλο ένα γραμμικό μοντέλο που αντιπροσωπεύει την κίνηση, η Προσαρμοσμένη Μαρκοβιανή Διαδικασία Poisson (Markov-Modulated Poisson Process - MMPP). Σύμφωνα με προηγούμενες έρευνες [33] θεωρήθηκε θεμιτή η χρήση ενός είδους MMPP που λέγεται Circulant Markov-Modulated Poisson Process (CMPP). Η διαδικασία $Y(t)$ έχει διάνυσμα ρυθμού $r = [r_1, \ldots, r_N]$ και μήτρα μετάβασης $Q = F\Lambda F^*$ όπου F είναι πίνακας Fourier και F^* ο συζυγής ανάστροφος του. Ακόμα, ισχύει $\Lambda = diag[\lambda_1, \ldots, \lambda_n]$ και στον πίνακα Fourier $W^{ij} = \frac{1}{\sqrt{N}} e^{(-2\pi(i-1)(j-1)/N)\sqrt{-1}}$. Σύμφωνα με την μαρκοβιανή ιδιότητα η βέλτιστη πρόβλεψη για το στοιχείο $Y(t+\tau)$ είναι:

$$\begin{aligned}\widehat{Y_i}(t+\tau) &\equiv \mathrm{E}[Y(t+\tau)|Y(t) = r_i] \\ &= \sum_{k=1}^{N} r_k [e^{Q\tau}]_{ik} \\ &= \frac{1}{N} \sum_{k=1}^{N} r_k \sum_{l=1}^{N} e^{\lambda_l \tau} W^{l(i-k)}\end{aligned}$$

Η συγκεκριμένη εργασία δεν επικεντρώθηκε στην αξιολόγηση των προβλέψεων του μοντέλου, οπότε δεν μπορεί να συγκριθεί με τα προηγούμενα αλλά αξίζει να αναφερθεί σαν γραμμικό μοντέλο.

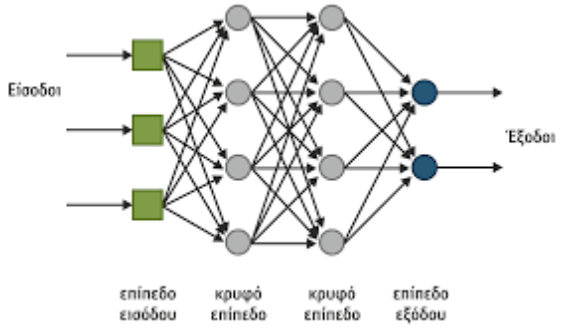
Οι συγγραφείς της [34] προτείνουν 3 νέα γραμμικά μοντέλα στα οποία χρησιμοποιούν το μεγαλύτερο μέρος των δεδομένων για πρόβλεψη και συγκεκριμένα κομμάτια δεδομένων για διόρθωση. Η πρόβλεψη γίνεται πάνω στην ένταση της κίνησης του κάθε κόμβου $y(i) = \sum x_{ij}$ και μετά η ένταση που εκτιμήθηκε κατανέμεται αναλόγως στα ζεύγη προέλευσης προορισμού. Το πρώτο και πιο απλό μοντέλο Independent Node Traffic (INP) θεωρεί ανεξάρτητη την κίνηση κάθε κόμβου και εκθετική την αύξηση της κίνησης. Αρχικά προβλέπει την μελλοντική κίνηση $y_{t+\Delta t}(i)$ με βάση την $y_t(i)$. Έπειτα, υπολογίζει τον συντελεστή κάθε ζεύγους συγκρίνοντας την μέση κίνηση του ζεύγους i, j με την μέση κίνηση του κόμβου i. Τέλος, κατανέμει την τιμή της συνολικής κίνησης που έχει προβλεφθεί στα ζεύγη ανάλογα με τον συντελεστή τους. Η μέθοδος

αυτή παρουσιάζει αρκετά μεγάλο σφάλμα διότι αγνοεί την συσχέτιση μεταξύ των ζευγαριών των κόμβων και είναι ευαίσθητη στην διακύμανση της έντασης της κυκλοφορίας. Το επόμενο μοντέλο που εξετάζεται στην ίδια εργασία ονομάζεται Total Matrix Prediction with Key Element Correction (TMP-KEC) το οποίο προβλέπει την μελλοντική κίνηση βασιζόμενο στην συνολική ένταση κίνησης όλου του δικτύου. Όπως και πριν η ένταση κατανέμεται στα ζεύγη με βάση τους συντελεστές τους. Το διαφορετικό χαρακτηριστικό αυτής της μεθόδου είναι ότι επιλέγονται οι κόμβοι με τη υψηλότερη ένταση εντός δικτύου και από αυτούς αποθηκεύονται τα ζεύγη με την μεγαλύτερη κίνηση σαν σημεία κλειδιά. Επαναλαμβάνεται η πρόβλεψη των συγκεκριμένων σημείων κλειδιών λαμβάνοντας υπόψη την χρονολογική εξέλιξη της κίνησης αυτών των ζευγών και έπειτα διορθώνεται η πρόβλεψη των άλλων στοιχείων των κόμβων κλειδιών. Έτσι μπορεί να παρουσιάζεται μικρή ευαισθησία στη διακύμανση της κίνησης μεταξύ των ζευγών, αλλά διορθώνεται η πρόβλεψη για τα ζεύγη και τους κόμβους με την μεγαλύτερη ένταση κι αυτό επιφέρει μείωση του συνολικού σφάλματος πρόβλεψης. Η τρίτη μέθοδος που παρουσιάζεται ονομάζεται Principle Component Prediction with Fluctuation Component Correction (PCP-FCC) και εστιάζει στις στιγμές συνολικής αυξημένης έντασης στο δίκτυο. Αρχικά, χωρίζεται η κίνηση των κόμβων στο κομμάτι της τάσης και το κομμάτι της διακύμανσης $y_t(i) = l_t(i) + f_t(i)$. Έπειτα, χωρίζουν το κομμάτι της τάσης σε συνιστώσες με συντελεστές για κάθε κόμβο $c_t(i) = u_{il}(1) + \dots + u_{iN} l_t(N)$ και υπολογίζουν τους συντελεστές u_{ij} από την μέθοδο Principle Component Analysis (PCA). Προβλέπουν την συνιστώσα $c_{t+\Delta t}(i)$ και με αντιστροφή υπολογίζουν την $l_{t+\Delta t}(i)$. Τέλος, το κομμάτι $f_t(i)$ υπολογίζεται με την χρήση του μοντέλου ARIMA. Έτσι έχει εκτιμηθεί η κίνηση των κόμβων $y_t(i)$ και η ένταση διανέμεται σε κάθε ζεύγος με την μέθοδο που περιγράψαμε στο μοντέλο INP. Το μοντέλο αυτό πετυχαίνει το μικρότερο σφάλμα πρόβλεψης κάθε στοιχείου, διότι λαμβάνει υπόψη την συσχέτιση των κόμβων άλλα και την εξέλιξη της κίνησης των κόμβων με την μεγαλύτερη ένταση.

2.4.2 Νευρωνικά Δίκτυα

Τα Νευρωνικά Δίκτυα (Neural Networks) αποτελούν μια από τις πιο διαδεδομένες τεχνικές Μηχανικής Μάθησης (Machine Learning). Πρόκειται για μια σειρά αλγορίθμων που επιχειρούν να αναγνωρίσουν αλληλουχίες και συσχετίσεις σε ένα σύνολο δεδομένων, μέσω διαδικασιών οι οποίες μιμούνται τον βιολογικό τρόπο λειτουργίας του ανθρώπινου εγκεφάλου. Είναι δίκτυα από απλούς υπολογιστικούς κόμβους (νευρώνες) διασυνδεδεμένους μεταξύ τους. Κάθε τέτοιος κόμβος δέχεται ένα σύνολο αριθμητικών εισόδων από διαφορετικές πηγές (είτε από άλλους νευρώνες, είτε από το περιβάλλον), επιτελεί έναν υπολογισμό με βάση αυτές τις εισόδους και παράγει μία έξοδο. Η εν λόγω έξοδος είτε κατευθύνεται στο περιβάλλον, είτε τροφοδοτείται ως είσοδος σε άλλους νευρώνες του δικτύου. Στο κεφάλαιο αυτό θα παρουσιαστούν τα μοντέλα νευρωνικών δικτύων που χρησιμοποιούνται για πρόβλεψη κίνησης. Η μηχανική μάθηση

γενικότερα, αλλά και ο τρόπος λειτουργίας των νευρωνικών δικτύων θα αναλυθούν λεπτομερώς στο κεφάλαιο 3.



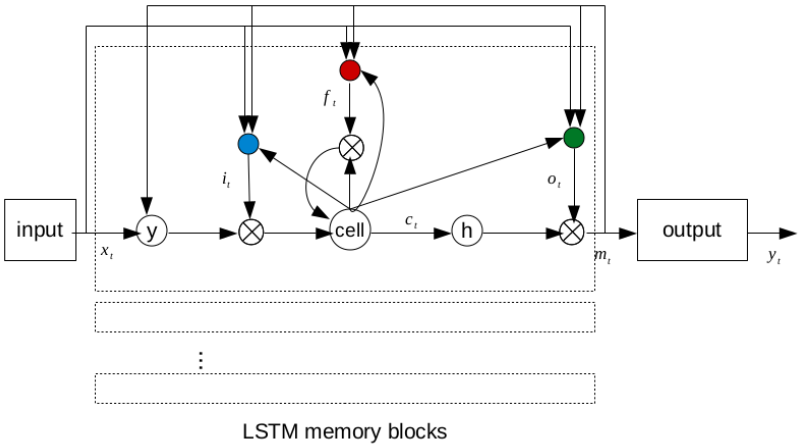
Εικόνα 2.4.2.1: Απλό Νευρωνικό Δίκτυο [35]

Η πιο διαδεδομένη μη γραμμική προσέγγιση στα προβλήματα πρόβλεψης είναι τα νευρωνικά δίκτυα (Εικόνα 2.4.2.1). Τα μοντέλα βασισμένα στα νευρωνικά δίκτυα έχουν καλύτερη επίδοση από τα γραμμικά μοντέλα όπως αποδείχθηκε [36] και η επιλογή της μεθόδου πρόβλεψης εξαρτάται από τα χαρακτηριστικά του προβλήματος, την πολυπλοκότητα της λύσης και την επιθυμητή ακρίβεια των προβλέψεων. Τα Νευρωνικά Δίκτυα χωρίζονται σε 3 κύριες κατηγορίες ανάλογα με την αρχιτεκτονική τους και τον τρόπο που συνδέονται οι νευρώνες μεταξύ τους σε:

- Νευρωνικά Δίκτυα με Πρόσθια Τροφοδότηση (Feedforward Neural Networks - FNN)
- Επαναλαμβανόμενα Νευρωνικά Δίκτυα (Recurrent Neural Networks - RNN)
- Συνελικτικά Νευρωνικά Δίκτυα (Convolutional Neural Networks - CNN)

Στο κεφάλαιο αυτό θα ασχοληθούμε κυρίως με τα δίκτυα RNN τα οποία περιλαμβάνουν βρόγχους ανατροφοδότησης. Με αυτόν τον τρόπο το σύστημα αποκτά κάποιου είδους μνήμη, ενώ παράλληλα εκτελεί πράξεις. Αυτή η ιδιότητα κάνει τα συγκεκριμένα μοντέλα κατάλληλα για προβλήματα μοντελοποίησης ακολουθιών όπως η πρόβλεψη χρονοσειρών. Παρ' όλα αυτά υπάρχουν δυσκολίες με τη λειτουργία τους, αφού χρειάζεται χρόνος να γίνει η εκπαίδευση και η μνήμη τους είναι περιορισμένη. Παρακάτω θα παρουσιαστούν ορισμένα μοντέλα νευρωνικών δικτύων κατάλληλα για πρόβλεψη χρονοσειρών.

Τα δίκτυα "Long Short-Term Memory" ή κοινώς LSTM είναι recurrent networks που έχουν σχεδιαστεί ώστε να ξεπεράσουν το πρόβλημα της γεφύρωσης των γεγονότων εισόδου χωρίς να έχουν απώλειες λόγω χρονικών κενών. Αποτελούνται από μπλοκ μνήμης που περιέχουν κελιά (cell) τα οποία επιλέγουν ποιες πληροφορίες θα αποθηκεύσουν με το να συνδυάζουν την προηγούμενη κατάσταση, την τρέχουσα μνήμη και την είσοδο.



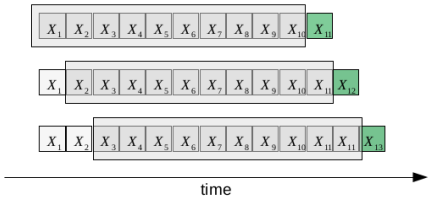
Εικόνα 2.4.2.2: Δομή LSTM block [37]

Το κάθε μπλοκ μνήμης (Εικόνα 7) περιέχει τμήματα προώθησης προς τα προηγούμενα αλλά και τα επόμενα μπλοκ. Παρακάτω θα αναπτυχθούν οι μαθηματικές σχέσεις που υπάρχουν στο μοντέλο LSTM όπως αυτό εμφανίζεται στην παραπάνω απεικόνιση 2.4.2.2 [37].

$$i_t = \sigma(W_{ix}x_t + W_{im}m_{t-1} + W_{ic}c_{t-1} + b_i)$$
$$f_t = \sigma(W_{fx}x_t + W_{fm}m_{t-1} + W_{fc}c_{t-1} + b_f)$$
$$c_t = f_t \odot c_{t-1} + i_t \odot g(W_{cx}x_t + W_{cm}m_{t-1} + b_c)$$
$$o_t = \sigma(W_{ox}x_t + W_{om}m_{t-1} + W_{oc}c_t + b_o)$$
$$m_t = o_t \odot h(c_t)$$
$$y_t = \varphi(W_{ym}m_t + b_y)$$

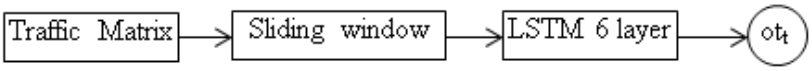
Να σημειώσουμε ότι με i συμβολίζουμε το input gate, με f το forget gate και με ο το output gate. Η συνάρτηση ενεργοποίησης του κελιού συμβολίζεται με c και m είναι η συνάρτηση ενεργοποίησης της εξόδου. Οι συναρτήσεις εισόδου και εξόδου του κελιού αποτυπώνονται με τα σύμβολα g, h και ισούνται με $tanh$. Φ είναι οι συνάρτηση ενεργοποίησης της εξόδου του δικτύου. W είναι τα βάρη, b είναι το διάνυσμα τιμών κατοφλίου και σ είναι η σιγμοειδής συνάρτηση (sigmoid function → $\sigma(x) = \frac{1}{1+e^{-x}}$). Τέλος, $\odot$ είναι το γινόμενο των διανυσμάτων στοιχείο προς στοιχείο.

Ο τρόπος πρόβλεψης του διανύσματος X^t επιτυγχάνεται με την χρήση ενός κινούμενου παραθύρου που αποτελείται από έναν αριθμό προηγούμενων χρονοθυρίδων.



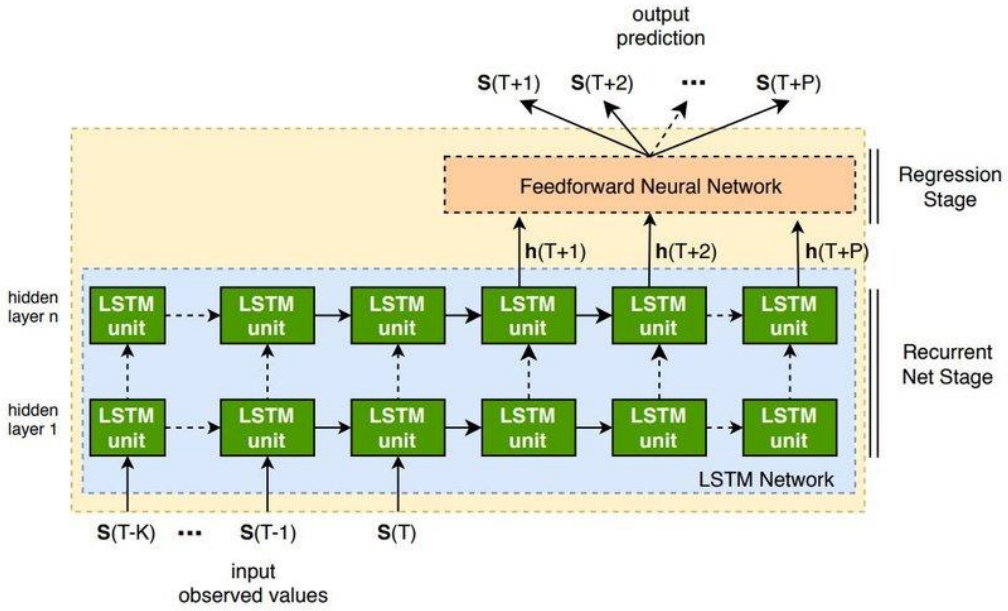
Εικόνα 2.4.2.3: Μέθοδος Κινούμενου Παραθύρου (Sliding Window) [21]

Εισάγουμε τον πίνακα κίνησης στο πρώτο επίπεδο του μοντέλου LSTM χρησιμοποιώντας το κινούμενο παράθυρο.



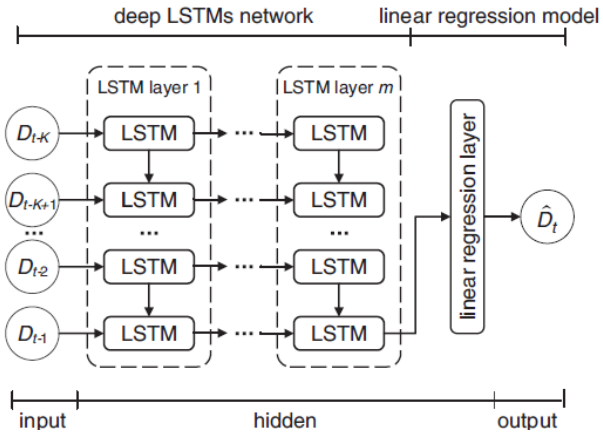
Εικόνα 2.4.2.4: Στάδια επεξεργασίας δεδομένων εισόδου [22]

Με αυτόν τον τρόπο το μοντέλο LSTM δέχεται ως είσοδο μεγάλο όγκο δεδομένων μέσω πολλών χρονικών βημάτων. Η χρήση περισσότερων επιπέδων LSTM (Stacked LSTM layers) επιφέρει καλύτερα αποτελέσματα και αποθηκεύει τις χρονικές και χωρικές συσχετίσεις των δεδομένων [22]. Στην παρακάτω απεικόνιση φαίνεται ένα μοντέλο LSTM n επιπέδων.



Εικόνα 2.4.2.5: Μοντέλο LSTM n επιπέδων [38]

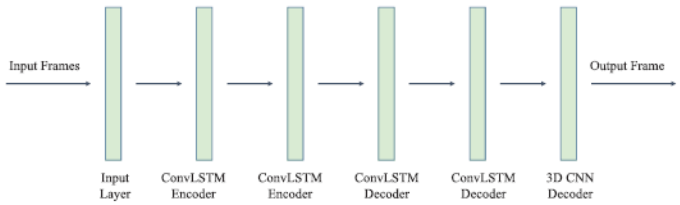
Στην εργασία [39] το μοντέλο DLSTM αποτελείται από m στοιβαγμένα LSTM στρώματα που χρησιμοποιούνται για την εύρεση των χωροχρονικών χαρακτηριστικών του δικτύου. Το στρώμα εξόδου περιέχει ένα μοντέλο γραμμικής παλινδρόμησης που υπολογίζει το μελλοντικό πίνακα κίνησης. Παρακάτω παρουσιάζεται η αρχιτεκτονική του μοντέλου DLSTM.



Εικόνα 2.4.2.6: Μοντέλο DLSTM [39]

Συγκεκριμένα η είσοδος είναι η $X_t = [D_{t-1}, \ldots, D_{t-K}]$ και η έξοδος η $Y_t = [D_t]$. Τροφοδοτώντας συνεχώς το μοντέλο με $T - K$ δεδομένα εκπαίδευσης αποκτάμε ένα αρκετά ακριβές μοντέλο πρόβλεψης. Το μοντέλο ανιχνεύει τα χωροχρονικά χαρακτηριστικά της κίνησης του δικτύου μέσω του πίνακα κυκλοφορίας και δημιουργεί μια συνάρτηση πρόβλεψης. Έτσι, όταν το μοντέλο τροφοδοτηθεί με άγνωστα δεδομένα κίνησης θα μπορεί να υπολογίσει την έξοδο γρήγορα και με ακρίβεια.

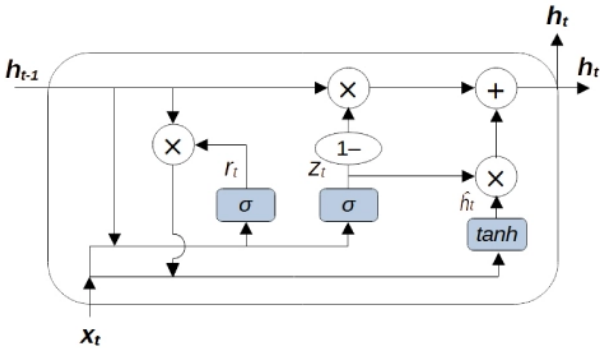
Μια διαφορετική προσέγγιση για πρόβλεψη δικτυακής κίνησης είναι τα μοντέλα Convolutional LSTM [40]. Τα μοντέλα αυτά είναι γνωστά σε προβλήματα πρόβλεψης βιντεο [41] [42] καθώς επεξεργάζονται L καρέ βιντεο προκειμένου να εκτιμηθεί το επόμενο καρέ. Γίνεται κανονικοποίηση των τιμών του πίνακα κίνησης μεταξύ των τιμών 0,1 ώστε να αντιμετωπίζεται ο πίνακας σαν ενα καρε ενός ασπρόμαυρου βίντεο, όπου κάθε πιξελ αντιστοιχεί σε μια τιμή του δισδιάστατου πίνακα. Το μοντέλο LSTM είναι σχεδιασμένο για πίνακα 1 διάστασης, ενώ το ConvLSTM εφαρμόζεται σε εικόνες που είναι 2 διαστάσεις. Επομένως, τα διανύσματα X_t, M_t, C_t πλέον αναφέρονται σε δισδιάστατους πίνακες. Οι σχέσεις που ισχύουν σε αυτό το δίκτυο είναι ίδιες με τις σχέσεις του μοντέλου LSTM. Στην εργασία [40] χρησιμοποιούνται δύο επίπεδα ConvLSTM ως κωδικοποιητές που εξάγουν τα χαρακτηριστικά του δικτύου και άλλα δύο επίπεδα ConvLSTM ως αποκωδικοποιητές που είναι υπεύθυνοι για την παραγωγή της εξόδου. Τέλος, απαραίτητο είναι ένα επίπεδο 3D-CNN που μετατρέπει την έξοδο στην επιθυμητή μορφή εικόνας.



Εικόνα 2.4.2.7: Μοντέλο ConvLSTM [40]

Μια άλλη αξιοσημείωτη προσπάθεια για πρόβλεψη επόμενου καρε έγινε στην εργασία [43], όπου αναπτύχθηκαν δυο αρχιτεκτονικές. Στην πρώτη αρχιτεκτονική (sequence-to-one) η είσοδος στο μοντέλο είναι μια ομάδα από καρε μεταξύ των χρονικών στιγμών t και $t + k$ όπου επεξεργάζονται με στόχο την πρόβλεψη του επόμενου καρε. Στην δεύτερη αρχιτεκτονική (sequence-to-sequence) η είσοδος είναι διαδοχικά καρέ που τροφοδοτούν το δίκτυο ξεχωριστά ένα προς ένα. Το καρέ της χρονικής στιγμης t προβλέπει το αμέσως επόμενο $t + 1$ και αυτό επαναλαμβάνεται μέχρι να γίνει πρόβλεψη του $t + m$. Στην πρώτη περίπτωση το μοντέλο εστιάζει κυρίως στα χωρικά χαρακτηριστικά των δεδομένων, ενώ στην δεύτερη περίπτωση εξετάζεται περισσότερο η χρονική ακολουθία.

Το μοντέλο "Gated Recurrent Units" ή κοινώς GRU είναι recurrent network που βασίζεται στο LSTM, επειδή και αυτό έχει πύλες που διαχειρίζεται την πληροφορία σε κάθε νευρώνα, όμως δεν έχουν ξεχωριστά κελιά για να αποθηκεύουν το cell state. Επίσης, οι πύλες είναι λιγότερες, μιας και λείπει η πύλη output gate και η forget gate, τις οποίες αντικαθιστα η πύλη update gate. Παρακάτω παρουσιάζεται η αρχιτεκτονική του μοντέλου GRU .



Εικόνα 2.4.2.8: Αρχιτεκτονική Μοντέλου GRU [44]

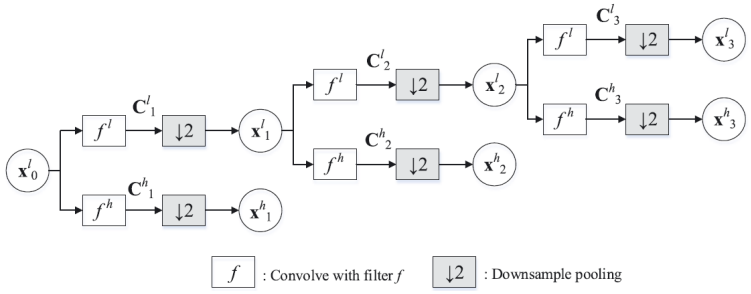
Οι μαθηματικές σχέσεις που υπάρχουν στο μοντέλο GRU όπως αυτό εμφανίζεται στην παραπάνω απεικόνιση είναι οι εξής [45]:

$$h_t = (1 - z_t)h_{t-1} + z_t\hat{h}_t$$
$$z_t = \sigma(W_z x_t + U_z h_{t-1})$$
$$\hat{h}_t = tanh(Wx_t + U(r_t \odot h_{t-1}))$$
$$r_t = \sigma(W_r x_t + U_r h_{t-1})$$

Να σημειωθεί U_z είναι ο πίνακας βαρών της προηγούμενης κατάστασης και W_z είναι ο πίνακας βαρών της εισόδου. Η update gate z_t αποφασίζει εάν η κατάσταση του κελιού πρέπει να ενημερωθεί με την υποψήφια κατάσταση (τρέχουσα τιμή ενεργοποίησης) ή όχι. Η reset gate r_t χρησιμοποιείται για να αποφασίσει εάν η προηγούμενη κατάσταση κελιού είναι σημαντική ή όχι. Η τελική κατάσταση κελιού h_t εξαρτάται από την update gate. Μπορεί ή όχι να ενημερωθεί

με την candidate activation. Αφαιρεί κάποιο περιεχόμενο από την τελευταία κατάσταση του κελιού και γράφει κάποιο νέο περιεχόμενο [46]. Τέλος, ⊙ είναι το γινόμενο των διανυσμάτων στοιχείο προς στοιχείο. Μετά από πειράματα έχει φανεί ότι το GRU τις περισσότερες φορές είναι ισάξιο με το LSTM και προτιμάται λόγω χαμηλότερης χρήσης της μνήμης και γιατί εκπαιδεύεται πιο γρήγορα. Στην εργασία [47] συγκρίνονται τα σφάλματα των δύο μοντέλων και άλλων δύο τεχνικών. Εξετάζεται η τεχνική LRCN (Long-term Recurrent Convolutional Networks) που χρησιμοποιεί στρώματα CNN για εξαγωγή χαρακτηριστικών εικόνας και στρώματα LSTM για πρόβλεψη ακολουθίας δεδομένων. Επίσης, εξετάζεται η τεχνική TCN (Temporal Convolutional Networks) που βασίζεται στην αρχιτεκτονική CNN. Αποτελείται από διασταλμένα (dilated) συνελικτικά στρώματα με τα ίδια μήκη εισόδου και εξόδου και από υπολειπόμενη (residual) σύνδεση [48].

Άλλη μια προσπάθεια για βελτίωση των προβλέψεων έγινε με το μοντέλο WSTNet βασισμένο στο wavelet multiscale analysis [49]. Αρχικά, ο πίνακας κίνησης αποσυντίθεται προκειμένου να δημιουργηθούν πολυεπίπεδες χρονικές σειρές με δεδομένα διαφορετικής χρονικής κλίμακας. Χρησιμοποιείται διακριτός μετασχηματισμος Wavelet (DWT) με σκοπό να ανιχνευθούν μοτίβα σε επίπεδο χρόνου και συχνότητας χωρίζοντας τον πίνακα σε χρονοσειρές χαμηλής και υψηλής συχνότητας. Για παράδειγμα η ροή X_0^l, όπου $t = 0$ και l το ζεύγος κόμβων, περνάει από χαμηλό f^l και υψηλό f^h φίλτρο και αποσυντίθεται στο προσεγγιστικό X_1^l και στο λεπτομερές μέρος X_1^h. Το X_1^l παρουσιάζει τις κυριότερες πληροφορίες σε υψηλότερο επίπεδο από τα αρχικά σήματα, ενώ το X_1^h παρέχει πληροφορίες υψηλότερων συχνοτήτων. Το παρακάτω σχήμα είναι βοηθητικό στην κατανόηση της λειτουργίας του μετασχηματισμού.



Εικόνα 2.4.2.9: Λειτουργία Διακριτού Μετασχηματισμού Wavelet (DWT) [49]

Όπου οι ακολουθίες C_λ^l, C_λ^h συνδέονται με τις χρονοσειρές X_λ^l, X_λ^h με τις σχέσεις:

$$C_{\lambda+1}^l(p) = \sum_{k=1}^{K} x_\lambda^l(p + k - 1) \cdot f_k^l$$

$$C_{\lambda+1}^h(p) = \sum_{k=1}^{K} x_\lambda^l(p + k - 1) \cdot f_k^h$$

Μετά τον μετασχηματισμό των δεδομένων εφαρμόζεται convolutional στρώμα για να εντοπιστούν τα χωρικά μοτίβα της δικτυακής κίνησης. Τέλος, εφαρμόζεται στρώμα LSTM προκειμένου να ανιχνευθούν και τα χρονικά μοτίβα του πίνακα. Μια άλλη τεχνική που βασίζεται κι αυτή στον μετασχηματισμό wavelet ονομάζεται Multiresolution Learning (MRL) και εξετάστηκε στην εργασία [36]. Το σύνολο των δεδομένων μετά τον μετασχηματισμό εκπαιδεύει το νευρωνικό δίκτυο. Κάθε επίπεδο του νευρωνικού δικτύου εκπαιδεύεται με το προσεγγιστικό μέρος των δεδομένων και σε κάθε σήμα τα βάρη του δικτύου ανανεώνονται. Έτσι επιτυγχάνεται ομαλή μετάβαση μεταξύ των σταδίων μάθησης.

Τέλος, θα αναφέρουμε δύο μοντέλα πρόβλεψης κίνησης που ανήκουν στην κατηγορία των Νευρωνικών Δικτύων με Πρόσθια Ανατροφοδότηση (Feedforward Neural Networks - FNN) [50 , 27]. Στα δίκτυα αυτά τα σήματα επιτρέπεται να ταξιδέψουν μόνο προς τα εμπρός. Στην [50] αξιολογούνται τα αποτελέσματα ενός Multilayer Perceptron, δηλαδή ενός πλήρους συνδεδεμένου FNN με 20 νευρώνες εισόδου, 12 και 6 νευρώνες στα κρυφά επίπεδα και 1 νευρώνα εξόδου. Στην εργασία [27] αναπτύσσεται μια τεχνική (Neural Network Ensemble) κατά την οποία 5 διαφορετικά δίκτυα εκπαιδεύονται και η τελική πρόβλεψη δίνεται από τον μέσο όρο των προβλέψεων των δικτύων. Έτσι, η επίδοση της εκπαίδευσης δεν θα εξαρτάται από τα αρχικά βάρη, αλλά μόνο από το κινούμενο χρονικό παράθυρο και από τους κρυφούς κόμβους των τοπολογιών. Οι παράμετροι αυτοί του δικτύου θα υπολογιστούν με ευρετική μέθοδο κι έπειτα όλα τα δίκτυα θα εκπαιδευτούν με το σύνολο των δεδομένων. Όπως αποδεικνύεται [27], η μέθοδος αυτή έχει τα επιθυμητά αποτελέσματα.

3. Θεωρητικό Υπόβαθρο Βαθιάς Μάθησης

Στην εργασία μας θα κάνουμε πρόβλεψη με Βαθιά Μάθηση (Deep Learning) για αυτόν τον λόγο στο κεφάλαιο αυτό θα αναλύσουμε το θεωρητικό υπόβαθρο της Βαθιάς Μάθησης. Επίσης, θα αναλύσουμε και άλλες κατηγορίες της Τεχνητής Νοημοσύνης όπως η Μηχανική Μάθηση (Machine Learning) και τα Νευρωνικά Δίκτυα (Neural Networks). Θα αναφέρουμε τις διαφορές των κατηγοριών και θα κάνουμε μια επισκόπηση στα μοντέλα και τις τεχνικές που θα χρησιμοποιήσουμε αργότερα στο κεφάλαιο 4.

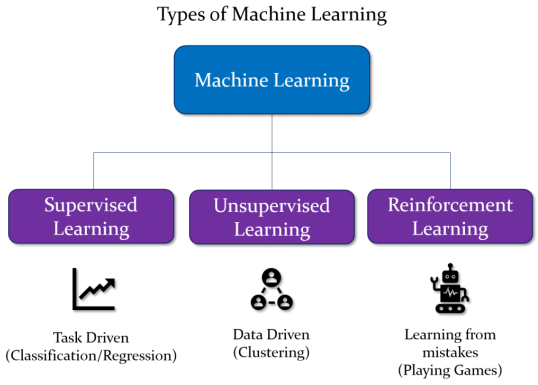
3.1 Τεχνητή Νοημοσύνη

Η Τεχνητή Νοημοσύνη (Artificial Intelligence - AI) αναφέρεται στον κλάδο της πληροφορικής ο οποίος ασχολείται με τη σχεδίαση και την υλοποίηση υπολογιστικών συστημάτων που μιμούνται στοιχεία της ανθρώπινης συμπεριφοράς τα οποία έχουν έστω και στοιχειώδη ευφυΐα. Επιχειρεί όχι μόνο να κατανοήσει, αλλά και να κατασκευάσει νοήμονες μηχανές συνδυάζοντας την επιστήμη των υπολογιστών με ισχυρά σύνολα δεδομένων, με στόχο την επίλυση προβλημάτων. Η τεχνητή νοημοσύνη μπορεί να συστηματοποιεί και να αυτοματοποιεί τις διανοητικές εργασίες, γι' αυτό και εφαρμόζεται σε κάθε πτυχή της ανθρώπινης δραστηριότητας. Η σύγχρονη τεχνητή νοημοσύνη αποτελεί ένα από τα πλέον «μαθηματικοποιημένα» και ταχέως εξελισσόμενα πεδία της πληροφορικής. Είναι δυνατό να επεξεργαστεί τεράστιες ποσότητες δεδομένων, κάτι που ο άνθρωπος δεν είναι ικανός, με σκοπό την αναγνώριση μοτίβων και την λήψη αποφάσεων. Η δύναμη των υπολογιστών χρησιμοποιείται για διεργασίες που παραδοσιακά απαιτούσαν ανθρώπινη νοημοσύνη. Περιλαμβάνει ένα τεράστιο πεδίο εφαρμογών, όπως την επεξεργασία φυσικής γλώσσας, την ανίχνευση αντικειμένων, την όραση υπολογιστών, τη ρομποτική. Άλλη μία δυνατότητα είναι ότι προσαρμόζεται μέσω προοδευτικών αλγορίθμων εκμάθησης ώστε να αφεθούν τα δεδομένα να κάνουν τον προγραμματισμό. Για παράδειγμα όπως ο αλγόριθμος μπορεί να διδάξει τον εαυτό του τον τρόπο να παίζει σκάκι , μπορεί και να τον διδάξει ποιο προϊόν να συστήσει στη συνέχεια διαδικτυακά. Και έτσι τα μοντέλα προσαρμόζονται όταν τους δίνονται νέα δεδομένα. Η οπισθο-διάδοση (Back-propagation) είναι μια τεχνική της τεχνητής νοημοσύνης που επιτρέπει στο μοντέλο να προσαρμόζεται μέσω εκπαίδευσης και επιπλέον δεδομένων όταν η πρώτη απάντηση δεν είναι η ενδεδειγμένη [51]. Υπάρχουν πολλές προσεγγίσεις για την επίτευξη αυτού του στόχου, κάποιες από τις οποίες είναι η μηχανική μάθηση (machine learning) και η βαθιά μάθηση (deep learning). Υπάρχουν επίσης διαφορετικοί τύποι τεχνητής νοημοσύνης, όσον αφορά την προσέγγιση του θέματος, για παράδειγμα η ισχυρή (General Intelligence – Strong AI) και η ασθενής τεχνητή νοημοσύνη (Narrow Intelligence – Weak AI). Η ισχυρή τεχνητή νοημοσύνη μας δίνει μια αίσθηση για το πως ο ανθρώπινος εγκέφαλος λειτουργεί, δεν ελέγχεται στη διαδικασία επίλυσης προβλημάτων και είναι σε θέση να διδαχθεί πράγματα, όπως στα παιχνίδια να δέχεται και να περιμένει κινήσεις. Αντίθετα, η ασθενής δεν μας δίνει αυτή τη

δυνατότητα, αλλά μπορεί και προσεγγίζει την ανθρώπινη συμπεριφορά. Η Siri της Apple, η Alexa της Amazon και η αναγνώριση προσώπου στα κινητά θεωρούνται ασθενή προγράμματα τεχνητής νοημοσύνης [52].

3.2 Μηχανική Μάθηση

Η Μηχανική Μάθηση (Machine Learning) είναι ένα υποσύνολο της τεχνητής νοημοσύνης, όπου επικεντρώνεται στο να κάνει τις μηχανές να μάθουν και να πάρουν αποφάσεις. Αφορά ένα σύνολο τεχνικών, όπου οι υπολογιστές εκπαιδεύονται από παραδείγματα και εμπειρία, να εκτελούν μια συγκεκριμένη εργασία αντί να προγραμματίζονται ρητά και με τη βοήθεια των ανθρώπων. Οι μηχανές συμπεριφέρονται όπως η ανθρώπινη νοημοσύνη και αυτό επιτυγχάνεται με το να τροφοδοτούνται με δεδομένα [52]. Είναι μια μορφή στατιστικής με έμφαση στην εκτίμηση περίπλοκων συναρτήσεων. Η μηχανική μάθηση ταιριάζει μαθηματικά μοντέλα σε δεδομένα προκειμένου να αντληθούν κάποιες γνώσεις ή να γίνει κάποια πρόβλεψη [53]. Μπορούμε να φανταστούμε τον αλγόριθμο της μηχανικής μάθησης, σαν μια εφαρμογή η οποία συντονίζει τις μεταβλητές ώστε να αντιστοιχίσει τις τιμές που λαμβάνει ως είσοδο με τις τιμές που δίνει ως έξοδο. Πλέον με την μηχανική μάθηση από τα δεδομένα εξαρτάται που θα οδηγηθεί η κατάσταση και ποιές ενέργειες θα γίνουν και όχι από τους προγραμματιστές. Η μηχανική μάθηση δεν είναι μόνο ένα προγραμματιστικό πρόβλημα άλλα είναι απαραίτητη για να έχει ένα σύστημα την ικανότητα της μάθησης. Το εκπαιδευόμενο πρόγραμμα βλέπει παραδείγματα και οι παράμετροι ανανεώνονται σε κάθε παράδειγμα ώστε η απόδοση να βελτιώνεται σταδιακά [54]. Οι αλγόριθμοι μηχανικής μάθησης μπορούν να χωριστούν κυρίως σε 3 μέρη. Αρχικά, το κομμάτι της απόφασης που βασίζεται στα δεδομένα εισόδου που αφορά πρόβλεψη δεδομένων ή ταξινόμηση δεδομένων. Έπειτα, το μέρος της συνάρτησης σφαλμάτων που αξιολογεί τα αποτελέσματα του αλγορίθμου και εκτιμά την ακρίβεια του μοντέλου. Τέλος, η διαδικασία βελτιστοποίησης του μοντέλου με την διαρκή ανανέωση των βαρών μέχρι να επιτευχθεί η επιθυμητή ακρίβεια [55]. Μπορούμε να διακρίνουμε τρεις κατηγορίες μηχανικής μάθησης [52] οι οποίες αναλύονται στις επόμενες ενότητες.



Εικόνα 3.2: Κατηγορίες Μηχανικής Μάθησης [56]

3.2.1 Επιβλεπόμενη Μάθηση - Supervised Machine Learning

Η επιβλεπόμενη μάθηση έχει ως είσοδο υπογεγραμμένα δεδομένα (Labeled Data), δηλαδή δεδομένα που είναι αποθηκευμένα μαζί με κάποια χαρακτηριστικά τους γνωρίσματα, που εκπαιδεύουν τον αλγόριθμο να κατηγοριοποιεί ή να προβλέπει την έξοδο. Δηλαδή, οι μηχανές προσπαθούν να βγάλουν συμπεράσματα αποκλειστικά βάσει παλαιότερων δεδομένων που συνέλεξαν. Supervised Learning είναι κάθε αλγόριθμος που χαρτογραφεί τις μεταβλητές εισόδου X σε μεταβλητές εξόδου Y. Στόχος είναι να παραχθεί μια ακριβής συνάρτηση χαρτογράφησης η οποία σε κάθε είσοδο θα μπορεί να δίνει μια καλή έξοδος. Από τη στιγμή που το μοντέλο τροφοδοτείται με δεδομένα τα βάρη προσαρμόζονται κατάλληλα ώστε να ταιριάζουν με αυτό. Για παράδειγμα, σε μια εφαρμογή επιβλεπόμενης μάθησης για επεξεργασία εικόνας, αρχικά παρέχουμε στο σύστημα εικόνες συγκεκριμένης επισήμανσης, όπως αυτοκίνητα, λεωφορεία και μηχανάκια (training data). Μετά από βαθιά επεξεργασία το σύστημα πρέπει να είναι σε θέση να μπορεί να διακρίνει και κατηγοριοποιεί μελλοντικές μη επισημασμένες εικόνες (test data). Ένα τέτοιο πρόβλημα, δηλαδή που η τιμή εξόδου είναι μια κατηγορία, «κόκκινο» ή «κίτρινο» και «αυτοκίνητο» ή «λεωφορείο», ονομάζεται πρόβλημα ταξινόμησης [52]. Ένα άλλο παράδειγμα Supervised Machine Learning είναι οι προσωποποιημένες προτάσεις προϊόντων που προτείνει σε κάθε χρήστη η Amazon, με βάση τα προϊόντα που αγόρασε ή απλά είδε στο παρελθόν. Κάποιες μέθοδοι που συναντάμε στην επιβλεπόμενη μάθηση είναι τα νευρωνικά δίκτυα, οι ταξινομητές naïve bayes, οι γραμμική παλινδρόμηση (linear regression), τα τυχαία δάση (random forest) και ο αλγόριθμος Support Vector Machine (SVM) [55].

3.2.2 Μάθηση χωρίς Επίβλεψη - Unsupervised Machine Learning

Τα δεδομένα εισόδου σε αυτή την περίπτωση, δεν είναι δομημένα και δεν φέρουν κάποια ετικέτα, γι' αυτό και δεν γίνεται να ταξινομηθούν σε κατηγορίες. Οι αλγόριθμοι εδώ προσπαθούν να εντοπίσουν διάφορα άγνωστα μοτίβα και ομοιότητες ή διαφορές στα δεδομένα, ώστε να γίνει ο συσχετισμός τους χωρίς την ανθρώπινη βοήθεια. Η ικανότητα αυτών των μεθόδων να βρίσκουν ομοιότητες και διαφορές σε άγνωστα δεδομένα είναι χρήσιμη στην ανάλυση δεδομένων, στην τμηματοποίηση των πελατών, στην διασταύρωση πωλήσεων και στην αναγνώριση εικόνων [55]. Τα προβλήματα μη επιτηρούμενης μάθησης, μπορούν να κατηγοριοποιηθούν σε προβλήματα clustering, που γίνεται η ομαδοποίηση των δεδομένων, και σε προβλήματα συσχέτισης, που ανακαλύπτονται οι κανόνες που περιγράφουν αυτά τα δεδομένα. Για παράδειγμα, η μέθοδος αυτή μπορεί να εφαρμοστεί όταν θέλει μια εταιρεία να προωθήσει στην αγορά ένα νέο προϊόν. Από τα δημογραφικά δεδομένα των πελατών της και τις προηγούμενες συναλλαγές τους θα δει τι είδους πελάτες συνήθως την προσεγγίζουν. Έπειτα θα τους ομαδοποιήσει και θα δημιουργήσει στρατηγικές για να παρέχει συγκεκριμένες υπηρεσίες και προϊόντα στις διάφορες ομάδες [54]. Κάποιες μέθοδοι που συναντάμε στην μη επιβλεπόμενη

μάθηση είναι τα νευρωνικά δίκτυα, η ομαδοποίηση k μέσων (k-means algorithm) και πιθανοτικές μέθοδοι ομαδοποίησης.

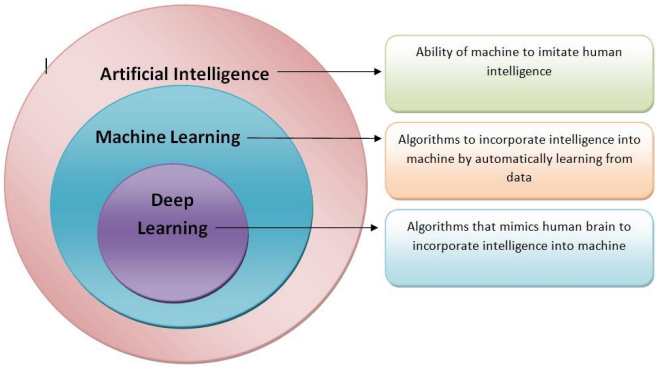
3.2.3 Ενισχυτική Μάθηση - Reinforcement Machine Learning

Η ενισχυτική μάθηση είναι ένα σύνολο τεχνικών στις οποίες το σύστημα μάθησης προσπαθεί να μάθει μέσα από την άμεση αλληλεπίδραση με το περιβάλλον. Το σύστημα δεν εκπαιδεύεται από την αρχή αλλά μαθαίνει να παίρνει αποφάσεις κάνοντας συγκεκριμένες κινήσεις. Η εκπαίδευση γίνεται μέσα από την ανταμοιβή ή την τιμωρία που θα δοθεί σε περίπτωση που οι κινήσεις βοηθούν ή εμποδίζουν την επίτευξη του στόχου. Σκοπός είναι με τις κινήσεις που θα γίνουν να μεγιστοποιηθεί η ανταμοιβή και να ελαχιστοποιηθεί η ποινή, έτσι ώστε να ληφθούν σωστές αποφάσεις σε νέα δεδομένα [52]. Αυτό το είδος μηχανικής μάθησης αποτελεί παραλλαγή της μάθησης με επίβλεψη. Εδώ δεν υπάρχει επιθυμητή απόκριση για κάθε είσοδο του συστήματος αλλά για κάθε απόκριση, το σύστημα λαμβάνει ένα σήμα ενίσχυσης (reinforcement signal) το οποίο υποδηλώνει κατά πόσο η απόκριση ήταν σωστή ή λανθασμένη. Το σύστημα δεν καθοδηγείται από κάποιον εξωτερικό επιβλέποντα για το ποια ενέργεια θα πρέπει να ακολουθήσει αλλά πρέπει να ανακαλύψει μόνο του ποιες ενέργειες είναι αυτές που θα του αποφέρουν το μεγαλύτερο κέρδος. Εφαρμόζεται στον έλεγχο κίνησης ρομπότ στη βελτιστοποίηση εργασιών σε εργοστάσια, στη μάθηση επιτραπέζιων παιχνιδιών, κτλ. Για την ανάπτυξη ενός μοντέλου ενισχυτικής μάθησης που θα ασχοληθεί με ένα παιχνίδι χρησιμοποιείται ένας νοήμων πράκτορας. Το περιβάλλον μέσα στο οποίο, βρίσκεται και αλληλεπιδρά, είναι το ίδιο το παιχνίδι και αυτός καλείται να πάρει τις σωστές αποφάσεις που θα τον οδηγήσουν στη νίκη. Χρησιμοποιεί ενισχυτική μάθηση για να βελτιώνει τις επιδόσεις του, γι' αυτό και ονομάζεται πράκτορας ενισχυτικής μάθησης (Reinforcement Learning Agent). Σε κάθε χρονική στιγμή ο πράκτορας μελετά την τρέχουσα κατάσταση του περιβάλλοντος και επιλέγει την πραγματοποίηση μιας ενέργειας, ακολουθώντας μια πολιτική. Ως αποτέλεσμα της ενέργειάς του, το περιβάλλον μεταπίπτει σε μία νέα κατάσταση και ο πράκτορας λαμβάνει μια ανταμοιβή [57].

3.3 Βαθιά Μάθηση

Η Βαθιά Μάθηση (Deep Learning) είναι μια υποκατηγορία της Μηχανικής Μάθησης (Εικόνα 3.3) η οποία λειτουργεί με παρόμοια δομή και λειτουργία με αυτή του ανθρώπινου εγκεφάλου. Καταλαμβάνει ένα μεγάλο μέρος των τεχνικών και των αρχιτεκτονικών της μηχανικής μάθησης με κύριο χαρακτηριστικό την χρήση πολλών στρωμάτων μη γραμμικών πληροφοριών [58]. Είναι μια διαδικασία στην οποία αναγνωρίζονται και ταξινομούνται εικόνες, κείμενο, ήχος, αριθμοί, όμως τα δεδομένα είναι πολύπλοκα και αδόμητα σε αντίθεση με τη μηχανική μάθηση που

χρησιμοποιεί πιο απλές δομές. Πρόκειται στην ουσία, για μια προσπάθεια να μιμηθούν οι μηχανές το πώς ο άνθρωπος σκέφτεται και μαθαίνει. Οι μέθοδοι που χρησιμοποιούνται εκπαιδεύουν τις υπολογιστικές μηχανές να επεξεργάζονται τα δεδομένα όπως ο ανθρώπινος εγκέφαλος. Η μηχανική μάθηση και η βαθιά μάθηση έχουν αλληλένδετα πεδία αλλά η βαθιά μάθηση χρησιμοποιείται κυρίως για να λύσει περίπλοκα προβλήματα [52]. Χρειάζεται περισσότερα δεδομένα και περισσότερο χρόνο εκπαίδευσης, μαθαίνει και βελτιώνεται το σύστημα από τα λάθη του και το κυριότερο, έχει μεγαλύτερη ακρίβεια από τα πιο απλοϊκά μοντέλα μηχανικής μάθησης. Μερικά πεδία εφαρμογών του deep learning είναι η εξυπηρέτηση πελατών, η ιατρική, οι χρηματοπιστωτικές υπηρεσίες και η αυτοκινητοβιομηχανία. Μερικές εφαρμογές που υπάγονται στο πεδίο της βαθιάς μάθησης είναι τα τηλεχειριστήρια με φωνητικό έλεγχο, τα αυτόνομα αυτοκίνητα, η ασφάλεια των πιστωτικών καρτών και τα chatbots.



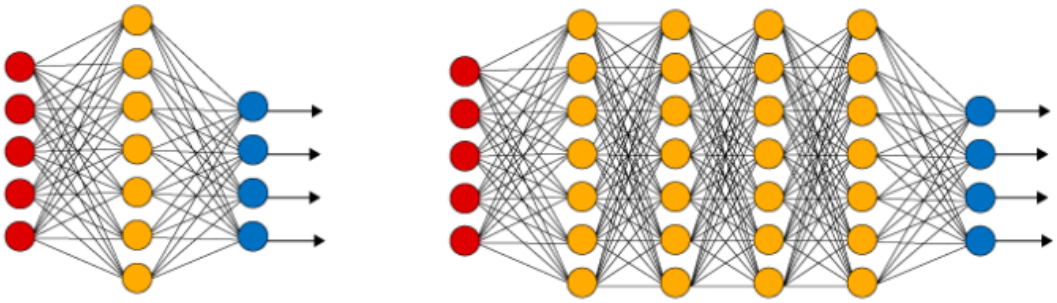
Εικόνα 3.3: Απεικόνιση της σχέσης AI – ML και DL [59]

3.4 Νευρωνικά Δίκτυα

3.4.1 Ορισμός και Δομή Νευρωνικών Δικτύων

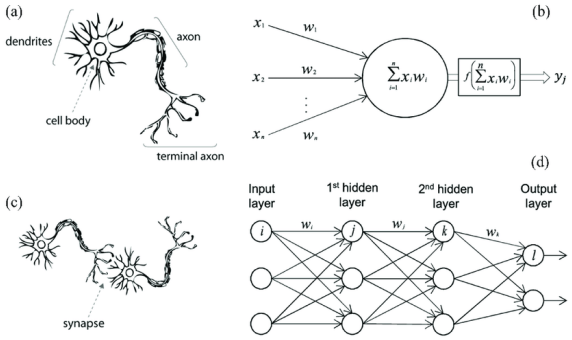
Νευρωνικό Δίκτυο (Neural Networks - NN) είναι ένα υπολογιστικό σύστημα το οποίο αποτελείται από υπολογιστικές μονάδες (νευρώνες) διασυνδεδεμένες μεταξύ τους. Οι νευρώνες επεξεργάζονται πληροφορία και με τη μεταξύ τους αλληλεπίδραση καταλήγουν σε μια έξοδο. Οι μεταξύ τους συνδέσεις έχουν αριθμητικά βάρη τα οποία συντονίζονται με βάση την απόκτηση εμπειρίας από την εκπαίδευση του συστήματος , πράγμα που καθιστά τα νευρωνικά δίκτυα ικανα να μαθαίνουν. Ένα νευρωνικό δίκτυο είναι βασικά ένα πρότυπο προγραμματισμού ή ένα σύνολο αλγορίθμων που επιτρέπει στον υπολογιστή να μάθει από τα δεδομένα παρατήρησης. Τα τεχνητά νευρωνικά δίκτυα αποτελούν μια οικογένεια στατιστικών μοντέλων μάθησης εμπνευσμένη από τα βιολογικά νευρωνικά δίκτυα και χρησιμοποιούνται για την προσέγγιση λειτουργιών οι οποίες εξαρτώνται από ένα μεγάλο αριθμό εισόδων. Στην απλούστερη μορφή του ένα νευρωνικό δίκτυο δέχεται μια πληροφορία στο στρώμα εισόδου την επεξεργάζεται και την

μεταβιβάζει σε ένα κρυφό στρώμα και εκεί μετά από επεξεργασία μεταφέρεται στο στρώμα εξόδου. Η βαθιά μάθηση που αναλύσαμε προηγουμένως αποτελείται από νευρωνικά δίκτυα αρκετών κρυφών στρωμάτων που επεξεργάζονται μεγάλο αριθμό δεδομένων και επιτελούν πολύπλοκες λειτουργίες. Στο σχήμα 3.4.1.1 φαίνεται η διαφορά που έχει ένα απλό και ένα βαθύ νευρωνικό δίκτυο.



Εικόνα 3.4.1.1: Απλό (αριστερά) και Βαθύ Νευρωνικό Δίκτυο (δεξια) [60]

Οι πληροφορίες που εισάγονται στα δίκτυα επεξεργάζονται από τους τεχνητούς νευρώνες οι οποίοι είναι υπολογιστικά μοντέλα που δέχονται σήματα εισόδου τα οποία μεταβάλλονται από τιμές βάρους. Είναι βασισμένοι στους βιολογικούς νευρώνες, καθώς τα μέρη τους και οι λειτουργίες τους αντιστοιχίζονται άμεσα. Έχουν ικανότητα στην μάθηση μέσω παραδειγμάτων, στην αναγνώριση προτύπων και στην ανοχή σε σφάλματα [61]. Η έξοδος ενός νευρώνα υπολογίζεται από την σχέση $y = g(\sum x_i w_i + b)$, όπου g είναι η συνάρτηση ενεργοποίησης, w είναι τα βάρη και b είναι η προκατάληψη. Παρακάτω απεικονίζονται οι βιολογικοί νευρώνες (a) και τα βιολογικά νευρωνικά δίκτυα (c) σε αντιπαράθεση με τους τεχνητούς νευρώνες (b) και τα τεχνητά νευρωνικά δίκτυα (d).



Εικόνα 3.4.1.2: (a) οι βιολογικός νευρώνας, (b) τεχνητός νευρώνας, (c) βιολογικό νευρωνικό δίκτυο και (d) τεχνητό νευρωνικό δίκτυο [62]

Η δομή των νευρωνικών δικτύων αποτελείται από τρια επίπεδα. Το επίπεδο εισόδου, το κρυφό επίπεδο και το επίπεδο εξοδου. Το επίπεδο εισόδου αποτελείται από τους κόμβους που λαμβάνουν την πληροφορία από τον εξωτερικό παράγοντα. Γενικότερα, η είσοδος σε ένα

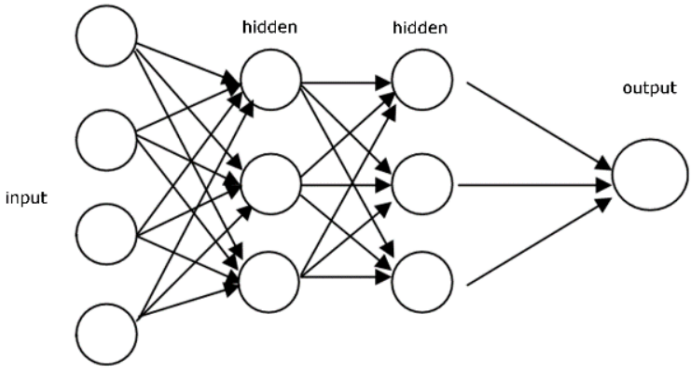
νευρωνικό δίκτυο εξαρτάται τον στόχο που θέλουμε να επιτύχουμε. Στο κρυφό επίπεδο γίνονται όλοι οι υπολογισμοί των δεδομένων. Η μαθηματική σχέση που περιγράφει έναν τεχνητό νευρώνα είναι η $y = \sum w_i^* x_i$. Ο αριθμός των κρυφών επιπέδων εξαρτάται από τον αλγόριθμο εκπαίδευσης, την πολυπλοκότητα της συνάρτησης του δικτύου και τον αριθμό των εισόδων και των εξόδων. Τέλος, το επίπεδο εξόδου που μπορεί να δομείται από περισσότερους του ενός κόμβους, εξαρτάται από την μορφή του αλγορίθμου, και μέσω αυτού εξέρχεται μια τιμή της επεξεργασμένης πληροφορίας στο περιβάλλον.

3.4.2 Κατηγορίες Νευρωνικών Δικτύων

Η αρχιτεκτονική ενός Νευρωνικού Δικτύου καθορίζεται από το πλήθος των στρωμάτων και τις συνδέσεις μεταξύ των νευρώνων. Τα ΝΔ με βάση την αρχιτεκτονική τους μπορούν να χωριστούν σε τρεις κατηγορίες. Στα Νευρωνικά Δίκτυα με Πρόσθια Τροφοδότηση (Feedforward Neural Networks), τα Συνελικτικά Νευρωνικά Δίκτυα (Convolutional Neural Networks) και τα Επαναλαμβανόμενα Νευρωνικά Δίκτυα (Recurrent Neural Networks).

3.4.2.1 Νευρωνικά Δίκτυα με Πρόσθια Τροφοδότηση (Feedforward Neural Networks)

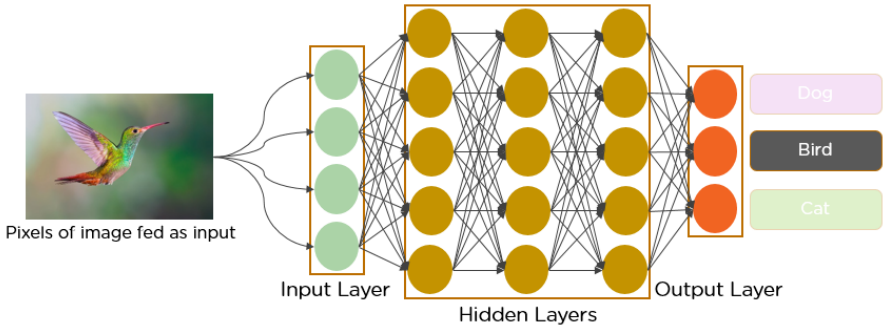
Τα δίκτυα με Πρόσθια Τροφοδότηση είναι η απλούστερη μορφή Νευρωνικών Δικτύων. Οι συνδέσεις των νευρώνων είναι προς μια κατεύθυνση και τα δεδομένα επεξεργάζονται και προωθούνται μόνο προς τα επόμενα στρώματα. Δεν υπάρχουν βρόχοι ανατροφοδότησης και η έξοδος ενος στρώματος επηρεάζει μόνο τα επόμενα στρώματα. Ανάλογα με τον αριθμό των κρυφών επιπέδων, διακρίνονται σε μονοεπίπεδα (Perceptron) και πολυεπίπεδα (Multilayer Perceptron) δίκτυα. Το Perceptron χρησιμοποιείται μόνο για την ταξινόμηση γραμμικά διαχωρίσιμων προτύπων. Ανήκει στην κατηγορία των γραμμικών ταξινομητών και μπορεί να διαχωρίσει γραμμικά τα δεδομένα μόνο σε δύο κλάσεις [63]. Τα πολυστρωματικά perceptron (MLP) αποτελούν μία από τις αρχιτεκτονικές μοντέλων που έχουν φανεί να έχουν καλή επίδοση σε προβλήματα πρόβλεψης χρονοσειρών. Συνήθως αποτελούν μία καλή αρχική εκτίμηση της λύσης και λόγω της μικρής υπολογιστικής πολυπλοκότητας που τα χαρακτηρίζει προσφέρονται ως εναρκτήρια σημεία σύγκρισης [64]. Τα νευρωνικά δίκτυα μπορούν να είναι πλήρως συνδεδεμένα, όταν κάθε κόμβος οποιουδήποτε στρώματος συνδέεται με όλους τους κόμβους του επόμενου στρώματος, ή μερικώς συνδεδεμένα. Στο παρακάτω σχήμα εμφανίζεται ένα feedforward πλήρως συνδεδεμένο δίκτυο.



Εικόνα 3.4.2.1: Feedforward πλήρως συνδεδεμένο νευρωνικό δίκτυο [65]

3.4.2.2 Συνελικτικά Νευρωνικά Δίκτυα (Convolutional Neural Networks)

Τα CNN δίκτυα, έχουν συνδεθεί κατά κύριο λόγο με εφαρμογές για εικόνες και βίντεο, γι' αυτό και χρησιμοποιούνται για ταξινόμηση εικόνων, ομαδοποίηση εικόνων και εντοπισμό αντικειμένων. Αυτό συμβαίνει γιατί η αρχιτεκτονική τους είναι τέτοια ώστε να μπορεί να γίνει σύνθετη οπτική ανάλυση. Χρησιμοποιούνται φίλτρα σε εικόνες για να εξάγουν χωρικά χαρακτηριστικά μεταξύ των pixels. Με αυτόν τον τρόπο τα δίκτυα προσδιορίζουν αντικείμενα στις εικόνες, την θέση τους και την σχέση τους με άλλα αντικείμενα εντός της εικόνας. Ακόμα ένα συνελικτικό δίκτυο μπορεί να αναγνωρίσει υποκείμενα μοτίβα και τις συσχετίσεις μεταξύ μελλοντικών και παρελθοντικών παρατηρήσεων και να παράξει ποιοτικές προβλέψεις [64].



Εικόνα 3.4.2.2.1: Κατηγοριοποίηση εικόνας με χρήση CNN [66]

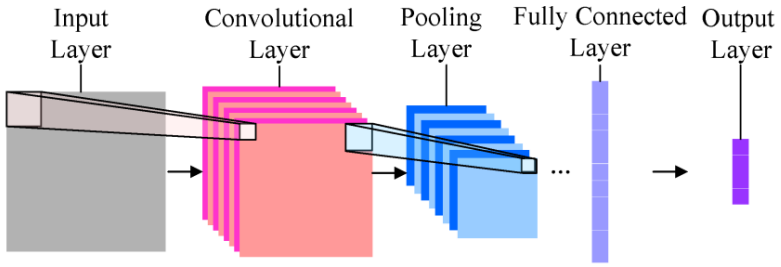
Είναι σχεδιασμένα ώστε να αναγνωρίσουν δισδιάστατα σχήματα με υψηλό βαθμό ανεκτικότητας στη μετατόπιση, την κλιμάκωση, τη στρέβλωση και σε άλλες μορφές παραμόρφωσης. Για να επιτευχθεί αυτός ο σκοπός κατά την εκπαίδευση του δικτύου ακολουθούνται τα ακόλουθα βήματα [67]:

- Εξαγωγή Χαρακτηριστικών: Κάθε νευρώνας συνδέεται με συγκεκριμένους κόμβους από το προηγούμενο επίπεδο και είναι υπεύθυνος να εξάγει ένα συγκεκριμένο χαρακτηριστικό από την εικόνα. Όταν γίνει η εξαγωγή του χαρακτηριστικού διατηρείται

η πληροφορία για την σχετική θέση του ως προς τα άλλα χαρακτηριστικά που δεν έχουν ακόμα εξαχθεί.

- Αντιστοίχιση Χαρακτηριστικών: Κάθε υπολογιστικό επίπεδο του δικτύου περιέχει πολλούς χάρτες χαρακτηριστικών, οι νευρώνες των οποίων ελέγχονται ώστε να μοιράζονται το ίδιο σύνολο συναπτικών βαρών. Έτσι το δίκτυο καθίσταται ανεξάρτητο από μετατοπίσεις.
- Υποδειγματοληψία: Μέσω της υποδειγματοληψίας μειώνεται η ανάλυση των χαρακτηριστικών στους χάρτες και η έξοδος γίνεται λιγότερο ευαίσθητη σε παραμορφώσεις.
- Εξαγωγή Προβλέψεων: Στο τέλος της δομής ενός συνελικτικού δικτύου προστίθεται ο κατάλληλος αριθμός από πλήρως συνδεδεμένα επίπεδα ώστε να εξαχθεί η επιθυμητή έξοδος του δικτύου.

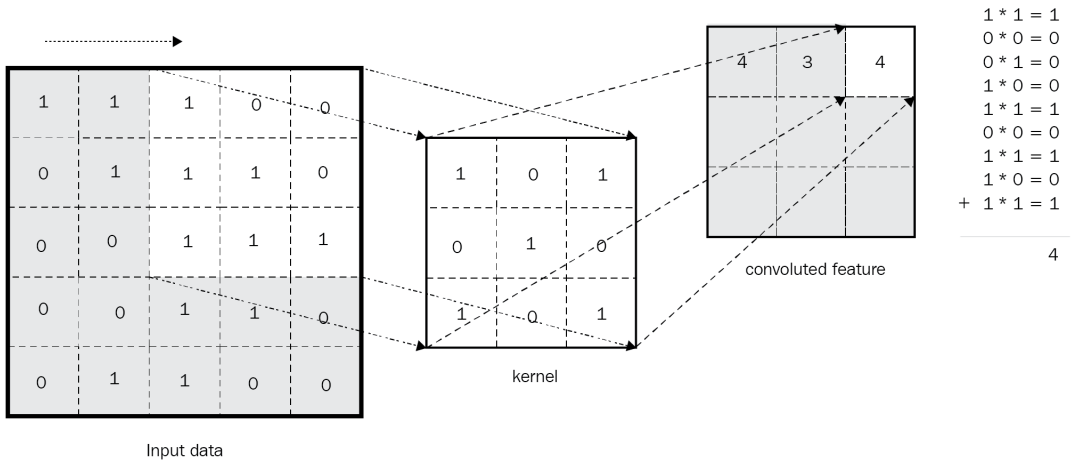
Προκειμένου να εκτελεστούν αυτές οι λειτουργίες είναι οι νευρώνες είναι ομαδοποιημένοι στα επίπεδα που βλέπουμε στην Εικόνα 3.4.2.2.2. Στη συνέχεια θα αναλύσουμε τον τρόπο κατασκευής και λειτουργίας της κάθε κατηγορίας επιπέδου:



Εικόνα 3.4.2.2.2: Σχηματική απεικόνιση της μεταβολής των δεδομένων εισόδου [68]

1. Convolutional Layer - Συνελικτικο Επίπεδο

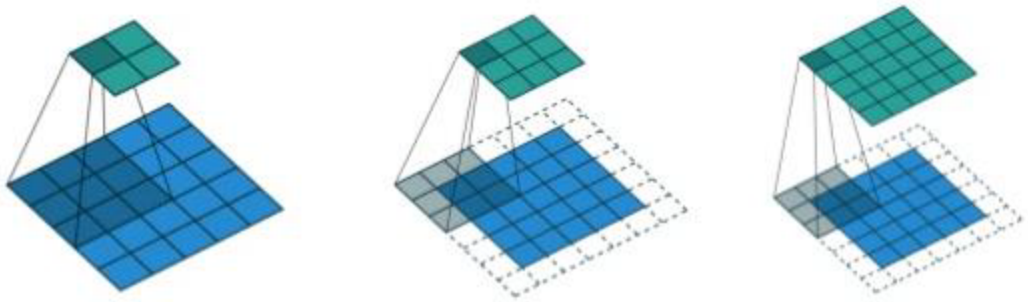
Είναι το πρώτο επίπεδο στο οποίο εξάγονται τα χαρακτηριστικά της εικόνας εισόδου, όπως είναι οι πλευρές, τα χρώματα και οι γωνίες. Σε ένα συνελικτικό επίπεδο οι παράμετροι του αντίστοιχου επιπέδου αποτελούνται από ένα σύνολο φίλτρων που μπορούν να μεταβάλουν τα βάρη τους ώστε να διδαχθούν. Ο αριθμός των φίλτρων που χρησιμοποιούνται καθορίζεται από τις παραμέτρους του συνελικτικού επιπέδου. Μετά την εφαρμογή των φίλτρων δημιουργείται ένας πίνακας που λέγεται χάρτης χαρακτηριστικών. Η διαδικασία αυτή απεικονίζεται στην παρακάτω εικόνα



Εικόνα 3.4.2.2.3: Εφαρμογή φίλτρου και εξαγωγή πίνακα χαρακτηριστικών [67]

Το φίλτρο ολισθαίνει στα δεδομένα εισόδου και παράγει συνελικτικά χαρακτηριστικά. Σε κάθε βήμα τα στοιχεία του φίλτρου πολλαπλασιάζονται ένα προς ένα με τα αντίστοιχα στοιχεία του πίνακα εισόδου. Υπάρχουν κάποιες παράμετροι που είναι σημαντικές σε αυτήν την διαδικασία και πρέπει να αναφερθούν [68, 69]. Το stride είναι μια από τις υπέρ-παραμέτρους που πρέπει να επιλέξει ο δημιουργός του νευρωνικού επιπέδου για κάθε ένα από τα συνελικτικά επίπεδα. Ουσιαστικά αφορά στο πόσο θα ολισθαίνει κάθε φορά το φίλτρο καθώς θα διασχίζει το σύνολο των δεδομένων εισοδου. Αν το stride είναι 1 τότε το φίλτρο θα μετακινείται στην εικόνα κατά ένα pixel την φορά. Αν όμως επιλεγεί μεγαλύτερη τιμή π.χ. 2 τότε μετακινούμαστε κατά 2 θέσεις τόσο κατά πλάτος όσο και κατά ύψος, οπότε η έξοδος που προκύπτει είναι μικρότερη. Όποτε ένας βασικός λόγος ύπαρξης του stride είναι η μείωση του επιπέδου εξόδου. Η άλλη σημαντική παράμετρος είναι το pad. Σε περιπτώσεις όπου οι χωρικές διαστάσεις των φίλτρων και αυτές της εισόδου δεν συμπίπτουν ακριβώς, ώστε να συνελισθούν τα πρώτα ομοιόμορφα στην δομή της εισόδου, τότε προσθέτουμε στην είσοδο κάποιες σειρές και στήλες ώστε το πρόβλημα να λυθεί. Πιο συγκεκριμένα, είναι ένα πλήθος από στοιχεία στα οποία δίνεται μία αρχική αυθαίρετη τιμή τα οποία επικολλώνται στα άκρα της εικόνας, μόνο στις κατευθύνσεις του πλάτους και του ύψους και η τιμή που δίνεται συνήθως είναι το 0. Ο γενικός τύπος για το μέγεθος της εξόδου που προκύπτει μετά την εφαρμογή του φίλτρου είναι:

$$\text{Διάσταση Εξόδου} = \frac{\text{Διάσταση Εισόδου} - \text{Διάσταση Φίλτρου} + 2 * padding}{stride} + 1$$



Εικόνα 3.4.2.2.4: Εφαρμογή διαφόρων μεγεθών φίλτρου με και χωρίς την εφαρμογή padding (1η περ. padding=0, stride=1, 2η περ. padding=1, stride=2, 3η περ. padding=1, stride=1) [68]

Πραγματοποιούνται πολλές συνελίξεις στην είσοδο, οι οποίες είναι τα διαφορετικά φίλτρα που εφαρμόζουμε. Ουσιαστικά το γεγονός ότι χρησιμοποιούνται μικρά φίλτρα τα οποία εφαρμόζονται σε όλο το μέγεθος της εικόνας προσφέρει τοπικότητα, η οποία είναι σημαντική αφού ένα χαρακτηριστικό θα εμφανιστεί σε μία περιοχή της εικόνας. Στο τέλος όλοι αυτοί οι πίνακες τοποθετούνται μαζί ως η τελική έξοδος του επιπέδου συνέλιξης.

2. Activation Function - Συνάρτηση Ενεργοποίησης (ReLU)

Σε κάθε Νευρωνικό Δίκτυο χρησιμοποιείται μια συνάρτηση ενεργοποίησης για να γίνει η έξοδος μη γραμμική. Στα δίκτυα CNN η συνάρτηση αυτή συνήθως είναι η Rectified Linear Unit (ReLU) [67].

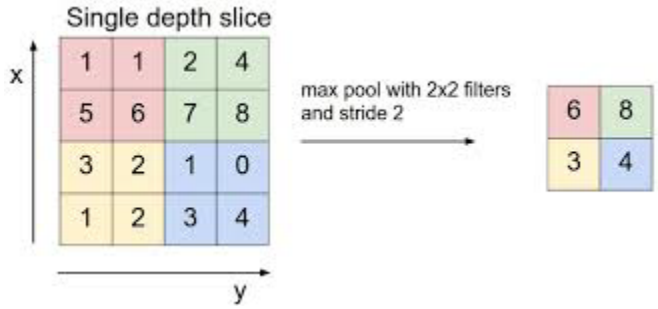
$$T(x) = \begin{cases} x, & \text{if } x \geq 0 \\ 0, & \text{if } x < 0 \end{cases}$$

Στην ουσία η συνάρτηση ReLU μετατρέπει κάθε αρνητική τιμή εισόδου σε μηδενική ώστε να αναδειχθούν οι μη γραμμικές συσχετίσεις των δεδομένων. Πρακτικά όταν σε μια περιοχή της εικόνας υπάρχει ένα μοτίβο τότε εκεί μαζεύονται πολλές θετικές τιμές, ενώ στις υπόλοιπες περιοχές που δεν εμφανίζεται αυτό το μοτίβο οι τιμές είναι αρνητικές ή μηδέν. Με την χρήση της ReLU ενεργοποιείται η έξοδος μόνο στις θετικές τιμές δηλαδή στις περιοχές που εμφανίζεται το συγκεκριμένο μοτίβο.

3. Pooling Layer - Συγκεντρωτικό Επίπεδο

Τα Συγκεντρωτικά Επίπεδα στοχεύουν στη μείωση της διάστασης του πίνακα χαρακτηριστικών. Έτσι μειώνεται συνεχώς ο χώρος αναπαράστασης των δεδομένων και των παραμέτρων του δικτύου. Πραγματοποιείται, δηλαδή, υποδειγματοληψία χωρίς όμως να υπάρχει απώλεια σημαντικών πληροφοριών. Η υποδειγματοληψία γίνεται με την βοήθεια ενός παραθύρου που ολισθαίνει πάνω στον πίνακα χαρακτηριστικών και διατηρεί του μέγιστο

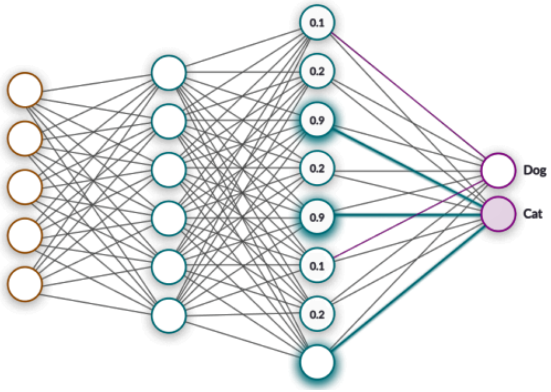
στοιχείο του χάρτη χαρακτηριστικών (Max Pooling) ή υπολογίζει το μέσο όρο των στοιχείων (Average Pooling) ή υπολογίζει το άθροισμα όλων το στοιχείων (Sum Pooling). Στην Εικόνα 3.4.2.2.5 παρατηρούμε την διαδικασία του Max Pooling.



Εικόνα 3.4.2.2.5: Λειτουργία Max Pooling [69]

4. Fully Connected Layer - Πλήρως Συνδεδεμένο Επίπεδο

Σε ένα πλήρως συνδεδεμένο επίπεδο κάθε νευρώνας συνδέεται με όλους τους νευρώνες του προηγούμενου επιπέδου. Σε ένα πρόβλημα ταξινόμησης ο σκοπός αυτού του επιπέδου είναι να χρησιμοποιήσει τα χαρακτηριστικά που εξάγονται από τα προηγούμενα επίπεδα, για να μπορέσει να ταξινομήσει την εικόνα σε διάφορες κλάσεις. Οι έξοδοι του επιπέδου αθροίζουν στην μονάδα και κάθε κλάση αντιστοιχεί σε τιμές ενός διανύσματος τιμών [0,1]. Χαρακτηριστικό είναι το παράδειγμα του σχήματος 3.4.2.2.6. Εκτός από την ταξινόμηση το επίπεδο αυτό προσφέρει έναν άμεσο τρόπο εκμάθησης μη γραμμικών συνδυασμών των χαρακτηριστικών.

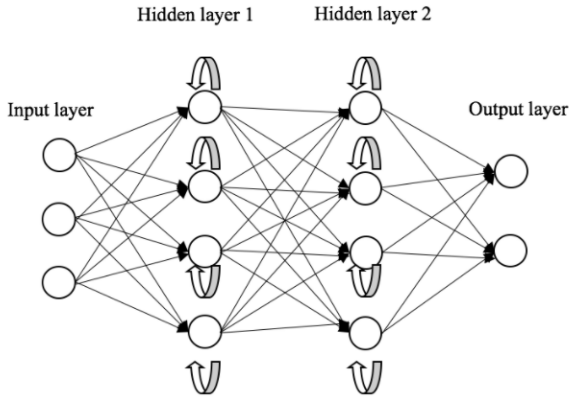


Εικόνα 3.4.2.2.6: Πλήρως Συνδεδεμένο Επίπεδο [67]

3.4.2.3 Επαναλαμβανόμενα Νευρωνικά Δίκτυα (Recurrent Neural Networks)

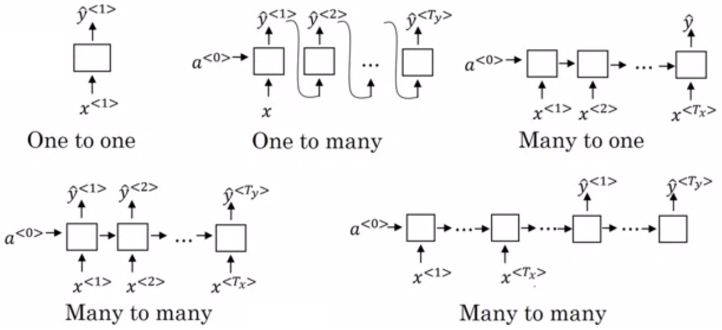
Τα Επαναλαμβανόμενα Νευρωνικά Δίκτυα περιλαμβάνουν βρόχους ανατροφοδότησης, ώστε το σήμα εξόδου να μπορεί να μεταφέρεται ως είσοδος του επιπέδου. Σε ένα δίκτυο RNN οι βρόγχοι ανατροφοδότησης μπορεί να είναι μεταξύ των κόμβων στα κρυφά στρώματα ή από την έξοδο προς τα κρυφά στρώματα [70]. Μπορούμε να τα σκεφτούμε σαν δίκτυα με μνήμη διότι

πέρα από τα εκάστοτε δεδομένα εισόδου συσσωρεύουν την πληροφορία του παρελθόντος σε εσωτερική κατάσταση για υπολογισμό της νέας. Τα RNN χρησιμεύουν σε επεξεργασία σειριακών δεδομένων, είτε χρονικών είτε τοπικών και εμφανίζονται κυρίως σε εφαρμογές αναγνώρισης λόγου και κειμένου, επεξεργασίας βίντεο, δημιουργίας εικόνων και πρόβλεψης χρονικων σειρών. Γενικά τα επαναλαμβανόμενα δίκτυα είναι πολύ ισχυρά και ιδιαίτερα πολύπλοκα. Παρ' όλα αυτά υπάρχουν δυσκολίες με τη λειτουργία τους, αφού χρειάζεται χρόνος να γίνει η εκπαίδευση και η μνήμη τους είναι περιορισμένη. Χαρακτηριστικό γνώρισμά τους είναι ότι θεωρούνται δυναμικά και η κατάσταση τους αλλάζει συνεχώς μέχρι να φτάσουν σε κατάσταση ισορροπίας. Κάθε φορά που αλλάζει η είσοδος αλλάζει και η κατάσταση μέχρι να βρεθεί η καινούργια κατάσταση ισορροπίας.



Εικόνα 3.4.2.3.1: Γενική μορφή RNN [71]

Υπάρχουν 4 κατηγορίες RNN αναλόγως με την είσοδο και την έξοδο τους [71, 72]:



Εικόνα 3.4.2.3.2: Κατηγορίες RNN [72]

<u>One to one</u>: Το πιο απλό είδος δικτύου από μοναδική είσοδο οδηγούμαστε σε μοναδική έξοδο ($Tx = Ty = 1$).

<u>One to Many</u>: Τα δίκτυα αυτά λαμβάνουν μια είσοδο και δημιουργούν μια ακολουθία εξόδων ($Tx = 1, Ty > 1$). Μια είσοδος εννοείται η είσοδος από μια χρονική στιγμή. Βασικές εφαρμογές αυτού του είδους είναι η δημιουργία μουσικής που από μια απλή νότα δημιουργείται

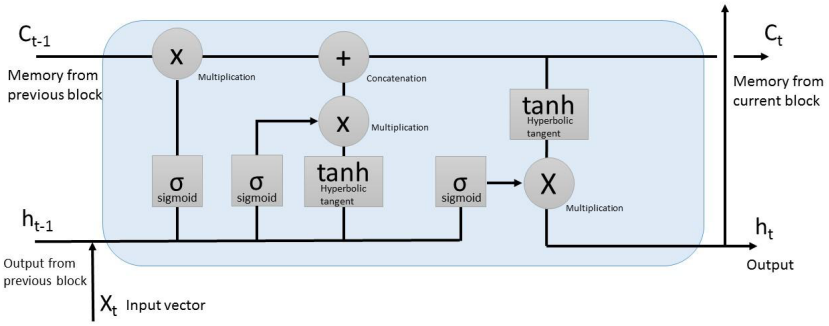
ένα ολόκληρο μουσικό κομμάτι και οι λεζάντες στις εικόνες όπου με είσοδο μια εικόνα έχουμε έξοδο περιγραφή της εικόνας (ακολουθία λέξεων).

Many to One: Η αρχιτεκτονική αυτή σε ένα δίκτυο RNN λαμβάνει ως είσοδο δεδομένα από μια ακολουθία χρονικών στιγμων αλλά παράγει μοναδική έξοδο ($Tx > 1, Ty = 1$). Τα δίκτυα αυτά χρησιμοποιούνται συχνά για ταξινόμηση διαδοχικών δεδομένων. Είναι αποτελεσματικά στην ανάλυση κριτικών πελατών με σκοπό την κατηγοριοποίηση τους σε καλές και κακές. Με την ίδια λογική είναι χρήσιμα στην επεξεργασία κριτικής μιας ταινίας και στην βαθμολόγηση της από το 1 μέχρι το 5. Τέλος, συχνά χρησιμοποιείται και στην αναγνώριση του είδους του τραγουδιού αναλύοντας όλη την ροή του ήχου.

Many to Many: Το είδος αυτό έχει ως είσοδο και ως έξοδο ακολουθίες δεδομένων. Υπάρχουν δύο υποκατηγορίες σε αυτό το είδος τα σύγχρονα και τα ασύγχρονα δίκτυα. Στις σύγχρονες αρχιτεκτονικές κάθε είσοδος έχει μια έξοδο ($Tx = Ty$). Κάθε έξοδος εξαρτάται μόνο από την είσοδο και όλες τις προηγούμενες εξόδους. Το είδος αυτό είναι πολύ χρήσιμο στην πρόβλεψη χρονικών σειρών όταν θέλουμε η πρόβλεψη σε κάθε χρονική στιγμή να είναι βασισμένη στα τρέχοντα και στα προηγούμενα δεδομένα. Μερικά παραδείγματα προβλέψεων είναι οι καθημερινές πωλήσεις σε ένα κατάστημα και η κατανάλωση ηλεκτρικού ρεύματος κάθε ώρα σε ένα εργοστάσιο. Τα ασύγχρονα δίκτυα παράγουν μια ακολουθία εξόδου αφού έχει επεξεργαστεί ολόκληρη η ακολουθία εισόδου. Το μήκος των ακολουθιών δεν χρειάζεται να είναι ίδιο. Το μοντέλο αυτό είναι χρήσιμο σε μεταφράσεις διότι πρώτα πρέπει να διαβαστεί μια ολόκληρη πρόταση και μετά να γίνει η μετάφραση. Επίσης συχνά χρησιμοποιείται και για μακροχρόνια πρόβλεψη όπως η πρόβλεψη πωλήσεων για τις επόμενες μέρες βασιζόμενοι στις πωλήσεις του προηγούμενου μήνα.

3.4.3 Δίκτυα Μακράς Βραχυπρόθεσμης Μνήμης (LSTM)

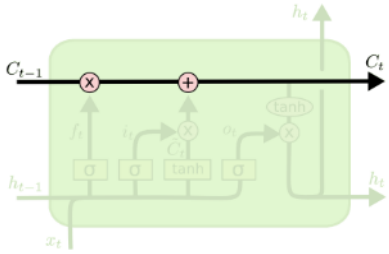
Τα Δίκτυα Μακράς Βραχυπρόθεσμης Μνήμης (Long-Short Term Memory) αποτελούν μια υποκατηγορία των Επαναλαμβανόμενων Νευρωνικών Δικτύων (RNN). Τα δίκτυα αυτά είναι ικανά να εκπαιδεύονται στην μάθηση μεγάλων ακολουθιών δεδομένων. Σε αντίθεση με τα περισσότερα RNN που έχουν προσωρινή μνήμη, τα δίκτυα LSTM έχουν την ικανότητα να απομνημονεύουν πληροφορία που βρίσκεται σε πολύ μακρινές χρονικές στιγμές. Αυτό επιτυγχάνεται με την προσθήκη πυλών στο δίκτυο LSTM [73]. Μια δομική μονάδα ενός δικτύου LSTM απαρτίζεται από το κελί μνήμης (cell), την πύλη εισόδου (input gate), την πύλη εξόδου (output gate) και την πύλη λήθης (forget gate).



Εικόνα 3.4.3.1: Δομική μονάδα LSTM [74]

Παρακάτω θα αναλύσουμε την δομική μονάδα LSTM (Εικόνα 3.4.3.1) και θα παρουσιάσουμε τις σχέσεις που έχουν οι πύλες με την κατάσταση του κελιού [74, 75].

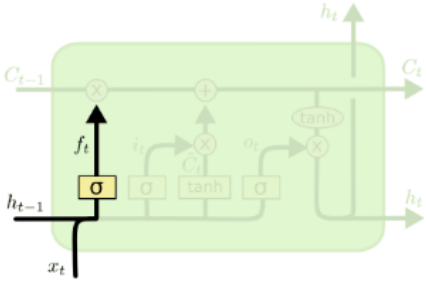
Cell State



Εικόνα 3.4.3.2: Κατάσταση κελιού μνήμης (cell state) [75]

Το κλειδί για τη λειτουργία του LSTM είναι η κατάσταση του κελιού μνήμης. Η κατάσταση του κελιού C_t είναι σαν μια ζώνη που διατρέχει την αλυσίδα. Οι πύλες είναι ειδικοί ρυθμιστικοί μηχανισμοί που επιτρέπουν την τροποποίηση των πληροφοριών της κατάστασης του κελιού.

Forget Gate

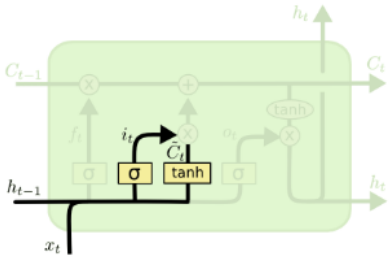


Εικόνα 3.4.3.3: Πύλη λήθης (forget gate) [75]

$$f_t = \sigma(W_{fx}x_t + W_{fh}h_{t-1} + W_{fc}C_{t-1} + b_f)$$

Η πρώτη πύλη είναι η πύλη λήθης (forget gate). Περιέχει ένα sigmoid activation που αποφασίζει αν η τιμή εξόδου της προηγούμενης κρυφής κατάστασης θα κρατηθεί ή όχι. Δημιουργεί ένα συνδυασμό της προηγούμενης κρυφής κατάστασης και του διανύσματος εισόδου και στη συνέχεια παράγει μια έξοδο για κάθε τιμή στην κατάσταση του κελιού. Η έξοδος αποτελείται από έναν αριθμό μεταξύ των 0 ή 1. Αν η έξοδος είναι 0 δίνει σήμα να μην κρατήσει την πληροφορία το κελί ενώ με έξοδο 1 συγκρατείται η πληροφορία. Αξίζει να αναφερθεί ότι τα αρχικά LSTM δίκτυα δεν είχαν forget gate [73], η πύλη προστέθηκε αργότερα για τους λόγους που αναφέρθηκαν παραπάνω.

Input Gate

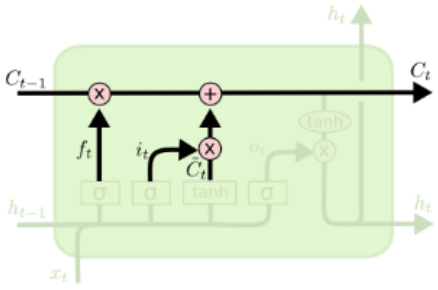


Εικόνα 3.4.3.4: Πύλη εισόδου (input gate) και tanh για παραγωγή υποψήφιων τιμών κελιού μνήμης [75]

$$i_t = \sigma(W_{ix} x_t + W_{ih} h_{t-1} + W_{ic} c_{t-1} + b_i)$$
$$C'_t = tanh(W_{cx} x_t + W_{ch} h_{t-1} + b_c)$$

Το επόμενο βήμα είναι να γίνει η απόφαση σχετικά με το ποια νέα πληροφορία πρόκειται να καταγραφεί στο κελί μνήμης. Αυτό υλοποιείται μέσω της πύλης εισόδου με sigmoid activation και μέσω ενός επιπέδου tanh που παράγει τιμές που θα μπορούσαν ενδεχομένως να προστεθούν στην κατάσταση κελιών. Ο συνδυασμός τους πρόκειται να ενημερώσει την κατάσταση του κελιού μνήμης. Παράλληλα μέσω ενός επιπέδου tanh παράγονται υποψήφιες τιμές C'_t που μπορεί να προστεθούν στην κατάσταση κελιού.

Ενημέρωση νέας κατάστασης

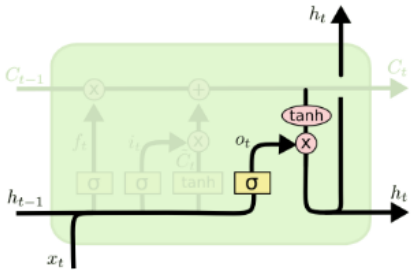


Εικόνα 3.4.3.5: Ενημέρωση νέας κατάστασης κελιού [75]

$$C_t = f_t \odot C_{t-1} + i_t \odot C'_t$$

Στο σημείο αυτό γίνεται ενημέρωση της προηγούμενης κατάστασης κελιού C_{t-1} στην νέα κατάσταση C_t. Πολλαπλασιάζεται η προηγούμενη κατάσταση C_{t-1} με την τιμή που προέκυψε από την πύλη επιλεκτικής συγκράτησης (forget gate) f_t, ξεχνώντας με αυτόν τον τρόπο τις πληροφορίες που θέλαμε να ξεχάσουμε. Έπειτα προσθέτουμε το $i_t \odot C'_t$ που είναι οι νέες υποψήφιες τιμές που εξαρτώνται από το πόσο θέλαμε να ανανεώσουμε κάθε τιμή της κατάστασης. $\odot$ είναι το γινόμενο των διανυσμάτων στοιχείο προς στοιχείο.

Output Gate



Εικόνα 3.4.3.6: Πύλη εξόδου (output gate) [75]

$$o_t = \sigma(W_{ox} x_t + W_{oh} h_{t-1} + W_{oc} C_t + b_o)$$
$$h_t = o_t \odot tanh(C_t)$$

Το Output Gate αποφασίζει ποια θα είναι η επόμενη κρυφή κατάσταση. Το αποτέλεσμα της εξόδου είναι ουσιαστικά μια φιλτραρισμένη έκδοση της κατάστασης του κελιού. Η κρυφή κατάσταση βασίζεται στην κατάσταση του κελιού όπως προκύπτει από sigmoid activation και μέσω ενός επιπέδου tanh ώστε να είναι οι τιμές στο εύρος [-1,1]. Η νέα κατάσταση κελιού καθώς και η νέα κρυφή κατάσταση μεταφέρονται στη συνέχεια στον επόμενο κόμβο LSTM που αντιστοιχεί στο επόμενο χρονικό βήμα.

Τα δίκτυα LSTM μπορούν να χρησιμοποιηθούν για διάφορες εφαρμογές. Ένας μεγάλος κλάδος που χρησιμοποιούνται κυρίως δίκτυα LSTM είναι η επεξεργασία ανάλυση και μετάφραση κειμένων [76, 77]. Η αναγνώριση γραφικού χαρακτήρα (handwriting gesture recognition) και η αναγνώριση φωνής και βιντεο (speech/video recognition) [78]. Επίσης, τα μοντέλα LSTM είναι αποτελεσματικά στην επεξεργασία και εξαγωγή προβλέψεων βάσει χρονικων ακολουθιων δεδομένων. Έχουν χρησιμοποιηθεί κυρίως για πρόβλεψη δικτυακής κίνησης, όπως είδαμε στο κεφάλαιο 2.4.2, αλλά και για πρόβλεψη κίνησης οχημάτων στους δρόμους [79]. Τέλος, συχνά χρησιμοποιείται σε συνδυασμό με συνελικτικά επίπεδα για επεξεργασία εικόνων και πρόβλεψη βίντεο, ώστε να επωφεληθούμε από τα πλεονεκτήματα των μοντέλων LSTM και των δικτύων CNN [40, 41, 43, 80].

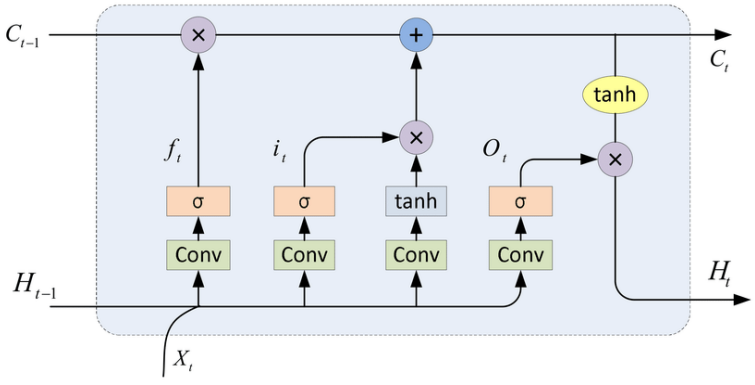
4. Μεθοδολογία

Στην εργασία αυτή θα ασχοληθούμε με την μελέτη και υλοποίηση μεθόδων Βαθιάς Μηχανικής Μάθησης για την Πρόβλεψη Δικτυακής Κίνησης. Το πρόβλημα της πρόβλεψης δικτυακής κίνησης αναπτύχθηκε στο κεφάλαιο 2.4, όπου αναφέρθηκαν αρκετές σχετικές εργασίες που προσπαθούν με διάφορες τεχνικές να πετύχουν την καλύτερη δυνατή πρόβλεψη. Βασιζόμενοι στο θεωρητικό υπόβαθρο που παρουσιάσαμε στο κεφάλαιο 3 αλλά και στις σχετικές εργασίες του κεφαλαίου 2.4, θα αναπτύξουμε μοντέλα τα οποία με τα κατάλληλα δεδομένα θα εκπαιδευτούν ώστε να προβλέπουν την μελλοντική κίνηση ενός δικτύου. Προκειμένου να πετύχουμε τα βέλτιστα αποτελέσματα, δοκιμάστηκαν και αξιολογήθηκαν 6 μοντέλα Βαθιάς Μάθησης τα οποία θα παρουσιαστούν αναλυτικά στις παρακάτω ενότητες.

4.1 Προτεινόμενα Μοντέλα

4.1.1 Conv-LSTM

Το πρώτο μοντέλο που θα αναλύσουμε στηρίζεται στα κελιά Conv-LSTM που είναι επέκταση των κλασσικών LSTM, όπως τα είδαμε αναλυτικά στο κεφάλαιο 3.4.3. Στην κλασική δομή του LSTM έχουν προστεθεί ενσωματωμένες συνελικτικές δομές στη μετάβαση από κατάσταση σε κατάσταση ή από εισόδους σε καταστάσεις, Ουσιαστικά, αντικαθίσταται η πράξη του πολλαπλασιασμού πινάκων με την πράξη της συνέλιξης σε κάθε πύλη του κελιού LSTM [81]. Παρακάτω παρουσιάζεται η δομή του κελιού Conv-LSTM και οι σχέσεις που υπάρχουν μεταξύ των πυλών και της κατάστασης του κελιού.

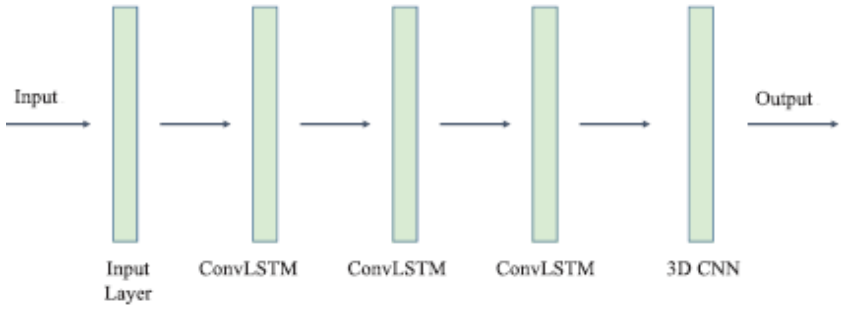


Εικόνα 4.1.1.1: Δομική Μονάδα Conv-LSTM [82]

$$f_t = \sigma(W_{fx} * x_t + W_{fh} * h_{t-1} + W_{fc} C_{t-1} + b_f)$$
$$i_t = \sigma(W_{ix} * x_t + W_{ih} * h_{t-1} + W_{ic} c_{t-1} + b_i)$$
$$C'_t = tanh(W_{cx} * x_t + W_{ch} * h_{t-1} + b_c)$$

$$C_t = f_t \odot C_{t-1} + i_t \odot C'_t$$
$$o_t = \sigma(W_{ox} * x_t + W_{oh} * h_{t-1} + W_{oc} C_t + b_o)$$
$$h_t = o_t \odot tanh(C_t)$$

Τα μοντέλα LSTM έχουν την ικανότητα να απομνημονεύουν πληροφορία που βρίσκεται σε πολύ μακρινές χρονικές στιγμές και να ανιχνεύουν την εξάρτηση των δεδομένων μεγάλων ακολουθιών. Ο λόγος που χρησιμοποιούνται επίπεδα ConvLSTM είναι επειδή κληρονομούν τα πλεονεκτήματα του LSTM, καθώς επίσης και τα πλεονεκτήματα του συνελικτικού επιπέδου. Για ευκολότερη κατανόηση μπορούμε να σκεφτούμε ότι τα δεδομένα αποθηκεύονται σε έναν τρισδιάστατο πίνακα με πρώτη διάσταση τη χρονική ακολουθία των δεδομένων και άλλες δύο χωρικές διαστάσεις. Δηλαδή, η είσοδος είναι διανύσματα πάνω σε ένα χωρικό πλέγμα. Το μοντέλο ConvLSTM προβλέπει την μελλοντική κατάσταση ενός κελιού του πλέγματος λαμβάνοντας υπόψη τις παλιές του καταστάσεις του ίδιου αλλά και των γειτονικών κελιών [80]. Αυτό επιτυγχάνεται με την χρήση της συνέλιξης. Με αυτόν τον τρόπο μειώνονται τα χωρικά δεδομένα και το μοντέλο εστιάζει σε συγκεκριμένα σημεία των ακολουθιών όπου συναντώνται οι χωρικές συσχετίσεις. Έτσι, το μοντέλο γίνεται κατάλληλο για προβλήματα πρόβλεψης χωροχρονικών ακολουθιών [83]. Στην Εικόνα 4.1.2 και στον πίνακα 4.1.1 παρουσιάζεται το πρώτο μοντέλο που αναπτύξαμε και χρησιμοποιήσαμε στο πείραμα μας για το πρόβλημα της πρόβλεψης δικτυακής κίνησης.



Εικόνα 4.1.1.2: Μοντέλο Conv-LSTM με 3 επίπεδα ConvLSTM και 1 CNN

Πίνακας 4.1.1: Δομή μοντέλου Conv-LSTM που αναπτύξαμε για το πείραμα μας

Layer Type	Filter	Kernel	Activation	Padding	Return Sequences	Output Shape
Input (x)						(?, Window, Nodes, Nodes, 1)
ConvLSTM2D	16	(5, 5)	tanh	same	true	(?, 10, Nodes, Nodes, 16)
ConvLSTM2D	16	(3, 3)	tanh	same	true	(?, 10, Nodes,

						Nodes, 16)
ConvLSTM2D	16	(1, 1)	tanh	same	true	(?, 10, Nodes, Nodes, 16)
Conv3D	1	(10,1,1)	tanh	valid	–	(?, 1,Nodes, Nodes, 1)
Reshape						*(?, Nodes*Nodes, 1)*

Οι παράμετροι των επιπέδων του συγκεκριμένου μοντέλου έχουν παρουσιαστεί και αναλυθεί στο κεφάλαιο 3.4.2.2 που περιγράψαμε τα συνελικτικά δίκτυα CNN. Περιληπτικά θα αναφέρουμε την χρησιμότητα τους [67].

- **Filter:** Είναι ο αριθμός των φίλτρων που εμφανίζονται στην έξοδο της συνέλιξης.
- **Kernel:** Χαρακτηρίζει το μέγεθος του φίλτρου συνέλιξης. Είναι διάνυσμα με 2 ακέραιους που εκφράζουν το ύψος και το πλάτος του παράθυρου συνέλιξης. Αν χρησιμοποιούσαμε ConvLSTM1D ή ConvLSTM3D θα είχε 1 ή 3 ακεραίους αντίστοιχα. Στο μοντέλο αυτό χρησιμοποιείται δισδιάστατο παράθυρο γιατί ο πίνακας δικτυακής κίνησης έχει 2 χωρικές διαστάσεις.
- **Activation:** Η συνάρτηση ενεργοποίησης που χρησιμοποιούμε στα επίπεδα.
- **Padding:** Η παράμετρος αυτή δέχεται δύο μεταβλητές “valid” ή “same”. Η μεταβλητή “valid” σημαίνει ότι μηδενικό padding. Αντίθετα, η μεταβλητή “same” σημαίνει ότι προσθέτουμε στην είσοδο κάποιες σειρές και στήλες, ώστε η έξοδος να έχει ίδιες διαστάσεις με την είσοδο.
- **Return Sequences:** Αν η παράμετρος αυτή έχει τιμή “True” σημαίνει ότι επιστρέφει στην έξοδο όλη την ακολουθία των δεδομένων. Αν η τιμή είναι “False” επιστρέφει μόνο την τελευταία έξοδο χωρίς κάποια χρονική ακολουθία και η διαστάσεις της εξόδου μειώνεται κατά μία.

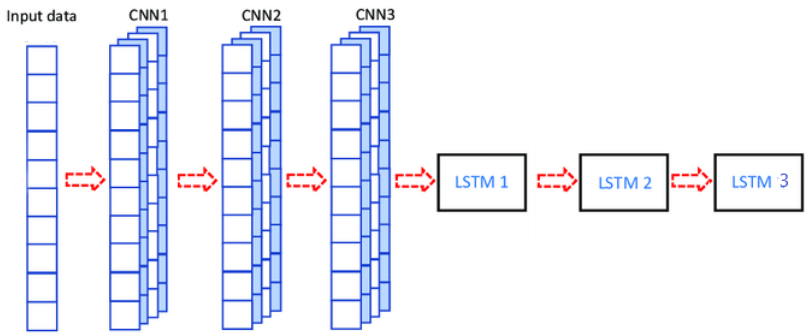
Αρχικά, στο μοντέλο μας χρησιμοποιούμε τρία επίπεδα ConvLSTM με 16 φίλτρα. Η είσοδος κάθε επιπέδου είναι της μορφής (samples, timesteps, rows, cols, channels). Έχει αποδειχθεί ότι τα πιο βαθιά δίκτυα με περισσότερα επίπεδα παρουσιάζουν καλύτερα αποτελέσματα από μοντέλα με 1 ή 2 επίπεδα [80]. Επίσης, εξετάστηκαν μοντέλα με περισσότερα επίπεδα ConvLSTM και παρατηρήθηκε ότι είχαν σχεδόν ίδια επίδοση με μεγαλύτερη πολυπλοκότητα και σημαντικά περισσότερο χρόνο εκπαίδευσης. Τα φίλτρα Kernel έχουν μεγέθη 5 - 3 - 1 διαδοχικά, διότι συνηθίζεται κάθε φίλτρο να έχει μικρότερο μέγεθος από το προηγούμενο του. Επίσης τα μεγάλα φίλτρα δεν είναι απαραίτητο ότι έχουν καλύτερη επίδοση [85]. Ως συνάρτηση ενεργοποίησης χρησιμοποιήθηκε η Tanh. Η Tanh αποφέρει μεγαλύτερες τιμές κλίσης (gradient) και σημαντικότερες ανανεώσεις στα βάρη του δικτύου κατά την διάρκεια της εκπαίδευσης. Συχνά χρησιμοποιείται το ReLU που αποτρέπει την εμφάνιση του προβλήματος της εξαφανιζόμενης κλίσης (vanishing gradient) και η συνάρτηση ενεργοποίησης sigmoid που

ενδείκνυται για προβλήματα δυαδικής ταξινόμησης. Η Tanh παρουσιάζει καλύτερα αποτελέσματα σε νευρωνικά δίκτυα με πολλαπλά επίπεδα. Παράλληλα, χρησιμοποιείται η παράμετρος padding ώστε να λαμβάνουμε υπόψη και τις πληροφορίες που υπάρχουν στις γωνίες του χωρικού πλέγματος των δεδομένων. Τέλος, η παράμετρος return sequences έχει τιμή true, διότι τα επίπεδα ConvLSTM είναι κρυφά επίπεδα και θέλουμε να διατηρείται η χρονική ακολουθία. Η έξοδος κάθε φίλτρου έχει μορφή (samples, timesteps, new_rows, new_cols, filters).

Το μοντέλο ολοκληρώνεται με ένα συνελικτικό επίπεδο τρισδιάστατης συνέλιξης. Το επίπεδο αυτό έχει ένα φίλτρο τριών διαστάσεων με μεγέθη 10-1-1. Το βάθος έχει μέγεθος 10 και εφαρμόζεται στο κυλιόμενο παράθυρο (sliding window) του μοντέλου που εξετάζει χρονική ακολουθία 10 στιγμών (το Window για όλα τα μοντέλα θα έχει μέγεθος 10). Έτσι, η δεύτερη διάσταση έχει έξοδο μεγέθους 1. Επίσης, δεν υπάρχει padding.

4.1.2 CNN-LSTM

Το δεύτερο μοντέλο που εξετάσαμε είναι το CNN-LSTM το οποίο αποτελεί συνδυασμό συνελικτικού και αναδρομικού δικτύου. Η επέκταση του σε σχέση με τα μοντέλα CNN είναι πως εκτός της επεξεργασίας της τοπικής χωροχρονικής πληροφορίας μέσω του συνελικτικού δικτύου, περιλαμβάνει και την ενσωμάτωσή της σε ένα αναδρομικό μοντέλο που λαμβάνει υπόψη την συνολική χρονική εξέλιξη των δεδομένων [86]. Τα CNN LSTMs αναπτύχθηκαν για προβλήματα πρόβλεψης οπτικών χρονοσειρών και τη δημιουργία περιγραφικών κειμένων από ακολουθίες εικόνων (π.χ. βίντεο), καθώς επίσης και στην επεξεργασία φωνής και φυσικής γλώσσας. Τα μοντέλα αυτά περιλαμβάνουν συνελικτικά επίπεδα των οποίων η έξοδος τροφοδοτείται σε δίκτυα LSTM. Ουσιαστικά, χρησιμοποιούνται αρχικά τα συνελικτικά επίπεδα για την εξαγωγή των σημαντικότερων χαρακτηριστικών από την χρονική ακολουθία των δεδομένων, τα οποία έπειτα εισάγονται στο δίκτυο LSTM προς την περαιτέρω εκμάθηση των μακροπρόθεσμων και βραχυπρόθεσμων μοτίβων σε αυτά [87]. Το πλεονέκτημα αυτού του μοντέλου είναι ότι μπορεί να υποστηρίξει πολύ μεγάλες ακολουθίες εισόδου, που μπορούν να διαβαστούν ως υπο-ακολουθίες από το μοντέλο CNN και στη συνέχεια να συγκεντρωθούν από το μοντέλο LSTM. Παρακάτω υπάρχει σχηματική απεικόνιση της αρχιτεκτονικής του μοντέλου που αναπτύξαμε.



Εικόνα 4.1.2: Μοντέλο CNN-LSTM με 3 επίπεδα CNN και 3 LSTM

Πίνακας 4.1.2: Δομή μοντέλου CNN-LSTM που αναπτύξαμε για το πείραμα μας

Layer Type	**Filter / LSTM Blocks**	**Kernel**	**Activation**	**Padding**	**Return Sequences**	**Output Shape**
Input(x)						(?, Nodes, Nodes, Window)
Conv2D	64	(5, 5)	tanh	same	–	(?, Nodes, Nodes, 64)
Conv2D	64	(3, 3)	tanh	same	–	(?, Nodes, Nodes, 64)
Conv2D	64	(1, 1)	tanh	valid	–	(?, Nodes, Nodes, 64)
Reshape						*(?, Nodes*Nodes, 64)*
LSTM	64	–	tanh	–	true	(?, Nodes*Nodes, 64)
LSTM	32	–	tanh	–	true	(?, Nodes*Nodes, 32)
LSTM	1	–	tanh	–	true	(?, Nodes*Nodes, 1)

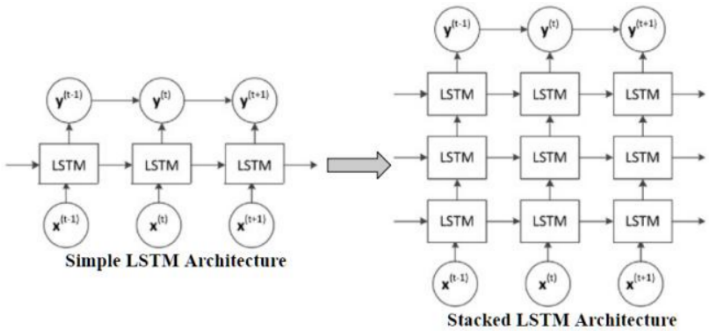
Το μοντέλο μας αρχικά αποτελείται από 3 συνελικτικά επίπεδα δισδιάστατης συνέλιξης. Η είσοδος κάθε συνελικτικού επιπέδου είναι της μορφής (batch size, rows, cols, channels). Τα επίπεδα CNN έχουν 64 φίλτρα. Τα φίλτρα Kernel έχουν μεγέθη 5 - 3 - 1 όπως και στο μοντέλο ConvLSTM και προτιμήθηκαν αυτά τα μεγέθη για τους ίδιους λόγους με πριν. Εξετάστηκαν και άλλοι συνδυασμοί για το μοντέλο με διαφορετικό αριθμό φίλτρων και διαφορετικό αριθμο συνελικτικών επιπέδων. Ο συγκεκριμένος συνδυασμός υπερτερούσε στην αποτελεσματικότητα σε συνδυασμό με το υπολογιστικό κόστος. Η μη γραμμική συνάρτηση Tanh εφαρμόζεται ως συνάρτηση ενεργοποίησης, διότι παρατηρήθηκαν καλύτερα αποτελέσματα από την ReLU που συνήθως χρησιμοποιείται στα συνελικτικά επίπεδα. Επίσης, χρησιμοποιούμε padding ώστε να είναι πιο ακριβής η χωρική ανάλυση των δεδομένων. Με αυτόν τον τρόπο ανιχνεύονται

περισσότερες χωρικές συσχετίσεις στα δεδομένα. Το τελευταίο συνελικτικό επίπεδο επειδή έχει φίλτρα μεγέθους 1 δεν χρειάζεται padding.

Τα πιο διακριτά χαρακτηριστικά που ανιχνεύθηκαν από τα συνελικτικά επίπεδα θα δοθούν ως είσοδος στα επίπεδα LSTM με σκοπό να εξεταστεί η χρονική ακολουθία αυτής της πληροφορίας. Τα 3 επίπεδα LSTM έχουν 64, 32, 1 δομικές μονάδες αντίστοιχα. Η χρήση 3 επιπέδων LSTM δίνει την δυνατότητα στο μοντέλο να δημιουργεί σε κάθε επίπεδο πιο πολύπλοκα χρονικά μοτίβα. Κάθε επίπεδο έχει ως είσοδο την ακολουθία εξόδου του προηγούμενου επιπεδου LSTM προσφέροντας την δυνατότητα για περισσότερες παρατηρήσεις σε διαφορετικές χρονικές κλίμακες. Επίσης, χρησιμοποιήθηκε συνάρτηση Tanh για τα επίπεδα LSTM, γιατί παρατηρήθηκε ότι βοηθάει στην ταχύτερη εκπαίδευση του μοντέλου. Τέλος, η παράμετρος return sequences έχει τιμή true, διότι θέλουμε να διατηρείται η χρονική ακολουθία.

4.1.3 Stacked LSTM

Το επόμενο μοντέλο που θα αναλύσουμε είναι ένα κλασσικό LSTM μοντέλο 3 επιπέδων. Τα δίκτυα LSTM έχουν την δυνατότητα να συγκρατούν και να ξεχνούν πληροφορίες μεταξύ των χρονικών στιγμών για αυτόν τον λόγο είναι κατάλληλα για πρόβλεψη χρονοσειρών. Η χρήση πολλών επιπέδων LSTM προσφέρει περισσότερες πληροφορίες σε διάφορες χρονικές κλίμακες δημιουργώντας πολύπλοκα χρονικά μοτίβα. Επίσης, είναι πιο αποτελεσματική η χρήση περισσότερων κρυφών στρωμάτων σε σχέση με την αύξηση των μπλοκ σε ένα επίπεδο. Με αυτόν τον τρόπο χαρτογραφούνται τα δεδομένα σε χώρο πολλών διαστάσεων και έτσι οι αναδρομικές μονάδες αποθηκεύουν τις πληροφορίες σχετικά με τις χρονικές εξαρτήσεις των δεδομένων [22]. Έχει αποδειχθεί ότι σε διάφορα προβλήματα η απόδοση των μοντέλων με πολλά επίπεδα LSTM είναι καλύτερη από τα μοντέλα με 1 επίπεδο LSTM [88]. Προσθέτοντας κρυφά επίπεδα LSTM αυξάνεται η ακρίβεια και η αξιοπιστία του μοντέλου. Ταυτόχρονα όσο αυξάνεται το βάθος του δικτύου, αυξάνεται και ο χρόνος εκπαίδευσης του, για αυτό και πρέπει να μειωθεί ο αριθμός των μπλοκ σε κάθε επίπεδο προκειμένου να επιτευχθεί ιδανική ισορροπία [89]. Στην Εικόνα 4.1.3 παρατηρούμε την διαφορά στην αρχιτεκτονική ενός απλού μοντέλου LSTM και του μοντέλου Stacked LSTM με 3 επίπεδα που αναπτύξαμε.



Εικόνα 4.1.3: Αρχιτεκτονική Απλού Μοντέλου LSTM και Πολυεπίπεδου LSTM που δημιουργήσαμε [90]

Πίνακας 4.1.3: Δομή μοντέλου LSTM που αναπτύξαμε για το πείραμα μας

Layer Type	LSTM Blocks	Dropout	Pool Size	Activation	Return Sequences	Output Shape
Input (x)						(?, Window, Nodes*Nodes)
LSTM	512	–	–	tanh	true	(?, 10, 512)
Dropout	–	0.2	–	–	–	(?, 10, 512)
MaxPooling1D	–	–	2	–	–	(?, 5, 512)
LSTM	256	–	–	tanh	true	(?, 5, 256)
Dropout	–	0.2	–	–	–	(?, 5, 256)
MaxPooling1D	–	–	5	–	–	(?, 1, 256)
LSTM	Nodes * Nodes	–	–	tanh	true	(?, 1, Nodes*Nodes)

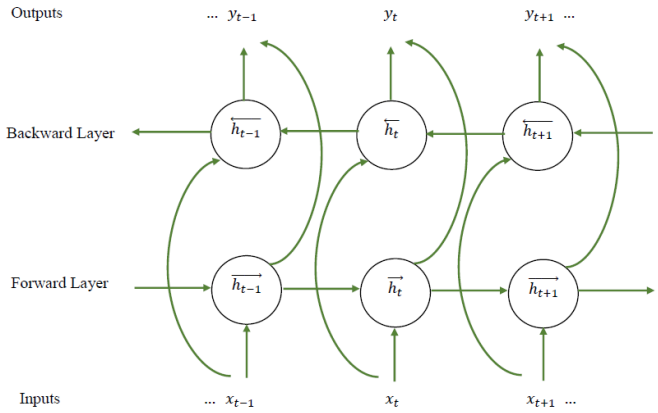
Η αρχιτεκτονική του νευρωνικού δικτύου που δημιουργήσαμε περιλαμβάνει 3 επίπεδα LSTM που παρεμβάλλονται από 2 επίπεδα Dropout και 2 επίπεδα MaxPooling. Αρχικά, τα επίπεδα LSTM έχουν 512, 256 και Nodes*Nodes (Abilene→12*12=144 / Geant→23*23=529) δομικές μονάδες. Το τελευταιο επίπεδο LSTM θέλουμε να έχει μέγεθος ίσο με τον αριθμό των ζευγών προέλευσης προορισμού (Nodes*Nodes), ώστε τα δεδομένα μετά την εκπαίδευση έχουν μέγεθος (?, 1, Nodes*Nodes). Επειδή χρησιμοποιούμε το ίδιο μοντέλο για 2 σύνολα δεδομένων δεν βάζουμε σταθερό αριθμό δομικών μονάδων στο τελευταίο επίπεδο για να λαμβάνουμε το αναμενόμενο μέγεθος εξόδου των δεδομένων και για τα 2 datasets. Στο μοντέλο αυτό χρησιμοποιήθηκαν οι προεπιλεγμένες συναρτήσεις ενεργοποίησης των κελιών LSTM. Όπως αναφέραμε στο κεφάλαιο 3.4.3 το κελί LSTM έχει 3 sigmoid activations στις πύλες και 1 tanh activation. Η παράμετρος Activation αντιστοιχεί στην συνάρτηση που χρησιμοποιείται για την παραγωγή υποψηφιων τιμών για την νέα κατάσταση του κελιού.

Επίσης, σε αυτό το μοντέλο χρησιμοποιούμε την μέθοδο της εξομάλυνσης (Dropout). Το Dropout είναι ένα επίπεδο για το μοντέλο LSTM το οποίο χρησιμοποιείται για απενεργοποίηση νευρώνων [91]. Η τεχνική αυτή ισοδυναμεί με την προσθήκη ενός επιπλέον στρώματος μετά από ένα στρώμα νευρώνων που απλώς ρυθμίζει τιμές στο μηδέν με κάποια πιθανότητα. Συγκεκριμένα, σε κάθε επανάληψη του αλγορίθμου εκπαίδευσης επιλέγονται τυχαία κάποιες συνδέσεις νευρώνων οι οποίες μηδενίζονται. Ο κάθε νευρώνας του δικτύου παραλείπεται με πιθανότητα p. Είναι μια τεχνική για την αποφυγή της υπερπροσαρμογής (overfitting), δηλαδή της τάσης του μοντέλου να προσαρμόζεται υπερβολικά στα δεδομένα εκπαίδευσης με αποτέλεσμα την μικρή ακρίβεια στα δεδομένα δοκιμής.

Τέλος, χρησιμοποιούνται Συγκεντρωτικά Επίπεδα (Pooling Layers) για υποδειγματοληψία των δεδομένων χωρίς να υπάρξει απώλεια σημαντικών χαρακτηριστικών. Στο κεφάλαιο 3.4.2.2 περιγράψαμε πως γίνεται η υποδειγματοληψία στον πίνακα χαρακτηριστικών των CNN δικτύων με την βοήθεια ενός κυλιόμενου παραθύρου. Αντίστοιχα, στην περίπτωση των LSTM μοντέλων τα συγκεντρωτικά επίπεδα χρησιμοποιούνται για την υποδειγματοληψία των ακολουθιών [92]. Το MaxPooling διατηρεί την μέγιστη τιμή κάθε υποακολουθίας. Έτσι, το μοντέλο δεν γίνεται υπερβολικά ευαίσθητο στις τιμές των χαρακτηριστικών και η ακρίβεια τους μειώνεται. Στην εργασία [93] αποδείχθηκε μετά από σύγκριση όλων των ειδών υποδειγματοληψίας ότι καλύτερη τεχνική για κατηγοριοποίηση δεδομένων με τη χρήση LSTM είναι η MaxPooling. Το ίδιο συμπέρασμα βγήκε και στην παρούσα εργασία για πρόβλεψη χρονοσειρών μετά από σύγκριση των αποτελεσμάτων των μοντέλων LSTM.

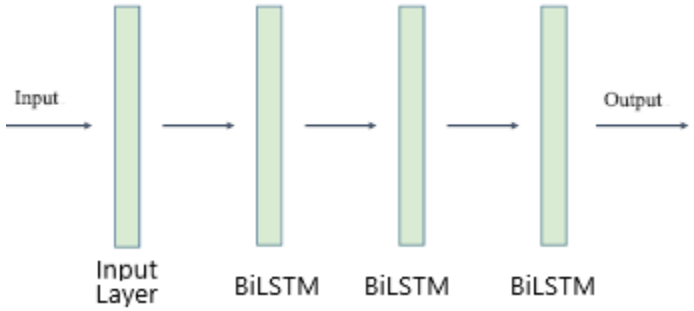
4.1.4 BiLSTM

Τα αμφίδρομα LSTM (Bidirectional LSTM ή BiLSTM) αποτελούν επέκταση των κλασσικών LSTM. Στα δεδομένα εισόδου τοποθετούνται 2 επίπεδα LSTM, όπου το καθένα επεξεργάζεται την ακολουθία με διαφορετική σειρά. Δηλαδή η ακολουθία εισόδου τροφοδοτείται σε ένα δίκτυο LSTM με την κανονική της κατεύθυνση και σε ένα δεύτερο δίκτυο με την αντίθετη κατεύθυνση. Σε κάθε χρονικό βήμα οι έξοδοι των δύο δικτύων συνενώνονται [94]. Με αυτόν τον τρόπο η πληροφορία και από τις δύο πλευρές της ακολουθίας χρησιμοποιείται για την εκτίμηση της εξόδου. Η τεχνική αυτή βασίζεται στην ιδέα ότι η έξοδος κάθε χρονική στιγμή εξαρτάται τόσο από τα προηγούμενα όσο και από τα επόμενα στοιχεία της ακολουθίας [95]. Έτσι, τα BiLSTM διπλασιάζουν τον όγκο της πληροφορίας που δέχονται. Αποτέλεσμα αυτής της τεχνικής είναι η βελτίωση της χρονικής συσχέτισης των δεδομένων σε μεγάλες ακολουθίες, άρα και της απόδοσης του μοντέλου σε σχέση με τα απλά LSTM [96]. Παρακάτω παρουσιάζεται η δομή ενός μοντέλου BiLSTM.



Εικόνα 4.1.4.1: Αρχιτεκτονική Μοντέλου BiLSTM [97]

Στην παραπάνω εικόνα ένα LSTM επεξεργάζεται την ακολουθία από αριστερά προς τα δεξιά, ενώ το δεύτερο LSTM την επεξεργάζεται από δεξιά προς αριστερά. Σε κάθε χρονική στιγμή t ένα δεξιόστροφο LSTM με κατάσταση h δέχεται ως είσοδο την είσοδο x_t και την προηγούμενη κατάσταση h_{t-1}. Επίσης, ένα αριστερόστροφο LSTM με κατάσταση h δέχεται ως είσοδο την είσοδο την είσοδο x_t και την μελλοντική κατάσταση h_{t+1} [97]. Στην Εικόνα 4.1.4.2 παρουσιάζουμε την αρχιτεκτονική του μοντέλου BiLSTM με 3 επίπεδα που αναπτύξαμε.



Εικόνα 4.1.4.2: Μοντέλο BiLSTM με 3 επίπεδα

Πίνακας 4.1.4: Δομή μοντέλου BiLSTM που αναπτύξαμε για το πείραμα μας

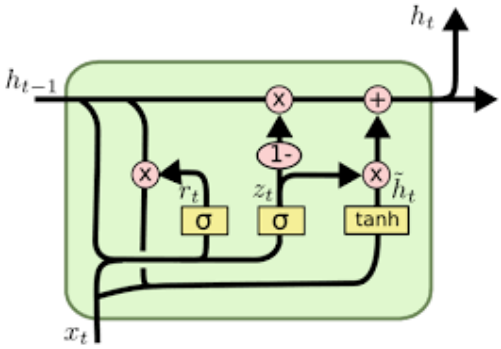
Layer Type	**LSTM Blocks**	**Dropout**	**Pool Size**	**Activation**	**Return Sequences**	**Output Shape**
Input (x)						(?, Window, Nodes*Nodes)
Bidirectional (LSTM)	256	–	–	tanh	true	(?, 10, 512)
Dropout	–	0.2	–	–	–	(?, 10, 512)
MaxPooling1D	–	–	2	–	–	(?, 5, 512)
Bidirectional (LSTM)	128	–	–	tanh	true	(?, 5, 256)
Dropout	–	0.2	–	–	–	(?, 5, 256)
MaxPooling1D	–	–	5	–	–	(?, 1, 256)
Bidirectional (LSTM)	106	–	–	tanh	true	(?, 1, 212)
Dense	Nodes*Nodes					(?, 1, Nodes*Nodes)

Η αρχιτεκτονική του μοντέλου που αναπτύξαμε περιλαμβάνει 3 επίπεδα BiLSTM. Η είσοδος της κλάσης Bidirectional περιλαμβάνει το είδος του επιπέδου και τον τρόπο συγχώνευσης των

ακολουθιών. Τα 3 επίπεδα είναι τύπου LSTM με 256, 128, 72 μπλοκ αντίστοιχα. Η ακολουθία εισόδου κάθε επιπέδου δημιουργεί ένα αντίγραφο για ανάποδη επεξεργασία. Αυτές οι ακολουθίες συγχωνεύονται με τρόπο "concat" και η έξοδος σε κάθε επίπεδο έχει διπλάσιο μέγεθος. Μεταξύ των επιπέδων αυτών παρεμβάλλονται από 2 επίπεδα Dropout και 2 επίπεδα MaxPooling. Οι παράμετροι αυτών των επιπέδων έμειναν ίδιες με το προηγούμενο μοντέλο διότι σε αυτές τις παραμέτρους μετρήθηκαν τα καλύτερα αποτελέσματα. Τέλος, προστίθεται ένα απλό επίπεδο dense πυκνά συνδεδεμένων νευρώνων προκειμένου η έξοδος να έχει μέγεθος ίσο με τον αριθμό των ζευγών προέλευσης προορισμού (Abilene→12*12=144 / Geant→23*23=529), ώστε τα δεδομένα μετά την εκπαίδευση έχουν μέγεθος (?, 1, Nodes*Nodes). Επειδή χρησιμοποιούμε το ίδιο μοντέλο για 2 σύνολα δεδομένων δεν βάζουμε σταθερό αριθμό δομικών μονάδων στο τελευταίο επίπεδο για να λαμβάνουμε το αναμενόμενο μέγεθος εξόδου των δεδομένων και για τα 2 datasets.

4.1.5 GRU

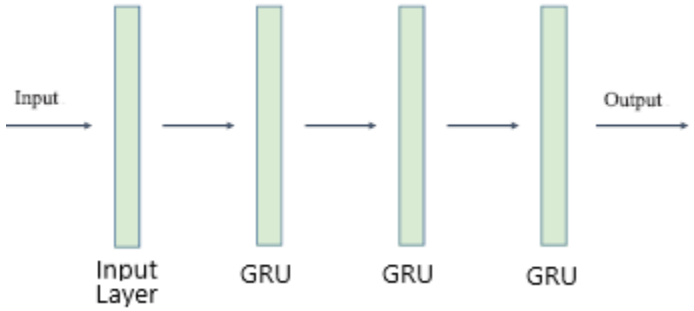
Τα μοντέλα GRU (Gated Recurrent Units) είναι άλλη μια δημοφιλής αρχιτεκτονική αναδρομικών νευρωνικών δικτύων. Είναι μια απλούστερη μορφή των μοντέλων LSTM. Βασίζονται στην ίδια λογική, γιατί έχουν πύλες που διαχειρίζονται τις πληροφορίες, αλλά δεν έχουν ξεχωριστά κελιά για να αποθηκεύουν την κατάσταση του κελιού. Η κυριότερη διαφορά τους είναι ότι τα κελιά LSTM έχουν 2 ξεχωριστές πύλες για την διαχείριση των πληροφοριών της κρυφής κατάστασης (Forget gate - Output gate). Στην δομή του GRU τις δύο πύλες αυτές αντικαθιστά η πύλη ενημέρωσης (Update gate), ενώ υπάρχει άλλη μια νέα πύλη που ονομάζεται πύλη επαναφοράς (Reset gate) [98]. Αρχικά, η Reset Gate (R) αποφασίζει αν η προηγούμενη κατάσταση κελιού είναι σημαντική ή όχι. Ουσιαστικά, αποφασίζει ποιες από τις προηγούμενες πληροφορίες θα ξεχαστούν [48]. Με τη χρήση της συνάρτησης ενεργοποίησης tanh αποθηκεύονται οι χρήσιμες πληροφορίες από το παρελθόν. Με τον τρόπο αυτό επιτρέπουμε στις πληροφορίες να παραμένουν περισσότερο χωρίς να εξαφανίζονται. Το γινόμενο αυτό θα αναφέρεται ως h΄. Παράλληλα, η Update Gate (Z) αποφασίζει αν η κατάσταση του κελιού πρέπει να ενημερωθεί με την υποψήφια κατάσταση ή οχι. Καθορίζει, δηλαδή, την ποσότητα των συγκεντρωμένων πληροφοριών από προηγούμενα χρονικά βήματα, που θα περάσει για μελλοντική χρήση [74]. Ως τελευταίο βήμα, το δίκτυο πρέπει να υπολογίσει το διάνυσμα που διατηρεί πληροφορίες για την τρέχουσα μονάδα και τις μεταβιβάζει στο δίκτυο (h_t). Η τελική κατάσταση του κελιού εξαρτάται από την update gate και το h'. Αφαιρεί κάποιες πληροφορίες από την προηγούμενη κατάσταση και προσθέτει κάποιες νέες. Στη συνέχεια παρουσιάζουμε την αρχιτεκτονική ενός κελιού GRU και τις μαθηματικές σχέσεις που συνδέουν τις καταστάσεις και τις πύλες.



Εικόνα 4.1.5.1: Δομική Μονάδα GRU [99]

$$r_t = \sigma(W_{rh} h_{t-1} + W_{rx} x_t + b_r)$$
$$z_t = \sigma(W_{zh} h_{t-1} + W_{rx} x_t + b_z)$$
$$h'_t = tanh(W_{h'h}(r_t \odot h_{t-1}) + W_{h'x} x_t + b_z)$$
$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot h'_t$$

Τα μοντέλα GRU συνήθως έχουν παρόμοια επίδοση με τα LSTM και οι διαφορές τους εξαρτώνται από το εκάστοτε πρόβλημα. Το GRU παρέχει λιγότερο περίπλοκες δομές και είναι πιο εύκολο στην υλοποίηση. Συνήθως, παρουσιάζει καλύτερα αποτελέσματα σε προβλήματα με λίγα δεδομένα λόγω των λίγων παραμέτρων, ενώ το LSTM είναι προτιμότερο για προβλήματα με μεγάλη ποσότητα δεδομένων. Τα μοντέλα LSTM είναι πιο δημοφιλή, γιατί είναι πιο παλιά και έχουν χρησιμοποιηθεί σε περισσότερα προβλήματα [98]. Για αυτό τον λόγο είναι πιο ασφαλής και αξιόπιστη επιλογή η χρήση των LSTM. Τα μοντέλα GRU όπως και τα LSTM είναι κατάλληλα για προβλήματα πρόβλεψης δικτυακής κίνησης. Μάλιστα σε συγκεκριμένα προβλήματα πρόβλεψης τα GRU μπορεί να εμφανίζουν και καλύτερα αποτελέσματα από τα LSTM [100]. Στην Εικόνα 4.1.5.2 παρουσιάζουμε την αρχιτεκτονική του μοντέλου GRU με 3 επίπεδα που αναπτύξαμε.



Εικόνα 4.1.5.2: Μοντέλο GRU με 3 επίπεδα

Πίνακας 4.1.5: Δομή μοντέλου GRU που αναπτύξαμε για το πείραμα μας

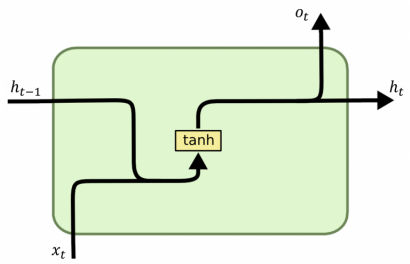
Layer Type	GRU Blocks	Dropout	Pool Size	Activation	Return Sequences	Output Shape
Input (x)						(?, Window, Nodes*Nodes)
GRU	512	–	–	tanh	true	(?, 10, 512)
Dropout	–	0.2	–	–	–	(?, 10, 512)
MaxPooling1D	–	–	2	–	–	(?, 5, 512)
GRU	256	–	–	tanh	true	(?, 5, 256)
Dropout	–	0.2	–	–	–	(?, 5, 256)
MaxPooling1D	–	–	5	–	–	(?, 1, 256)
GRU	Nodes * Nodes	–	–	tanh	true	(?, 1, Nodes*Nodes)

Η αρχιτεκτονική του μοντέλου που αναπτύξαμε βασίζεται κυρίως σε 3 επίπεδα GRU. Τα 3 επίπεδα αποτελούνται από 512, 256 και Nodes*Nodes (Abilene→12*12=144 / Geant→23*23=529) δομικές μονάδες. Το τελευταιο επίπεδο GRU θέλουμε να έχει μέγεθος ίσο με τον αριθμό των ζευγών προέλευσης προορισμού (Nodes*Nodes), ώστε τα δεδομένα μετά την εκπαίδευση έχουν μέγεθος (?, 1, Nodes*Nodes). Επειδή χρησιμοποιούμε το ίδιο μοντέλο για 2 σύνολα δεδομένων δεν βάζουμε σταθερό αριθμό δομικών μονάδων στο τελευταίο επίπεδο για να λαμβάνουμε το αναμενόμενο μέγεθος εξόδου των δεδομένων και για τα 2 datasets. Στο μοντέλο αυτό χρησιμοποιήθηκαν οι προεπιλεγμένες συναρτήσεις ενεργοποίησης των κελιών GRU. Όπως παρατηρούμε στην Εικόνα 4.1.5 το κελί GRU έχει 2 sigmoid activations στις πύλες και 1 tanh activation. Η παράμετρος Activation, όπως και στα LSTM, αντιστοιχεί στην συνάρτηση που χρησιμοποιείται για την παραγωγή υποψηφιων τιμών για την νέα κατάσταση του κελιού. Μεταξύ των επιπέδων GRU παρεμβάλλονται από 2 επίπεδα Dropout και 2 επίπεδα MaxPooling. Οι παράμετροι αυτών των επιπέδων έμειναν ίδιες με τα μοντέλα LSTM και BiLSTM, διότι σε αυτές τις παραμέτρους μετρήθηκαν τα καλύτερα αποτελέσματα.

4.1.6 SimpleRNN

Το τελευταίο μοντέλο που θα αναλύσουμε είναι ένα απλό αναδρομικό μοντέλο RNN που δημιουργήθηκε με την βοήθεια της κλάσης SimpleRNN του keras. Όπως είπαμε και στο κεφάλαιο 3.4.2.3 τα RNN διαθέτουν βρόχους ανατροφοδότησης προκειμένου να επεξεργαστούν ακολουθίες δεδομένων. Η μνήμη που διαθέτει ένα απλό μοντέλο RNN είναι μικρής διάρκειας

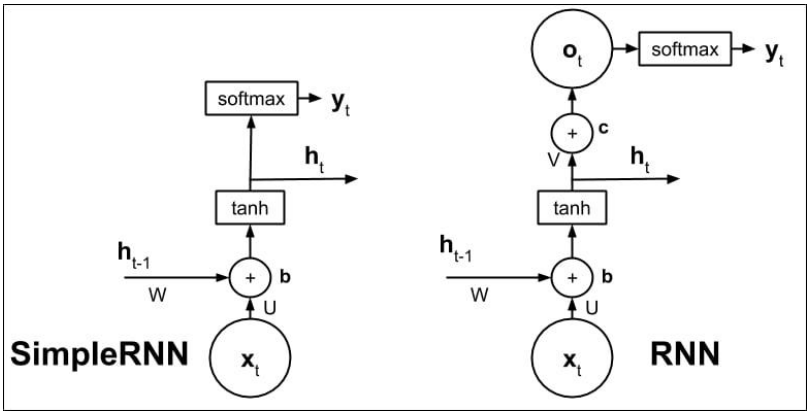
και οι πληροφορίες αποθηκεύονται στην μεταβλητή h, η οποία ανανεώνεται σε κάθε χρονικό βήμα από τα νέα δεδομένα εισόδου [101]. Λόγω αυτού του προβλήματος της μικρής μνήμης δημιουργήθηκαν πιο πολύπλοκα μοντέλα RNN [102], όπως αυτά που περιγράψαμε προηγουμένως. Στη συνέχεια παρουσιάζουμε την αρχιτεκτονική του SimpleRNN και τις μαθηματικές σχέσεις που συνδέουν την κρυφή κατάσταση του κελιού με την είσοδο και την προηγούμενη κατάσταση.



Εικόνα 4.1.6.1: Αρχιτεκτονική Αναδρομικού Μοντέλου SimpleRNN

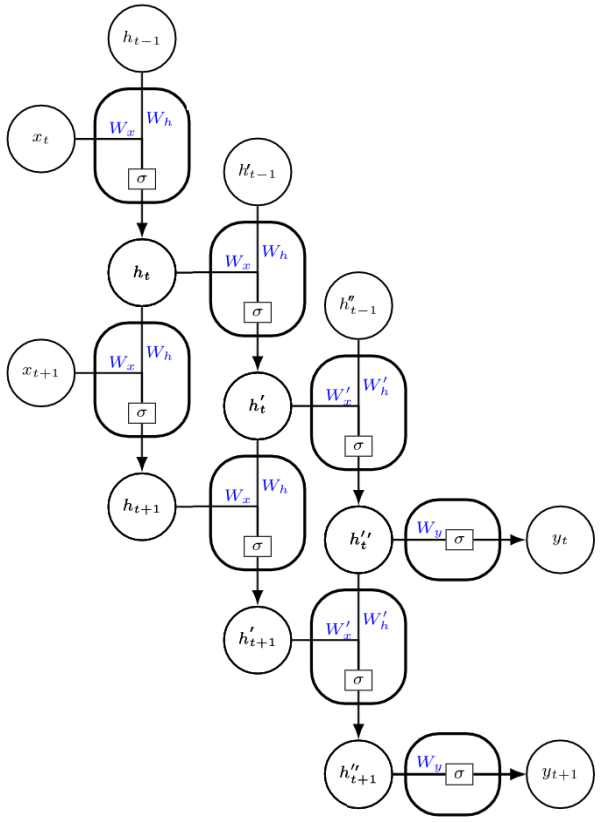
$$h_t = tanh(W_x x_t + W_h h_{t-1} + b_h)$$

Στα κελιά SimpleRNN χρησιμοποιείται συνάρτηση ενεργοποίησης tanh, διότι η ReLU σε μεγάλες ακολουθίες αντιμετωπίζει το πρόβλημα των εξαφανιζόμενων κλίσεων [103]. Η διαφορά του SimpleRNN που το καθιστά πιο απλό σε σχέση με κάποιο άλλο RNN είναι η αφαίρεση του υπολογισμού των τιμών εξόδου O_t. Όπως βλέπουμε παρακάτω στο SimpleRNN, παραλείπεται η σχέση $O_t = Vh_t + c$ πριν από το softmax [104].



Εικόνα 4.1.6.2: Διαφορά μοντέλου SimpleRNN από RNN [104]

Προκειμένου να γίνει κατανοητή η σύνδεση των επιπέδων μεταξύ τους, παρακάτω θα παρουσιάσουμε το μοντέλο που αναπτύξαμε με 3 επίπεδα SimpleRNN. Τα διανύσματα h, h΄, h΄΄ εχουν μέγεθος ίσο με τον αριθμό των units κάθε επιπέδου [105].



Εικόνα 4.1.6.3: Αρχιτεκτονική του μοντέλου SimpleRNN που αναπτύξαμε με 3 επίπεδα

$$h_t = tanh(W_x x_t + W_h h_{t-1} + b_h)$$

$$h'_t = tanh(W'_x h_t + W'_h h'_{t-1} + b'_h)$$

$$h''_t = tanh(W''_x h'_t + W''_h h''_{t-1} + b''_h)$$

Πίνακας 4.1.6: Δομή μοντέλου SimpleRNN που αναπτύξαμε για το πείραμα μας

Layer Type	**Filter**	**Dropout**	**Pool Size**	**Activation**	**Return Sequences**	**Output Shape**
Input (x)						(?, Window, Nodes*Nodes)
SimpleRNN	512			tanh	true	(?, 10, 512)
Dropout		0.2				(?, 10, 512)
MaxPooling1D			2			(?, 5, 512)
SimpleRNN	256			tanh	true	(?, 5, 256)
Dropout		0.2				(?, 5, 256)

MaxPooling1D			5			(?, 1, 256)
SimpleRNN	144			tanh	true	(?, 1, Nodes*Nodes)

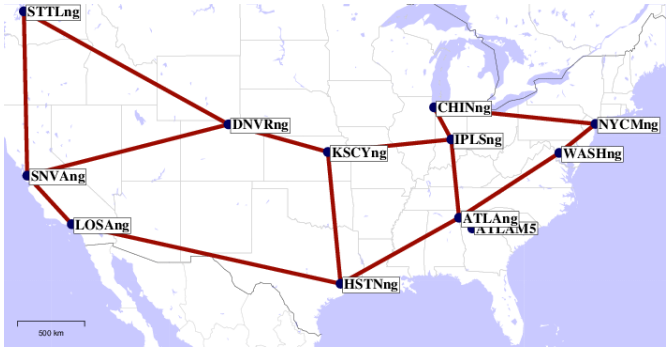
Το μοντέλο μας περιλαμβάνει 3 επίπεδα SimpleRNN με 512, 256 και Nodes*Nodes (Abilene→12*12=144 / Geant→23*23=529) units. Το τελευταίο επίπεδο SimpleRNN θέλουμε να έχει μέγεθος ίσο με τον αριθμό των ζευγών προέλευσης προορισμού (Nodes*Nodes), ώστε τα δεδομένα μετά την εκπαίδευση έχουν μέγεθος (?, 1, Nodes*Nodes). Επειδή χρησιμοποιούμε το ίδιο μοντέλο για 2 σύνολα δεδομένων δεν βάζουμε σταθερό αριθμό δομικών μονάδων στο τελευταίο επίπεδο για να λαμβάνουμε το αναμενόμενο μέγεθος εξόδου των δεδομένων και για τα 2 datasets.Στο μοντέλο αυτό χρησιμοποιήθηκε η προεπιλεγμένη συνάρτηση ενεργοποίησης tanh για τον λόγο που αναφέραμε προηγουμένως. Γενικά, είναι προτιμότερο ένα μοντέλο με περισσότερα επίπεδα και σχετικά λίγα units από ένα μοντέλο με 1-2 επίπεδα και αυξημένο αριθμό units. Αυτό αποδείχτηκε και στην εργασία [106] η οποία συγκρίνει τα Simple RNN με 1 και 2 επίπεδα με τα αντίστοιχα LSTM. Μεταξύ των SimpleRNN χρησιμοποιήθηκαν dropout, ως μια τεχνική για την αποφυγή της υπερπροσαρμογής (overfitting) [107]. Τέλος, όπως και στα προηγούμενα μοντέλα LSTM, BiLSTM και GRU προστέθηκαν 2 επίπεδα MaxPooling. Οι παράμετροι αυτών των επιπέδων έμειναν ίδιες με τα προηγούμενα μοντέλα, διότι σε αυτές τις παραμέτρους μετρήθηκαν τα καλύτερα αποτελέσματα.

4.2 Ρυθμίσεις Πειράματος

4.2.1 Σύνολα Δεδομένων - Προεπεξεργασία Δεδομένων

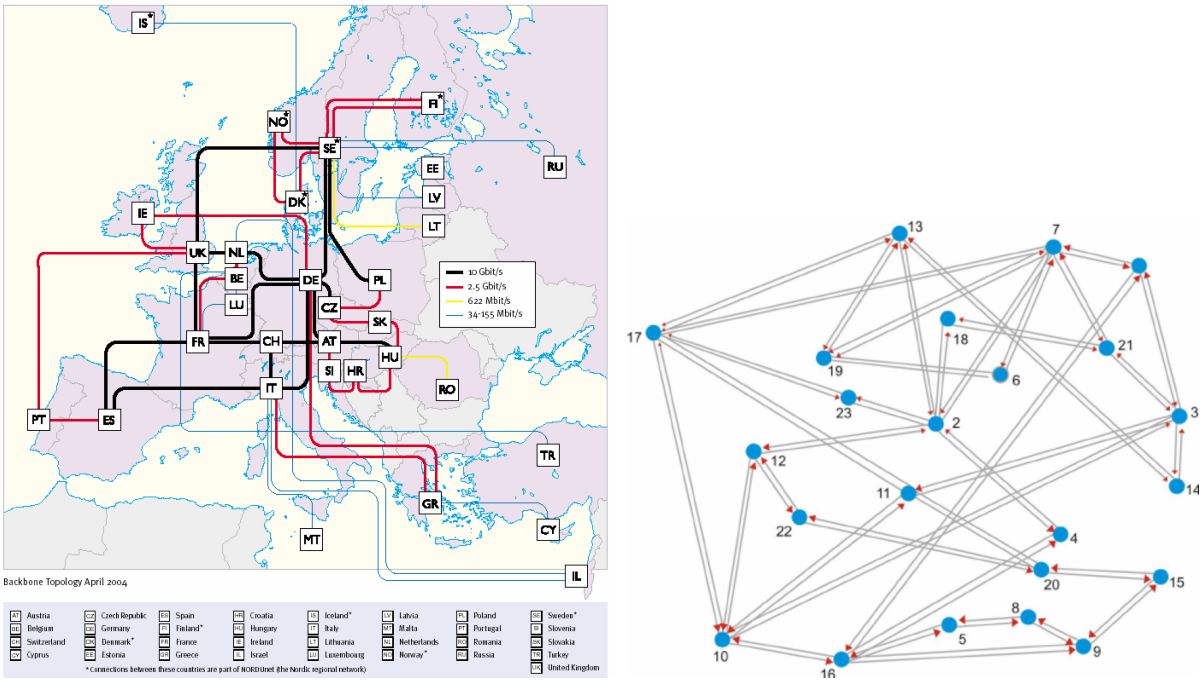
Για την ορθή αξιολόγηση τους τα μοντέλα έπρεπε να εξεταστούν σε αληθινά δίκτυα. Προϋπόθεση για την χρήση ενός δικτύου είναι η γνώση της τοπολογίας του και της χωρητικότητας των ζεύξεων. Επίσης, προκειμένου να εφαρμοστούν τα μοντέλα πρέπει να έχει γίνει μέτρηση της κίνησης του δικτύου σε ένα συγκεκριμένο χρονικό διάστημα. Για τον λόγο αυτό χρησιμοποιήθηκαν τα δεδομένα από τα δίκτυα Abilene [2] και GEANT [1]. Τα δεδομένα των δύο δικτύων είναι ελεύθερα προς χρήση για ερευνητικούς σκοπούς [108]. Το Αμερικανικό Ερευνητικό και Εκπαιδευτικό Δίκτυο Abilene είναι ένα δίκτυο κορμού που βρίσκεται στη Βόρεια Αμερική και αποτελείται από 12 κύριους κόμβους εκ των οποίων τα περισσότερα είναι μεγάλα πανεπιστήμια των ΗΠΑ. Η κίνηση γίνεται μεταξύ και των 12 κόμβων για αυτό και υπάρχουν $12 \times 12 = 144$ ροές προέλευσης προορισμού. Επίσης, η τοπολογία του δικτύου αποτελείται από $15 \times 2 = 30$ κύριες ζεύξεις καθορισμένης κατεύθυνσης που συνδέουν αυτούς τους κόμβους μεταξύ τους. Κάθε κόμβος έχει και άλλες 2 επιπλέον εξωτερικές ζεύξεις, 1 για είσοδο δεδομένων από εξωτερικό κόμβο και 1 για έξοδο δεδομένων προς κόμβο εκτός δικτύου [2]. Άρα συνολικά οι ζεύξεις είναι $30 + 12 \times 2 = 54$. Η χωρητικότητα όλων των εσωτερικών ζεύξεων είναι 9920000 kbps, εκτός από τις ζεύξεις των πόλεων Ατλάντα-Ιντιανάπολη που έχουν

χωρητικότητα 2480000 kbps. Το σύνολο δεδομένων του δικτύου Abilene αποτελείται από τους μέσους όρους (kbps) δεδομένων κίνησης που μετρήθηκαν σε διάστημα 5 λεπτών για 24 εβδομάδες, από την 1η Μαρτίου έως τις 10 Σεπτεμβρίου 2004 [2]. Το σύνολο αυτό περιέχει 48096 χρονικά στιγμιότυπα, δηλαδή 48096 πίνακες κίνησης μεγέθους 12*12. Το δίκτυο αυτό είναι κατάλληλο για εκπαίδευση μοντέλων πρόβλεψης και εκτίμησης δικτυακής κίνησης λόγω του μεγάλου όγκου πληροφοριών που μας παρέχει [10, 13, 14, 23, 40, 109, 110, 111]. Παράλληλα, τα δεδομένα του Abilene έχουν χρησιμοποιηθεί για ανάλυση των ροών προέλευσης προορισμού [112], για εξισορρόπηση φορτίου μεταξύ των ζεύξεων [113] και για δρομολόγηση των ροών κίνησης [114].



Εικόνα 4.2.1: Τοπολογία Δικτύου Abilene [115]

Το Ευρωπαϊκό Ερευνητικό και Εκπαιδευτικό Δίκτυο GEANT είναι ένα δίκτυο κορμού που βρίσκεται στην Ευρώπη και αποτελείται από 23 κύριους κόμβους εκ των οποίων τα περισσότερα είναι μεγάλα πανεπιστήμια και ερευνητικά κέντρα. Η δικτυακή κίνηση μεταξύ των κόμβων αποτελείται από 23 * 23 = 529 ροές προέλευσης προορισμού. Οι κόμβοι είναι ανώνυμοι και χαρακτηρίζονται με αριθμούς από το 1 έως το 23. Επίσης, η τοπολογία του δικτύου αποτελείται από 74 εσωτερικές ζεύξεις που συνδέουν αυτούς τους κόμβους μεταξύ τους. Το σύνολο δεδομένων του δικτύου GEANT αποτελείται από τους μέσους όρους (kbps) δεδομένων κίνησης που μετρήθηκαν σε διάστημα 15 λεπτών για 4 μήνες, από την 1η Ιανουαρίου έως τις 29 Απριλίου 2005. Το σύνολο αυτό περιέχει 10772 χρονικά στιγμιότυπα, δηλαδή 10772 πίνακες κίνησης μεγέθους 23*23. Το σύνολο δεδομένων του δικτύου GEANT λόγω του μεγάλου όγκου πληροφοριών που μας παρέχει έχει χρησιμοποιηθεί για εκπαίδευση μοντέλων πρόβλεψης και εκτίμησης δικτυακής κίνησης [13, 14, 23, 40, 109, 110, 111] και και για δρομολόγηση των ροών κίνησης [114].



Εικόνα 4.2.2: Τοπολογία Δικτύου GEANT (Φυσική Τοποθεσία - Γράφος) [116, 117]

Για να χρησιμοποιηθούν τα δεδομένα των δικτύων χρειάζεται μια επεξεργασία, ώστε να μετατραπούν οι μετρήσεις της δικτυακής κίνησης σε πίνακες κίνησης. Το αποτέλεσμα αυτής της επεξεργασίας των πληροφοριών για T χρονικά βήματα και N κόμβους είναι ο πίνακας δεδομένων $T \times N \times N$ με το στοιχείο $x_{i,j}(t)$ να είναι η ένταση της δικτυακής κίνησης με πηγή τον κόμβο i και προορισμό τον κόμβο j την χρονική στιγμή t. Αναλόγως το μοντέλο ο πίνακας μπορεί να μετατραπεί σαν πίνακας διαστάσεων $T \times N^2$ με το στοιχείο $x_{j,t}$ να είναι η ένταση της δικτυακής κίνησης του ζεύγους j την χρονική στιγμή t. Για παράδειγμα στα μοντέλα Conv-LSTM και CNN-LSTM ο πίνακας κίνησης έχει τη μορφή $T \times N \times N$, ενώ για τα μοντέλα Stacked LSTM, BiLSTM, GRU και SimpleRNN ο πίνακας κίνησης έχει τη μορφή TxN^2. Έπειτα, κανονικοποιούμε τα δεδομένα σε τιμές μεταξύ των 0-1 προκειμένου να είναι πιο γρήγορη η εκπαίδευση. Για να είναι εφικτή και αποδοτική η κανονικοποίηση πρέπει να απομακρύνουμε κάποιες ακραίες τιμές των δεδομένων των δικτύων. Συγκεκριμένα υπάρχει μια τιμή στο Abilene που είναι 5 τάξεις μεγέθους μεγαλύτερη από όλες τις υπόλοιπες γεγονός που μας προϊδεάζει ότι είναι σφάλμα μέτρησης. Αντίστοιχα στο δίκτυο GEANT υπάρχουν τιμές με 2-3 τάξεις μεγέθους μεγαλύτερες. Για τον λόγο αυτό έχει τοποθετηθεί ένα κατώφλι τιμών και στα δύο σύνολα δεδομένων (Abilene → 99.99998%, GEANT → 99.9%). Με τον τρόπο αυτό αντικαθίστανται οι ακραίες τιμές των δεδομένων με τις μέγιστες "φυσιολογικές" τιμές που είναι εντός ορίων. Προκειμένου να γίνει η εκπαίδευση και η αξιολόγηση των μοντέλων, χωρίζουμε τον πίνακα δεδομένων σε training set 80% και test set 20%. Τέλος, για την υλοποίηση της εκπαίδευσης χρησιμοποιούνται 10 χρονικά στιγμιότυπα ως είσοδος με το ακριβώς επόμενο χρονικό στιγμιότυπο να χρησιμοποιείται σαν στόχος. Αυτή είναι η τεχνική κυλιόμενου

παραθύρου (sliding window) με μέγεθος παραθύρου 10, όπου σε κάθε επανάληψη το παράθυρο μετατοπίζεται κατά 1 θέση (overlap window με stride=1). Η τεχνική αυτή χρησιμοποιείται ευρέως σε προβλήματα ανάλυσης χρονοσειρών.

4.2.2 Εκτέλεση

I. Ρυθμίσεις Πειράματος - Βιβλιοθήκες

Τα πειράματα προσομοίωσης έγιναν σε φορητό υπολογιστή laptop με επεξεργαστή Intel(R) Core(TM) i5-7200U CPU και 8GB μνήμη RAM, ενώ η εκπαίδευση έγινε σε έναν server μέσω ενός commercial VPS provider με 16GB RAM και 6 vCPUs. Τα μοντέλα υλοποιήθηκαν σε Python3 και οι κύριες βιβλιοθήκες που χρησιμοποιήθηκαν για την υλοποίηση των μοντέλων είναι οι `Tensorflow` [118] και `Keras` [119]. Το `Tensorflow` είναι μια βιβλιοθήκη λογισμικού ανοιχτού κώδικα για αριθμητικούς υπολογισμούς που χρησιμοποιείται ευρέως στον τομέα της μηχανικής μάθησης. Αναπτύχθηκε από την Google το 2015 για να καλύψει τις ανάγκες της σε συστήματα ικανά να δημιουργήσουν και να εκπαιδεύσουν νευρωνικά δίκτυα για να ανιχνεύσουν και να αποκρυπτογραφήσουν μοτίβα και συσχετισμούς, ανάλογα με τη μάθηση και τη λογική που χρησιμοποιούν οι άνθρωποι. Το `TensorFlow` προσφέρεται για προγραμματισμό σε χαμηλό και σε υψηλό επίπεδο μέσω των API που διαθέτει. Ένα τέτοιο υποστηριζόμενο API είναι το `Keras`, το οποίο επιτρέπει την μοντελοποίηση δικτύων σε υψηλό επίπεδο. Το `Keras` είναι πιο εύκολο σε χρήση γιατί είναι φτιαγμένο σε Python. Χρησιμοποιείται για την εκπαίδευση του δικτύου μέσω συναρτήσεων βελτιστοποίησης και απώλειας αλλά και για την αξιολόγησή του μέσω κάποιας μετρικής. Επίσης, επιτρέπει την παραμετροποίηση των νευρώνων κάθε επιπέδου. Για την κατασκευή των μοντέλων στο `keras` χρησιμοποιήσαμε `Sequential API` και την μέθοδο add για να προσθέσουμε layers. Επίσης, χρησιμοποιήσαμε την μέθοδο compile που ρυθμίζει την εκπαίδευση του μοντέλου. Για την ανάλυση και επεξεργασία των δεδομένων χρησιμοποιήθηκαν και οι βιβλιοθήκες `matplotlib`, `NumPy`, `Scikit-learn (sklearn)`. Η βιβλιοθήκη `matplotlib` χρησιμοποιήθηκε για την παρουσίαση των αποτελεσμάτων σε γραφικές παραστάσεις, ενώ η `sklearn` χρησιμοποιήθηκε για την αξιοποίηση κάποιων έτοιμων συναρτήσεων όπως της mean_squared_error. Τέλος, η `NumPy` χρησιμοποιήθηκε για μαθηματικούς υπολογισμούς (max, min, percentile, sqrt, absolute, subtract κλπ). Ο πλήρης κώδικας της εργασίας παρατίθεται στο Παράρτημα.

II. Υπερ-παράμετροι

Οι υπερ-παράμετροι χρησιμοποιούνται στην μηχανική μάθηση και καθορίζουν τη δομή και τον τρόπο εκπαίδευσης του δικτύου. Στα μοντέλα που υλοποιήσαμε χρησιμοποιήθηκαν οι εξής κοινές παράμετροι:

• batch size = 128
• epochs = 100
• optimizer = Adam
• loss function = MAE

Το batch size καθορίζει τον αριθμό των δειγμάτων (samples) του δικτύου που θα επεξεργαστούν πριν ανανεωθούν οι εσωτερικές παράμετροι του μοντέλου. Epochs είναι η παράμετρος που παρουσιάζει τον αριθμό των επαναλήψεων της εκπαίδευσης όλων των δεδομένων εκπαίδευσης (training set). Σε κάθε εποχή κάθε sample εχει την δυνατότητα να διαφοροποιήσει τις εσωτερικές παραμέτρους του μοντέλου. Οι βελτιστοποιητές (optimizers) είναι οι μέθοδοι που χρησιμοποιούμε για να τροποποιήσουμε τις παραμέτρους του δικτύου μας. Είναι μαθηματικές συναρτήσεις που στοχεύουν στην ελαχιστοποίηση των λαθών. Οι πιο διαδεδομένοι optimizers είναι οι RMSprop, Adam, Adagrad, Adadelta, Adamax, Nadam, Gradient Descent, Stochastic Gradient Descent [120]. Στα μοντέλα μας χρησιμοποιούμε τον Adam (Adaptive Momentum Estimation) ο οποίος είναι από τους καλύτερους optimizer γιατί έχει γρήγορους υπολογισμούς, χρησιμοποιεί λιγότερες παραμέτρους και τις περισσότερες φορές έχει τα καλύτερα αποτελέσματα. Ο Adam έχει ξεχωριστά learning rates για τα διαφορετικά βάρη και χρησιμοποιεί εκτιμήσεις της πρώτης και της δεύτερης ροπής της κλίσης (Μέση Τιμή και Διακύμανση) για να προσαρμόσει το ρυθμό μάθησης του κάθε βάρους του δικτύου [121]. Ο Adam αποθηκεύει έναν εκθετικά μειούμενο μέσο όρο των προηγούμενων τετραγωνικών κλίσεων u_t και έναν εκθετικά μειούμενο μέσο όρο των προηγούμενων κλίσεων m_t.

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$$

$$u_t = \beta_2 u_{t-1} + (1 - \beta_2) g_t^2$$

Τα m_t και u_t είναι η μέση τιμή και η διακύμανση της κλίσης. Όταν τα m_t και u_t αρχικοποιούνται σαν μηδενικά διανύσματα τα β_1 και β_2 τείνουν στη μονάδα. Μετά υπολογίζουμε τα $\widehat{m_t}$, $\widehat{u_t}$:

$$\widehat{m_t} = \frac{m_t}{1-\beta_1^t}$$

$$\widehat{u_t} = \frac{u_t}{1-\beta_2^t}$$

Αυτά μετά χρησιμοποιούνται για να ενημερώσουν τις παραμέτρους, οι οποίες μας δίνουν τον κανόνα ενημέρωσης Adam:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\widehat{u_t}}+\varepsilon} \cdot \widehat{m_t}$$

Τέλος, κατά την εκπαίδευση η συνάρτηση που υπολογίζει το σφάλμα είναι η MAE (Mean Absolute Error). Στη συνάρτηση αυτή υπολογίζεται η απόλυτη τιμή της διαφοράς μεταξύ πραγματικής τιμής και τιμής πρόβλεψης για κάθε στοιχείο της δικτυακής κίνησης. Από αυτές τις τιμές υπολογίζεται ο μέσος όρος ο οποίος ανανεώνεται σε κάθε νέο πακέτο δειγμάτων. Το MAE είναι δίνεται από τη σχέση:

$$MAE(t) = \frac{\sum_{i=1}^{N} \left| \widehat{x_t}(i) - x_t(i) \right|}{N}$$

III. Early Stopping

Το Early Stopping [122] είναι η πιο συχνά χρησιμοποιούμενη μέθοδος για την αποφυγή του overfitting. Η τεχνική του Early Stopping είναι απλή και ξεκινάει με τον διαχωρισμό του συνόλου δεδομένων σε σύνολο εκπαίδευσης και σύνολο αξιολόγησης. Επειτα, εκπαιδεύουμε μόνο το σύνολο εκπαίδευσης και παρατηρούμε το σφάλμα κάθε εποχής στα δεδομένα εκπαίδευσης. Σταματάμε την εκπαίδευση μόλις παρατηρήσουμε ότι το σφάλμα είναι ψηλότερο από την προηγούμενη φορά που το ελέγξαμε και τέλος παρουσιάζουμε τα βάρη του προηγούμενου βήματος σαν αποτέλεσμα της εκπαίδευσης [122]. Είναι σημαντικό να αντιληφθούμε ότι το σφάλμα αξιολόγησης δεν αποτελεί καλή εκτίμηση του σφάλματος γενίκευσης για αυτό και πρέπει να δοκιμάσουμε το μοντέλο σε δεδομένα που δεν έλαβαν μέρος στην εκπαίδευση. Η αξιολόγηση της εκπαίδευσης γίνεται κάθε φορά από την συνάρτηση σφάλματος (monitor='loss'). Συχνά η πρώτη επανάληψη που δεν παρατηρείται βελτίωση του σφάλματος δεν είναι η καλύτερη στιγμή να σταματήσει η εκπαίδευση. Είναι πιθανό για διάφορους λόγους να καθυστερήσει η βελτίωση για έναν αριθμό εποχών. Για τον λόγο αυτό υπάρχει η παράμετρος ‘patience’ που καθορίζει τον αριθμό των εποχών που δεν μειώνεται το σφάλμα πριν διακοπεί η εκπαίδευση. Η παράμετρος αυτή εξαρτάται από το πρόβλημα και τον αριθμό των εποχών [123]. Γενικά, αν το σφάλμα παραμένει σταθερό για πολλές επαναλήψεις ή αρχίζει να αυξάνεται, σημαίνει ότι πιθανώς το μοντέλο μας αρχίζει να υπερεκπαιδεύεται και χάνει την ικανότητα γενίκευσης σε νέα δεδομένα, ενώ μαθαίνει πολύ καλά τα δεδομένα εκπαίδευσης. Στο πρόβλημα μας χρησιμοποιούμε την έτοιμη κλάση EarlyStopping που μας παρέχει το keras και χρησιμοποιείται σαν callback κατά την εκπαίδευση των δεδομένων. Έπειτα από αρκετούς συνδυασμούς και δοκιμές παρατηρήθηκαν τα καλύτερα αποτελέσματα σε 100 εποχές με patience 10.

4.3 Αξιολόγηση Απόδοσης

Τα 6 μοντέλα που αναπτύχθηκαν αξιολογήθηκαν κατά την διάρκεια της εκπαίδευσης με τη μετρική Accuracy. Η μετρική accuracy υπολογίζει πόσο συχνά η πρόβλεψη ταυτίζεται με την πραγματική τιμή. Η κυριότερη αξιολόγηση των μοντέλων έγινε στα testing sets όπου το εκπαιδευμένο μοντέλο δοκιμάστηκε σε άγνωστα δεδομένα. Οι μετρικές που αξιολόγησαν τα μοντέλα είναι οι εξής:

- **Root Mean Square Error (RMSE)**

Το RMSE είναι η Ρίζα του Μέσου Τετραγωνικού Σφάλματος (MSE) και δίνεται από τη σχέση:

$$RMSE(t) = \sqrt{\frac{\sum_{i=1}^{N} (\widehat{x_t}(i) - x_t(i))^2}{N}}$$

όπου x είναι ο πραγματικός Πίνακας Κίνησης και $\widehat{x}$ είναι ο Πίνακας Κίνησης που προβλέψαμε. Το $i = 1, 2, \ldots, N$ αντιπροσωπεύει την κάθε ροή κίνησης και το $t = 1, 2, \ldots, T$ δηλώνει το κάθε χρονικό σημείο. Η μετρική αυτή είναι βασισμένη στο MSE που παίρνει το μέσο όρο του τετραγώνου της διαφοράς μεταξύ των διανυσμάτων παρατηρήσεων και των προβλέψεων. Το πλεονέκτημα του RMSE σε σχέση με το MSE είναι ότι βρίσκεται στην ίδια κλίμακα με τα δεδομένα οπότε είναι πιο εύκολο να το κατανοήσει κάποιος. Εξαιτίας του τετραγώνου τα σφάλματα αντίθετου προσήμου δεν ακυρώνονται μεταξύ τους, κάτι που θα μείωνε την τιμή της μετρικής. Η μετρική αυτή χρησιμοποιείται κυρίως όταν ενδιαφερόμαστε για τα μεγάλα σφάλματα, διότι το αποτέλεσμα της επηρεάζεται πολύ από αυτά. Καθώς παίρνουμε το τετράγωνο του σφάλματος, η επίδραση των μεγαλύτερων σφαλμάτων γίνεται πιο έντονη σε σχέση με τα μικρότερα, επομένως το μοντέλο μαθαίνοντας μπορεί να εστιάσει περισσότερο στα μεγαλύτερα λάθη. Τα μεγαλύτερα λάθη μπορούν να προκύψουν είτε όταν η προβλεπόμενη τιμή είναι αρκετά μεγαλύτερη από την πραγματική, είτε όταν είναι αρκετά μικρότερη.

- **Normalized Mean Absolute Error (NMAE)**

Το NMAE είναι η Κανονικοποιημένη τιμή του Μέσου Απόλυτου Σφάλματος (MAE) και δίνεται από τη σχέση:

$$NMAE(t) = \frac{\sum_{i=1}^{N} \left| \widehat{x_t}(i) - x_t(i) \right|}{\sum_{i=1}^{N} \left| x_t(i) \right|}$$

όπου x είναι ο πραγματικός Πίνακας Κίνησης και $\widehat{x}$ είναι ο Πίνακας Κίνησης που προβλέψαμε. Το $i = 1, 2, ..., N$ αντιπροσωπεύει την κάθε ροή κίνησης και το $t = 1, 2, ..., T$ δηλώνει το κάθε χρονικό σημείο. Η μετρική αυτή είναι βασισμένη στο MAE που μετράει τη μέση απόλυτη απόκλιση των προβλεπόμενων τιμών από τις πραγματικές για όλο το σύνολο της πρόβλεψης. Το μέσο απόλυτο σφάλμα είναι ο μέσος όρος της διαφοράς μεταξύ των πραγματικών τιμών και των προβλέψεων. Λόγω της απόλυτης τιμής του δεν υπάρχουν αρνητικά σφάλματα. Επίσης το σφάλμα είναι στην ίδια κλίμακα με τα δεδομένα. Το πλεονέκτημα του NMAE σε σχέση με το MAE είναι ότι η κανονικοποιημένη τιμή βοηθάει στις συγκρίσεις της απόδοσης για χρονοσειρές με διαφορετικές χρονικές κλίμακες.

- **Temporal Relative Error (TRE)**

Το TRE είναι το Χρονικό Σχετικό Σφάλμα που εκφράζει το σφάλμα όλων των ροών προέλευσης - προορισμού σε ένα συγκεκριμένο χρονικό σημείο και δίνεται από τη σχέση:

$$TRE(t) = \frac{\sqrt{\sum_{i=1}^{N} (\widehat{x_t}(i) - x_t(i))^2}}{\sqrt{\sum_{i=1}^{N} (x_t(i))^2}}$$

όπου x είναι ο πραγματικός Πίνακας Κίνησης και $\widehat{x}$ είναι ο Πίνακας Κίνησης που προβλέψαμε. Το $i = 1, 2, ..., N$ αντιπροσωπεύει την κάθε ροή κίνησης και το $t = 1, 2, ..., T$ δηλώνει το κάθε χρονικό σημείο. Αυτή η μετρική εξυπηρετεί στην παρατήρηση του σφάλματος όλων των δεδομένων σε κάθε χρονική στιγμή. Έτσι γνωρίζουμε ποιές χρονικές στιγμές τα σφάλματα της πρόβλεψης ήταν μεγαλύτερα από τον μέσο όρο και ποιές στιγμές τα σφάλματα ήταν ελάχιστα.

- **Spatial Relative Error (SRE)**

Το SRE είναι το Χωρικό Σχετικό Σφάλμα που εκφράζει το σφάλμα κάθε ροής προέλευσης - προορισμού σε όλη την διάρκεια ζωής της και δίνεται από τη σχέση:

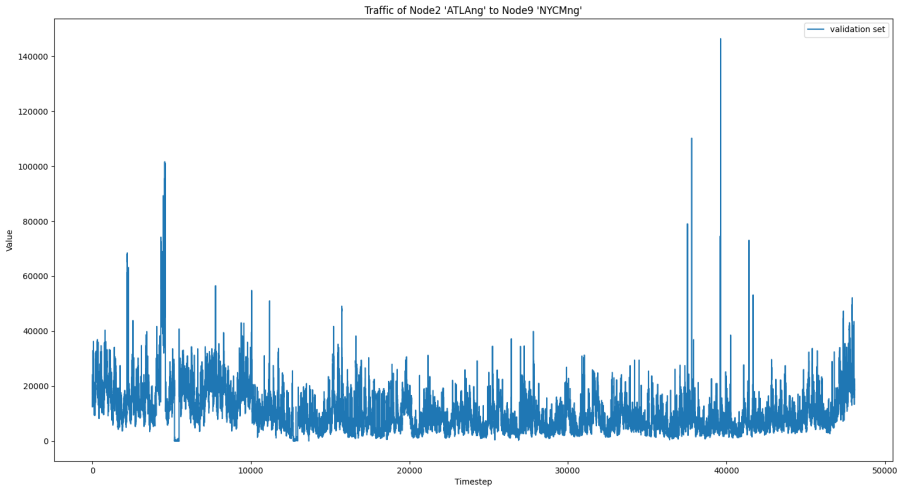
$$SRE(i) = \frac{\sqrt{\sum_{t=1}^{T} (\widehat{x_t}(i) - x_t(i))^2}}{\sqrt{\sum_{t=1}^{T} (x_t(i))^2}}$$

όπου x είναι ο πραγματικός Πίνακας Κίνησης και $\widehat{x}$ είναι ο Πίνακας Κίνησης που προβλέψαμε. Το $i = 1, 2, ..., N$ αντιπροσωπεύει την κάθε ροή κίνησης και το $t = 1, 2, ..., T$ δηλώνει το κάθε χρονικό σημείο. Αυτή η μετρική όπως και η TRE είναι ειδικές μετρικές που εξυπηρετούν σε

συγκεκριμένες παρατηρήσεις. Η SRE σε αντίθεση με τις προηγούμενες μετρικές έχει μεταβλητή την ροή κίνησης και όχι τον χρόνο. Εξυπηρετεί στην παρατήρηση του σφάλματος κάθε ροής για όλη την διάρκεια ζωής της. Με αυτόν τον τρόπο γνωρίζουμε ποια ζεύγη προέλευσης προορισμού έχουν συνολικό σφάλμα μεγαλύτερο ή μικρότερο από το συνηθισμένο.

Παρακάτω παρουσιάζουμε αναλυτικά τους συγκεντρωτικούς πίνακες και τα διαγράμματα με τα σφάλματα (RMSE, NMAE, SRE, TRE) των μοντέλων (ConvLSTM, CNNLSTM, Stacked LSTM, GRU, BiLSTM, SimpleRNN) για τα σύνολα δεδομένων Abilene και GEANT.

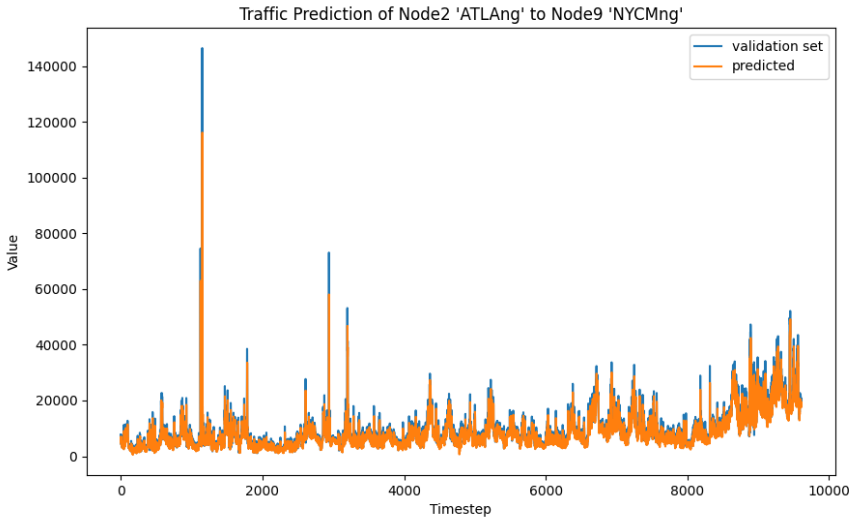
ABILENE



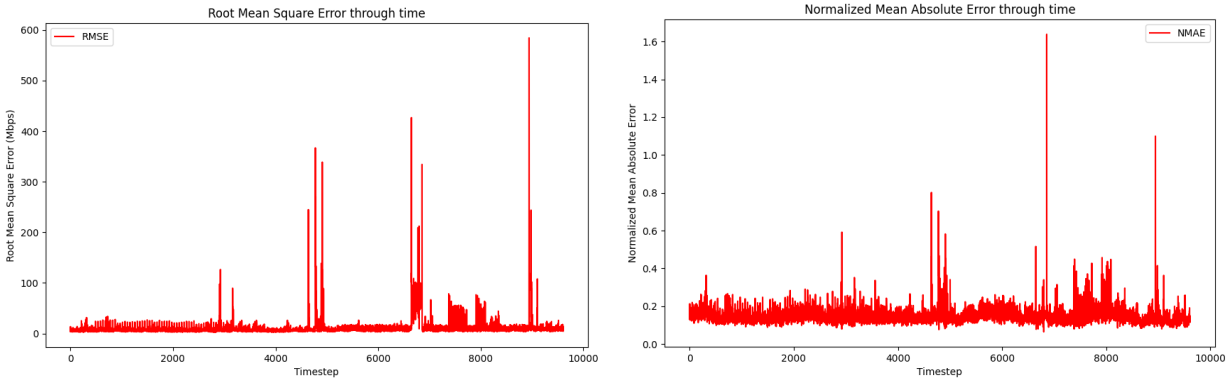
Εικόνα 4.3.1: Ένταση δικτυακής κίνησης του ζεύγους Atlang-NYCMng του Abilene σε όλη την χρονική διάρκεια

Πίνακας 4.3.1: Σφάλματα Πρόβλεψης μοντέλου ConvLSTM

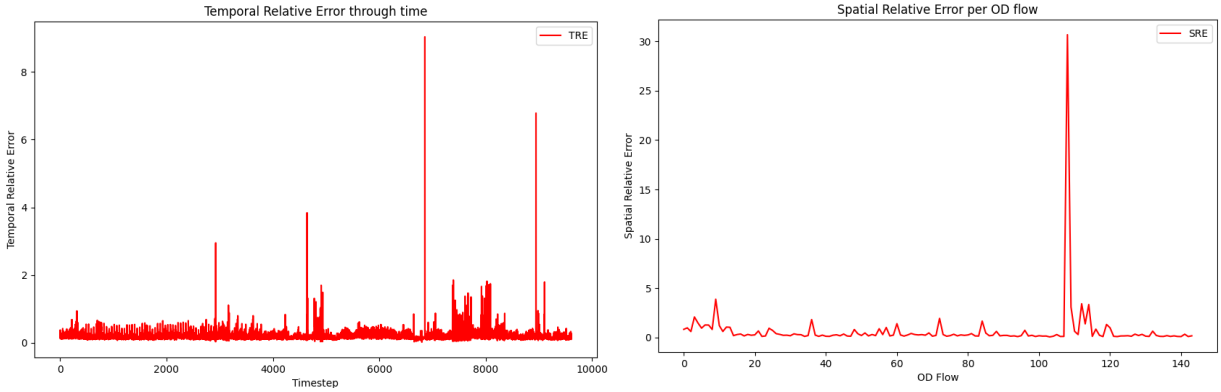
ConvLSTM				
Σφάλμα	Μέση Τιμή	Διάμεσος	Τυπική Απόκλιση	Μέγιστο
RMSE (Mbps)	10.4970	7.0144	17.0821	584.4747
NMAE	0.1498	0.1436	0.0418	1.6379
TRE	0.2051	0.1722	0.1702	9.0353
SRE	0.7049	0.2449	2.5873	30.6735



Εικόνα 4.3.2: Πρόβλεψη δικτυακής κίνησης του ζεύγους Atlang-NYCMng με τη χρήση του μοντέλου ConvLSTM



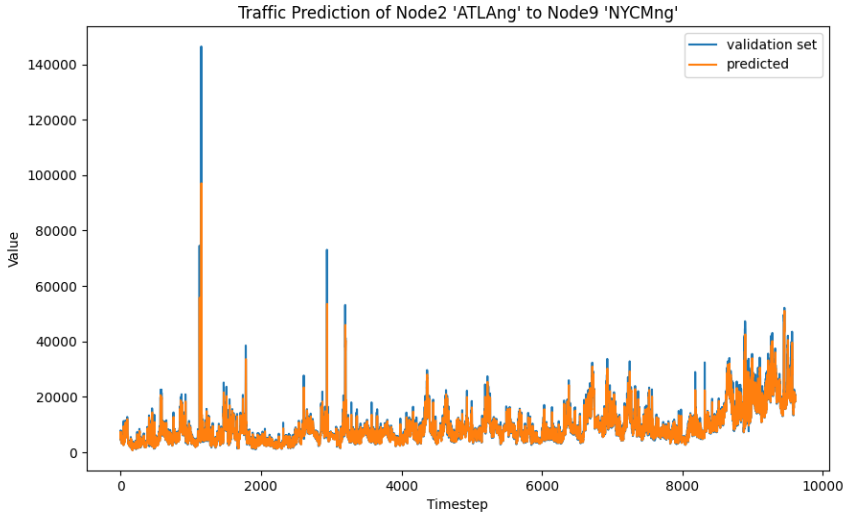
Εικόνα 4.3.3: RMSE - NMAE του μοντέλου ConvLSTM



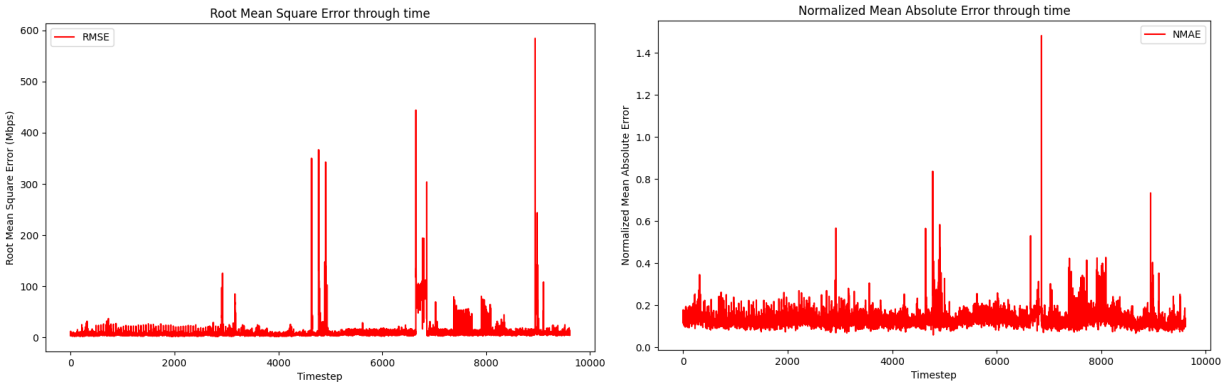
Εικόνα 4.3.4: TRE - SRE του μοντέλου ConvLSTM

Πίνακας 4.3.2: Σφάλματα Πρόβλεψης μοντέλου CNNLSTM

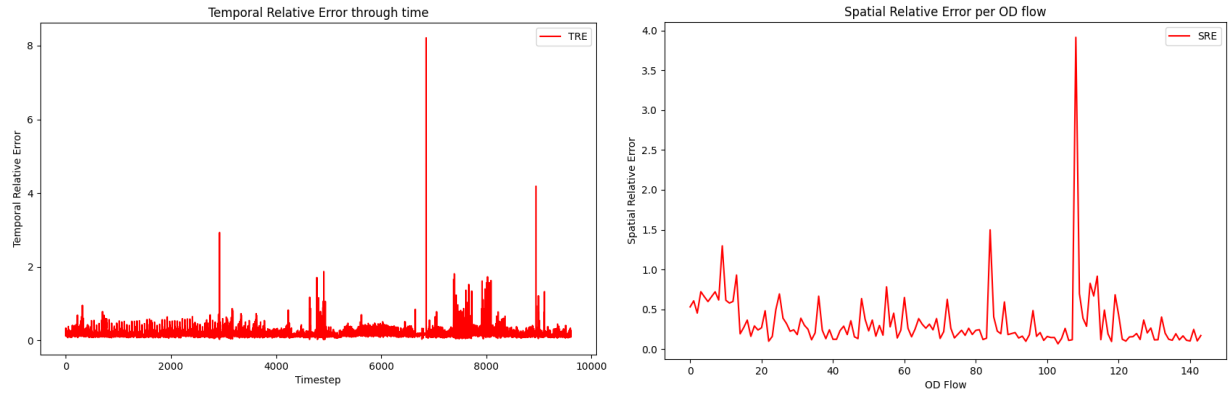
CNNLSTM				
Σφάλμα	Μέση Τιμή	Διάμεσος	Τυπική Απόκλιση	Μέγιστο
RMSE (Mbps)	10.8410	6.7129	19.4586	584.4225
NMAE	0.1352	0.1285	0.0403	1.4821
TRE	0.1980	0.1681	0.1512	8.2126
SRE	0.3416	0.2383	0.3799	3.9133



Εικόνα 4.3.5: Πρόβλεψη δικτυακής κίνησης του ζεύγους Atlang-NYCMng με τη χρήση του μοντέλου CNNLSTM



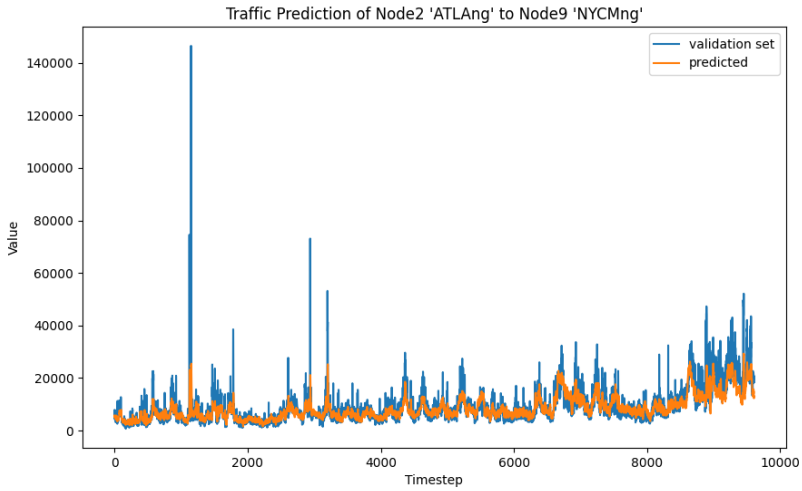
Εικόνα 4.3.6: RMSE - NMAE του μοντέλου CNNLSTM



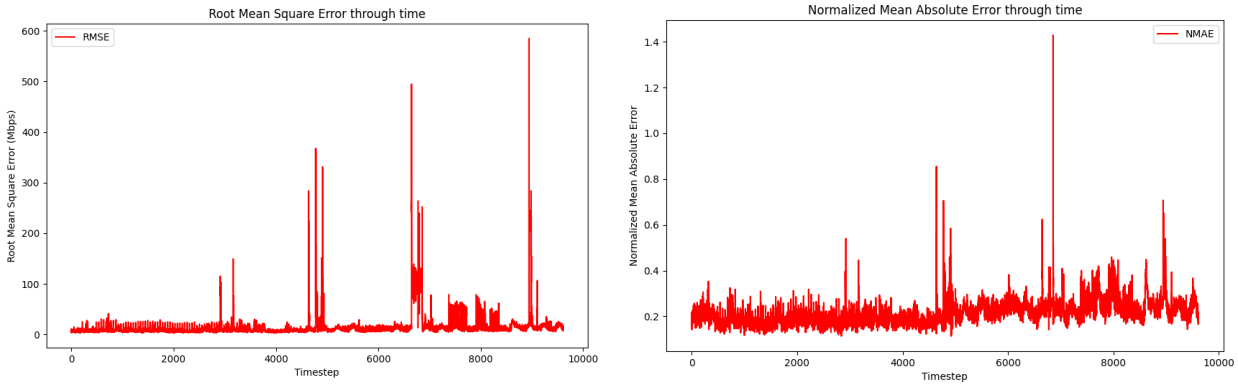
Εικόνα 4.3.7: TRE - SRE του μοντέλου CNNLSTM

Πίνακας 4.3.3: Σφάλματα Πρόβλεψης μοντέλου LSTM

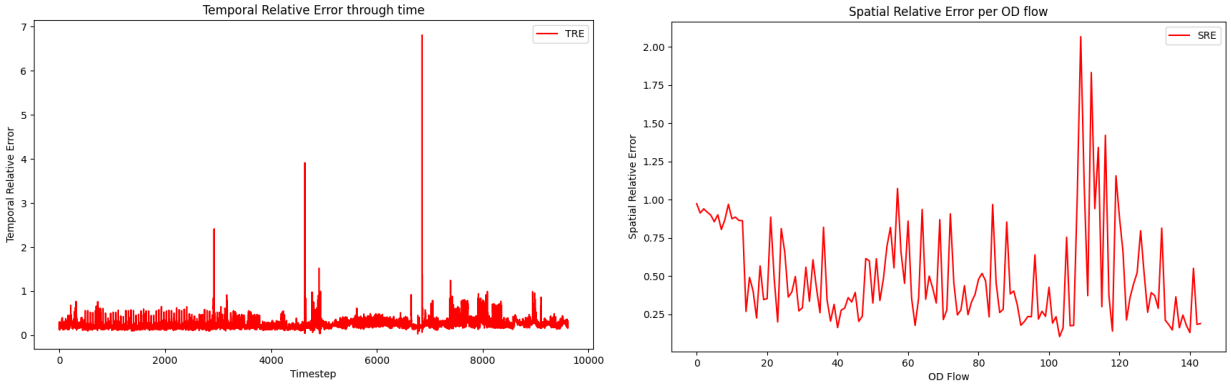
Stacked LSTM				
Σφάλμα	Μέση Τιμή	Διάμεσος	Τυπική Απόκλιση	Μέγιστο
RMSE (Mbps)	14.0897	8.7121	22.3670	584.9954
NMAE	0.2138	0.2060	0.0521	1.4288
TRE	0.2506	0.2216	0.1354	6.8107
SRE	0.5093	0.3912	0.3340	2.0667



Εικόνα 4.3.8: Πρόβλεψη δικτυακής κίνησης του ζεύγους Atlang-NYCMng με τη χρήση του μοντέλου LSTM



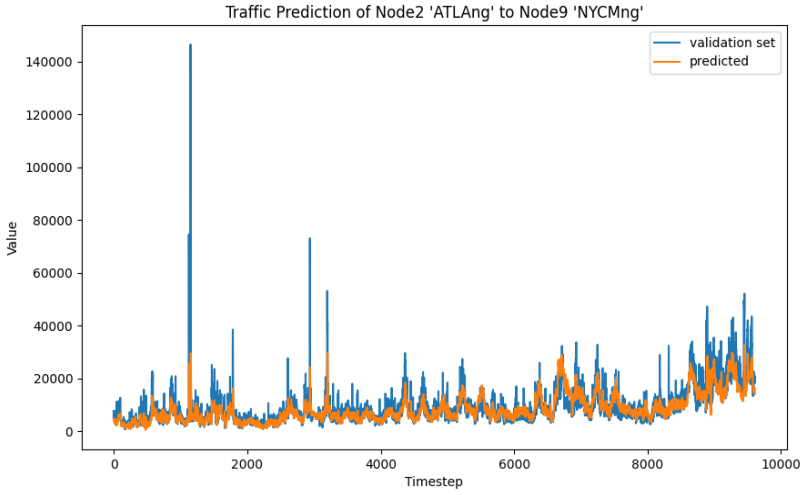
Εικόνα 4.3.9: RMSE - NMAE του μοντέλου LSTM



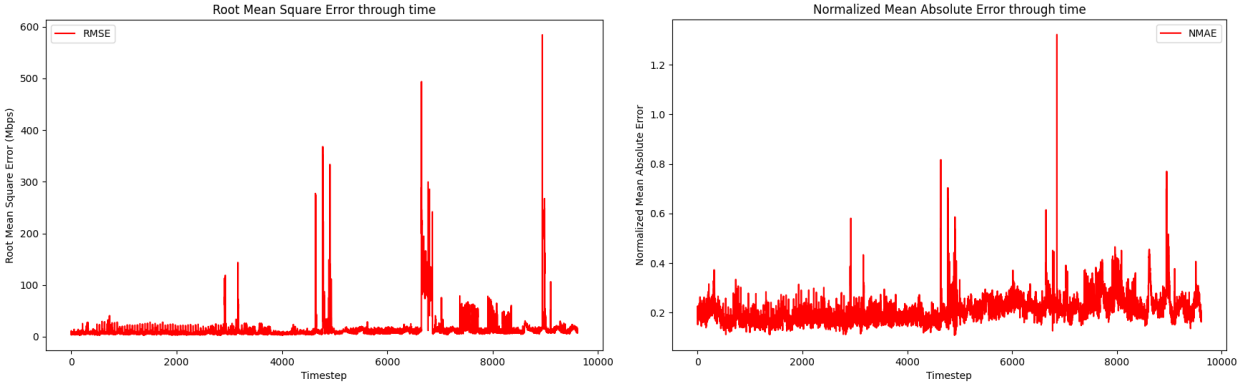
Εικόνα 4.3.10: TRE - SRE του μοντέλου LSTM

Πίνακας 4.3.4: Σφάλματα Πρόβλεψης μοντέλου GRU

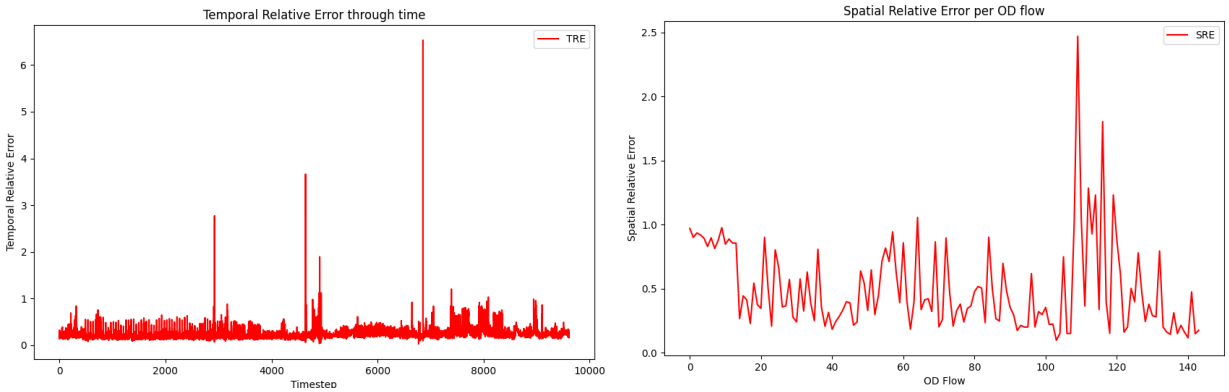
GRU				
Σφάλμα	Μέση Τιμή	Διάμεσος	Τυπική Απόκλιση	Μέγιστο
RMSE (Mbps)	13.7867	8.3903	22.9708	584.1502
NMAE	0.2092	0.2012	0.0529	1.3223
TRE	0.2428	0.2130	0.1347	6.5324
SRE	0.5041	0.3882	0.3453	2.4701



Εικόνα 4.3.11: Πρόβλεψη δικτυακής κίνησης του ζεύγους Atlang-NYCMng με τη χρήση του μοντέλου GRU



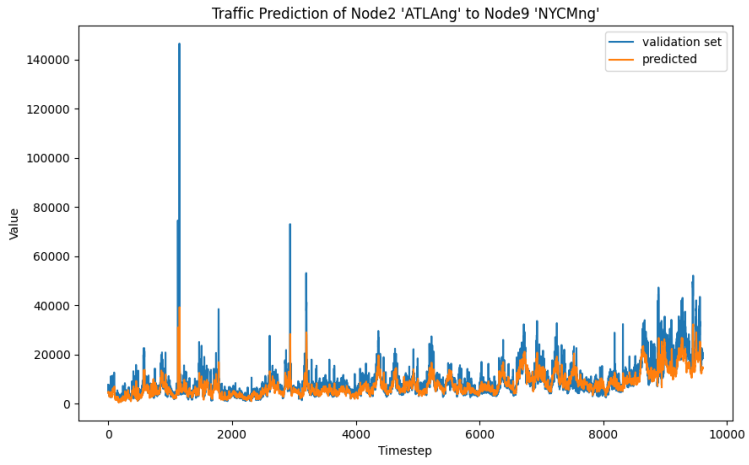
Εικόνα 4.3.12: RMSE - NMAE του μοντέλου GRU



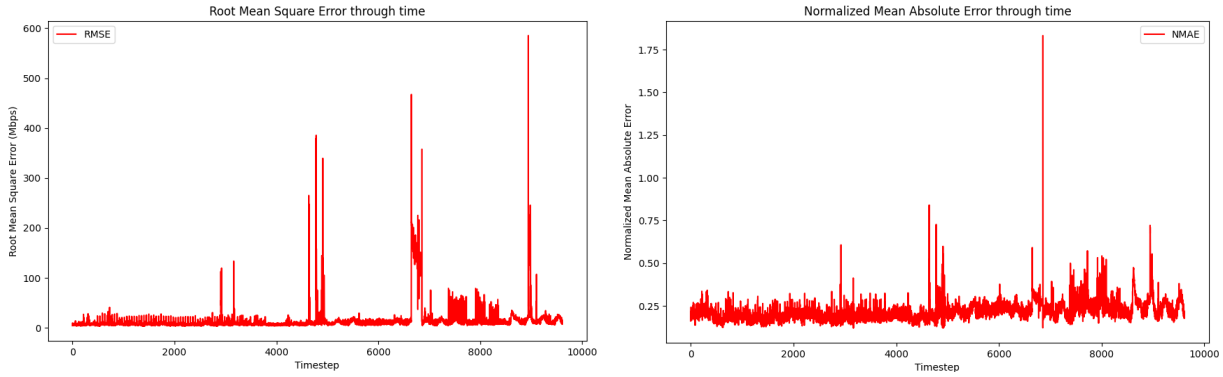
Εικόνα 4.3.13: TRE - SRE του μοντέλου GRU

Πίνακας 4.3.5: Σφάλματα Πρόβλεψης μοντέλου BiLSTM

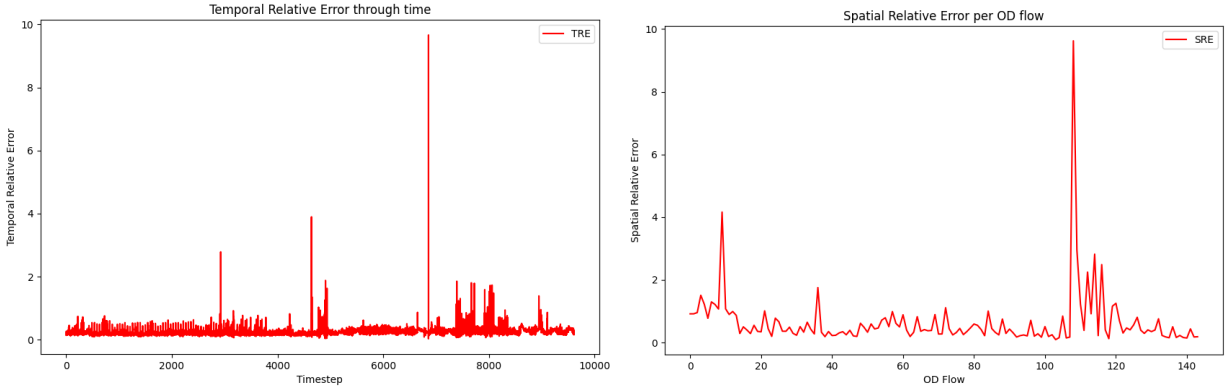
BiLSTM				
Σφάλμα	Μέση Τιμή	Διάμεσος	Τυπική Απόκλιση	Μέγιστο
RMSE (Mbps)	14.4352	8.4581	26.5813	585.3008
NMAE	0.2171	0.2080	0.0526	1.8317
TRE	0.2468	0.2208	0.1557	9.6659
SRE	0.6448	0.4034	0.9374	9.6254



Εικόνα 4.3.14: Πρόβλεψη δικτυακής κίνησης του ζεύγους Atlang-NYCMng με τη χρήση του μοντέλου BiLSTM



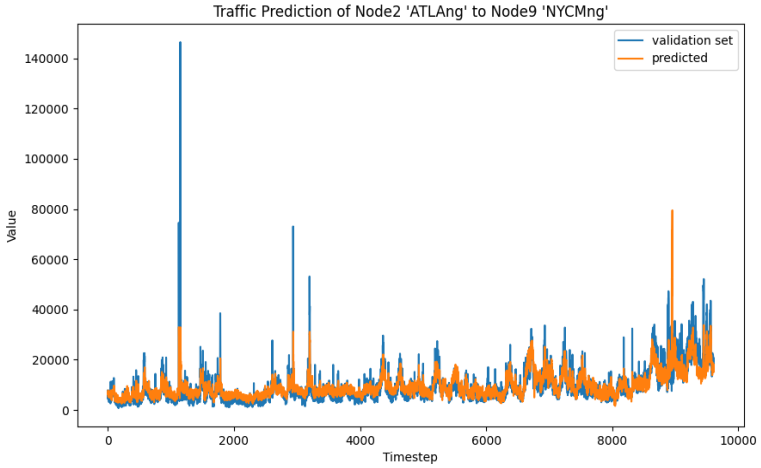
Εικόνα 4.3.15: RMSE - NMAE του μοντέλου BiLSTM



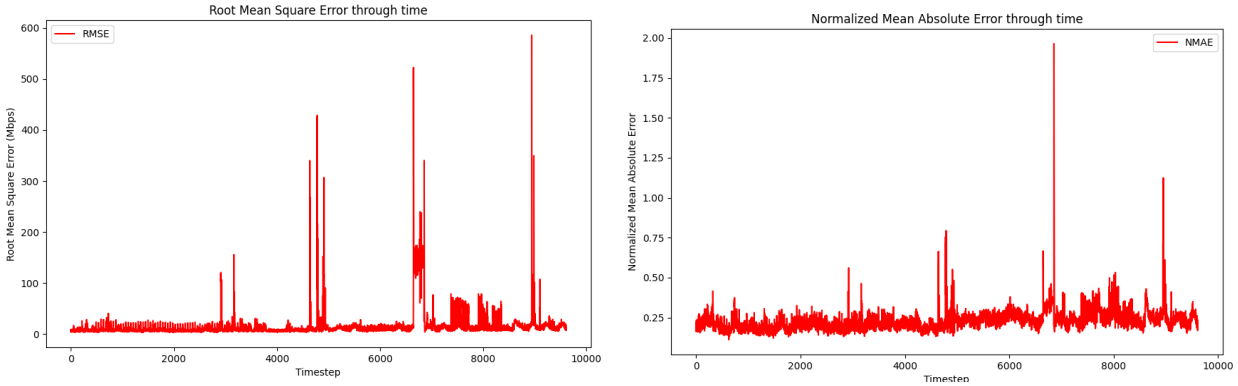
Εικόνα 4.3.16: TRE - SRE του μοντέλου BiLSTM

Πίνακας 4.3.6: Σφάλματα Πρόβλεψης μοντέλου SimpleRNN

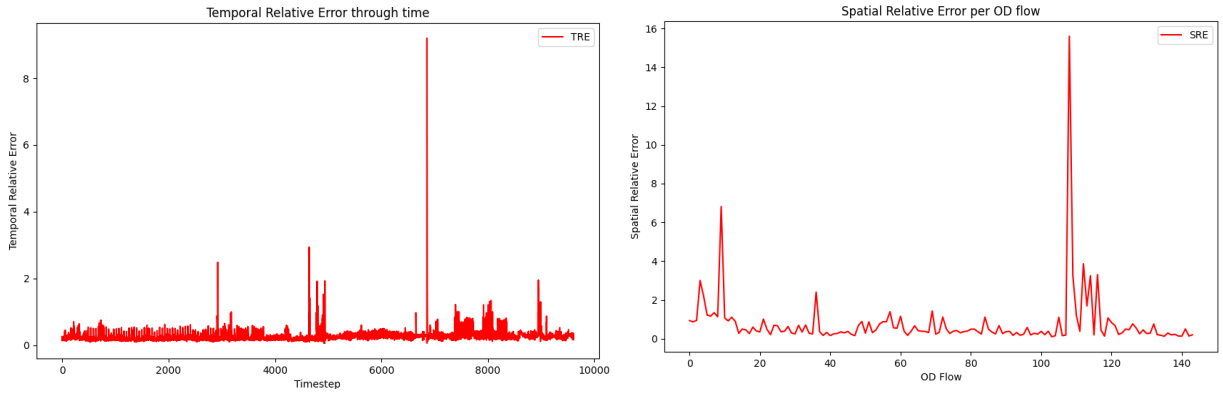
SimpleRNN				
Σφάλμα	Μέση Τιμή	Διάμεσος	Τυπική Απόκλιση	Μέγιστο
RMSE (Mbps)	15.8247	8.8944	29.5157	585.8422
NMAE	0.2295	0.2201	0.0642	1.9640
TRE	0.2658	0.2325	0.1873	9.1972
SRE	0.7763	0.3915	1.4891	15.5987



Εικόνα 4.3.17: Πρόβλεψη δικτυακής κίνησης του ζεύγους Atlang-NYCMng με τη χρήση του μοντέλου SimpleRNN

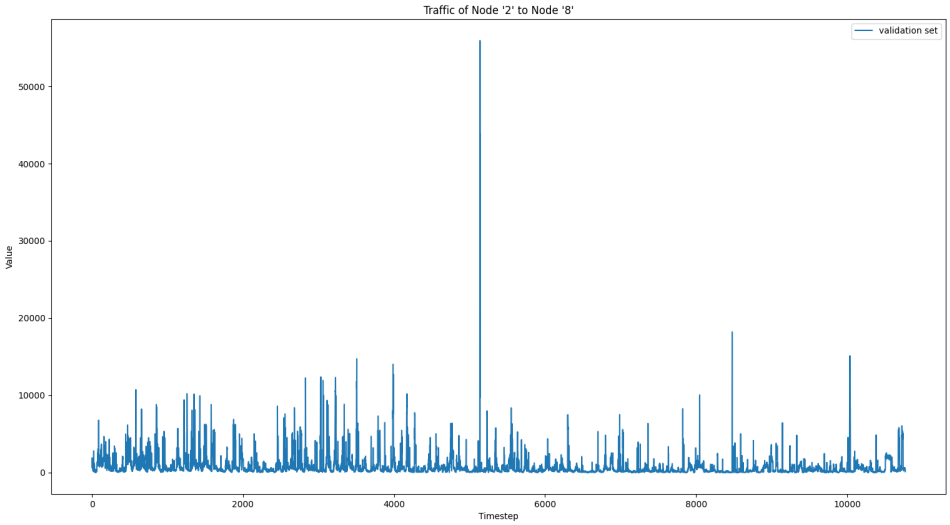


Εικόνα 4.3.18: RMSE - NMAE του μοντέλου SimpleRNN



Εικόνα 4.3.19: TRE - SRE του μοντέλου SimpleRNN

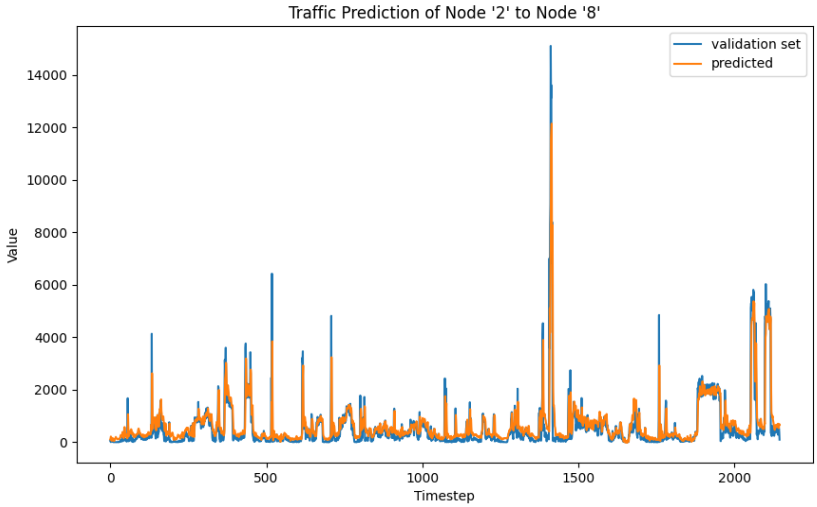
GEANT



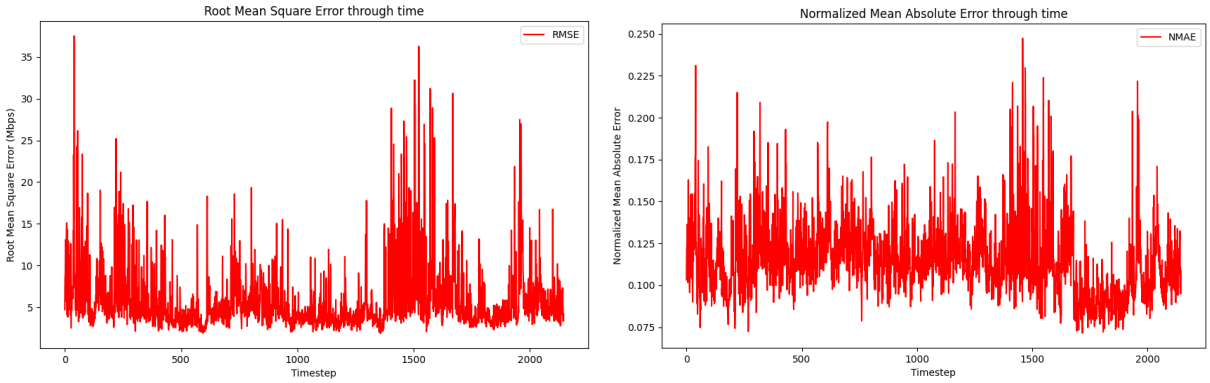
Εικόνα 4.3.20: Ένταση δικτυακής κίνησης του ζεύγους Node ‘2’ - Node ‘9’ του GEANT σε όλη την χρονική διάρκεια

Πίνακας 4.3.7: Σφάλματα Πρόβλεψης μοντέλου ConvLSTM

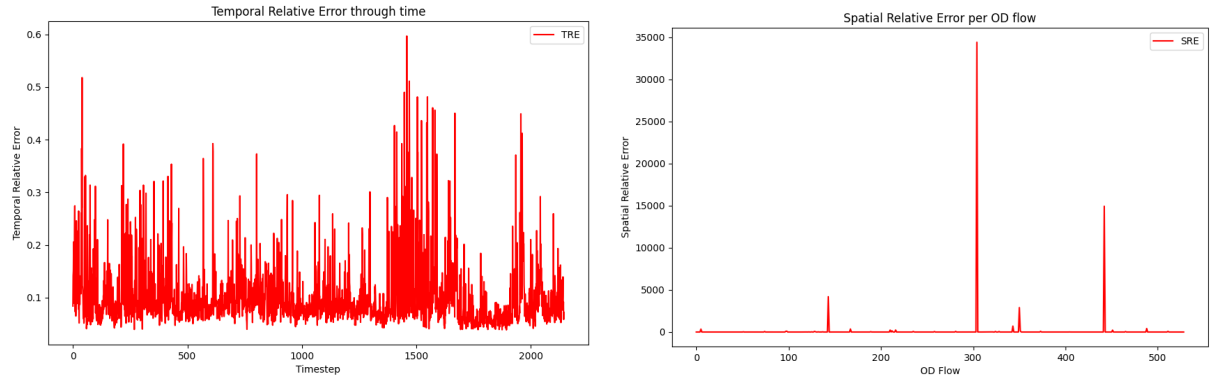
ConvLSTM				
Σφάλμα	Μέση Τιμή	Διάμεσος	Τυπική Απόκλιση	Μέγιστο
RMSE (Mbps)	5.9420	4.6108	4.0057	37.5260
NMAE	0.1159	0.1133	0.0224	0.2473
TRE	0.1049	0.0845	0.0642	0.5967
SRE	114.7740	0.6396	1641.8680	34393.1994



Εικόνα 4.3.21: Πρόβλεψη δικτυακής κίνησης του ζεύγους Node ‘2’ - Node ‘8’ με τη χρήση του μοντέλου ConvLSTM



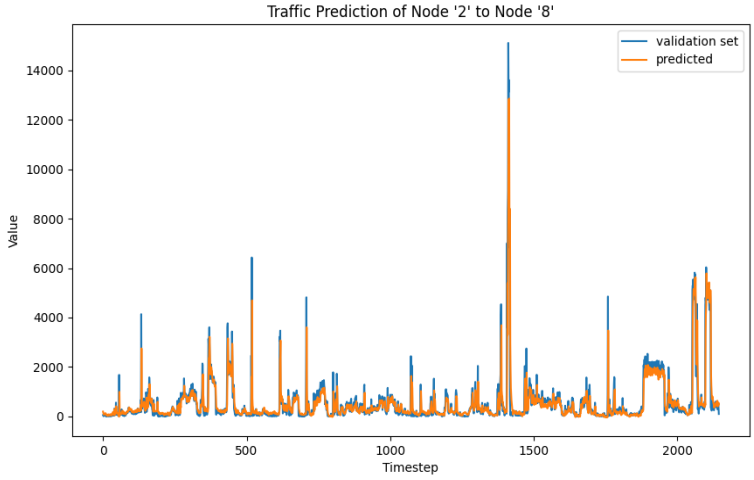
Εικόνα 4.3.22: RMSE - NMAE του μοντέλου ConvLSTM



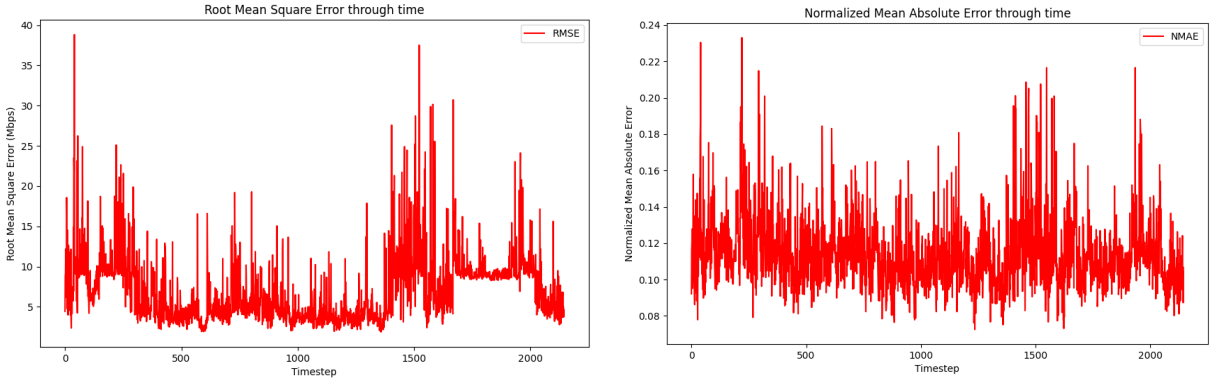
Εικόνα 4.3.23: TRE - SRE του μοντέλου ConvLSTM

Πίνακας 4.3.8: Σφάλματα Πρόβλεψης μοντέλου CNNLSTM

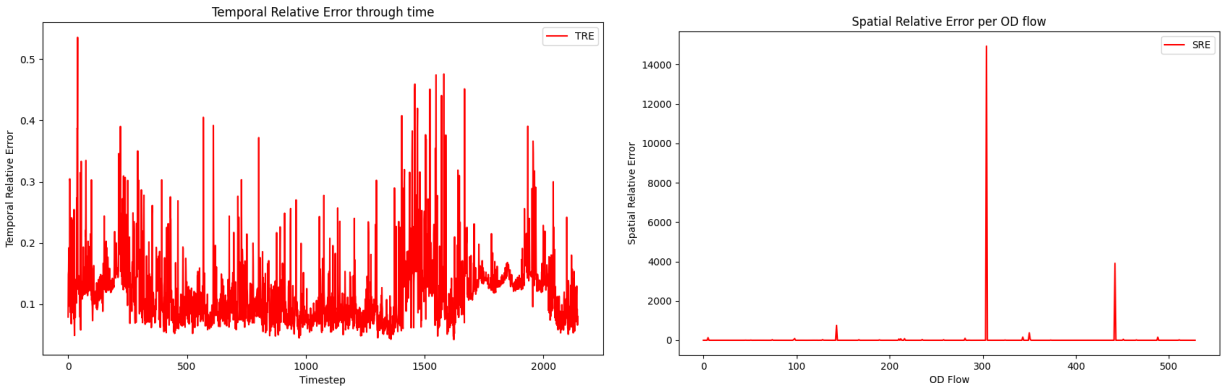
CNNLSTM				
Σφάλμα	Μέση Τιμή	Διάμεσος	Τυπική Απόκλιση	Μέγιστο
RMSE (Mbps)	7.1096	5.9935	4.0871	38.8220
NMAE	0.1143	0.1109	0.0198	0.2331
TRE	0.1222	0.1131	0.0591	0.5356
SRE	40.6926	0.6026	671.4452	14941.3431



Εικόνα 4.3.24: Πρόβλεψη δικτυακής κίνησης του ζεύγους Node ‘2’ - Node ‘8’ με τη χρήση του μοντέλου CNNLSTM



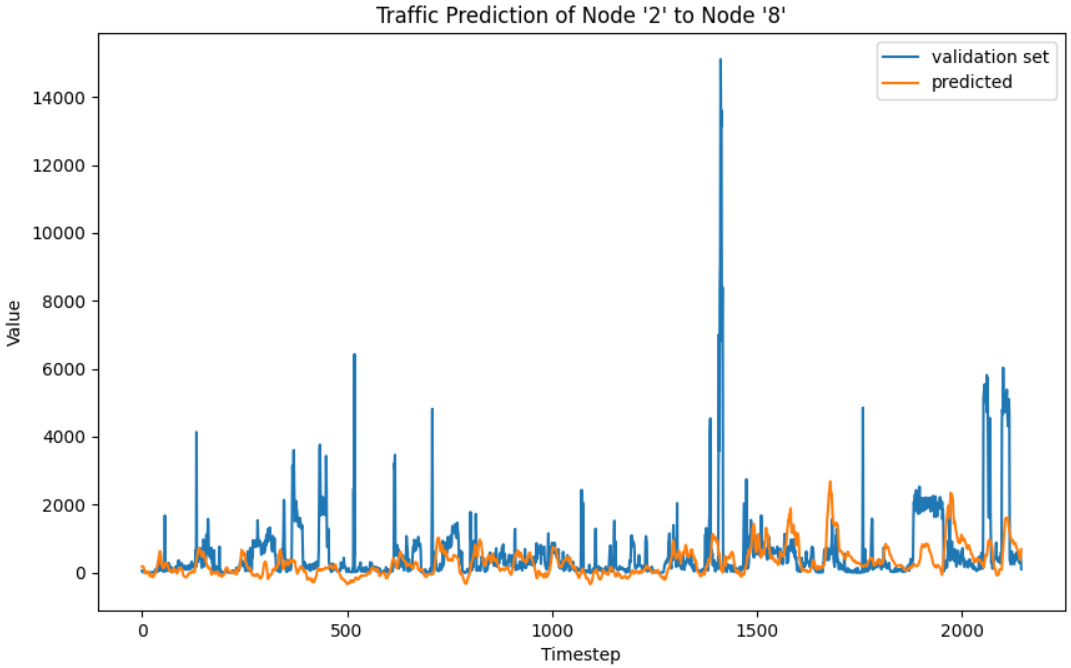
Εικόνα 4.3.25: RMSE - NMAE του μοντέλου CNNLSTM



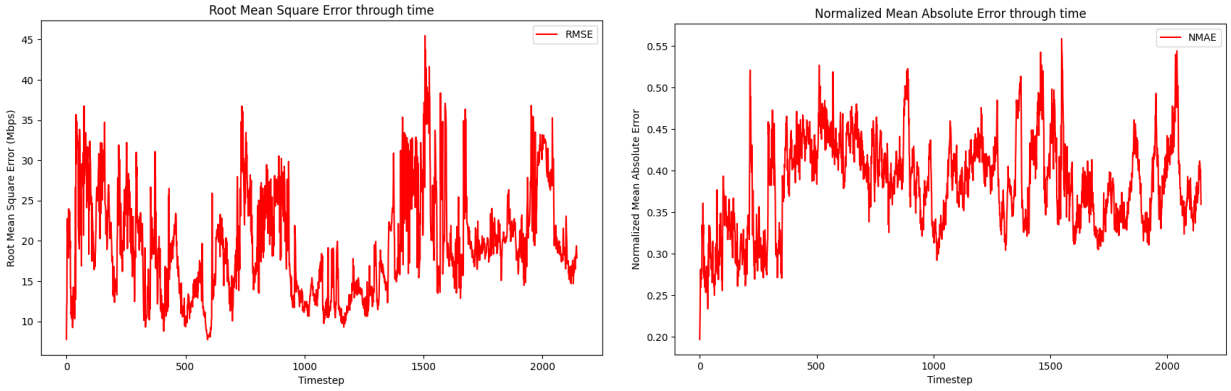
Εικόνα 4.3.26: TRE - SRE του μοντέλου CNNLSTM

Πίνακας 4.3.9: Σφάλματα Πρόβλεψης μοντέλου LSTM

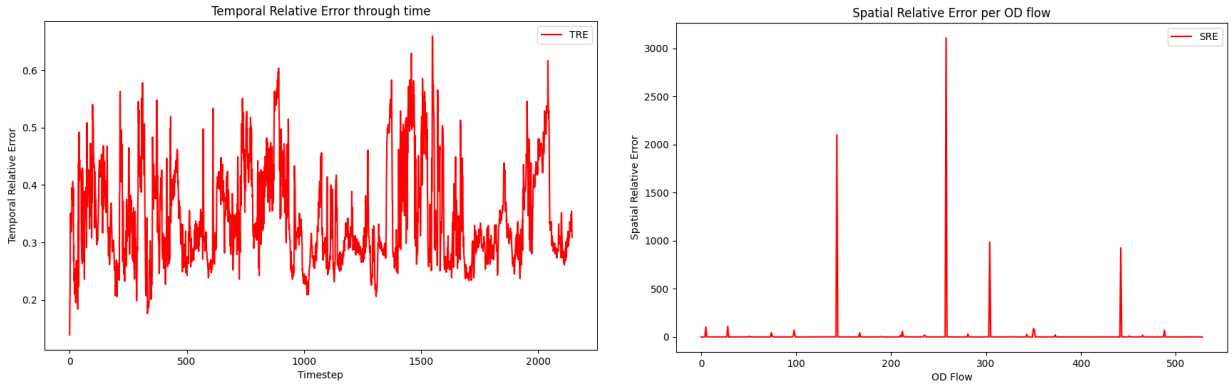
Stacked LSTM				
Σφάλμα	Μέση Τιμή	Διάμεσος	Τυπική Απόκλιση	Μέγιστο
RMSE (Mbps)	19.6061	18.8033	6.5421	45.4851
NMAE	0.3843	0.3864	0.0551	0.5588
TRE	0.3450	0.3189	0.0860	0.6600
SRE	15.9768	0.9497	173.0576	3110.0019



Εικόνα 4.3.27: Πρόβλεψη δικτυακής κίνησης του ζεύγους Node ‘2’ - Node ‘8’ με τη χρήση του μοντέλου LSTM



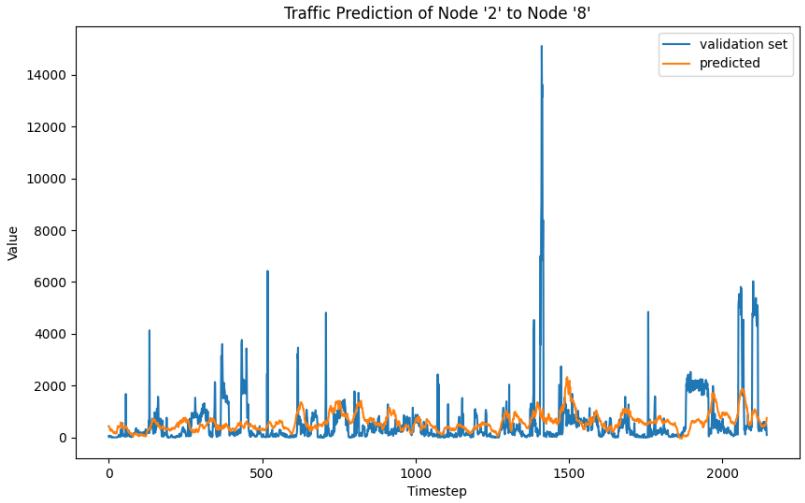
Εικόνα 4.3.28: RMSE - NMAE του μοντέλου LSTM



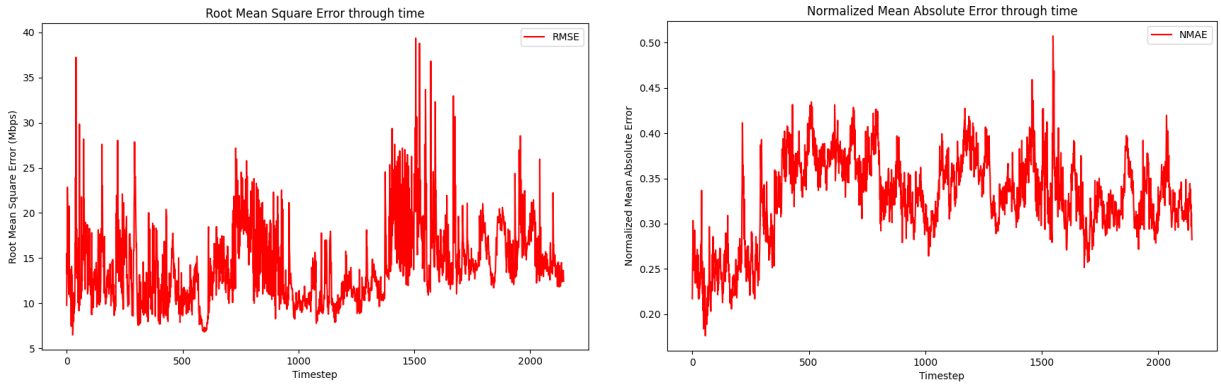
Εικόνα 4.3.29: TRE - SRE του μοντέλου LSTM

Πίνακας 4.3.10: Σφάλματα Πρόβλεψης μοντέλου GRU

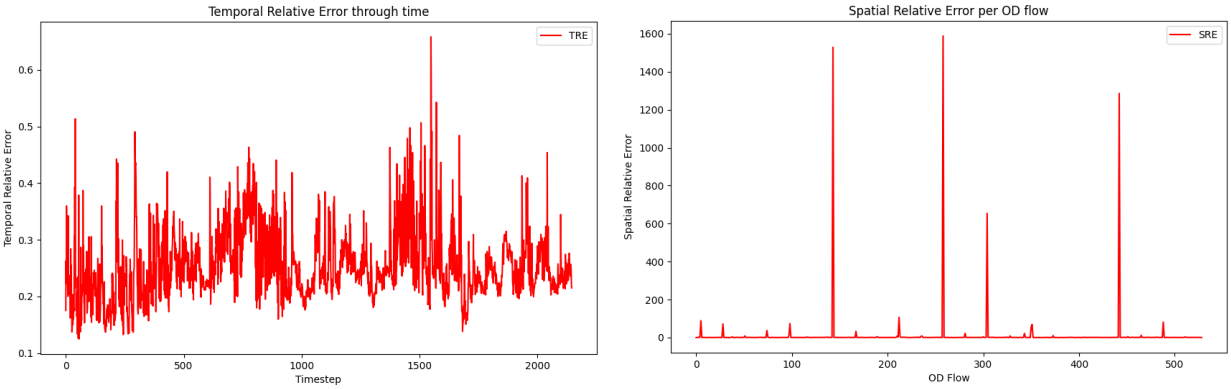
GRU				
Σφάλμα	Μέση Τιμή	Διάμεσος	Τυπική Απόκλιση	Μέγιστο
RMSE (Mbps)	14.2486	13.4644	4.2571	39.3464
NMAE	0.3291	0.3313	0.0470	0.5072
TRE	0.2535	0.2418	0.0602	0.6584
SRE	11.8624	0.9326	114.3384	1588.2242



Εικόνα 4.3.30: Πρόβλεψη δικτυακής κίνησης του ζεύγους Node ‘2’ - Node ‘8’ με τη χρήση του μοντέλου GRU



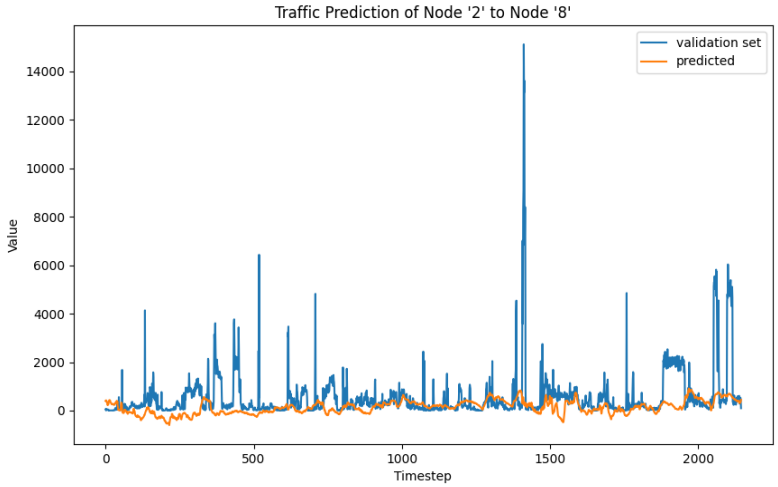
Εικόνα 4.3.31: RMSE - NMAE του μοντέλου GRU



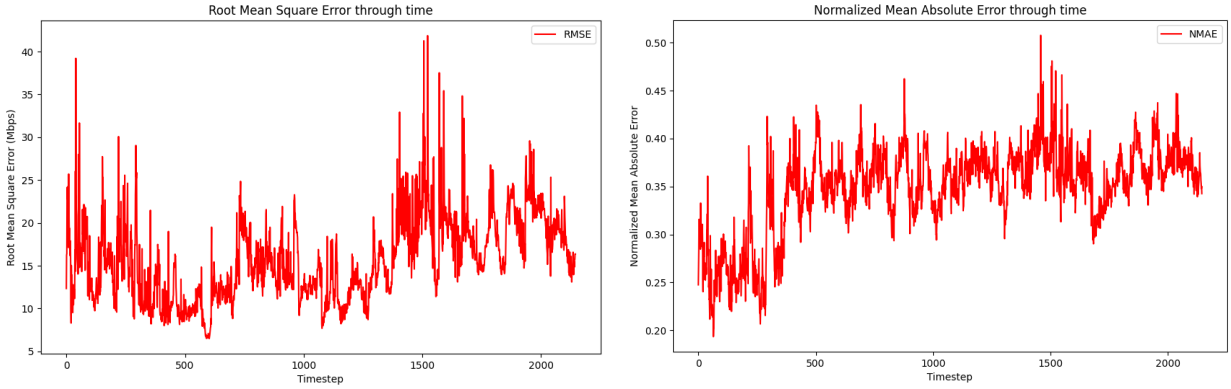
Εικόνα 4.3.32: TRE - SRE του μοντέλου GRU

Πίνακας 4.3.11: Σφάλματα Πρόβλεψης μοντέλου BiLSTM

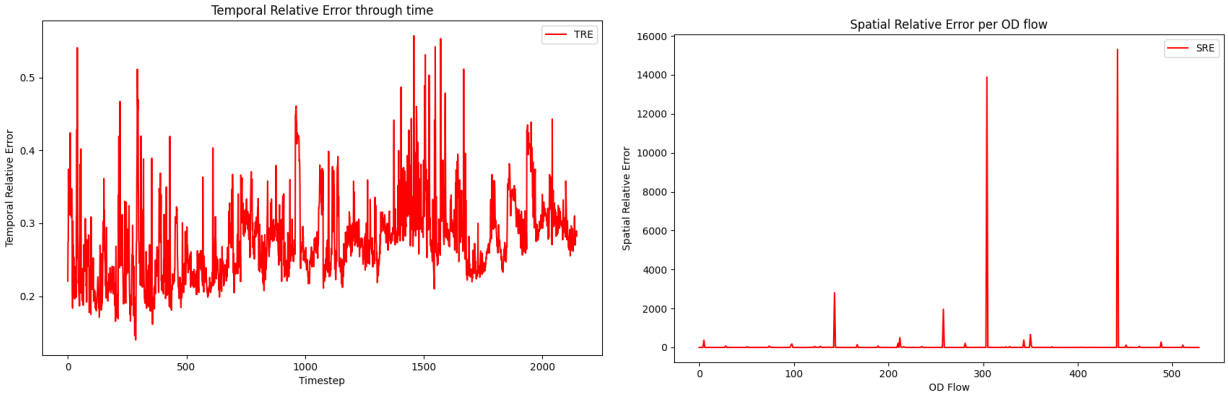
BiLSTM				
Σφάλμα	Μέση Τιμή	Διάμεσος	Τυπική Απόκλιση	Μέγιστο
RMSE (Mbps)	15.7076	15.1272	4.7803	41.8530
NMAE	0.3476	0.3564	0.0449	0.5076
TRE	0.2761	0.2711	0.0552	0.5574
SRE	73.1475	0.9486	909.2851	15309.8341



Εικόνα 4.3.33: Πρόβλεψη δικτυακής κίνησης του ζεύγους Node ‘2’ - Node ‘8’ με τη χρήση του μοντέλου BiLSTM



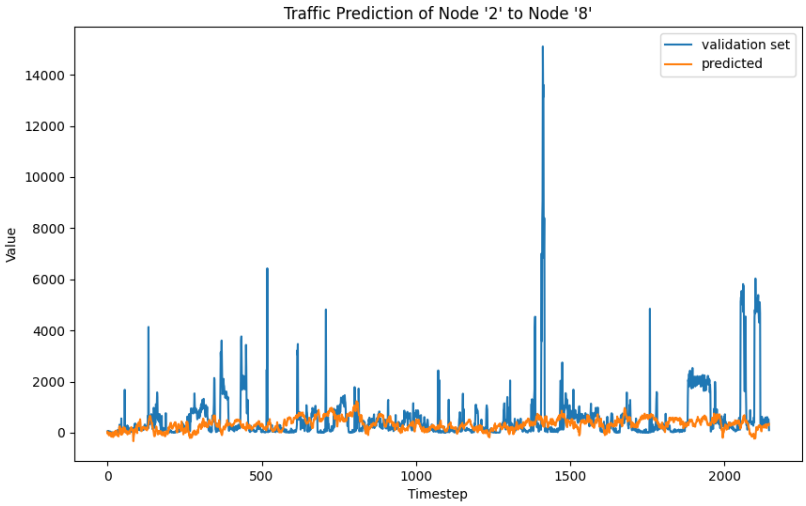
Εικόνα 4.3.34: RMSE - NMAE του μοντέλου BiLSTM



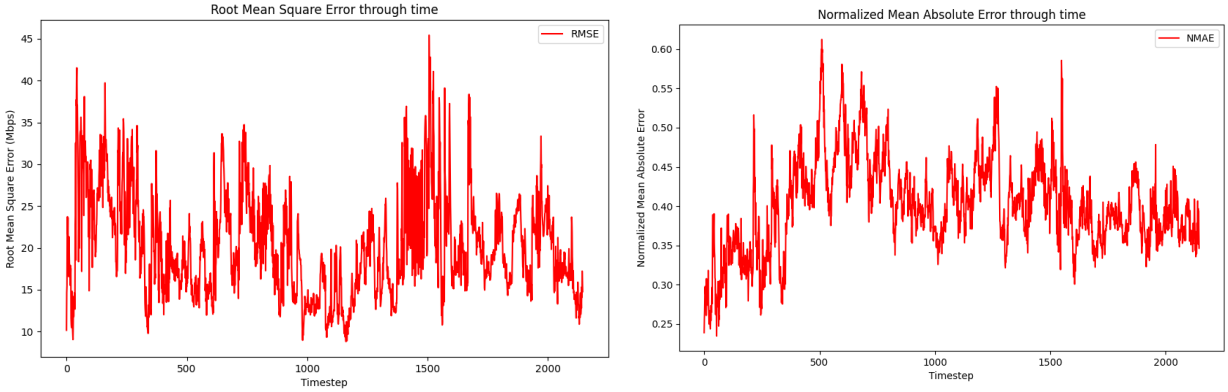
Εικόνα 4.3.35: TRE - SRE του μοντέλου BiLSTM

Πίνακας 4.3.12: Σφάλματα Πρόβλεψης μοντέλου SimpleRNN

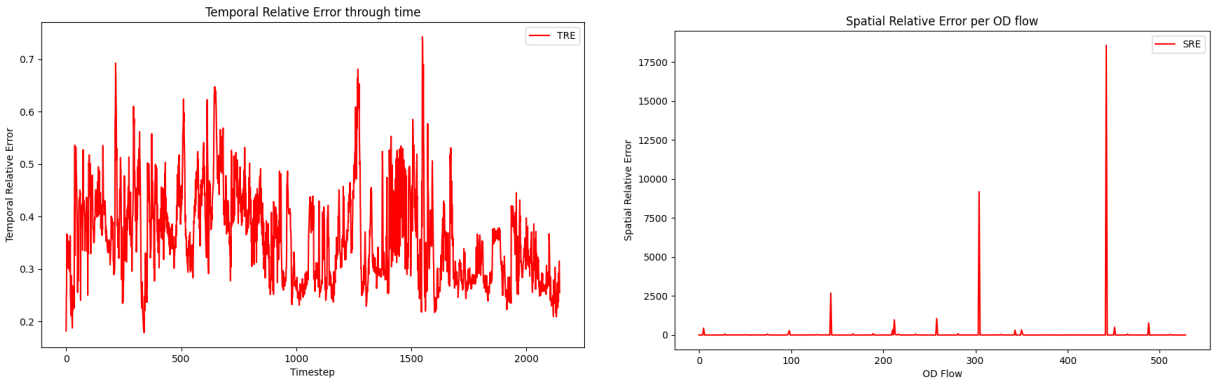
SimpleRNN				
Σφάλμα	Μέση Τιμή	Διάμεσος	Τυπική Απόκλιση	Μέγιστο
RMSE (Mbps)	20.3721	19.3559	6.1234	45.4355
NMAE	0.4018	0.3985	0.0581	0.6124
TRE	0.3624	0.3534	0.0885	0.7426
SRE	70.2050	0.9630	909.0213	18562.0507



Εικόνα 4.3.36: Πρόβλεψη δικτυακής κίνησης του ζεύγους Node ‘2’ - Node ‘8’ με τη χρήση του μοντέλου SimpleRNN



Εικόνα 4.3.37: RMSE - NMAE του μοντέλου SimpleRNN



Εικόνα 4.3.38: TRE - SRE του μοντέλου SimpleRNN

Πίνακας 4.3.13: Συνοπτικά αποτελέσματα όλων των μοντέλων

	Μέση Τιμή	ConvLSTM	CNNLSTM	Stacked LSTM	GRU	BiLSTM	SimpleRNN
Abilene	RMSE (Mbps)	10.4970	10.8410	14.0897	13.7867	14.4352	15.8247
	NMAE	0.1498	0.1352	0.2138	0.2092	0.2171	0.2295
	TRE	0.2051	0.1980	0.2506	0.2428	0.2468	0.2658
	SRE	0.7049	0.3416	0.5093	0.5041	0.6448	0.7763
GEANT	RMSE (Mbps)	5.9420	7.1096	19.6061	14.2486	15.7076	20.3721
	NMAE	0.1159	0.1143	0.3843	0.3291	0.3476	0.4018
	TRE	0.1049	0.1222	0.3450	0.2535	0.2761	0.3624
	SRE	114.7740	40.6926	15.9768	11.8624	73.1475	70.2050

Πίνακας 4.3.14: Χρόνος για το Training χωρίς την χρήση GPU

		ConvLSTM	CNNLSTM	Stacked LSTM	GRU	BiLSTM	SimpleRNN
Abilene	TIME (sec)	25600 50/100 epochs	8900	5500	4200	3900	1800
GEANT	TIME (sec)	29000 69/100 epochs	7600	1600	1250	1100	500

Σύγκριση και σχολιασμός μοντέλων

Παρατηρούμε ότι τα δύο μοντέλα που έχουν συνδυασμό Recurrent και Convolutional layers έχουν πολύ καλύτερα αποτελέσματα από τα μοντέλα που έχουν μόνο Recurrent layers. Συμπεραίνουμε δηλαδή ότι συνδυάζοντας τα δύο είδη επιπέδων τα μοντέλα αποκτούν τις ιδιότητες των δικτύων RNN και CNN, τα δεδομένα προσαρμόζονται στα μοντέλα και η εκπαίδευση είναι πιο αποδοτική. Αν προσέξουμε τις εικόνες 4.3.21 και 4.3.24 παρατηρούμε ότι η πρόβλεψη των δύο αυτών μοντέλων για την κίνηση ενός ζεύγους είναι πολύ ικανοποιητική. Επίσης, αυτά τα δύο μοντέλα χρειάζονται και τον περισσότερο χρόνο για εκπαίδευση λόγω της αυξημένης πολυπλοκότητας τους. Ο λόγος που τα συνδυαστικά μοντέλα έχουν καλύτερη απόδοση είναι επειδή στους Πίνακες Κίνησης των δικτύων υπάρχουν χωροχρονικές εξαρτήσεις. Οι χωρικές εξαρτήσεις των δεδομένων κίνησης προκύπτουν από τις σχέσεις που υπάρχουν μεταξύ των κόμβων, ενώ οι χρονικές εξαρτήσεις είναι οι σχέσεις και τα μοτίβα που επικρατούν σε διάφορα χρονικά διαστήματα. Οι χωροχρονικές σχέσεις σε ένα δίκτυο μπορούν να αναπτυχθούν λόγω πολλών παραγόντων. Αρχικά, είναι συχνό φαινόμενο στη δικτυακή κίνηση να εμφανίζονται επαναλαμβανόμενες συμπεριφορές από τα δεδομένα. Δημιουργούνται δηλαδή, χωρικά και χρονικά μοτίβα. Για παράδειγμα, τις ώρες εργασίας εμφανίζεται αυξημένη κίνηση σε περιοχές που υπάρχουν γραφεία, ενώ τις βραδινές ώρες υπάρχει αυξημένη κίνηση σε κατοικημένες περιοχές. Επίσης, σε ένα δίκτυο πανεπιστημίου παρατηρείται μεγάλη κίνηση κατά τις πρωινές ώρες, ενώ τα σαββατοκύριακα η κίνηση είναι ελάχιστη. Αυτά τα μοτίβα που παρατηρούνται στη δικτυακή κίνηση οδηγούν στις χωροχρονικές συσχετίσεις των πινάκων κίνησης. Άλλος ένας λόγος που παρατηρούνται χωρικές εξαρτήσεις είναι η τοπολογία του δικτύου. Οι συνδέσεις μεταξύ των κόμβων και η φυσική τοπολογία του δικτύου δημιουργούν μοτίβα στη δικτυακή κίνηση. Κόμβοι που είναι άμεσα συνδεδεμένοι ή είναι μέρος ενός υποδικτύου είναι αναμενόμενο να έχουν πιο ισχυρές σχέσεις και η κίνηση μεταξύ τους να είναι αυξημένη. Αντίθετα κόμβοι που έχουν πολλούς ενδιάμεσους θα έχουν μικρότερη εξάρτηση και λιγότερη επικοινωνία. Η τοπολογία του δικτύου είναι πολύ σημαντική στη δρομολόγηση του δικτύου και στην κατανομή των πόρων. Τέλος, οι διάφορες τεχνικές για εξισορρόπηση του φορτίου κατανέμουν την κίνηση σε διαθέσιμα μονοπάτια και έχουν στόχο στην αποφυγή της

συμφόρησης του δικτύου. Οι τεχνικές αυτές χρησιμοποιούν τους παροντικούς και παρελθοντικούς πίνακες κυκλοφορίας προκειμένου να δρομολογήσουν την κυκλοφορία. Οι χρονικές και χωρικές εξαρτήσεις των πινάκων κίνησης παίζουν καθοριστικό ρόλο στις τεχνικές εξισορρόπησης φορτίου και στη διαχείριση της χωρητικότητας. Επίσης, με την βοήθεια της μηχανικής μάθησης μοντέλα μπορούν να ανιχνεύσουν αυτές τις εξαρτήσεις και να επιτευχθούν πιο ακριβείς προβλέψεις δικτυακής κίνησης.

Παραπάνω αναλύσαμε τους λόγους που τα συνδυαστικά μοντέλα CNN-LSTM και ConvLSTM παρουσιάζουν τα καλύτερα αποτελέσματα. Στην παράγραφο αυτή θα επικεντρωθούμε στις διαφορές μεταξύ των δύο μοντέλων. Όπως αναφέραμε στο κεφάλαιο 4.1.1 το μοντέλο ConvLSTM είναι μια τροποποιημένη έκδοση του μοντέλου LSTM. Στην δομή του LSTM έχουν προστεθεί συνελικτικές λειτουργίες, καθώς έχει αντικατασταθεί η πράξη του πολλαπλασιασμού πινάκων με την πράξη της συνέλιξης σε κάθε πύλη του κελιού LSTM. Η κύρια ικανότητα του μοντέλου ConvLSTM είναι ότι απομνημονεύει χωροχρονικές πληροφορίες μεταξύ κόμβων του δικτύου ακόμα και σε πολύ μακρινές χρονικές στιγμές. Αυτό το καταφέρνει χωρίς την προσθήκη ξεχωριστού βήματος εξαγωγής χαρακτηριστικών. Αντίθετα, το μοντέλο CNN-LSTM αποτελεί συνδυασμό συνελικτικού και αναδρομικού δικτύου. Το μοντέλο αυτό εξάγει τα χωρικά χαρακτηριστικά όπως ένα μοντέλο CNN και μετά η πληροφορία μεταβιβάζεται σε ένα μοντέλο LSTM προκειμένου να ληφθεί υπόψη η συνολική χρονική εξέλιξη των δεδομένων και να ανιχνευθούν οι βραχυπρόθεσμες ή μακροπρόθεσμες χρονικές ακολουθίες. Το πλεονέκτημα αυτού του μοντέλου είναι ότι μπορεί να υποστηρίξει πολύ μεγάλες ακολουθίες εισόδου, που μπορούν να διαβαστούν ως υπο-ακολουθίες από το μοντέλο CNN και στη συνέχεια να συγκεντρωθούν από το μοντέλο LSTM.

Γενικά τα δυο αυτά μοντέλα παρουσιάζουν πολύ κοντινά αποτελέσματα. Βασιζόμενοι στο κυριότερο κριτήριο αξιολόγησης που είναι το RMSE μπορούμε να πούμε ότι το μοντέλο ConvLSTM παρουσιάζει λίγο καλύτερη αποτελεσματικότητα και στα δύο δίκτυα. Ωστόσο, αξιοσημείωτο είναι το αρκετά υψηλό χωρικό σφάλμα (SRE) που εμφανίζεται σε αυτό το μοντέλο. Παρόλα αυτά η επιλογή μοντέλου δεν γίνεται αποκλειστικά με βάση την αποτελεσματικότητα του. Πρέπει να λάβουμε υπόψη και τον χρόνο εκπαίδευσης του κάθε μοντέλου. Για παράδειγμα τα δύο συνδυαστικά μοντέλα επειδή είναι πιο πολύπλοκα δίνουν καλύτερες προβλέψεις αλλά έχουν αρκετά μεγαλύτερο χρόνο εκπαίδευσης. Σημαντική διαφορά έχουν και τα δύο μοντέλα αυτά μεταξύ τους. Το ConvLSTM απαιτεί περίπου τριπλάσιο χρόνο εκπαίδευσης από το CNN-LSTM, οπότε σε αρκετά προβλήματα ίσως συμφέρει η επιλογή του μοντέλου CNN-LSTM που έχει παρόμοια αποτελέσματα με το ConvLSTM, αλλά χρειάζεται πολύ λιγότερο χρόνο εκπαίδευσης. Αν αντιθέτως σε κάποιο πρόβλημα δεν μας απασχολεί καθόλου ο χρόνος εκπαίδευσης και επικεντρωνόμαστε μόνο στην ακρίβεια, το ConvLSTM είναι το κατάλληλο μοντέλο για πρόβλεψη της δικτυακής κίνησης.

Σχετικά με τα μοντέλα GRU, LSTM και BiLSTM μπορούμε να αναφέρουμε ότι παρουσιάζουν παρόμοια αποτελέσματα, με το GRU να είναι ελάχιστα πιο αποδοτικό, ενώ όπως ήταν αναμενόμενο το μοντέλο SimpleRNN παρουσιάζει τα χειρότερα αποτελέσματα. Αυτό

συμβαίνει γιατί το SimpleRNN είναι το πιο απλό μοντέλο για αυτό και αδυνατεί να ανιχνεύσει μακροχρόνιες εξαρτήσεις. Το LSTM αντίθετα έχει ικανότητα να εντοπίζει τις μακροχρόνιες εξαρτήσεις των ακολουθιών για αυτό και είναι κατάλληλο για πρόβλεψη χρονοσειρών. Το BiLSTM αποτελεί προέκταση του LSTM, όπου η πληροφορία χρησιμοποιείται και από τις δύο πλευρές της ακολουθίας για την εκτίμηση της εξόδου. Η τεχνική αυτή βασίζεται στην ιδέα ότι η έξοδος κάθε χρονική στιγμή εξαρτάται τόσο από τα προηγούμενα όσο και από τα επόμενα στοιχεία της ακολουθίας. Το GRU έχει απλούστερη αρχιτεκτονική από το LSTM, αλλά η απόδοση του είναι παρόμοια με το LSTM.

Τέλος, στο δίκτυο GEANT παρατηρούνται πολύ μεγάλες τιμές στα σφάλματα SRE σε όλα τα μοντέλα. Αυτό συμβαίνει επειδή υπάρχουν κάποιες τεράστιες τιμές σε συγκεκριμένα ζευγάρια κόμβων, όπως παρατηρούμε και στις εικόνες 4.3.23, 4.3.26, 4.3.29, 4.3.32, 4.3.35, 4.3.38. Όπως αναφέραμε και στο κεφάλαιο 4.2.1, στο δίκτυο GEANT υπάρχουν τιμές με 2-3 τάξεις μεγέθους μεγαλύτερες και για τον λόγο αυτό τοποθετήθηκε ένα κατώφλι τιμών 99.9%. Με τον τρόπο αυτό αντικαθίστανται οι ακραίες τιμές των δεδομένων με τις μέγιστες “φυσιολογικές” τιμές που είναι εντός ορίων. Υπήρχε η δυνατότητα να χαμηλώσουμε αρκετά το κατώφλι αυτό προκειμένου το σφάλμα SRE να μην παρουσιάζει ορισμένες ακραίες τιμές. Ωστόσο, θεωρήσαμε ότι το κατώφλι 99.9% είναι ικανοποιητικό, διότι με ένα μικρότερο κατώφλι αλλοιωνόταν αρκετά το σύνολο δεδομένων και τα υπόλοιπα σφάλματα δεν θα ανταποκρίνονταν στις πραγματικές μετρήσιμες τιμές.

5. Συμπεράσματα και Μελλοντικές Προεκτάσεις

Στην εργασία αυτή αναπτύξαμε 6 μοντέλα μηχανικής μάθησης και τα εφαρμόσαμε σε δύο σύνολα δεδομένων προκειμένου να προβλέψουμε τη μελλοντική Δικτυακή Κίνηση. Τα μοντέλα αυτά ήταν το ConvLSTM, το CNN-LSTM, το LSTM, το BiLSTM, το GRU και το SimpleRNN. Το κύριο συμπέρασμα που βγάλαμε είναι ότι τα συνδυαστικά μοντέλα ConvLSTM και CNN-LSTM παρουσιάζουν φανερά καλύτερα αποτελέσματα. Αυτό συμβαίνει διότι αυτά τα δύο μοντέλα μπορούν να ανιχνεύσουν χωροχρονικές εξαρτήσεις που επικρατούν στους πίνακες δικτυακής κίνησης. Οι χωρικές εξαρτήσεις των δεδομένων κίνησης προκύπτουν από τις σχέσεις που υπάρχουν μεταξύ των κόμβων, ενώ οι χρονικές εξαρτήσεις είναι τα μοτίβα που επικρατούν σε διάφορα χρονικά διαστήματα. Αντίθετα, τα υπόλοιπα μοντέλα ανιχνεύουν κυρίως χρονικές εξαρτήσεις με αποτέλεσμα να έχουν μεγαλύτερα σφάλματα στην πρόβλεψη της κίνησης. Ανάμεσα στα δύο καλύτερα μοντέλα που συνδυάζουν τα οφέλη των CNN και των RNN μοντέλων, το ConvLSTM είναι αυτό που φαίνεται να έχει πιο ακριβή πρόβλεψη. Ωστόσο, το CNN-LSTM έχει πολύ κοντινά αποτελέσματα και παρουσιάζει ελάχιστα μεγαλύτερο σφάλμα. Το πλεονέκτημα του μοντέλου αυτού είναι η μικρότερη πολυπλοκότητα του σε σχέση με το ConvLSTM. Το CNN-LSTM χρειάζεται πολύ λιγότερο χρόνο εκπαίδευσης, περίπου το ⅓, οπότε σε πολλά προβλήματα είναι καλύτερη η επιλογή αυτού του μοντέλου. Αντίθετα, σε προβλήματα που δεν υπάρχει περιορισμός στους πόρους και στον χρόνο εκπαίδευσης η καλύτερη επιλογή είναι το ConvLSTM.

Τα μοντέλα που έχουν ήδη αναφερθεί στην παρούσα εργασία παρουσιάζουν ικανοποιητικά αποτελέσματα στα προβλήματα πρόβλεψης δικτυακής κίνησης. Επίσης, όπως είδαμε στο κεφάλαιο 2.4, υπάρχουν αρκετές εργασίες που ανέπτυξαν παρόμοια μοντέλα με διαφορετικές παραμέτρους που είχαν ως στόχο την ελαχιστοποίηση του σφάλματος πρόβλεψης. Η μελλοντική έρευνα σε αυτόν τον τομέα θα μπορούσε να επικεντρωθεί στην ανάπτυξη αποδοτικών μοντέλων που έχουν ως στόχο την ελαχιστοποίηση των πόρων και του χρόνου εκπαίδευσης. Η ανίχνευση και επεξεργασία των χωροχρονικών εξαρτήσεων των δεδομένων είναι απαραίτητη προκειμένου να είναι αποτελεσματική η πρόβλεψη της μελλοντικής κίνησης. Για τον λόγο αυτό οι μελλοντικές έρευνες πρέπει να συνδυάσουν CNN και LSTM μοντέλα λαμβάνοντας υπόψη την αποτελεσματικότητα των μοντέλων σε συνδυασμό με την χρήση πόρων και τον χρόνο εκπαίδευσης. Τα συνελικτικά επίπεδα (CNN) εξάγουν τα σημαντικότερα χωρικά χαρακτηριστικών από τα δεδομένα και τα επίπεδα LSTM ανιχνεύουν τα μοτίβα και τις χρονικές ακολουθίες που επικρατούν στα δεδομένα. Ένα μοντέλο που μπορεί να προστεθεί σε αυτήν την τεχνική στην θέση των επιπέδων LSTM, προκειμένου να ελαχιστοποιηθούν οι πόροι και ο χρόνος εκπαίδευσης, είναι το LSTMP (LSTM with Projection). Το LSTMP είναι μια προέκταση του LSTM και χρησιμοποιεί ένα επίπεδο projection το οποίο μειώνει τις διαστάσεις των κρυφών στρωμάτων LSTM. Έτσι μειώνονται οι παράμετροι κάθε επιπέδου με αποτέλεσμα να μειώνεται η πολυπλοκότητα και να αυξάνεται η ταχύτητα εκπαίδευσης [124]. Αυτό επιτυγχάνεται χωρίς να μειώνεται σημαντικά η επίδοση του μοντέλου [125]. Με αυτή την τεχνική μπορεί να επιτευχθεί η ισορροπία ανάμεσα στην ανίχνευση των χωροχρονικών εξαρτήσεων και την βελτιστοποίηση

της χρήσης των πόρων. Μελλοντικά η επεξεργασία και δοκιμή αυτού του μοντέλου σε σύνολα δεδομένων μπορεί να επιφέρει ακριβή πρόβλεψη δικτυακής κίνησης με καλύτερη χρήση των πόρων και ελαχιστοποίηση του χρόνου εκπαίδευσης.

Παράρτημα Κώδικας Εργασίας

ConvLSTM

```
import matplotlib.pyplot as plt
import numpy as np
import tensorflow as tf
import statistics
import sys
from tensorflow import keras
from keras.models import Sequential
from keras.layers import ConvLSTM2D, Conv3D, Reshape
from keras.callbacks import EarlyStopping
from numpy import array
from sklearn.metrics import mean_squared_error

#Choose dataset Geant or Abilene
if len(sys.argv)==1:
   print('Choose dataset between GEANT or ABILENE')
   exit()
else:
   dataset = sys.argv[1]

# Load the data
if dataset=='GEANT':
   traffic_matrix=np.load('data/GEANT/X.npy')
elif dataset=='ABILENE':
   traffic_matrix=np.load('data/ABILENE/X.npy')
else:
   print('WRONG dataset! Choose between GEANT or ABILENE')
   exit()
data = traffic_matrix[:, :, :].copy()

# Parameters of our time series
SPLIT_TIME = int(len(data) * 0.8)
WINDOW_IN = 10
WINDOW_OUT = 1
NODES=data.shape[2]

# Normalize the data
if dataset==('GEANT'):
   data = np.clip(data, 0.0, np.percentile(data.flatten(), 99.9))#geant
elif dataset==('ABILENE'):
   data = np.clip(data, 0.0, np.percentile(data.flatten(),
99.99998))#abilene
data_split = data[:SPLIT_TIME]
max_list = np.max(data)
min_list = np.min(data)
data = (data - min_list) / (max_list - min_list)
data[np.isnan(data)] = 0  # fill the abnormal data with 0
```

```
data[np.isinf(data)] = 0

# define input sequence
# split a multivariate sequence into x=[samples, window_in, rows, columns]
y=[samples, window_out, rows, columns]
def split_sequences(sequences, WINDOW_IN, WINDOW_OUT):
   X, y = list(), list()
   for i in range(len(sequences)):
       # find the end of this pattern
       end_ix = i + WINDOW_IN
       out_end_ix = end_ix + WINDOW_OUT
       # check if we are beyond the dataset
       if out_end_ix > len(sequences):
           break
       # gather input and output parts of the pattern
       seq_x, seq_y = sequences[i:end_ix, :, :],
sequences[end_ix:out_end_ix, :, :]
       X.append(seq_x)
       y.append(seq_y)
   return array(X), array(y)

# convert into input/output all the samples
X, y = split_sequences(data, WINDOW_IN, WINDOW_OUT)

# reshape from [samples, window, rows, columns] into [samples, window,
rows, columns, 1]
X = X.reshape(X.shape[0], WINDOW_IN, NODES, NODES,1)
y = np.swapaxes(y, 1, 2)
y = np.swapaxes(y, 2, 3)
y = y.reshape(y.shape[0], NODES*NODES,1)

# Split into training data and test data
series_train_x = X[:SPLIT_TIME]
series_train_y = y[:SPLIT_TIME]
series_test_x = X[SPLIT_TIME:]
series_test_y = y[SPLIT_TIME:]

# Define model
model = Sequential()
model.add(ConvLSTM2D(filters=16,
   kernel_size=(5, 5),
   padding="same",
   return_sequences=True,
   input_shape=(WINDOW_IN,NODES, NODES,1)))
model.add(ConvLSTM2D(filters=16,
   kernel_size=(3, 3),
   padding="same",
   return_sequences=True))
model.add(ConvLSTM2D(filters=16,
   kernel_size=(1, 1),
```

```
    padding="same",
    return_sequences=True))
model.add(Conv3D(filters=1, kernel_size=(10,1,1)))
model.add(Reshape((NODES*NODES,1)))

model.compile(loss='mae', optimizer='adam', metrics=["accuracy"])
model.summary()
early_stopping = EarlyStopping(monitor='loss',patience = 10)

# fit model (data = batch_size*samples per epoch)
model.fit(series_train_x, series_train_y, epochs=100, batch_size=128,
verbose=1, callbacks=[early_stopping])

# Prediction on the whole series
val_forecast = model.predict(series_test_x, verbose=0)

series_test_y=series_test_y*(max_list - min_list) + min_list
val_forecast=val_forecast*(max_list - min_list) + min_list

# calculate RMSE, NMAE, TRE, SRE
series_test_y=series_test_y.reshape(series_test_y.shape[0],series_test_y.s
hape[1]*series_test_y.shape[2])
val_forecast=val_forecast.reshape(val_forecast.shape[0],val_forecast.shape
[1]*val_forecast.shape[2])
rmse=list()
nmae=list()
tre=list()
sre=list()

for t in range(series_test_y.shape[0]):
    rmse.append(np.sqrt(mean_squared_error(series_test_y[t,:],
val_forecast[t,:]))/1000)

nmae.append(sum(np.absolute(np.subtract(series_test_y[t,:],val_forecast[t,
:])))/sum(np.absolute(series_test_y[t,:])))

tre.append(np.sqrt(sum((np.subtract(series_test_y[t,:],val_forecast[t,:]))
**2))/np.sqrt(sum(series_test_y[t,:]**2)))

for i in range(series_test_y.shape[1]):
    if(np.sqrt(sum(series_test_y[:,i]**2))!=0):
        result =
np.sqrt(sum((np.subtract(series_test_y[:,i],val_forecast[:,i]))**2))/np.sq
rt(sum(series_test_y[:,i]**2))
        sre.append(result)
    elif(np.sqrt(sum(series_test_y[:,i]**2))==0):
        result = 0
        sre.append(result)

# Printing the mean
print('Mean RMSE Test Score: %.4f Mbps' % (np.mean(rmse)))
```

```
print('Median RMSE Test Score: %.4f Mbps' % (statistics.median(rmse)))
print('Std RMSE Test Score: %.4f Mbps' % (np.std(rmse)))
print('Max RMSE Test Score: %.4f Mbps' % (max(rmse)))

print('Mean NMAE Test Score: %.4f' % (np.mean(nmae)))
print('Median NMAE Test Score: %.4f' % (statistics.median(nmae)))
print('Std NMAE Test Score: %.4f' % (np.std(nmae)))
print('Max NMAE Test Score: %.4f' % (max(nmae)))

print('Mean TRE Test Score: %.4f' % (np.mean(tre)))
print('Median TRE Test Score: %.4f' % (statistics.median(tre)))
print('Std TRE Test Score: %.4f' % (np.std(tre)))
print('Max TRE Test Score: %.4f' % (max(tre)))

print('Mean SRE Test Score: %.4f' % (np.mean(sre)))
print('Median SRE Test Score: %.4f' % (statistics.median(sre)))
print('Std SRE Test Score: %.4f' % (np.std(sre)))
print('Max SRE Test Score: %.4f' % (max(sre)))

# Plot 1
plt.figure(1,figsize=(10, 6))
if dataset==('ABILENE'):
   plt.title("Traffic Prediction of Node2 'ATLAng' to Node9
'NYCMng'")#Abilene
   plt.plot(np.squeeze(series_test_y[:,20]), label="validation
set")#Abilene
   plt.plot(np.squeeze(val_forecast[:,20]), label="predicted")#Abilene
elif dataset==('GEANT'):
   plt.title("Traffic Prediction of Node '2' to Node '8'")#GEANT
   plt.plot(np.squeeze(series_test_y[:,30]), label="validation set")#GEANT
   plt.plot(np.squeeze(val_forecast[:,30]), label="predicted")#GEANT
plt.xlabel("Timestep")
plt.ylabel("Value")
plt.legend()
plt.show()

# Plot 2
plt.figure(2,figsize=(10, 6))
plt.title("Root Mean Square Error through time")
plt.plot(rmse, 'r', label="RMSE")
plt.xlabel("Timestep")
plt.ylabel("Root Mean Square Error (Mbps)")
plt.legend()
plt.show()

# Plot 3
plt.figure(3,figsize=(10, 6))
plt.title("Normalized Mean Absolute Error through time")
plt.plot(nmae,'r', label="NMAE")
plt.xlabel("Timestep")
```

```
plt.ylabel("Normalized Mean Absolute Error")
plt.legend()
plt.show()

# Plot 4
plt.figure(4,figsize=(10, 6))
plt.title("Temporal Relative Error through time")
plt.plot(tre, 'r', label="TRE")
plt.xlabel("Timestep")
plt.ylabel("Temporal Relative Error")
plt.legend()
plt.show()

# Plot 5
plt.figure(5,figsize=(10, 6))
plt.title("Spatial Relative Error per OD flow")
plt.plot(sre, 'r', label="SRE")
plt.xlabel("OD Flow")
plt.ylabel("Spatial Relative Error")
plt.legend()
plt.show()
```

CNN-LSTM

```
import matplotlib.pyplot as plt
import numpy as np
import tensorflow as tf
import statistics
import sys
from tensorflow import keras
from keras.models import Sequential
from keras.layers import LSTM, Reshape, Conv2D
from keras.callbacks import EarlyStopping
from numpy import array
from sklearn.metrics import mean_squared_error

#Choose dataset Geant or Abilene
if len(sys.argv)==1:
   print('Choose dataset between GEANT or ABILENE')
   exit()
else:
   dataset = sys.argv[1]

# Load the data
if dataset=='GEANT':
   traffic_matrix=np.load('data/GEANT/X.npy')
elif dataset=='ABILENE':
   traffic_matrix=np.load('data/ABILENE/X.npy')
else:
```

```
   print('WRONG dataset! Choose between GEANT or ABILENE')
   exit()
data = traffic_matrix[:, :, :].copy()

# Parameters of our time series
SPLIT_TIME = int(len(data) * 0.8)
WINDOW_IN = 10
WINDOW_OUT = 1
NODES=data.shape[2]

# Normalize the data
if dataset==('GEANT'):
   data = np.clip(data, 0.0, np.percentile(data.flatten(), 99.9))#geant
elif dataset==('ABILENE'):
   data = np.clip(data, 0.0, np.percentile(data.flatten(),
99.99998))#abilene
data_split = data[:SPLIT_TIME]
max_list = np.max(data)
min_list = np.min(data)
data = (data - min_list) / (max_list - min_list)
data[np.isnan(data)] = 0  # fill the abnormal data with 0
data[np.isinf(data)] = 0

# define input sequence
# split a multivariate sequence into x=[samples, window_in, rows, columns]
y=[samples, window_out, rows, columns]
def split_sequences(sequences, WINDOW_IN, WINDOW_OUT):
   X, y = list(), list()
   for i in range(len(sequences)):
       # find the end of this pattern
       end_ix = i + WINDOW_IN
       out_end_ix = end_ix + WINDOW_OUT
       # check if we are beyond the dataset
       if out_end_ix > len(sequences):
           break
       # gather input and output parts of the pattern
       seq_x, seq_y = sequences[i:end_ix, :, :],
sequences[end_ix:out_end_ix, :, :]
       X.append(seq_x)
       y.append(seq_y)
   return array(X), array(y)

# convert into input/output all the samples
X, y = split_sequences(data, WINDOW_IN, WINDOW_OUT)

# reshape from [samples, window, rows, columns] into [samples, rows,
columns, window]
X = X.reshape(X.shape[0],WINDOW_IN, NODES,NODES)
X = np.swapaxes(X, 1, 2)
X = np.swapaxes(X, 2, 3)
```

```
y = y.reshape(y.shape[0],WINDOW_OUT,NODES,NODES)
y = np.swapaxes(y, 1, 2)
y = np.swapaxes(y, 2, 3)
y = y.reshape(y.shape[0], NODES*NODES,1)

# Split into training data and test data
series_train_x = X[:SPLIT_TIME]
series_train_y = y[:SPLIT_TIME]
series_test_x = X[SPLIT_TIME:]
series_test_y = y[SPLIT_TIME:]

# Define model
model = Sequential()
model.add(Conv2D(filters=64, kernel_size=5, activation='tanh',
padding="same", input_shape=(NODES,NODES,WINDOW_IN)))
model.add(Conv2D(filters=64, kernel_size=3, activation='tanh',
padding="same"))
model.add(Conv2D(filters=64, kernel_size=1, activation='tanh'))
model.add(Reshape((NODES*NODES,64)))
model.add(LSTM(64, return_sequences = True))
model.add(LSTM(32, return_sequences = True))
model.add(LSTM(1, return_sequences = True))

model.compile(loss='mae', optimizer='adam', metrics=["accuracy"])
model.summary()
early_stopping = EarlyStopping(monitor='loss',patience = 10)

# fit model (data = batch_size*samples per epoch)
model.fit(series_train_x, series_train_y, epochs=100, batch_size=128,
verbose=1, callbacks=[early_stopping])

# Prediction on the test series
val_forecast = model.predict(series_test_x, verbose=0)

series_test_y=series_test_y*(max_list - min_list) + min_list
val_forecast=val_forecast*(max_list - min_list) + min_list

# calculate RMSE, NMAE, TRE, SRE
series_test_y=series_test_y.reshape(series_test_y.shape[0],series_test_y.s
hape[1]*series_test_y.shape[2])
val_forecast=val_forecast.reshape(val_forecast.shape[0],val_forecast.shape
[1]*val_forecast.shape[2])
rmse=list()
nmae=list()
tre=list()
sre=list()

for t in range(series_test_y.shape[0]):
   rmse.append(np.sqrt(mean_squared_error(series_test_y[t,:],
val_forecast[t,:]))/1000)
```

```
nmae.append(sum(np.absolute(np.subtract(series_test_y[t,:],val_forecast[t,
:])))/sum(np.absolute(series_test_y[t,:])))

tre.append(np.sqrt(sum((np.subtract(series_test_y[t,:],val_forecast[t,:]))
**2))/np.sqrt(sum(series_test_y[t,:]**2)))

for i in range(series_test_y.shape[1]):
   if(np.sqrt(sum(series_test_y[:,i]**2))!=0):
       result =
np.sqrt(sum((np.subtract(series_test_y[:,i],val_forecast[:,i]))**2))/np.sq
rt(sum(series_test_y[:,i]**2))
       sre.append(result)
   elif(np.sqrt(sum(series_test_y[:,i]**2))==0):
       result = 0
       sre.append(result)

# Printing the mean
print('Mean RMSE Test Score: %.4f Mbps' % (np.mean(rmse)))
print('Median RMSE Test Score: %.4f Mbps' % (statistics.median(rmse)))
print('Std RMSE Test Score: %.4f Mbps' % (np.std(rmse)))
print('Max RMSE Test Score: %.4f Mbps' % (max(rmse)))

print('Mean NMAE Test Score: %.4f' % (np.mean(nmae)))
print('Median NMAE Test Score: %.4f' % (statistics.median(nmae)))
print('Std NMAE Test Score: %.4f' % (np.std(nmae)))
print('Max NMAE Test Score: %.4f' % (max(nmae)))

print('Mean TRE Test Score: %.4f' % (np.mean(tre)))
print('Median TRE Test Score: %.4f' % (statistics.median(tre)))
print('Std TRE Test Score: %.4f' % (np.std(tre)))
print('Max TRE Test Score: %.4f' % (max(tre)))

print('Mean SRE Test Score: %.4f' % (np.mean(sre)))
print('Median SRE Test Score: %.4f' % (statistics.median(sre)))
print('Std SRE Test Score: %.4f' % (np.std(sre)))
print('Max SRE Test Score: %.4f' % (max(sre)))

# Plot 1
plt.figure(1,figsize=(10, 6))
if dataset==('ABILENE'):
   plt.title("Traffic Prediction of Node2 'ATLAng' to Node9
'NYCMng'")#Abilene
   plt.plot(np.squeeze(series_test_y[:,20]), label="validation
set")#Abilene
   plt.plot(np.squeeze(val_forecast[:,20]), label="predicted")#Abilene
elif dataset==('GEANT'):
   plt.title("Traffic Prediction of Node '2' to Node '8'")#GEANT
   plt.plot(np.squeeze(series_test_y[:,30]), label="validation set")#GEANT
   plt.plot(np.squeeze(val_forecast[:,30]), label="predicted")#GEANT
```

```
plt.xlabel("Timestep")
plt.ylabel("Value")
plt.legend()
plt.show()

# Plot 2
plt.figure(2,figsize=(10, 6))
plt.title("Root Mean Square Error through time")
plt.plot(rmse, 'r', label="RMSE")
plt.xlabel("Timestep")
plt.ylabel("Root Mean Square Error (Mbps)")
plt.legend()
plt.show()

# Plot 3
plt.figure(3,figsize=(10, 6))
plt.title("Normalized Mean Absolute Error through time")
plt.plot(nmae,'r', label="NMAE")
plt.xlabel("Timestep")
plt.ylabel("Normalized Mean Absolute Error")
plt.legend()
plt.show()

# Plot 4
plt.figure(4,figsize=(10, 6))
plt.title("Temporal Relative Error through time")
plt.plot(tre, 'r', label="TRE")
plt.xlabel("Timestep")
plt.ylabel("Temporal Relative Error")
plt.legend()
plt.show()

# Plot 5
plt.figure(5,figsize=(10, 6))
plt.title("Spatial Relative Error per OD flow")
plt.plot(sre, 'r', label="SRE")
plt.xlabel("OD Flow")
plt.ylabel("Spatial Relative Error")
plt.legend()
plt.show()
```

LSTM

```
import matplotlib.pyplot as plt
import numpy as np
import tensorflow as tf
import statistics
import sys
from tensorflow import keras
```

```
from keras.models import Sequential
from keras.layers import LSTM, MaxPooling1D, Dropout
from keras.callbacks import EarlyStopping
from numpy import array
from sklearn.metrics import mean_squared_error

#Choose dataset Geant or Abilene
if len(sys.argv)==1:
   print('Choose dataset between GEANT or ABILENE')
   exit()
else:
   dataset = sys.argv[1]

# Load the data
if dataset=='GEANT':
   traffic_matrix=np.load('data/GEANT/X.npy')
elif dataset=='ABILENE':
   traffic_matrix=np.load('data/ABILENE/X.npy')
else:
   print('WRONG dataset! Choose between GEANT or ABILENE')
   exit()
data = traffic_matrix[:, :, :].copy()

# Parameters of our time series
SPLIT_TIME = int(len(data) * 0.8)
WINDOW_IN = 10
WINDOW_OUT = 1
NODES=data.shape[2]

# Normalize the data
if dataset==('GEANT'):
   data = np.clip(data, 0.0, np.percentile(data.flatten(), 99.9))#geant
elif dataset==('ABILENE'):
   data = np.clip(data, 0.0, np.percentile(data.flatten(),
99.99998))#abilene
data_split = data[:SPLIT_TIME]
max_list = np.max(data)
min_list = np.min(data)
data = (data - min_list) / (max_list - min_list)
data[np.isnan(data)] = 0  # fill the abnormal data with 0
data[np.isinf(data)] = 0

# define input sequence
# split a multivariate sequence into x=[samples, window_in, rows, columns]
y=[samples, window_out, rows, columns]
def split_sequences(sequences, WINDOW_IN, WINDOW_OUT):
   X, y = list(), list()
   for i in range(len(sequences)):
       # find the end of this pattern
       end_ix = i + WINDOW_IN
```

```
        out_end_ix = end_ix + WINDOW_OUT
        # check if we are beyond the dataset
        if out_end_ix > len(sequences):
            break
        # gather input and output parts of the pattern
        seq_x, seq_y = sequences[i:end_ix, :, :],
sequences[end_ix:out_end_ix, :, :]
        X.append(seq_x)
        y.append(seq_y)
    return array(X), array(y)

# convert into input/output all the samples
X, y = split_sequences(data, WINDOW_IN, WINDOW_OUT)

# reshape from [samples, window, rows, columns] into [samples, window,
rows*columns]
X = X.reshape(X.shape[0], WINDOW_IN,NODES*NODES)
y = y.reshape(y.shape[0], WINDOW_OUT,NODES*NODES)

# Split into training data and test data
series_train_x = X[:SPLIT_TIME]
series_train_y = y[:SPLIT_TIME]
series_test_x = X[SPLIT_TIME:]
series_test_y = y[SPLIT_TIME:]

# Define model
model = Sequential()
model.add(LSTM(512, return_sequences = True))
model.add(Dropout(0.2))
model.add(MaxPooling1D(pool_size=2))
model.add(LSTM(256, return_sequences = True))
model.add(Dropout(0.2))
model.add(MaxPooling1D(pool_size=5))
model.add(LSTM(NODES*NODES, return_sequences = True))

model.compile(loss='mae', optimizer='adam', metrics=["accuracy"])
early_stopping = EarlyStopping(monitor='loss',patience = 10)

# fit model (data = batch_size*samples per epoch)
model.fit(series_train_x, series_train_y, epochs=100, batch_size=128,
verbose=1, callbacks=[early_stopping])
model.summary()

# Prediction on the test series
val_forecast = model.predict(series_test_x, verbose=0)

# Rescale to original values
series_test_y=series_test_y*(max_list - min_list) + min_list
val_forecast=val_forecast*(max_list - min_list) + min_list
```

```
# calculate RMSE, NMAE, TRE, SRE
series_test_y=series_test_y.reshape(series_test_y.shape[0],series_test_y.s
hape[1]*series_test_y.shape[2])
val_forecast=val_forecast.reshape(val_forecast.shape[0],val_forecast.shape
[1]*val_forecast.shape[2])
rmse=list()
nmae=list()
tre=list()
sre=list()

for t in range(series_test_y.shape[0]):
   rmse.append(np.sqrt(mean_squared_error(series_test_y[t,:],
val_forecast[t,:]))/1000)

nmae.append(sum(np.absolute(np.subtract(series_test_y[t,:],val_forecast[t,
:])))/sum(np.absolute(series_test_y[t,:])))

tre.append(np.sqrt(sum((np.subtract(series_test_y[t,:],val_forecast[t,:]))
**2))/np.sqrt(sum(series_test_y[t,:]**2)))

for i in range(series_test_y.shape[1]):
   if(np.sqrt(sum(series_test_y[:,i]**2))!=0):
       result =
np.sqrt(sum((np.subtract(series_test_y[:,i],val_forecast[:,i]))**2))/np.sq
rt(sum(series_test_y[:,i]**2))
       sre.append(result)
   elif(np.sqrt(sum(series_test_y[:,i]**2))==0):
       result = 0
       sre.append(result)

# Printing the mean
print('Mean RMSE Test Score: %.4f Mbps' % (np.mean(rmse)))
print('Median RMSE Test Score: %.4f Mbps' % (statistics.median(rmse)))
print('Std RMSE Test Score: %.4f Mbps' % (np.std(rmse)))
print('Max RMSE Test Score: %.4f Mbps' % (max(rmse)))

print('Mean NMAE Test Score: %.4f' % (np.mean(nmae)))
print('Median NMAE Test Score: %.4f' % (statistics.median(nmae)))
print('Std NMAE Test Score: %.4f' % (np.std(nmae)))
print('Max NMAE Test Score: %.4f' % (max(nmae)))

print('Mean TRE Test Score: %.4f' % (np.mean(tre)))
print('Median TRE Test Score: %.4f' % (statistics.median(tre)))
print('Std TRE Test Score: %.4f' % (np.std(tre)))
print('Max TRE Test Score: %.4f' % (max(tre)))

print('Mean SRE Test Score: %.4f' % (np.mean(sre)))
print('Median SRE Test Score: %.4f' % (statistics.median(sre)))
print('Std SRE Test Score: %.4f' % (np.std(sre)))
print('Max SRE Test Score: %.4f' % (max(sre)))
```

```
# Plot 1
plt.figure(1,figsize=(10, 6))
if dataset==('ABILENE'):
   plt.title("Traffic Prediction of Node2 'ATLAng' to Node9
'NYCMng'")#Abilene
   plt.plot(np.squeeze(series_test_y[:,20]), label="validation
set")#Abilene
   plt.plot(np.squeeze(val_forecast[:,20]), label="predicted")#Abilene
elif dataset==('GEANT'):
   plt.title("Traffic Prediction of Node '2' to Node '8'")#GEANT
   plt.plot(np.squeeze(series_test_y[:,30]), label="validation set")#GEANT
   plt.plot(np.squeeze(val_forecast[:,30]), label="predicted")#GEANT
plt.xlabel("Timestep")
plt.ylabel("Value")
plt.legend()
plt.show()

# Plot 2
plt.figure(2,figsize=(10, 6))
plt.title("Root Mean Square Error through time")
plt.plot(rmse, 'r', label="RMSE")
plt.xlabel("Timestep")
plt.ylabel("Root Mean Square Error (Mbps)")
plt.legend()
plt.show()

# Plot 3
plt.figure(3,figsize=(10, 6))
plt.title("Normalized Mean Absolute Error through time")
plt.plot(nmae,'r', label="NMAE")
plt.xlabel("Timestep")
plt.ylabel("Normalized Mean Absolute Error")
plt.legend()
plt.show()

# Plot 4
plt.figure(4,figsize=(10, 6))
plt.title("Temporal Relative Error through time")
plt.plot(tre, 'r', label="TRE")
plt.xlabel("Timestep")
plt.ylabel("Temporal Relative Error")
plt.legend()
plt.show()

# Plot 5
plt.figure(5,figsize=(10, 6))
plt.title("Spatial Relative Error per OD flow")
plt.plot(sre, 'r', label="SRE")
plt.xlabel("OD Flow")
plt.ylabel("Spatial Relative Error")
```

```
plt.legend()
plt.show()
```

BiLSTM

```
import matplotlib.pyplot as plt
import numpy as np
import tensorflow as tf
import statistics
import sys
from tensorflow import keras
from keras.models import Sequential
from keras.layers import LSTM, MaxPooling1D, Bidirectional, Dense, Dropout
from keras.callbacks import EarlyStopping
from numpy import array
from sklearn.metrics import mean_squared_error

#Choose dataset Geant or Abilene
if len(sys.argv)==1:
   print('Choose dataset between GEANT or ABILENE')
   exit()
else:
   dataset = sys.argv[1]

# Load the data
if dataset=='GEANT':
   traffic_matrix=np.load('data/GEANT/X.npy')
elif dataset=='ABILENE':
   traffic_matrix=np.load('data/ABILENE/X.npy')
else:
   print('WRONG dataset! Choose between GEANT or ABILENE')
   exit()
data = traffic_matrix[:, :, :].copy()

# Parameters of our time series
SPLIT_TIME = int(len(data) * 0.8)
WINDOW_IN = 10
WINDOW_OUT = 1
NODES=data.shape[2]

# Normalize the data
if dataset==('GEANT'):
   data = np.clip(data, 0.0, np.percentile(data.flatten(), 99.9))#geant
elif dataset==('ABILENE'):
   data = np.clip(data, 0.0, np.percentile(data.flatten(),
99.99998))#abilene
data_split = data[:SPLIT_TIME]
max_list = np.max(data)
min_list = np.min(data)
```

```
data = (data - min_list) / (max_list - min_list)
data[np.isnan(data)] = 0  # fill the abnormal data with 0
data[np.isinf(data)] = 0

# define input sequence
# split a multivariate sequence into x=[samples, window_in, rows, columns]
y=[samples, window_out, rows, columns]
def split_sequences(sequences, WINDOW_IN, WINDOW_OUT):
   X, y = list(), list()
   for i in range(len(sequences)):
       # find the end of this pattern
       end_ix = i + WINDOW_IN
       out_end_ix = end_ix + WINDOW_OUT
       # check if we are beyond the dataset
       if out_end_ix > len(sequences):
           break
       # gather input and output parts of the pattern
       seq_x, seq_y = sequences[i:end_ix, :, :],
sequences[end_ix:out_end_ix, :, :]
       X.append(seq_x)
       y.append(seq_y)
   return array(X), array(y)

# convert into input/output all the samples
X, y = split_sequences(data, WINDOW_IN, WINDOW_OUT)

# reshape from [samples, window, rows, columns] into [samples, window,
rows*columns]
X = X.reshape(X.shape[0], WINDOW_IN,NODES*NODES)
y = y.reshape(y.shape[0], WINDOW_OUT,NODES*NODES)

# Split into training data and test data
series_train_x = X[:SPLIT_TIME]
series_train_y = y[:SPLIT_TIME]
series_test_x = X[SPLIT_TIME:]
series_test_y = y[SPLIT_TIME:]

# Define model
model = Sequential()
model.add(Bidirectional(LSTM(256, return_sequences = True)))
model.add(Dropout(0.2))
model.add(MaxPooling1D(pool_size=2))
model.add(Bidirectional(LSTM(128, return_sequences = True)))
model.add(Dropout(0.2))
model.add(MaxPooling1D(pool_size=5))
model.add(Bidirectional(LSTM(106, return_sequences = True)))
model.add(Dense(NODES*NODES))

model.compile(loss='mae', optimizer='adam', metrics=["accuracy"])
```

```
early_stopping = EarlyStopping(monitor='loss',patience = 10)

# fit model (data = batch_size*samples per epoch)
model.fit(series_train_x, series_train_y, epochs=100, batch_size=128,
verbose=1, callbacks=[early_stopping])
model.summary()

# Prediction on the test series
val_forecast = model.predict(series_test_x, verbose=0)

# Rescale to original values
series_test_y=series_test_y*(max_list - min_list) + min_list
val_forecast=val_forecast*(max_list - min_list) + min_list

# calculate RMSE, NMAE, TRE, SRE
series_test_y=series_test_y.reshape(series_test_y.shape[0],series_test_y.s
hape[1]*series_test_y.shape[2])
val_forecast=val_forecast.reshape(val_forecast.shape[0],val_forecast.shape
[1]*val_forecast.shape[2])
rmse=list()
nmae=list()
tre=list()
sre=list()

for t in range(series_test_y.shape[0]):
   rmse.append(np.sqrt(mean_squared_error(series_test_y[t,:],
val_forecast[t,:]))/1000)

nmae.append(sum(np.absolute(np.subtract(series_test_y[t,:],val_forecast[t,
:])))/sum(np.absolute(series_test_y[t,:])))

tre.append(np.sqrt(sum((np.subtract(series_test_y[t,:],val_forecast[t,:]))
**2))/np.sqrt(sum(series_test_y[t,:]**2)))

for i in range(series_test_y.shape[1]):
   if(np.sqrt(sum(series_test_y[:,i]**2))!=0):
       result =
np.sqrt(sum((np.subtract(series_test_y[:,i],val_forecast[:,i]))**2))/np.sq
rt(sum(series_test_y[:,i]**2))
       sre.append(result)
   elif(np.sqrt(sum(series_test_y[:,i]**2))==0):
       result = 0
       sre.append(result)

# Printing the mean
print('Mean RMSE Test Score: %.4f Mbps' % (np.mean(rmse)))
print('Median RMSE Test Score: %.4f Mbps' % (statistics.median(rmse)))
print('Std RMSE Test Score: %.4f Mbps' % (np.std(rmse)))
print('Max RMSE Test Score: %.4f Mbps' % (max(rmse)))

print('Mean NMAE Test Score: %.4f' % (np.mean(nmae)))
```

```
print('Median NMAE Test Score: %.4f' % (statistics.median(nmae)))
print('Std NMAE Test Score: %.4f' % (np.std(nmae)))
print('Max NMAE Test Score: %.4f' % (max(nmae)))

print('Mean TRE Test Score: %.4f' % (np.mean(tre)))
print('Median TRE Test Score: %.4f' % (statistics.median(tre)))
print('Std TRE Test Score: %.4f' % (np.std(tre)))
print('Max TRE Test Score: %.4f' % (max(tre)))

print('Mean SRE Test Score: %.4f' % (np.mean(sre)))
print('Median SRE Test Score: %.4f' % (statistics.median(sre)))
print('Std SRE Test Score: %.4f' % (np.std(sre)))
print('Max SRE Test Score: %.4f' % (max(sre)))

# Plot 1
plt.figure(1,figsize=(10, 6))
if dataset==('ABILENE'):
   plt.title("Traffic Prediction of Node2 'ATLAng' to Node9
'NYCMng'")#Abilene
   plt.plot(np.squeeze(series_test_y[:,20]), label="validation
set")#Abilene
   plt.plot(np.squeeze(val_forecast[:,20]), label="predicted")#Abilene
elif dataset==('GEANT'):
   plt.title("Traffic Prediction of Node '2' to Node '8'")#GEANT
   plt.plot(np.squeeze(series_test_y[:,30]), label="validation set")#GEANT
   plt.plot(np.squeeze(val_forecast[:,30]), label="predicted")#GEANT
plt.xlabel("Timestep")
plt.ylabel("Value")
plt.legend()
plt.show()

# Plot 2
plt.figure(2,figsize=(10, 6))
plt.title("Root Mean Square Error through time")
plt.plot(rmse, 'r', label="RMSE")
plt.xlabel("Timestep")
plt.ylabel("Root Mean Square Error (Mbps)")
plt.legend()
plt.show()

# Plot 3
plt.figure(3,figsize=(10, 6))
plt.title("Normalized Mean Absolute Error through time")
plt.plot(nmae,'r', label="NMAE")
plt.xlabel("Timestep")
plt.ylabel("Normalized Mean Absolute Error")
plt.legend()
plt.show()

# Plot 4
```

```
plt.figure(4,figsize=(10, 6))
plt.title("Temporal Relative Error through time")
plt.plot(tre, 'r', label="TRE")
plt.xlabel("Timestep")
plt.ylabel("Temporal Relative Error")
plt.legend()
plt.show()

# Plot 5
plt.figure(5,figsize=(10, 6))
plt.title("Spatial Relative Error per OD flow")
plt.plot(sre, 'r', label="SRE")
plt.xlabel("OD Flow")
plt.ylabel("Spatial Relative Error")
plt.legend()
plt.show()
```

GRU

```
import matplotlib.pyplot as plt
import numpy as np
import tensorflow as tf
import statistics
import sys
from tensorflow import keras
from keras.models import Sequential
from keras.layers import LSTM, MaxPooling1D, GRU, Dropout
from keras.callbacks import EarlyStopping
from numpy import array
from sklearn.metrics import mean_squared_error

#Choose dataset Geant or Abilene
if len(sys.argv)==1:
   print('Choose dataset between GEANT or ABILENE')
   exit()
else:
   dataset = sys.argv[1]

# Load the data
if dataset=='GEANT':
   traffic_matrix=np.load('data/GEANT/X.npy')
elif dataset=='ABILENE':
   traffic_matrix=np.load('data/ABILENE/X.npy')
else:
   print('WRONG dataset! Choose between GEANT or ABILENE')
   exit()
data = traffic_matrix[:, :, :].copy()

# Parameters of our time series
```

```
SPLIT_TIME = int(len(data) * 0.8)
WINDOW_IN = 10
WINDOW_OUT = 1
NODES=data.shape[2]

# Normalize the data
if dataset==('GEANT'):
   data = np.clip(data, 0.0, np.percentile(data.flatten(), 99.9))#geant
elif dataset==('ABILENE'):
   data = np.clip(data, 0.0, np.percentile(data.flatten(),
99.99998))#abilene
data_split = data[:SPLIT_TIME]
max_list = np.max(data)
min_list = np.min(data)
data = (data - min_list) / (max_list - min_list)
data[np.isnan(data)] = 0  # fill the abnormal data with 0
data[np.isinf(data)] = 0

# define input sequence
# split a multivariate sequence into x=[samples, window_in, rows, columns]
y=[samples, window_out, rows, columns]
def split_sequences(sequences, WINDOW_IN, WINDOW_OUT):
   X, y = list(), list()
   for i in range(len(sequences)):
       # find the end of this pattern
       end_ix = i + WINDOW_IN
       out_end_ix = end_ix + WINDOW_OUT
       # check if we are beyond the dataset
       if out_end_ix > len(sequences):
           break
       # gather input and output parts of the pattern
       seq_x, seq_y = sequences[i:end_ix, :, :],
sequences[end_ix:out_end_ix, :, :]
       X.append(seq_x)
       y.append(seq_y)
   return array(X), array(y)

# convert into input/output all the samples
X, y = split_sequences(data, WINDOW_IN, WINDOW_OUT)

# reshape from [samples, window, rows, columns] into [samples, window,
rows*columns]
X = X.reshape(X.shape[0], WINDOW_IN,NODES*NODES)
y = y.reshape(y.shape[0], WINDOW_OUT,NODES*NODES)

# Split into training data and test data
series_train_x = X[:SPLIT_TIME]
series_train_y = y[:SPLIT_TIME]
series_test_x = X[SPLIT_TIME:]
series_test_y = y[SPLIT_TIME:]
```

```
# Define model
model = Sequential()
model.add(GRU(512, return_sequences = True))
model.add(Dropout(0.2))
model.add(MaxPooling1D(pool_size=2))
model.add(GRU(256, return_sequences = True))
model.add(Dropout(0.2))
model.add(MaxPooling1D(pool_size=5))
model.add(GRU(NODES*NODES, return_sequences = True))

model.compile(loss='mae', optimizer='adam', metrics=["accuracy"])

early_stopping = EarlyStopping(monitor='loss',patience = 10)

# fit model (data = batch_size*samples per epoch)
model.fit(series_train_x, series_train_y, epochs=100, batch_size=128,
verbose=1, callbacks=[early_stopping])
model.summary()

# Prediction on the test series
val_forecast = model.predict(series_test_x, verbose=0)

# Rescale to original values
series_test_y=series_test_y*(max_list - min_list) + min_list
val_forecast=val_forecast*(max_list - min_list) + min_list

# calculate RMSE, NMAE, TRE, SRE
series_test_y=series_test_y.reshape(series_test_y.shape[0],series_test_y.s
hape[1]*series_test_y.shape[2])
val_forecast=val_forecast.reshape(val_forecast.shape[0],val_forecast.shape
[1]*val_forecast.shape[2])
rmse=list()
nmae=list()
tre=list()
sre=list()

for t in range(series_test_y.shape[0]):
   rmse.append(np.sqrt(mean_squared_error(series_test_y[t,:],
val_forecast[t,:]))/1000)

nmae.append(sum(np.absolute(np.subtract(series_test_y[t,:],val_forecast[t,
:])))/sum(np.absolute(series_test_y[t,:])))

tre.append(np.sqrt(sum((np.subtract(series_test_y[t,:],val_forecast[t,:]))
**2))/np.sqrt(sum(series_test_y[t,:]**2)))

for i in range(series_test_y.shape[1]):
   if(np.sqrt(sum(series_test_y[:,i]**2))!=0):
```

```
        result =
np.sqrt(sum((np.subtract(series_test_y[:,i],val_forecast[:,i]))**2))/np.sq
rt(sum(series_test_y[:,i]**2))
        sre.append(result)
    elif(np.sqrt(sum(series_test_y[:,i]**2))==0):
        result = 0
        sre.append(result)

# Printing the mean
print('Mean RMSE Test Score: %.4f Mbps' % (np.mean(rmse)))
print('Median RMSE Test Score: %.4f Mbps' % (statistics.median(rmse)))
print('Std RMSE Test Score: %.4f Mbps' % (np.std(rmse)))
print('Max RMSE Test Score: %.4f Mbps' % (max(rmse)))

print('Mean NMAE Test Score: %.4f' % (np.mean(nmae)))
print('Median NMAE Test Score: %.4f' % (statistics.median(nmae)))
print('Std NMAE Test Score: %.4f' % (np.std(nmae)))
print('Max NMAE Test Score: %.4f' % (max(nmae)))

print('Mean TRE Test Score: %.4f' % (np.mean(tre)))
print('Median TRE Test Score: %.4f' % (statistics.median(tre)))
print('Std TRE Test Score: %.4f' % (np.std(tre)))
print('Max TRE Test Score: %.4f' % (max(tre)))

print('Mean SRE Test Score: %.4f' % (np.mean(sre)))
print('Median SRE Test Score: %.4f' % (statistics.median(sre)))
print('Std SRE Test Score: %.4f' % (np.std(sre)))
print('Max SRE Test Score: %.4f' % (max(sre)))

# Plot 1
plt.figure(1,figsize=(10, 6))
if dataset==('ABILENE'):
    plt.title("Traffic Prediction of Node2 'ATLAng' to Node9
'NYCMng'")#Abilene
    plt.plot(np.squeeze(series_test_y[:,20]), label="validation
set")#Abilene
    plt.plot(np.squeeze(val_forecast[:,20]), label="predicted")#Abilene
elif dataset==('GEANT'):
    plt.title("Traffic Prediction of Node '2' to Node '8'")#GEANT
    plt.plot(np.squeeze(series_test_y[:,30]), label="validation set")#GEANT
    plt.plot(np.squeeze(val_forecast[:,30]), label="predicted")#GEANT
plt.xlabel("Timestep")
plt.ylabel("Value")
plt.legend()
plt.show()

# Plot 2
plt.figure(2,figsize=(10, 6))
plt.title("Root Mean Square Error through time")
plt.plot(rmse, 'r', label="RMSE")
```

```
plt.xlabel("Timestep")
plt.ylabel("Root Mean Square Error (Mbps)")
plt.legend()
plt.show()

# Plot 3
plt.figure(3,figsize=(10, 6))
plt.title("Normalized Mean Absolute Error through time")
plt.plot(nmae,'r', label="NMAE")
plt.xlabel("Timestep")
plt.ylabel("Normalized Mean Absolute Error")
plt.legend()
plt.show()

# Plot 4
plt.figure(4,figsize=(10, 6))
plt.title("Temporal Relative Error through time")
plt.plot(tre, 'r', label="TRE")
plt.xlabel("Timestep")
plt.ylabel("Temporal Relative Error")
plt.legend()
plt.show()

# Plot 5
plt.figure(5,figsize=(10, 6))
plt.title("Spatial Relative Error per OD flow")
plt.plot(sre, 'r', label="SRE")
plt.xlabel("OD Flow")
plt.ylabel("Spatial Relative Error")
plt.legend()
plt.show()
```

SimpleRNN

```
import matplotlib.pyplot as plt
import numpy as np
import tensorflow as tf
import statistics
import sys
from tensorflow import keras
from keras.models import Sequential
from keras.layers import MaxPooling1D, SimpleRNN,Dropout
from keras.callbacks import EarlyStopping
from numpy import array
from sklearn.metrics import mean_squared_error

#Choose dataset Geant or Abilene
if len(sys.argv)==1:
   print('Choose dataset between GEANT or ABILENE')
```

```
    exit()
else:
    dataset = sys.argv[1]

# Load the data
if dataset=='GEANT':
    traffic_matrix=np.load('data/GEANT/X.npy')
elif dataset=='ABILENE':
    traffic_matrix=np.load('data/ABILENE/X.npy')
else:
    print('WRONG dataset! Choose between GEANT or ABILENE')
    exit()
data = traffic_matrix[:, :, :].copy()

# Parameters of our time series
SPLIT_TIME = int(len(data) * 0.8)
WINDOW_IN = 10
WINDOW_OUT = 1
NODES=data.shape[2]

# Normalize the data
if dataset==('GEANT'):
    data = np.clip(data, 0.0, np.percentile(data.flatten(), 99.9))#geant
elif dataset==('ABILENE'):
    data = np.clip(data, 0.0, np.percentile(data.flatten(),
99.99998))#abilene
data_split = data[:SPLIT_TIME]
max_list = np.max(data)
min_list = np.min(data)
data = (data - min_list) / (max_list - min_list)
data[np.isnan(data)] = 0  # fill the abnormal data with 0
data[np.isinf(data)] = 0

# define input sequence
# split a multivariate sequence into x=[samples, window_in, rows, columns]
y=[samples, window_out, rows, columns]
def split_sequences(sequences, WINDOW_IN, WINDOW_OUT):
    X, y = list(), list()
    for i in range(len(sequences)):
        # find the end of this pattern
        end_ix = i + WINDOW_IN
        out_end_ix = end_ix + WINDOW_OUT
        # check if we are beyond the dataset
        if out_end_ix > len(sequences):
            break
        # gather input and output parts of the pattern
        seq_x, seq_y = sequences[i:end_ix, :, :],
sequences[end_ix:out_end_ix, :, :]
        X.append(seq_x)
        y.append(seq_y)
```

```
   return array(X), array(y)

# convert into input/output all the samples
X, y = split_sequences(data, WINDOW_IN, WINDOW_OUT)

# reshape from [samples, window, rows, columns] into [samples, window,
rows*columns]
X = X.reshape(X.shape[0], WINDOW_IN,NODES*NODES)
y = y.reshape(y.shape[0], WINDOW_OUT,NODES*NODES)

# Split into training data and test data
series_train_x = X[:SPLIT_TIME]
series_train_y = y[:SPLIT_TIME]
series_test_x = X[SPLIT_TIME:]
series_test_y = y[SPLIT_TIME:]

# Define model
model = Sequential()
model.add(SimpleRNN(512, return_sequences = True))
model.add(Dropout(0.2))
model.add(MaxPooling1D(pool_size=2))
model.add(SimpleRNN(256, return_sequences = True))
model.add(Dropout(0.2))
model.add(MaxPooling1D(pool_size=5))
model.add(SimpleRNN(NODES*NODES, return_sequences = True))

model.compile(loss='mae', optimizer='adam', metrics=["accuracy"])

early_stopping = EarlyStopping(monitor='loss',patience = 10)

# fit model (data = batch_size*samples per epoch)
model.fit(series_train_x, series_train_y, epochs=100, batch_size=128,
verbose=1, callbacks=[early_stopping])
model.summary()

# Prediction on the test series
val_forecast = model.predict(series_test_x, verbose=0)

# Rescale to original values
series_test_y=series_test_y*(max_list - min_list) + min_list
val_forecast=val_forecast*(max_list - min_list) + min_list

# calculate RMSE, NMAE, TRE, SRE
series_test_y=series_test_y.reshape(series_test_y.shape[0],series_test_y.s
hape[1]*series_test_y.shape[2])
val_forecast=val_forecast.reshape(val_forecast.shape[0],val_forecast.shape
[1]*val_forecast.shape[2])
rmse=list()
nmae=list()
tre=list()
```

```
sre=list()

for t in range(series_test_y.shape[0]):
   rmse.append(np.sqrt(mean_squared_error(series_test_y[t,:],
val_forecast[t,:]))/1000)

nmae.append(sum(np.absolute(np.subtract(series_test_y[t,:],val_forecast[t,
:])))/sum(np.absolute(series_test_y[t,:])))

tre.append(np.sqrt(sum((np.subtract(series_test_y[t,:],val_forecast[t,:]))
**2))/np.sqrt(sum(series_test_y[t,:]**2)))

for i in range(series_test_y.shape[1]):
   if(np.sqrt(sum(series_test_y[:,i]**2))!=0):
       result =
np.sqrt(sum((np.subtract(series_test_y[:,i],val_forecast[:,i]))**2))/np.sq
rt(sum(series_test_y[:,i]**2))
       sre.append(result)
   elif(np.sqrt(sum(series_test_y[:,i]**2))==0):
       result = 0
       sre.append(result)

# Printing the mean
print('Mean RMSE Test Score: %.4f Mbps' % (np.mean(rmse)))
print('Median RMSE Test Score: %.4f Mbps' % (statistics.median(rmse)))
print('Std RMSE Test Score: %.4f Mbps' % (np.std(rmse)))
print('Max RMSE Test Score: %.4f Mbps' % (max(rmse)))

print('Mean NMAE Test Score: %.4f' % (np.mean(nmae)))
print('Median NMAE Test Score: %.4f' % (statistics.median(nmae)))
print('Std NMAE Test Score: %.4f' % (np.std(nmae)))
print('Max NMAE Test Score: %.4f' % (max(nmae)))

print('Mean TRE Test Score: %.4f' % (np.mean(tre)))
print('Median TRE Test Score: %.4f' % (statistics.median(tre)))
print('Std TRE Test Score: %.4f' % (np.std(tre)))
print('Max TRE Test Score: %.4f' % (max(tre)))

print('Mean SRE Test Score: %.4f' % (np.mean(sre)))
print('Median SRE Test Score: %.4f' % (statistics.median(sre)))
print('Std SRE Test Score: %.4f' % (np.std(sre)))
print('Max SRE Test Score: %.4f' % (max(sre)))

# Plot 1
plt.figure(1,figsize=(10, 6))
if dataset==('ABILENE'):
   plt.title("Traffic Prediction of Node2 'ATLAng' to Node9
'NYCMng'")#Abilene
   plt.plot(np.squeeze(series_test_y[:,20]), label="validation
set")#Abilene
   plt.plot(np.squeeze(val_forecast[:,20]), label="predicted")#Abilene
```

```
elif dataset==('GEANT'):
   plt.title("Traffic Prediction of Node '2' to Node '8'")#GEANT
   plt.plot(np.squeeze(series_test_y[:,30]), label="validation set")#GEANT
   plt.plot(np.squeeze(val_forecast[:,30]), label="predicted")#GEANT
plt.xlabel("Timestep")
plt.ylabel("Value")
plt.legend()
plt.show()

# Plot 2
plt.figure(2,figsize=(10, 6))
plt.title("Root Mean Square Error through time")
plt.plot(rmse, 'r', label="RMSE")
plt.xlabel("Timestep")
plt.ylabel("Root Mean Square Error (Mbps)")
plt.legend()
plt.show()

# Plot 3
plt.figure(3,figsize=(10, 6))
plt.title("Normalized Mean Absolute Error through time")
plt.plot(nmae,'r', label="NMAE")
plt.xlabel("Timestep")
plt.ylabel("Normalized Mean Absolute Error")
plt.legend()
plt.show()

# Plot 4
plt.figure(4,figsize=(10, 6))
plt.title("Temporal Relative Error through time")
plt.plot(tre, 'r', label="TRE")
plt.xlabel("Timestep")
plt.ylabel("Temporal Relative Error")
plt.legend()
plt.show()

# Plot 5
plt.figure(5,figsize=(10, 6))
plt.title("Spatial Relative Error per OD flow")
plt.plot(sre, 'r', label="SRE")
plt.xlabel("OD Flow")
plt.ylabel("Spatial Relative Error")
plt.legend()
plt.show()
```

Κατάλογος Εικόνων

Κατάλογος Πινάκων

Βιβλιογραφικές Αναφορές

[1] S. Uhlig, B. Quoitin, J. Lepropre, and S. Balon, “Providing public intradomain traffic matrices to the research community,” *ACM SIGCOMM Comput. Commun. Rev.*, vol. 36, no. 1, pp. 83–86, Jan. 2006, doi: 10.1145/1111322.1111341.

[2] “Index of /~yzhang/research/AbileneTM.” https://www.cs.utexas.edu/~yzhang/research/AbileneTM/ (accessed Mar. 29, 2023).

[3] A. Medina, N. Taft, K. Salamatian, S. Bhattacharyya, and C. Diot, “Traffic matrix estimation: existing techniques and new directions,” *ACM SIGCOMM Comput. Commun. Rev.*, vol. 32, no. 4, pp. 161–174, Aug. 2002, doi: 10.1145/964725.633041.

[4] A. Tootoonchian, M. Ghobadi, and Y. Ganjali, “OpenTM: Traffic Matrix Estimator for OpenFlow Networks,” in *Passive and Active Measurement*, A. Krishnamurthy and B. Plattner, Eds., in Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2010, pp. 201–210. doi: 10.1007/978-3-642-12334-4_21.

[5] G. Kakkavas, M. Kalntis, V. Karyotis, and S. Papavassiliou, “Future Network Traffic Matrix Synthesis and Estimation Based on Deep Generative Models,” in *2021 International Conference on Computer Communications and Networks (ICCCN)*, Jul. 2021, pp. 1–8. doi: 10.1109/ICCCN52240.2021.9522222.

[6] P. Tune and M. Roughan, “Internet Traffic Matrices: A Primer,” 2013. Accessed: Jan. 25, 2023. [Online]. Available: https://www.semanticscholar.org/paper/Internet-Traffic-Matrices%3A-A-Primer-Tune-Roughan/62bf61f513d388211f99b3a9ad2c9f76bfd5886b

[7] L. Nie, H. Wang, S. Gong, Z. Ning, M. S. Obaidat, and K.-F. Hsiao, “Anomaly Detection Based on Spatio-Temporal and Sparse Features of Network Traffic in VANETs,” in *2019 IEEE Global Communications Conference (GLOBECOM)*, Dec. 2019, pp. 1–6. doi: 10.1109/GLOBECOM38437.2019.9013915.

[8] “Social Network Analysis Prof. Dr. Daning Hu Department of Informatics University of Zurich Mar 5th, ppt download.” https://slideplayer.com/slide/8171533/ (accessed May 23, 2023).

[9] “Constructing an O/D Matrix | The Geography of Transport Systems,” Jan. 19, 2018. https://transportgeography.org/contents/methods/spatial-interactions-gravity-model/od-matrix-construction/ (accessed May 23, 2023).

[10] Μ. Καλντής and M. Kalntis, “Μελέτη και υλοποίηση εργαλείων μηχανικής μάθησης στην τομογραφία δικτύων υπολογιστών,” Jun. 2021, doi: 10.26240/heal.ntua.21238.

[11] A. Soule *et al.*, “Traffic matrices: balancing measurements, inference and modeling,” in *Proceedings of the 2005 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, in SIGMETRICS ’05. New York, NY, USA: Association for Computing Machinery, Jun. 2005, pp. 362–373. doi: 10.1145/1064212.1064259.

[12] K. Papagiannaki, N. Taft, and A. Lakhina, “A distributed approach to measure IP traffic matrices,” in *Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, in IMC ’04. New York, NY, USA: Association for Computing Machinery, Oct. 2004, pp. 161–174. doi: 10.1145/1028788.1028808.

[13] L. Nie, D. Jiang, and Z. Lv, “Modeling network traffic for traffic matrix estimation and anomaly detection based on Bayesian network in cloud computing networks,” *Ann. Telecommun.*, vol. 72, no. 5, pp. 297–305, Jun. 2017, doi: 10.1007/s12243-016-0546-3.

[14] S. Xu, M. Kodialam, T. V. Lakshman, and S. Panwar, “Learning Based Methods for Traffic Matrix Estimation from Link Measurements.” arXiv, Aug. 03, 2020. doi: 10.48550/arXiv.2008.00905.

[15] Y. Vardi, “Network Tomography: Estimating Source-Destination Traffic Intensities from Link Data,” *J. Am. Stat. Assoc.*, vol. 91, no. 433, pp. 365–377, 1996, doi: 10.2307/2291416.

[16] reia, “Traffic Matrix Estimation in Non-Stationary Environments,” *SlideServe*, Sep. 17, 2014. https://www.slideserve.com/reia/traffic-matrix-estimation-in-non-stationary-environments (accessed May 23, 2023).

[17] T. Hong, F. Tongliang, and Z. Guogeng, “An Assignment Model on Traffic Matrix Estimation,” in *Advances in Natural Computation*, L. Jiao, L. Wang, X. Gao, J. Liu, and F. Wu, Eds., in Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2006, pp. 295–304. doi: 10.1007/11881223_36.

[18] J. Cao, D. Davis, S. Vander Wiel, and B. Yu, “Time-Varying Network Tomography: Router Link Data,” *J. Am. Stat. Assoc.*, vol. 95, no. 452, pp. 1063–1075, Dec. 2000, doi: 10.1080/01621459.2000.10474303.

[19] C. Tebaldi and M. West, “Bayesian Inference on Network Traffic Using Link Count Data,” *J. Am. Stat. Assoc.*, vol. 93, no. 442, pp. 557–573, 1998, doi: 10.2307/2670105.

[20] J. Zhang, “Origin-Destination Network Tomography with Bayesian Inversion Approach,” in *2006 IEEE/WIC/ACM International Conference on Web Intelligence (WI 2006 Main Conference Proceedings)(WI’06)*, Dec. 2006, pp. 38–44. doi: 10.1109/WI.2006.126.

[21] A. Azzouni and G. Pujolle, “A Long Short-Term Memory Recurrent Neural Network Framework for Network Traffic Matrix Prediction.” arXiv, Jun. 08, 2017. doi: 10.48550/arXiv.1705.05690.

[22] R. Vinayakumar, K. P. Soman, and P. Poornachandran, “Applying deep learning approaches for network traffic prediction,” in *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, Sep. 2017, pp. 2353–2358. doi: 10.1109/ICACCI.2017.8126198.

[23] L. Nie, D. Jiang, L. Guo, and S. Yu, “Traffic matrix prediction and estimation based on deep learning in large-scale IP backbone networks,” *J. Netw. Comput. Appl.*, vol. 76, pp.

16–22, Dec. 2016, doi: 10.1016/j.jnca.2016.10.006.

[24] C.-J. Lan and S.-P. Miaou, “Real-Time Prediction of Traffic Flows Using Dynamic Generalized Linear Models,” *Transp. Res. Rec.*, vol. 1678, no. 1, pp. 168–178, Jan. 1999, doi: 10.3141/1678-21.

[25] “Introduction to forecasting - Skforecast Docs.” https://skforecast.org/0.5.1/quick-start/introduction-forecasting.html (accessed May 23, 2023).

[26] H. Feng and Y. Shu, “Study on network traffic prediction techniques,” in *Proceedings. 2005 International Conference on Wireless Communications, Networking and Mobile Computing, 2005.*, Sep. 2005, pp. 1041–1044. doi: 10.1109/WCNM.2005.1544219.

[27] P. Cortez, M. Rio, M. Rocha, and P. Sousa, “Internet Traffic Forecasting using Neural Networks,” in *The 2006 IEEE International Joint Conference on Neural Network Proceedings*, Jul. 2006, pp. 2635–2642. doi: 10.1109/IJCNN.2006.247142.

[28] R. J. Chetty Priya, “Introduction to the Autoregressive Integrated Moving Average (ARIMA) model,” *Knowledge Tank*, Sep. 29, 2020. https://www.projectguru.in/introduction-to-the-autoregressive-integrated-moving-average-arima-model/ (accessed May 23, 2023).

[29] “Συγκριτική ανάλυση και εφαρμογή γραμμικών, μη-γραμμικών και νευρο-ασαφών μεθόδων, για τη βραχυπρόθεσμη προβλέψη παραγωγής ενέργειας από αιολικά πάρκα.” https://dias.library.tuc.gr/view/11817?locale=el (accessed Feb. 17, 2023).

[30] “E-class Πανεπιστημίου Πειραιώς | Ποσοτικές Μέθοδοι | Έγγραφα.” https://eclass.unipi.gr/modules/document/index.php?course=DES103&openDir=/5739e8183Doj/5e90c78cmafZ (accessed Feb. 17, 2023).

[31] “Dimitris Kugiumtzis.” https://users.auth.gr/dkugiu/Teach/TimeSeriesTHMMY/index.html#notes (accessed Feb. 17, 2023).

[32] A. Sang and S. Li, “A predictability analysis of network traffic,” *Comput. Netw.*, vol. 39, no. 4, pp. 329–345, Jul. 2002, doi: 10.1016/S1389-1286(01)00304-8.

[33] S.-Q. Li and C.-L. Hwang, “On the convergence of traffic measurement and queueing analysis: a statistical-matching and queueing (SMAQ) tool,” *IEEEACM Trans. Netw.*, vol. 5, no. 1, pp. 95–110, Feb. 1997, doi: 10.1109/90.554725.

[34] W. Liu, A. Hong, L. Ou, W. Ding, and G. Zhang, “Prediction and correction of traffic matrix in an IP backbone network,” in *2014 IEEE 33rd International Performance Computing and Communications Conference (IPCCC)*, Dec. 2014, pp. 1–9. doi: 10.1109/PCCC.2014.7017051.

[35] “ΚΕΦΑΛΑΙΟ 4 - Μηχανική Μάθηση.” http://repfiles.kallipos.gr/html_books/93/04a-main.html (accessed May 23, 2023).

[36] M. Barabas, G. Boanea, A. B. Rus, V. Dobrota, and J. Domingo-Pascual, “Evaluation of network traffic prediction based on neural networks with multi-task learning and multiresolution decomposition,” in *2011 IEEE 7th International Conference on Intelligent Computer Communication and Processing*, Aug. 2011, pp. 95–102. doi: 10.1109/ICCP.2011.6047849.

[37] A. Azzouni and G. Pujolle, “NeuTM: A neural network-based framework for traffic matrix prediction in SDN,” in *NOMS 2018 - 2018 IEEE/IFIP Network Operations and Management Symposium*, Apr. 2018, pp. 1–5. doi: 10.1109/NOMS.2018.8406199.

[38] “Fig. 4: Proposed architecture for the mobile traffic prediction.,” *ResearchGate*. https://www.researchgate.net/figure/Proposed-architecture-for-the-mobile-traffic-prediction_fig4_326773789 (accessed May 23, 2023).

[39] J. Zhao, H. Qu, J. Zhao, and D. Jiang, “Towards traffic matrix prediction with LSTM recurrent neural networks,” *Electron. Lett.*, vol. 54, no. 9, pp. 566–568, 2018, doi: 10.1049/el.2018.0336.

[40] W. Jiang, “Internet traffic matrix prediction with convolutional LSTM neural network,” *Internet Technol. Lett.*, vol. 5, no. 2, p. e322, 2022, doi: 10.1002/itl2.322.

[41] B. Liu, Y. Chen, S. Liu, and H.-S. Kim, “Deep Learning in Latent Space for Video Prediction and Compression,” presented at the Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 701–710. Accessed: Feb. 08, 2023. [Online]. Available: https://openaccess.thecvf.com/content/CVPR2021/html/Liu_Deep_Learning_in_Latent_Space_for_Video_Prediction_and_Compression_CVPR_2021_paper.html

[42] M. Hosseini, A. S. Maida, M. Hosseini, and G. Raju, “Inception LSTM for Next-frame Video Prediction (Student Abstract),” *Proc. AAAI Conf. Artif. Intell.*, vol. 34, no. 10, Art. no. 10, Apr. 2020, doi: 10.1609/aaai.v34i10.7176.

[43] Y. Zhou, H. Dong, and A. El Saddik, “Deep Learning in Next-Frame Prediction: A Benchmark Review,” *IEEE Access*, vol. 8, pp. 69273–69283, 2020, doi: 10.1109/ACCESS.2020.2987281.

[44] A. R. Yuliani *et al.*, “Remaining Useful Life Prediction of Lithium-Ion Battery Based on LSTM and GRU,” in *Proceedings of the 2021 International Conference on Computer, Control, Informatics and Its Applications*, in IC3INA ’21. New York, NY, USA: Association for Computing Machinery, Feb. 2022, pp. 21–25. doi: 10.1145/3489088.3489092.

[45] S. Troia, R. Alvizu, Y. Zhou, G. Maier, and A. Pattavina, “Deep Learning-Based Traffic Prediction for Network Optimization,” in *2018 20th International Conference on Transparent Optical Networks (ICTON)*, Jul. 2018, pp. 1–4. doi: 10.1109/ICTON.2018.8473978.

[46] Ε.-Ι. Κουλέτου, “Αυτοματοποιημένη διαχείριση πόρων με χρήση τεχνικών πρόβλεψης

χρονοσειρών και βαθιά ενισχυτική μάθηση," Mar. 2021, doi: 10.26240/heal.ntua.20828.

[47] Z. Liu, Z. Wang, X. Yin, X. Shi, Y. Guo, and Y. Tian, "Traffic Matrix Prediction Based on Deep Learning for Dynamic Traffic Engineering," in *2019 IEEE Symposium on Computers and Communications (ISCC)*, Jun. 2019, pp. 1–7. doi: 10.1109/ISCC47284.2019.8969631.

[48] Μ. Ποντίκη and M. Pontiki, "Πρόβλεψη της τιμής του κρυπτονομίσματος με αλγόριθμους βαθιάς μηχανικής μάθησης," Sep. 2022, doi: 10.26240/heal.ntua.23410.

[49] J. Zhao, H. Qu, J. Zhao, and D. Jiang, "Spatiotemporal traffic matrix prediction: A deep learning approach with wavelet multiscale analysis," *Trans. Emerg. Telecommun. Technol.*, vol. 30, no. 12, p. e3640, 2019, doi: 10.1002/ett.3640.

[50] V. B. Dharmadhikari and J. D. Gavade, "An NN approach for MPEG video traffic prediction," in *2010 2nd International Conference on Software Technology and Engineering*, Oct. 2010, pp. V1-57-V1-61. doi: 10.1109/ICSTE.2010.5608912.

[51] Ε. Αναστασοπούλου and E. Anastasopoulou, "Η Τεχνητή Νοημοσύνη και οι εφαρμογές της," Jul. 2019, doi: 10.26240/heal.ntua.16707.

[52] Α. Ι. Μπατζογιάννη, "Μέθοδοι μηχανικής μάθησης και νευρωνικών δικτύων για ανίχνευση ποιότητας τροφίμων," bachelorThesis, 2021. Accessed: Mar. 08, 2023. [Online]. Available: http://ir.lib.uth.gr/xmlui/handle/11615/55279

[53] A. Zheng and A. Casari, *Feature Engineering for Machine Learning: Principles and Techniques for Data Scientists*. O'Reilly Media, Inc., 2018.

[54] E. Alpaydin, *Machine Learning, revised and updated edition*. MIT Press, 2021.

[55] "What is Machine Learning? | IBM." https://www.ibm.com/topics/machine-learning (accessed Mar. 08, 2023).

[56] "3 Types of Machine Learning," *New Tech Dojo*, Jun. 22, 2020. https://www.newtechdojo.com/3-types-of-machine-learning/ (accessed May 23, 2023).

[57] Α. Νικολακακης, "ΕΝΙΣΧΥΤΙΚΗ ΜΑΘΗΣΗ ΣΕ ΠΑΙΧΝΙΔΙΑ ΣΤΡΑΤΗΓΙΚΗΣ: ΑΝΑΠΤΥΞΗ ΠΑΙΧΝΙΔΙΟΥ ΤΥΠΟΥ «ΣΚΑΚΙ»", Accessed: Mar. 08, 2023. [Online]. Available: https://apothesis.eap.gr/archive/item/78034

[58] L. Deng and D. Yu, "Deep Learning: Methods and Applications," *Found. Trends® Signal Process.*, vol. 7, no. 3–4, pp. 197–387, Jun. 2014, doi: 10.1561/2000000039.

[59] Saumyab271, "Machine Learning vs Deep Learning vs Artificial Intelligence| Know in-depth Difference," *Analytics Vidhya*, Jun. 14, 2021. https://www.analyticsvidhya.com/blog/2021/06/machine-learning-vs-artificial-intelligence-vs-deep-learning/ (accessed May 23, 2023).

[60] "Deep Learning in Digital Pathology," Feb. 13, 2018. https://www.global-engage.com/life-science/deep-learning-in-digital-pathology/ (accessed May 23, 2023).

[61] Γ. Β. Σομπόνης, “Ευφυής Προσδιορισμός Βέλτιστων Διαδρομών Βάσει Μεθόδων Μηχανικής Μάθησης,” Jul. 2012, Accessed: Mar. 09, 2023. [Online]. Available: http://artemis.cslab.ece.ntua.gr:8080/jspui/handle/123456789/16317

[62] “Figure 3. A biological neuron in comparison to an artificial neural...,” *ResearchGate*. https://www.researchgate.net/figure/A-biological-neuron-in-comparison-to-an-artificial-neural-network-a-human-neuron-b_fig2_339446790 (accessed May 23, 2023).

[63] Ν. Δ. Μάκαρης, “Αναγνώριση είδους μουσικής από συμβολικά δεδομένα (MIDI) με τεχνικές μηχανικής μάθησης,” Nov. 2020, Accessed: Mar. 09, 2023. [Online]. Available: http://artemis.cslab.ece.ntua.gr:8080/jspui/handle/123456789/17799

[64] Ε. Σταθά and Ε. Κριτσωτάκης, “Εφαρμογή τεχνικών μεταφοράς μάθησης για τη βελτίωση της προβλεπτικής ακρίβειας νευρωνικών δικτύων,” Jul. 2021, Accessed: Mar. 09, 2023. [Online]. Available: http://artemis.cslab.ece.ntua.gr:8080/jspui/handle/123456789/18026

[65] “Figure 3. Structure of a feedforward neural network,” *ResearchGate*. https://www.researchgate.net/figure/Structure-of-a-feedforward-neural-network_fig2_343452701 (accessed May 23, 2023).

[66] M. Mandal, “Introduction to Convolutional Neural Networks (CNN),” *Analytics Vidhya*, May 01, 2021. https://www.analyticsvidhya.com/blog/2021/05/convolutional-neural-networks-cnn/ (accessed May 23, 2023).

[67] Φ. Στάμου, “Αναγνώριση ανθρώπινων ενεργειών σε βίντεο με την χρήση Βαθιών Νευρωνικών δικτύων,” Nov. 2019, Accessed: Mar. 09, 2023. [Online]. Available: http://artemis.cslab.ece.ntua.gr:8080/jspui/handle/123456789/17464

[68] “Πέργαμος - Βιβλιοθήκη και Κέντρο Πληροφόρησης Εθνικού και Καποδιστριακού Πανεπιστημίου Αθηνών.” https://pergamos.lib.uoa.gr/uoa/dl/object/2896736 (accessed Mar. 10, 2023).

[69] Ε. Τσατσαρώνης and E. Tsatsaronis, “Ανάλυση επίδοσης συνελικτικών νευρωνικών δικτύων σε πολυπύρηνα συστήματα,” Feb. 2018, doi: 10.26240/heal.ntua.15112.

[70] “Deep Learning.” https://www.deeplearningbook.org/ (accessed Mar. 08, 2023).

[71] “Understanding the Mechanism and Types of Recurrent Neural Networks,” *Open Data Science - Your News Source for AI, Machine Learning & more*, Dec. 15, 2020. https://opendatascience.com/understanding-the-mechanism-and-types-of-recurring-neural-networks/ (accessed Mar. 11, 2023).

[72] “Types of RNN (Recurrent Neural Network),” *OpenGenus IQ: Computing Expertise & Legacy*, Feb. 04, 2020. https://iq.opengenus.org/types-of-rnn/ (accessed Mar. 11, 2023).

[73] Y. Yu, X. Si, C. Hu, and J. Zhang, “A Review of Recurrent Neural Networks: LSTM Cells and Network Architectures,” *Neural Comput.*, vol. 31, no. 7, pp. 1235–1270, Jul. 2019,

doi: 10.1162/neco_a_01199.

[74] Σ. Τσανάκας and S. Tsanakas, “Αυτοματοποίηση της Εκπαίδευσης Νευρωνικών Δικτύων μέσω Εξελικτικών Αλγορίθμων,” Feb. 2022, doi: 10.26240/heal.ntua.22214.

[75] Α. Ζάχος, “Το νευρωνικό δίκτυο LSTM ως υδρολογικό μοντέλο βροχής απορροής,” Mar. 2021, Accessed: Mar. 01, 2023. [Online]. Available: http://artemis.cslab.ece.ntua.gr:8080/jspui/handle/123456789/17882

[76] C. Zhou, C. Sun, Z. Liu, and F. C. M. Lau, “A C-LSTM Neural Network for Text Classification.” arXiv, Nov. 30, 2015. doi: 10.48550/arXiv.1511.08630.

[77] F. A. Gers and E. Schmidhuber, “LSTM recurrent networks learn simple context-free and context-sensitive languages,” *IEEE Trans. Neural Netw.*, vol. 12, no. 6, pp. 1333–1340, Nov. 2001, doi: 10.1109/72.963769.

[78] T. He and J. Droppo, “Exploiting LSTM structure in deep neural networks for speech recognition,” in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Mar. 2016, pp. 5445–5449. doi: 10.1109/ICASSP.2016.7472718.

[79] X. Du, H. Zhang, H. V. Nguyen, and Z. Han, “Stacked LSTM Deep Learning Model for Traffic Prediction in Vehicle-to-Vehicle Communication,” in *2017 IEEE 86th Vehicular Technology Conference (VTC-Fall)*, Sep. 2017, pp. 1–5. doi: 10.1109/VTCFall.2017.8288312.

[80] X. SHI, Z. Chen, H. Wang, D.-Y. Yeung, W. Wong, and W. WOO, “Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting,” in *Advances in Neural Information Processing Systems*, Curran Associates, Inc., 2015. Accessed: Mar. 12, 2023. [Online]. Available: https://proceedings.neurips.cc/paper/2015/hash/07563a3fe3bbe7e3ba84431ad9d055af-Abstract.html

[81] Ε. Καμπιτάκης and E. Kampitakis, “Σύγκριση Στατιστικών Μεθόδων και Μεθόδων Υπολογιστικής Νοημοσύνης για Βραχυπρόθεσμη Πρόβλεψη Μονομεταβλητών Κυκλοφοριακών Χρονοσειρών,” Oct. 2020, doi: 10.26240/heal.ntua.19258.

[82] “Figure 7. The structure of ConvLSTM. The new memory C t and output H t...,” *ResearchGate*. https://www.researchgate.net/figure/The-structure-of-ConvLSTM-The-new-memory-C-t-and-output-H-t-will-be-generated-by_fig7_335238114 (accessed May 23, 2023).

[83] B. Li, B. Tang, L. Deng, and M. Zhao, “Self-Attention ConvLSTM and Its Application in RUL Prediction of Rolling Bearings,” *IEEE Trans. Instrum. Meas.*, vol. 70, pp. 1–11, 2021, doi: 10.1109/TIM.2021.3086906.

[84] K. Team, “Keras documentation: ConvLSTM2D layer.” https://keras.io/api/layers/recurrent_layers/conv_lstm2d/ (accessed Mar. 20, 2023).

[85] “About Convolutional Layer and Convolution Kernel.”

https://shhd.storychief.io/en/2019-10-31-convolutional-layer-convolution-kernel (accessed Mar. 20, 2023).

[86] Θ. Πίσσας and T. Pissas, “Αναγνώριση Ανθρώπινης Δράσης και Χειρονομιών χρησιμοποιώντας Συνελικτικά και Αναδρομικά Νευρωνικά Δίκτυα,” Oct. 2017, doi: 10.26240/heal.ntua.14732.

[87] Ε. Νταρουίς, “Διάγνωση της νόσου Πάρκινσον μέσω βάδισης με τεχνικές μηχανικής μάθησης,” Dec. 2020, doi: 10.26240/heal.ntua.19913.

[88] A. G. Salman, Y. Heryadi, E. Abdurahman, and W. Suparta, “Single Layer & Multi-layer Long Short-Term Memory (LSTM) Model with Intermediate Variables for Weather Forecasting,” *Procedia Comput. Sci.*, vol. 135, pp. 89–98, Jan. 2018, doi: 10.1016/j.procs.2018.08.153.

[89] A. Sahar and D. Han, “An LSTM-based Indoor Positioning Method Using Wi-Fi Signals,” in *Proceedings of the 2nd International Conference on Vision, Image and Signal Processing*, in ICVISP 2018. New York, NY, USA: Association for Computing Machinery, Aug. 2018, pp. 1–5. doi: 10.1145/3271553.3271566.

[90] “Figure 4. Simple LSTM Vs Stacked LSTM.,” *ResearchGate*. https://www.researchgate.net/figure/Simple-LSTM-Vs-Stacked-LSTM_fig4_328819708 (accessed May 23, 2023).

[91] Α. Σαρρής and A. Sarris, “Πρόβλεψη αποτελεσμάτων αθλητικών γεγονότων με χρήση δικτύων μακράς βραχύχρονης μνήμης (LSTM),” Feb. 2023, doi: 10.26240/heal.ntua.24860.

[92] Y. Shi, K. Yao, L. Tian, and D. Jiang, “Deep LSTM based Feature Mapping for Query Classification,” in *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, San Diego, California: Association for Computational Linguistics, Jun. 2016, pp. 1501–1511. doi: 10.18653/v1/N16-1176.

[93] C.-C. Kao, M. Sun, W. Wang, and C. Wang, “A Comparison of Pooling Methods on LSTM Models for Rare Acoustic Event Classification,” in *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2020, pp. 316–320. doi: 10.1109/ICASSP40776.2020.9053150.

[94] Κ. Καραΐσκου and K. Karaiskou, “Χρήση τεχνικών βαθιάς μηχανικής μάθησης για την εύρεση δεδομένων από το Twitter που σχετίζονται με καταστροφές,” Jul. 2021, doi: 10.26240/heal.ntua.21405.

[95] A. Aziz Sharfuddin, Md. Nafis Tihami, and Md. Saiful Islam, “A Deep Recurrent Neural Network with BiLSTM model for Sentiment Classification,” in *2018 International Conference on Bangla Speech and Language Processing (ICBSLP)*, Sep. 2018, pp. 1–4. doi: 10.1109/ICBSLP.2018.8554396.

[96] S. Siami-Namini, N. Tavakoli, and A. S. Namin, “The Performance of LSTM and

BiLSTM in Forecasting Time Series," in *2019 IEEE International Conference on Big Data (Big Data)*, Dec. 2019, pp. 3285–3292. doi: 10.1109/BigData47090.2019.9005997.

[97] Λ. Ηλίας, "Ανίχνευση κακόβουλων χρηστών σε κοινωνικά δίκτυα μέσω μεθόδων βαθιάς μάθησης," Dec. 2020, doi: 10.26240/heal.ntua.20200.

[98] C. C. Aggarwal, *Neural Networks and Deep Learning: A Textbook*. Cham: Springer International Publishing, 2018. doi: 10.1007/978-3-319-94463-0.

[99] "Figure 4.2: Gated Recurrent Unit (GRU)," *ResearchGate*. https://www.researchgate.net/figure/Gated-Recurrent-Unit-GRU_fig4_328462205 (accessed May 23, 2023).

[100] R. Fu, Z. Zhang, and L. Li, "Using LSTM and GRU neural network methods for traffic flow prediction," in *2016 31st Youth Academic Annual Conference of Chinese Association of Automation (YAC)*, Nov. 2016, pp. 324–328. doi: 10.1109/YAC.2016.7804912.

[101] N. M. Shahani, M. Kamran, X. Zheng, and C. Liu, "Predictive modeling of drilling rate index using machine learning approaches: LSTM, simple RNN, and RFA," *Pet. Sci. Technol.*, vol. 40, no. 5, pp. 534–555, Mar. 2022, doi: 10.1080/10916466.2021.2003386.

[102] Ο. Μπούρχας and O. Bourchas, "Neural networks for the prediction of fuel oil consumption for containerships," Feb. 2021, doi: 10.26240/heal.ntua.20625.

[103] J. Korstanje, *Advanced Forecasting with Python: With State-of-the-Art-Models Including LSTMs, Facebook's Prophet, and Amazon's DeepAR*. Berkeley, CA: Apress, 2021. doi: 10.1007/978-1-4842-7150-6.

[104] "Recurrent neural networks (RNNs) | Advanced Deep Learning with Keras." https://subscription.packtpub.com/book/big-data-and-business-intelligence/9781788629416/1/ch01lvl1sec09/recurrent-neural-networks-rnns (accessed Mar. 26, 2023).

[105] A. Huet, "RNN with Keras: Understanding computations." https://ahstat.github.io/RNN-Keras-understanding-computations/ (accessed Mar. 26, 2023).

[106] M. C. Sorkun, C. Paoli, and Ö. D. Incel, "Time series forecasting on solar irradiation using deep learning," in *2017 10th International Conference on Electrical and Electronics Engineering (ELECO)*, Nov. 2017, pp. 151–155.

[107] Priyanka, A. Kumari, and M. Sood, "Implementation of SimpleRNN and LSTMs based prediction model for coronavirus disease (Covid-19)," *IOP Conf. Ser. Mater. Sci. Eng.*, vol. 1022, no. 1, p. 012015, Jan. 2021, doi: 10.1088/1757-899X/1022/1/012015.

[108] "TOTEM (TOolbox for Traffic Engineering Methods) Project." https://totem.run.montefiore.uliege.be/ (accessed Mar. 29, 2023).

[109] W. Zheng, Y. Li, M. Hong, X. Fan, and G. Zhao, "Flow-by-flow traffic matrix prediction methods: Achieving accurate, adaptable, low cost results," *Comput. Commun.*, vol. 194, pp. 348–360, Oct. 2022, doi: 10.1016/j.comcom.2022.07.052.

[110] N. Ramakrishnan and T. Soni, “Network Traffic Prediction Using Recurrent Neural Networks,” in *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, Dec. 2018, pp. 187–193. doi: 10.1109/ICMLA.2018.00035.

[111] S. Qazi, S. Atif, and M. Kadri, “A Novel Compressed Sensing Technique for Traffic Matrix Estimation of Software Defined Cloud Networks,” *KSII Trans. Internet Inf. Syst.*, vol. 12, pp. 4678–4702, Oct. 2018, doi: 10.3837/tiis.2018.10.004.

[112] A. Lakhina, K. Papagiannaki, M. Crovella, C. Diot, E. D. Kolaczyk, and N. Taft, “Structural analysis of network traffic flows,” in *Proceedings of the joint international conference on Measurement and modeling of computer systems*, in SIGMETRICS ’04/Performance ’04. New York, NY, USA: Association for Computing Machinery, Jun. 2004, pp. 61–72. doi: 10.1145/1005686.1005697.

[113] J. Zhang, K. Xi, M. Luo, and H. J. Chao, “Load balancing for multiple traffic matrices using SDN hybrid routing,” in *2014 IEEE 15th International Conference on High Performance Switching and Routing (HPSR)*, Jul. 2014, pp. 44–49. doi: 10.1109/HPSR.2014.6900880.

[114] J. Reis, M. Rocha, T. K. Phan, D. Griffin, F. Le, and M. Rio, “Deep Neural Networks for Network Routing,” in *2019 International Joint Conference on Neural Networks (IJCNN)*, Jul. 2019, pp. 1–8. doi: 10.1109/IJCNN.2019.8851733.

[115] “Figure 1: Physical supply topology of the Abilene network.,” *ResearchGate*. https://www.researchgate.net/figure/Physical-supply-topology-of-the-Abilene-network_fig1_261483544 (accessed May 23, 2023).

[116] D. Chadwick *et al.*, “One step ahead, The 20th Trans European Research and Education Networking Conference, June 7-10, 2004, Rhodes, Greece, Selected Papers,” Jan. 2004.

[117] “Fig. 1 . Topology of GEANT network,” *ResearchGate*. https://www.researchgate.net/figure/Topology-of-GEANT-network_fig1_220707854 (accessed May 23, 2023).

[118] “TensorFlow,” *TensorFlow*. https://www.tensorflow.org/ (accessed Apr. 03, 2023).

[119] K. Team, “Keras documentation: Keras API reference.” https://keras.io/api/ (accessed Apr. 03, 2023).

[120] D. Choi, C. J. Shallue, Z. Nado, J. Lee, C. J. Maddison, and G. E. Dahl, “On Empirical Comparisons of Optimizers for Deep Learning.” arXiv, Jun. 15, 2020. doi: 10.48550/arXiv.1910.05446.

[121] D. P. Kingma and J. Ba, “Adam: A Method for Stochastic Optimization.” arXiv, Jan. 29, 2017. doi: 10.48550/arXiv.1412.6980.

[122] L. Prechelt, “Early Stopping — But When?,” in *Neural Networks: Tricks of the Trade: Second Edition*, G. Montavon, G. B. Orr, and K.-R. Müller, Eds., in Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2012, pp. 53–67. doi:

10.1007/978-3-642-35289-8_5.

[123] J. Brownlee, “Use Early Stopping to Halt the Training of Neural Networks At the Right Time,” *MachineLearningMastery.com*, Dec. 09, 2018. https://machinelearningmastery.com/how-to-stop-training-deep-neural-networks-at-the-right-time-using-early-stopping/ (accessed Apr. 03, 2023).

[124] H. Sak, A. Senior, and F. Beaufays, “Long Short-Term Memory Based Recurrent Neural Network Architectures for Large Vocabulary Speech Recognition.” arXiv, Feb. 05, 2014. doi: 10.48550/arXiv.1402.1128.

[125] Y. Jia, Z. Wu, Y. Xu, D. Ke, and K. Su, “Long Short-Term Memory Projection Recurrent Neural Network Architectures for Piano’s Continuous Note Recognition,” *J. Robot.*, vol. 2017, p. e2061827, Sep. 2017, doi: 10.1155/2017/2061827.