Ατομική Διπλωματική Εργασία

Σύστημα πρόβλεψης ζήτησης φυσικού αερίου με τη χρήση νευρωνικών δικτύων με ανάδραση (τύπου LSTM και GRU)

Μιχαήλ-Παναγιώτης Μπόφος

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Μάιος 2022

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Σύστημα πρόβλεψης ζήτησης φυσικού αερίου με τη χρήση νευρωνικών δικτύων με ανάδραση (τύπου LSTM και GRU)

Μιχαήλ-Παναγιώτης Μπόφος

Επιβλέπων Καθηγητής
Δρ. Χρίστος Χριστοδούλου

Η Ατομική Διπλωματική Εργασία υποβλήθηκε προς μερική εκπλήρωση των απαιτήσεων απόκτησης του πτυχίου Πληροφορικής του Τμήματος Πληροφορικής του Πανεπιστημίου Κύπρου

Μάιος 2022

Ευχαριστίες

Θα ήθελα να ευχαριστήσω θερμά τον επιβλέποντα καθηγητή μου Δρ. Χρίστο Χριστοδούλου καθώς και τον Δρ. Μιχάλη Αγαθοκλέους για την εμπιστοσύνη τους και τη συνεχή τους υποστήριξη και βοήθεια. Επίσης θα ήθελα να ευχαριστήσω τους φίλους και την οικογένειά μου για τη συμπαράσταση και τη βοήθειά τους κατά τη διάρκεια των σπουδών μου.

Περίληψη

Στόχος αυτής της διπλωματικής εργασίας είναι η εφαρμογή προχωρημένων νευρωνικών δικτύων με ανάδραση στη δημιουργία συστημάτων πρόβλεψης πραγματικού κόσμου. Πιο συγκεκριμένα, το πρόβλημα που κληθήκαμε να μοντελοποιήσουμε, είναι η ωριαία πρόβλεψη ζήτησης φυσικού αερίου. Στόχος αυτής της μελέτης ήταν η σύγκριση των προχωρημένων νευρωνικών δικτύων με ανάδραση με άλλα πιο απλά δίκτυα με ανάδραση στην επίλυση δυναμικών προβλημάτων σε ένα θορυβώδες περιβάλλον.

Στην εργασία αυτή εκμεταλλευτήκαμε δεδομένα από μια Βρετανική εταιρεία παροχής φυσικού αερίου τα οποία καλύπτουν ένα φάσμα επτά ετών και μας παρείχαν πληροφορίες όπως η ζήτηση φυσικού αερίου ανά ώρα, η θερμοκρασία ανά δύο ώρες και η ταχύτητα του ανέμου ανά τέσσερις ώρες. Τα δεδομένα αυτά είναι πραγματικά και καταγράφηκαν την περίοδο 1990 με 1997. Για να μπορέσουμε να προβλέψουμε την ζήτηση φυσικού αερίου, υλοποιήσαμε ένα νευρωνικό δίκτυο Long short term memory[5] (LSTM) και ένα νευρωνικό δίκτυο Gated recurrent unit[4] (GRU). Τα δύο αυτά νευρωνικά δίκτυα εκμεταλλεύονται τις λογικές πύλες για να ορίσουν την βαρύτητα κάθε ψήγματος γνώσης που αποκτά το δίκτυό μας. Με αυτόν το τρόπο, δημιουργείται η τεχνητή μνήμη των δύο αυτών δικτύων, τα οποία ανάλογα με τα δεδομένα εισόδου, ανακαλούν από την μνήμη τους τα αντίστοιχα δεδομένα. Τα στοιχεία που δίνουμε στα δίκτυα αυτά ως είσοδο είναι: η ζήτηση φυσικού αερίου, η θερμοκρασία, η ταχύτητα του ανέμου και αν μια μέρα είναι αργία ή όχι.

Για να μπορέσουμε να βρούμε ποια σύνθεση του δικτύου μας επιλύει πιο αποτελεσματικά το πρόβλημα αυτό πειραματιστήκαμε με τη δομή και τις παραμέτρους των δικτύων μας και καταλήξαμε σε κάποιες βέλτιστες, για αυτό το πρόβλημα, τιμές. Τέλος, συγκρίναμε τα αποτελέσματά μας με υλοποιήσεις άλλων νευρωνικών δικτύων με ανάδραση τα οποία αν και πιο απλά είχαν υπό συνθήκες καλύτερες επιδόσεις από τα LSTM και GRU. Πιο συγκεκριμένα, η πιο ακριβής πρόβλεψη μέχρι στιγμής δίνεται από μια παραλλαγή του δικτύου Jordan[10], η οποία είχε ποσοστό πρόβλεψης ίσο με 98.034%[7], απέναντι στα 95.39% και 94.93% των LSTM και GRU αντίστοιχα.

Περιεχόμενα

Κεφάλαιο 1

Εισαγωγή

1.1 Περιγραφή προβλήματος

Είναι γνωστό ότι το φυσικό αέριο αποτελεί μία από τις σημαντικότερες πηγές ενέργειας παγκοσμίως, ως εκ τούτου η αξία του είναι αδιαπραγμάτευτη. Ωστόσο, ως πηγή ενέργειας κατατάσσεται στις μη ανανεώσιμες, γεγονός που υπαγορεύει την ανάγκη για καλή διαχείριση των αποθεμάτων μας. Η διαδικασία που ακολουθείται για να μεταφερθεί το φυσικό αέριο από τις αποθήκες του εκάστοτε πάροχου μέχρι τα σπίτια μας είναι απλή στη φιλοσοφία της. Τεράστιες ποσότητες φυσικού αερίου (σε υγρή κατάσταση) ταξιδεύουν από την αποθήκη προς τα σπίτια που εξυπηρετεί η κάθε αποθήκη, μέσω ενός συστήματος σωλήνων. Το μόνο πρόβλημα με αυτή τη διαδικασία είναι η χαμηλή ταχύτητα με την οποία ταξιδεύει το φυσικό αέριο, η οποία είναι 30 μίλια την ώρα[1] (~48 χλμ. την ώρα). Επομένως, για να μπορούμε να παρέχουμε φυσικό αέριο στους καταναλωτές πρέπει να βρούμε κάποιο τρόπο να φτάνει έγκαιρα ικανοποιητική ποσότητα στον χώρο τους. Το πρόβλημα αυτό θεωρητικά μπορεί να λυθεί με δύο τρόπους, είτε με την ύπαρξη πολλαπλών υποσταθμών φυσικού αερίου, κάτι που είναι ανέφικτο να υλοποιηθεί, είτε με την σωστή πρόβλεψη των αναγκών κάθε περιοχής και την έγκαιρη αποστολή καυσίμων.

Για να καταφέρουμε να επιλύσουμε ουσιαστικά το παραπάνω πρόβλημα θα πρέπει να είμαστε σε θέση να προβλέψουμε αποτελεσματικά τις ανάγκες για κατανάλωση της κάθε περιοχής και για το επιτύχουμε αυτό θα χρησιμοποιήσουμε τη μηχανική μάθηση και τα νευρωνικά δίκτυα. Σκοπός μας είναι να ελαχιστοποιήσουμε το σφάλμα που στη δική μας περίπτωση είναι η οποιαδήποτε απόκλιση από την επιθυμητή ποσότητα καυσίμων (είτε έχουμε έλλειμα είτε περίσσεια καυσίμων). Ωστόσο, υπάρχουν διάφοροι περιβαλλοντικοί και μη παράγοντες που επηρεάζουν την επιθυμητή ποσότητα. Συνοπτικά αυτοί οι παράγοντες είναι η ταχύτητα των ανέμων, η θερμοκρασία, η μέρα της εβδομάδας, η ώρα της ημέρας και τέλος ο μήνας που βρισκόμαστε. Εν κατακλείδι, για να είμαστε σε θέση να παρέχουμε μια έγκυρη πρόβλεψη σε κάθε σταθμό φυσικού αερίου οφείλουμε να λάβουμε υπόψιν μας πολλούς παράγοντες που δύνανται να επηρεάσουν τη ζητούμενη ποσότητα φυσικού αερίου.

1.2 Προηγούμενες έρευνες

Για το ίδιο πρόβλημα έχουν γίνει αρκετές μελέτες, πιο συγκεκριμένα με τα ίδια σύνολα δεδομένων έχουν εκπονηθεί δύο διπλωματικές εργασίες. Και στις δύο περιπτώσεις είχαμε στη διάθεσή μας δεδομένα από μια μεγάλη Βρετανική εταιρεία παροχής φυσικού αερίου που καλύπτουν την περίοδο Μαΐου 1990 – Σεπτεμβρίου 1997. Πιο συγκεκριμένα, σε αυτή τη χρονική περίοδο μας δίδεται ανά μία ώρα η ζήτηση φυσικού αερίου (εκατομμύρια κυβικά μέτρα), ανά δύο ώρες η θερμοκρασία (βαθμοί Κελσίου) της περιοχής και ανά τέσσερις ώρες η ταχύτητα των ανέμων (Κόμβοι). Ωστόσο υπήρχαν αρκετές τιμές που έλειπαν από τα αρχικά δεδομένα, αλλά ο Χρίστος – Αιμίλιος Πραστίτης[2] και ο Κυπριανός Χατζηδημητρίου[3] χρησιμοποίησαν ένα εργαλείο που βοήθησε στη συμπλήρωση των κενών αλλά και στη δημιουργία ενός ενιαίου αρχείου, με αποτέλεσμα να έχουμε ένα αρχείο με την κατάλληλη δομή το οποίο μπορεί να χρησιμοποιηθεί εύκολα και αποδοτικά ως σύνολο δεδομένων. Στην εργασία του Χρίστου – Αιμίλιου Πραστίτη, ερευνήθηκαν οι λύσεις με χρήση του αλγόριθμου Back Propagation[4][5] με την προσέγγιση του κινούμενου παραθύρου αλλά και μια παραλλαγή με στατικό δίκτυο. Τα αποτελέσματα αυτής της έρευνας ήταν αρκετά ενθαρρυντικά καθώς με την τεχνική του κινούμενου παραθύρου σημειώθηκε ακρίβεια πρόβλεψης της τάξης του 96%, ενώ με το στατικό δίκτυο 86%. Βάσει της θεωρίας η τεχνική του κινούμενου παραθύρου είναι μια αποδεκτή λύση για προβλήματα πρόβλεψης, ενώ η παραλλαγή με το στατικό δίκτυο δεν αναμενόταν να έχει ιδιαίτερη αποτελεσματικότητα σε τέτοιου είδους προβλήματα, ωστόσο η υλοποίηση του Πραστίτη παρουσίασε αξιοπρεπέστατα αποτελέσματα. Συμπερασματικά η διπλωματική του Πραστίτη, αποτελεί μια πάρα πολύ καλή βάση για τη λύση αυτού του προβλήματος, καθώς μας καθορίζει τους στόχους και τις απαιτήσεις που έχουμε από τις μελλοντικές μας λύσεις. Ως συνέχεια στην έρευνα αυτού του προβλήματος, ο Κυπριανός Χατζηδημητρίου, υλοποίησε και χρησιμοποίησε δίκτυα συναρτήσεων αξονικών βάσεων[6] και δίκτυα με ανάδραση με την αρχιτεκτονική Jordan[7]. Πιο συγκεκριμένα, για το δίκτυο συναρτήσεων αξονικών βάσεων υλοποιήθηκαν δύο παραλλαγές και οι δύο βασισμένες στον αλγόριθμο των μεταβλητών κέντρων, στη μία η αρχική θέση των κέντρων επιλεγόταν τυχαία ενώ στη δεύτερη χρησιμοποιείτο ένας χάρτης αυτό-οργάνωσης Kohonen[8] , ο οποίος απεικόνιζε τη θέση των εισόδων στο χώρο. Όσον αφορά το δίκτυο Jordan, η υλοποίηση του αποτελείτο από ένα πολυεπίπεδο δίκτυο perceptron, του οποίου η τρέχουσα έξοδος ήταν μια από τις εισόδους για την επόμενη χρονική στιγμή, ο αλγόριθμος που χρησιμοποιήθηκε ήταν παραλλαγή του αλγόριθμου ανάστροφης μετάδοσης σφάλματος, η Back Propagation Through Time (BPTT) [9]. Τα αποτελέσματα που προέκυψαν κρίθηκαν ικανοποιητικά για τη φύση του προβλήματος.

Υλοποίηση	Ποσοστό επιτυχίας
Απλή υλοποίηση RBF	85.057%
Συνδυασμός Kohonen και RBF	83.56%
Jordan με ανατροφοδότηση επιθυμητής τιμής	98.034%
Jordan με ανατροφοδότηση υπολογισμένης τιμής	94.013%

Σχεδιάγραμμα 1.2.1 Αποτελέσματα προηγούμενων ερευνών

Το συμπέρασμα που προκύπτει είναι ότι, τα δίκτυα με ανάδραση, φέρνουν πολύ πιο ακριβή αποτελέσματα στις προβλέψεις που θέλουμε να κάνουμε. Με βάση αυτό, εμείς θα αναλύσουμε δύο πιο προχωρημένα και σύνθετα νευρωνικά δίκτυα με ανάδραση, με στόχο να επιβεβαιώσουμε τα παραπάνω αποτελέσματα και ιδανικά να βελτιώσουμε το ποσοστό επιτυχίας της πρόβλεψής μας. Παράλληλα, άλλες δύο έρευνες έχουν συνταχθεί πάνω στην πρόβλεψη ζήτησης φυσικού αερίου με τη χρήση Long Short Term Memory (LSTM) δικτύων[10]. Πιο συγκεκριμένα, οι Laib et al. ανέπτυξαν ένα υβριδικό νευρωνικό δίκτυο[11] το οποίο με τη χρήση κάποιων ημερήσιων προφίλ και κατηγοριοποίησης δημιουργούν μοντέλα LSTM για να υπολογίσουν την ακριβή τιμή της πρόβλεψής τους, έχοντας ακρίβεια 98%. Αντίστοιχα οι Αναγνώστης et al. ανέπτυξαν μια λύση βασισμένη στα LSTM δίκτυα[12] και έφτασαν σε επίπεδα ακρίβειας της τάξεως του 95%. Η βασική διαφορά των δύο αυτών ερευνών είναι ότι η τιμή που προβλέπουν είναι ημερήσια, σε αντίθεση με τις έρευνες των Πραστίτη και Χατζηδημητρίου όπου η πρόβλεψη γίνεται ανά ώρα.

1.3 Περιγραφή μεθοδολογίας

Σκοπός αυτής της διπλωματικής εργασίας είναι η υλοποίηση ενός συστήματος πρόβλεψης αναγκών φυσικού αερίου με τη χρήση νευρωνικών δικτύων με ανάδραση και πιο συγκεκριμένα δίκτυα LSTM[10] και Gated Recurrent Unit [13] (GRU). Δυστυχώς εξαιτίας της μεγάλης διάρκειας συλλογής δεδομένων κάποιες τιμές έλειπαν, επί τούτου βασιστήκαμε στις διορθώσεις που έγιναν στο σύνολο δεδομένων από τις προηγούμενες έρευνες καθώς και σε κάποιες διορθώσεις που προσθέσαμε οι ίδιοι. Το πρόβλημα που εντοπίσαμε αφορούσε κυρίως τις ημερομηνίες που εμφανίζονταν σε κάθε γραμμή του συνόλου δεδομένων καθώς υπήρχε ένα κενό μερικών μηνών, που εν τέλει οδηγούσε σε δεδομένα εκτός του ορισμένου χρονικού πλαισίου.

1.4 Σύντομη περιγραφή δικτύων

Η επιλογή αυτών των δικτύων έγινε καθώς και τα δύο αποτελούν προχωρημένα αναδρομικά δίκτυα. Και τα δύο αυτά δίκτυα λύνουν το πρόβλημα της εξαφανιζόμενης κλίσης[14] (vanishing gradient problem), δίνοντας μας έτσι τη δυνατότητα να συσχετίσουμε εισόδους με εξόδους με μεγαλύτερο παράθυρο επίδρασης από ότι τα παραδοσιακά δίκτυα με ανάδραση όπως τα Jordan και Elman[15]. Σε πιο τεχνικό επίπεδο και τα LSTM αλλά και τα GRU, χρησιμοποιούν πύλες για ελέγξουν τη ροή πληροφορίας μέσα στο δίκτυό μας. Αυτές οι πύλες ουσιαστικά μαθαίνουν ποιες πληροφορίες είναι σημαντικές και ποιες όχι, επομένως και ποιες πρέπει να επηρεάσουν την πρόβλεψή μας. Πιο συγκεκριμένα ένα LSTM cell αποτελείται από τρεις πύλες, την forget gate, την input gate και την output gate. Η forget gate, καθορίζει ποιες πληροφορίες κρατάμε και ποιες πετάμε, η input gate καθορίζει ποια δεδομένα εισόδου θα αλλάξουν και τέλος η output gate διαμορφώνει την κρυφή κατάσταση του δικτύου που χρησιμοποιείται για προβλέψεις αλλά και για την επεξεργασία των δεδομένων εισόδου. Αντίστοιχα, τα GRU που είναι μια μετεξέλιξη των LSTM, χρησιμοποιούν το update gate για να επιλέξουν ποιες πληροφορίες απορρίπτουμε και ποιες διατηρούμε, και το reset gate για να καθορίσουμε ποια πρότερη γνώση θα ξεχάσουμε. Συμπερασματικά, αυτά τα δίκτυα καταφέρνουν να αυξήσουν τη διάρκεια της μνήμης μας επιτρέποντάς μας να λύνουμε πιο αποτελεσματικά δυναμικά προβλήματα όπως το δικό μας. Στη δική μας περίπτωση έχουμε δεδομένα επταετίας τα οποία παρουσιάζουν διάφορα μοτίβα. Όπως είναι φυσικό διάφορες συμπεριφορές επαναλαμβάνονται ανά τακτά χρονικά διαστήματα, παραδείγματος χάριν ο καιρός ανά εποχή ή ανά ώρα της ημέρας, επομένως είναι αρκετά σημαντικό να μπορούμε να χρησιμοποιούμε πρότερη γνώση η οποία απέχει αρκετά χρονικά.

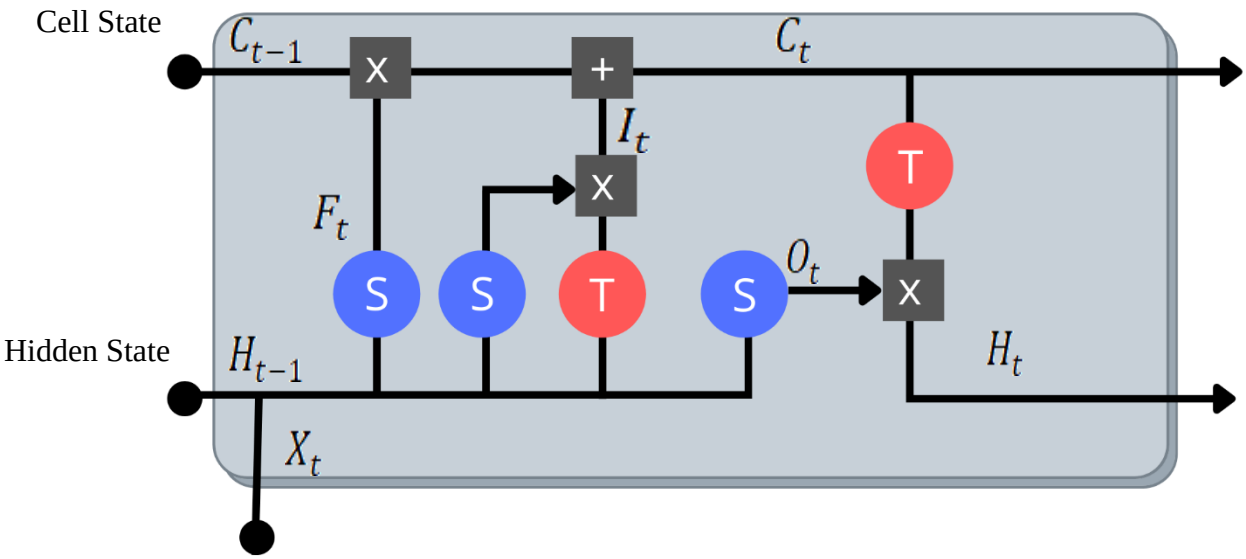
Κεφάλαιο 2

Θεωρητικό υπόβαθρο

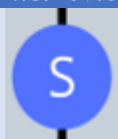
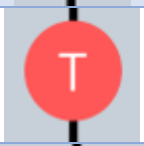
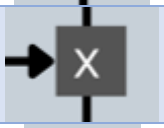
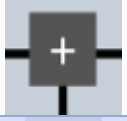
Τα νευρωνικά δίκτυα LSTM και GRU, αποτελούν μετεξέλιξη των απλών δικτύων με ανάδραση όπως τα Jordan's and Elman's. Η κύρια διαφορά τους είναι ότι τα LSTM και GRU δεν επηρεάζονται από το πρόβλημα της εξαφανιζόμενης κλίσης, δηλαδή το φαινόμενο όπου το σφάλμα που χρησιμοποιείται για να αλλάξουν οι τιμές των βαρών φθίνει μέχρι να φτάσει σε όλους τους νευρώνες με αποτέλεσμα να μην έχει ιδιαίτερη επίδραση στην αλλαγή των τιμών των βαρών. Ο αλγόριθμος που χρησιμοποιούν είναι μια παραλλαγή του Back Propagation Through Time[9], κατά τον οποίο ξεδιπλώνουμε το δίκτυό μας, υπολογίζουμε το σφάλμα της κάθε εκτέλεσης και έπειτα το ξαναδιπλώνουμε ανανεώνοντας τις τιμές των βαρών. Παράλληλα, τα LSTM και GRU παρέχουν την αίσθηση βραχυπρόθεσμης μνήμης. Για να το επιτύχουν αυτό, τα δίκτυα αυτά χρησιμοποιούν λογικές πύλες, οι οποίες ελέγχουν τη ροή της πληροφορίας, καθορίζοντας επομένως ποια κομμάτια πληροφορίας είναι σημαντικά και ποια ασήμαντα στις προβλέψεις μας.

2.1 Long short-term memory

Ένα δίκτυο LSTM, αποτελείται από έναν αριθμό από λογικές μονάδες LSTM, όπως αυτή που βλέπουμε στο σχεδιάγραμμα 2.1.1. Η παραγόμενη έξοδος της κάθε λογικής μονάδας καθορίζεται από τρεις παράγοντες, τις τιμές εισόδου, την κατάσταση μνήμης του δικτύου (cell state) και την προηγούμενη έξοδο της μονάδας (hidden state). Οι τιμές των δύο μεταβλητών cell και hidden state καθορίζονται από τις λογικές πύλες forget, input και output που θα εξηγηθούν παρακάτω.



Σχεδιάγραμμα 2.1.1 Εσωτερική δομή λογικής μονάδας LSTM

Απεικόνιση	Εξήγηση
	Σιγμοειδής συνάρτηση
	Υπερβολή της εφαπτομένης
	Πολλαπλασιασμός διανυσμάτων
	Πρόσθεση διανυσμάτων
	Ένωση διανυσμάτων

Σχεδιάγραμμα 2.1.2 Υπόμνημα δομής λογικής μονάδας LSTM

2.1.1 Forget Gate

Η πύλη forget, όπως υποδηλώνει και το όνομά της, καθορίζει ποια κομμάτια πληροφορίας θα απορριφθούν. Για να επιτευχθεί αυτό χρησιμοποιούμε την προηγούμενη τιμή του hidden state και τα τωρινά δεδομένα εισόδου, τα οποία τα ενώνουμε σε ένα καινούργιο κοινό διάνυσμα. Το διάνυσμα αυτό περνάει μέσα από την σιγμοειδή συνάρτηση (έστω F_t) και πλέον οι τιμές του κυμαίνονται μεταξύ του μηδέν και του ένα. Όσο πιο κοντά στο ένα είναι οι τιμές μας τόσο πιο χρήσιμες κρίνονται.

$$F_t = sigmoid(W_f * X_t + U_f * H_{t-1} + b_f)$$

Όπου W_f , U_f και b_f είναι διανύσματα βαρών και bias για το forget gate, X_t τα τωρινά δεδομένα εισόδου και H_{t-1} η προηγούμενη τιμή του hidden state.

2.1.2 Input Gate

Η πύλη input, επηρεάζει την τιμή του cell state, χρησιμοποιώντας την προηγούμενη τιμή του hidden state, και τα τωρινά δεδομένα εισόδου που ενωμένα πάλι δημιουργούν ένα καινούργιο διάνυσμα το οποίο περνάμε και πάλι μέσα από την σιγμοειδή συνάρτηση. Παράλληλα, περνάμε το καινούργιο διάνυσμα και από την υπερβολή της εφαπτομένης και πολλαπλασιάζουμε τα δύο διανύσματα που δημιουργήσαμε (έστω I_t), το οποίο περιέχει τις τιμές που κρίνεται σκόπιμο να χρησιμοποιήσουμε.

$$I_t = sigmoid\,(W_i * X_t + U_i * H_{t-1} + b_i) * \tanh\,(W_c * X_t + U_c * H_{t-1} + b_c)$$

Όπου W_i , W_c , U_i , U_c και b_i, b_c είναι διανύσματα βαρών και bias για το input gate, X_t τα τωρινά δεδομένα εισόδου και H_{t-1} η προηγούμενη τιμή του hidden state.

2.1.3 Output Gate

Η πύλη output, καθορίζει την επόμενη τιμή του hidden state και του cell state, χρησιμοποιώντας την προηγούμενη τιμή του hidden state, και τα τωρινά δεδομένα εισόδου που ενωμένα πάλι δημιουργούν ένα καινούργιο διάνυσμα το οποίο περνάμε και πάλι μέσα από την σιγμοειδή συνάρτηση. Παράλληλα, περνάμε την ανανεωμένη τιμή του cell state από την υπερβολή της εφαπτομένης.

$$O_t = sigmoid(W_o * X_t + U_o * H_{t-1} + b_o)$$

Όπου W_o , U_o και b_o είναι διανύσματα βαρών και bias για το forget gate, X_t τα τωρινά δεδομένα εισόδου και H_{t-1} η προηγούμενη τιμή του hidden state.

2.1.4 Cell State

Η τιμή του cell state ορίζεται ως εξής:

$$C_t = F_t * C_{t-1} + I_t$$

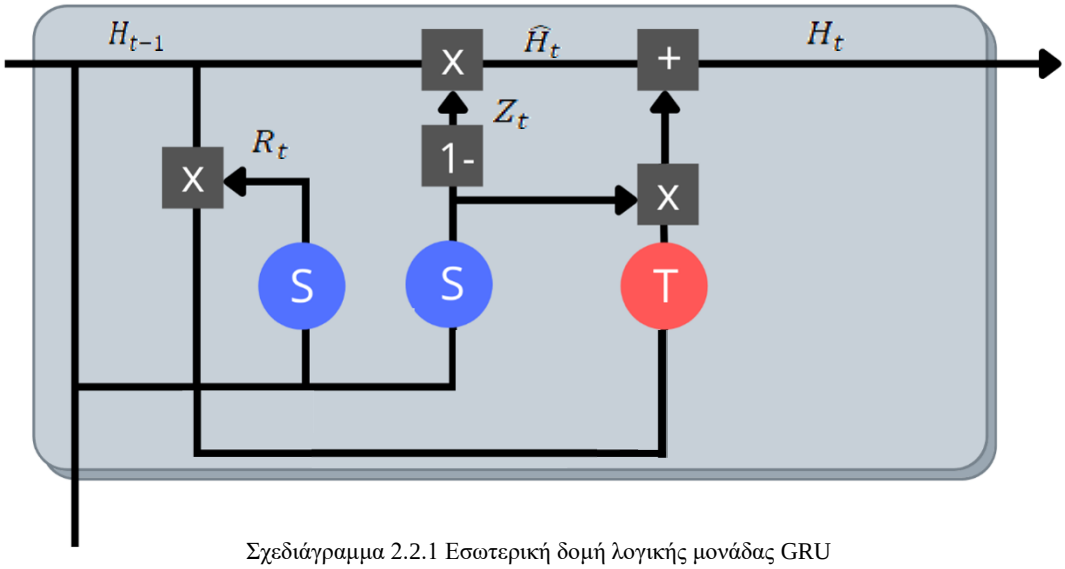
2.1.5 Hidden State

Η τιμή του hidden state ορίζεται ως εξής:

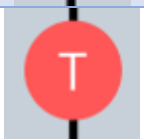
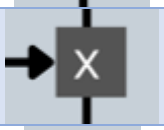
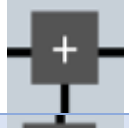
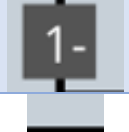
$$H_t = O_t * \tanh\,(C_t)$$

2.2 Gated recurrent unit

Ένα δίκτυο GRU, αποτελείται από έναν αριθμό από λογικές μονάδες GRU, όπως αυτή που βλέπουμε στο σχεδιάγραμμα 2.2.1. Η παραγόμενη έξοδος της κάθε λογικής μονάδας καθορίζεται από τις τιμές εισόδου και από την προηγούμενη έξοδο της μονάδας (**hidden state**). Η τιμή του hidden state καθορίζεται από τις λογικές πύλες reset και update που θα εξηγηθούν παρακάτω.



Σχεδιάγραμμα 2.2.1 Εσωτερική δομή λογικής μονάδας GRU

Απεικόνιση	Εξήγηση
	Σιγμοειδής συνάρτηση
	Υπερβολή της εφαπτομένης
	Πολλαπλασιασμός διανυσμάτων
	Πρόσθεση διανυσμάτων
	Αφαίρεση διανύσματος από το διάνυσμα ίσο με ένα
	Ένωση διανυσμάτων

Σχεδιάγραμμα 2.2.2 Υπόμνημα δομής λογικής μονάδας GRU

2.1.1 Reset Gate

Η πύλη reset καθορίζει ποια κομμάτια πληροφορίας θα απορριφθούν. Για να επιτευχθεί αυτό χρησιμοποιούμε την προηγούμενη τιμή του hidden state και τα τωρινά δεδομένα εισόδου τα οποία τα ενώνουμε σε ένα καινούργιο κοινό διάνυσμα. Το διάνυσμα αυτό περνάει μέσα από την σιγμοειδή συνάρτηση (έστω R_t) και πλέον οι τιμές του κυμαίνονται μεταξύ του μηδέν και του ένα. Όσο πιο κοντά στο ένα είναι οι τιμές μας τόσο πιο χρήσιμες κρίνονται.

$$R_t = sigmoid\,(W_r * X_t + U_r * H_{t-1} + b_r)$$

Όπου W_r , U_r και b_r είναι διανύσματα βαρών και bias για το reset gate, X_t τα τωρινά δεδομένα εισόδου και H_{t-1} η προηγούμενη τιμή του hidden state.

2.1.2 Update Gate

Η πύλη update, επηρεάζει την τιμή του hidden state, χρησιμοποιώντας την προηγούμενη τιμή του, και τα τωρινά δεδομένα εισόδου που ενωμένα πάλι δημιουργούν ένα καινούργιο διάνυσμα το οποίο περνάμε και πάλι μέσα από την σιγμοειδή συνάρτηση. Παράλληλα, αφαιρούμε το καινούργιο μας διάνυσμα από το μοναδιαίο διάνυσμα και πολλαπλασιάζουμε το αποτέλεσμα με την προηγούμενη τιμή του hidden state (έστω $\widehat{H}_t$).

$$Z_t = sigmoid\,(W_u * X_t + U_u * H_{t-1} + b_u\,)$$

$$\widehat{H}_t = H_{t-1} * (\vec{1} - U_t)$$

Όπου W_u , U_u και b_z είναι διανύσματα βαρών και bias για το input gate, X_t τα τωρινά δεδομένα εισόδου και H_{t-1} η προηγούμενη τιμή του hidden state.

2.1.3 Hidden State

Η τιμή του hidden state ορίζεται ως εξής:

$$H_t = \widehat{H}_t + (Z_t * \tanh(W_h * X_t + U_h * (H_{t-1} * R_t) + b_h))$$

2.3 Χρήσιμα εργαλεία και στατιστικές μέθοδοι

Sigmoid Function

Η σιγμοειδής συνάρτηση, είναι μια ευρέως χρησιμοποιούμενη συνάρτηση ενεργοποίησης. Η σιγμοειδής είναι παραγωγίσιμη και έχει πεδίο τιμών $[0,1]$. Στην υλοποίησή μας η σιγμοειδής χρησιμοποιείται στις υλοποιήσεις των λογικών πυλών των μονάδων LSTM και GRU.

$$S(t) = \frac{1}{1 - e^{-t}}$$

Hyperbolic Tangent Function

Η υπερβολή της εφαπτομένης ορίζεται ως το πηλίκο του υπερβολικού ημιτόνου προς το υπερβολικό συνημίτονο. Στην υλοποίησή μας χρησιμοποιείται στις υλοποιήσεις των λογικών πυλών των μονάδων LSTM και GRU.

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

Mean Squared Error

Το μέσο τετραγωνικό σφάλμα, είναι μια μέθοδος υπολογισμού απόκλισης της πραγματικής τιμής προς την προβλεπόμενη τιμή. Χρησιμοποιεί τη μέση τιμή της τετραγωνικής απόκλισης. Στην υλοποίησή μας το χρησιμοποιούμε για να υπολογίσουμε το σφάλμα εκπαίδευσης και επαλήθευσης.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \hat{Y}_i\right)^2$$

Όπου Y_i είναι η προβλεπόμενη τιμή και $\hat{Y}_i$ η πραγματική τιμή.

Mean Absolute Percentage Error

Το μέσο απόλυτο ποσοστιαίο σφάλμα είναι μια μέθοδος υπολογισμού σφάλματος σε ποσοστιαία μορφή. Στην υλοποίησή μας το χρησιμοποιούμε για να υπολογίσουμε την ποσοστιαία επιτυχία του δικτύου μας αφαιρώντας το ποσοστιαίο σφάλμα από το εκατό.

$$MAPE = \frac{100\%}{n}\sum_{i=1}^{n}\left|\frac{A_i - F_i}{A_i}\right|$$

Όπου A_i είναι η προβλεπόμενη τιμή και F_i η πραγματική τιμή.

Glorot Uniform

Η συνάρτηση Glorot uniform[16] ή αλλιώς Xavier, χρησιμοποιείται για να δίνει τυχαίες τιμές με κανονική κατανομή επιλέγοντας τιμές από το σύνολο τιμών [-limit, limit].

$$limit = \sqrt{\frac{6}{(fan_{in} + fan_{out})}}$$

Όπου fan_{in} είναι το πλήθος των input units στο weight tensor και fan_{out} το πλήθος των output units στο weight tensor.

2.4 Δεδομένα

Όπως αναφέραμε, τα δεδομένα μας καλύπτουν μια περίοδο επτά ετών στη δεκαετία του ενενήντα και χρειάστηκε να κάνουμε κάποια μορφοποίηση. Πιο συγκεκριμένα, ο μήνας εκφραζόταν σε δυαδική μορφή, χωρίς να εμφανίζεται η μέρα ή η ώρα. Επιπρόσθετα, υπήρχε ένα κενό επτά μηνών στα δεδομένα το οποίο έπειτα από ανάλυση καταλήξαμε πως προέκυψε από λανθασμένη αρίθμηση των μηνών. Για να επιλύσουμε τα παραπάνω θέματα επαναλάβαμε την αρίθμηση των μηνών προσθέτοντας δεδομένα για την ημέρα και την ώρα της κάθε μέτρησης. Η μορφή των δεδομένων μας πλέον είναι ως εξής:

timestamp,demand,m1,m2,m3,m4,temp,wind,holiday
1990-5-2 01:00:00,0.282,0,1,0,1,0.262,0.129,0
1990-5-2 02:00:00,0.352,0,1,0,1,0.262,0.129,0
1990-5-2 03:00:00,0.32,0,1,0,1,0.381,0.129,0
1990-5-2 04:00:00,0.274,0,1,0,1,0.381,0.129,0
1990-5-2 05:00:00,0.218,0,1,0,1,0.476,0.129,0
1990-5-2 06:00:00,0.216,0,1,0,1,0.476,0.129,0
1990-5-2 07:00:00,0.207,0,1,0,1,0.5,0.129,0
1990-5-2 08:00:00,0.191,0,1,0,1,0.5,0.129,0
1990-5-2 09:00:00,0.171,0,1,0,1,0.524,0.097,0
1990-5-2 10:00:00,0.181,0,1,0,1,0.524,0.097,0
1990-5-2 11:00:00,0.235,0,1,0,1,0.524,0.097,0

Πρακτικά προσθέσαμε τη στήλη timestamp, η οποία μας βοηθάει στην ανάλυση των δεδομένων μας. Στον παρακάτω πίνακα βλέπουμε ανάλυση των στηλών και των μονάδων μέτρησης.

Όνομα	Τύπος - Μονάδα Μέτρησης	Συχνότητα αλλαγής
demand	εκατομμύρια κυβικά μέτρα	ανά ώρα
m1	bit	ανά μήνα
m2	bit	ανά μήνα
m3	bit	ανά μήνα
m4	bit	ανά μήνα
temp	βαθμοί Κελσίου	ανά δύο ώρες
wind	Κόμβοι	ανά τέσσερις ώρες
holiday	bit	ανά μέρα

Σχεδιάγραμμα 2.4.1 Πίνακας ανάλυσης δεδομένων

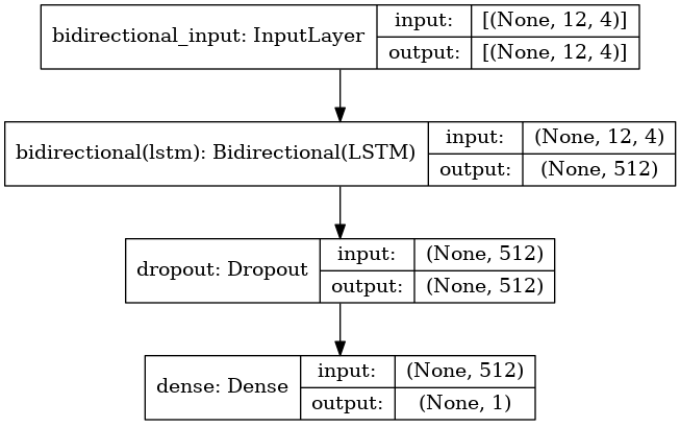
Κεφάλαιο 3

Σχεδιασμός και Υλοποίηση

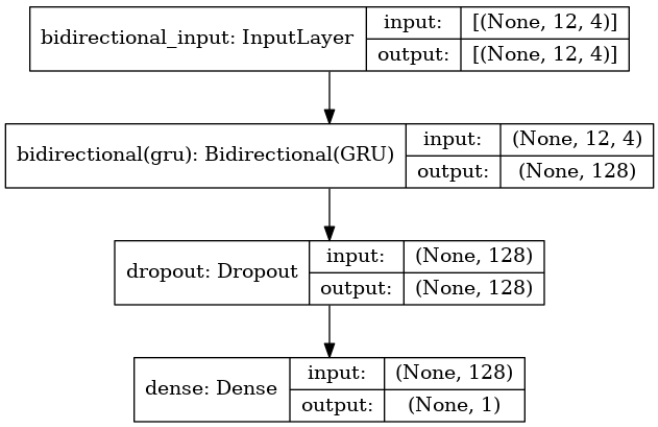
3.1 Σχεδιασμός

3.1.1 Δομή νευρωνικών δικτύων

Όπως αναφέραμε και προηγουμένως, για την επίλυση αυτού του προβλήματος υλοποιήσαμε δύο νευρωνικά δίκτυα με ανάδραση, το δίκτυο LSTM και το δίκτυο GRU. Τα δίκτυα μας δομήθηκαν όπως φαίνεται στα σχεδιαγράμματα 3.1.1.1 και 3.1.1.2.



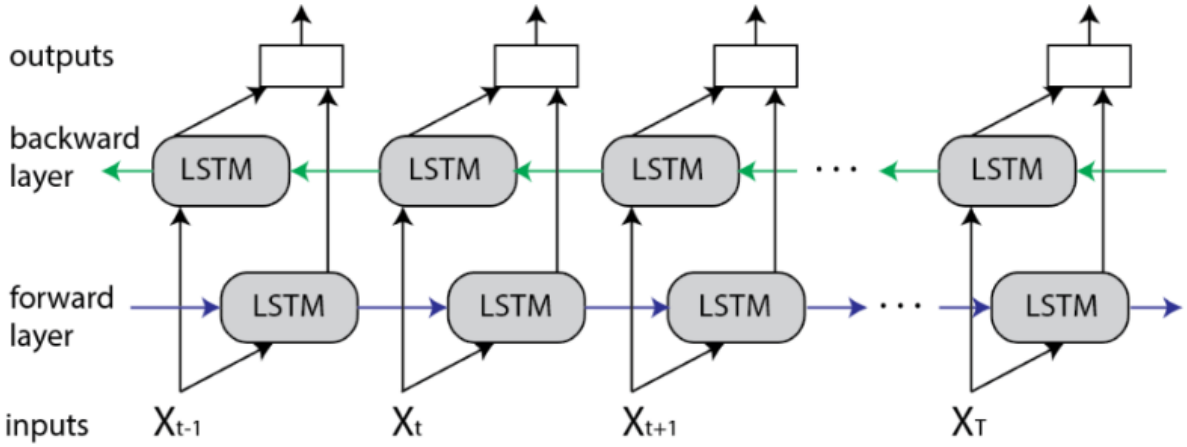
Σχεδιάγραμμα 3.1.1.1 Δομή νευρωνικού δικτύου LSTM σε Keras



Σχεδιάγραμμα 3.1.1.2 Δομή νευρωνικού δικτύου GRU σε Keras

Αρχικά έχουμε ένα επίπεδο εισόδου το οποίο εισάγει στο δίκτυο ένα στιγμιότυπο της χρονοσειράς που χρησιμοποιούμε, στη συνέχεια έχουμε ένα πλέγμα από LSTM & GRU blocks τα οποία παράγουν ένα διάνυσμα από τιμές (μία τιμή για κάθε block). Το διάνυσμα αυτό έπειτα περνάει στο Dropout επίπεδο, το οποίο επιλέγει τυχαία κάποιες τιμές του διανύσματος οι οποίες δεν θα ληφθούν υπόψιν σε αυτή την εποχή. Τέλος, το διάνυσμα αυτό περνάει στο τελευταίο επίπεδο, το επίπεδο Dense, το οποίο ουσιαστικά μετατρέπει την έξοδο του δικτύου από διάνυσμα σε μία μοναδική τιμή, στην δική μας περίπτωση την τιμή της ζήτησης φυσικού αερίου για μια συγκεκριμένη ώρα.

Όπως βλέπουμε στα σχεδιαγράμματα 3.1.1.1 και 3.1.1.2, για τα επίπεδα LSTM και GRU χρησιμοποιούμε αμφίδρομα επίπεδα. Η δομή αυτή επιτρέπει στο δίκτυό μας να χρησιμοποιεί δεδομένα και από το παρελθόν αλλά και από το μέλλον για να προβλέψει την τιμή που επιθυμούμε.



Σχεδιάγραμμα 3.1.1.3 Bidirectional LSTM layer [24]

3.1.2 Αρχικοποίηση βαρών από το Keras

Και τα δύο δίκτυα χρησιμοποιούν τον αρχικοποιητή Glorot Uniform. Ο αρχικοποιητής Glorot Uniform αρχικοποιεί τα βάρη του δικτύου χρησιμοποιώντας την γκαουσιανή συνάρτηση με άκρα το ± √(6 / (fan_in + fan_out)), όπου fan_in είναι το πλήθος των μονάδων εισόδου στο επίπεδο των βαρών (weight tensor) και fan_out το πλήθος των μονάδων εξόδου.

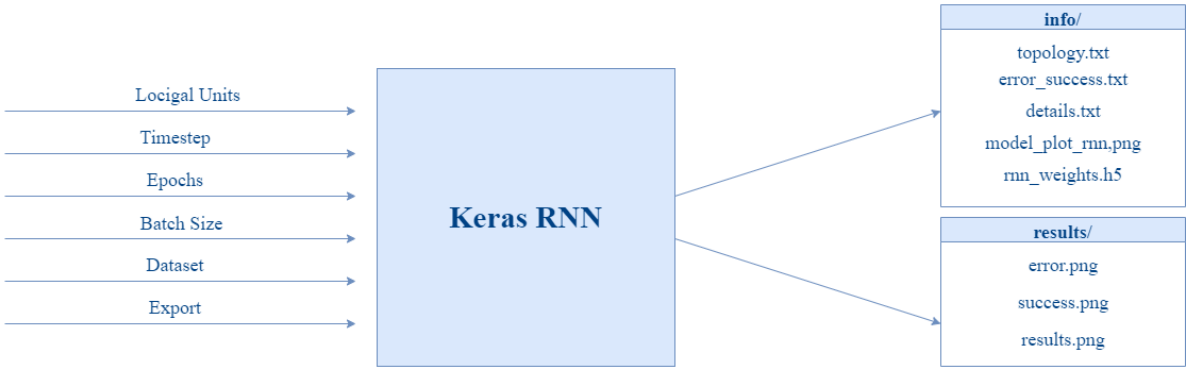
3.2 Υλοποίηση

Η υλοποίηση και των δύο νευρωνικών δικτύων έγινε με τη χρήση της προγραμματιστικής γλώσσας Python (έκδοση 3.6.8) και πιο συγκεκριμένα έγινε χρήση της βιβλιοθήκης Keras η οποία θα εξηγηθεί στη συνέχεια. Όσον αφορά τη γενική δομή του κώδικα, η υλοποίηση κάθε δικτύου έχει γίνει σε ένα ανεξάρτητο αρχείο, το οποίο μοντελοποιεί και την διαδικασία της εκπαίδευσης, αλλά και τον έλεγχο των αποτελεσμάτων.

3.2.1 Περιγραφή ροής δεδομένων

Το πρόγραμμά μας μόλις ξεκινήσει διαβάζει τις παραμέτρους, οι οποίες είναι: το πλήθος των μονάδων LSTM ή GRU, το πλήθος των ωρών που απαρτίζουν μια είσοδο στο δίκτυο (timestep), το πλήθος των εποχών εκπαίδευσης, το μέγεθος δέσμης, το αρχείο εισόδου και μια μεταβλητή ελέγχου που καθορίζει αν θα γίνει εξαγωγή του δικτύου. Στη συνέχεια το πρόγραμμα διαβάζει το αρχείο εισόδου, το οποίο πρέπει να έχει μορφή csv (comma-separated values), με σκοπό να δημιουργήσει μια δομή δεδομένων συμβατή με τη βιβλιοθήκη Pandas (dataframe). Έπειτα διαμορφώνει τη χρονοσειρά που θα χρησιμοποιηθεί για την εκπαίδευση του δικτύου μας, επιλέγοντας από τα συνολικά δεδομένα την ζήτηση φυσικού

αερίου, την θερμοκρασία, την ένταση του αέρα και το αν η εκάστοτε μέρα είναι αργία ή όχι. Με αυτά τα δεδομένα, δημιουργείται ένα διάνυσμα το οποίο θα χρησιμοποιήσει το δίκτυό μας ως είσοδο. Οι διαστάσεις του διανύσματος αυτού καθορίζονται από τη μεταβλητή timestep και από το πλήθος των δεδομένων που χρησιμοποιούμε. Πιο συγκεκριμένα, κάθε γραμμή του διανύσματος προσομοιάζει μια ώρα δεδομένων (ζήτηση φυσικού αερίου, ταχύτητα ανέμων, θερμοκρασία, αργία), καλύπτοντας τις προηγούμενες ώρες πριν από την ώρα της οποίας θέλουμε να προβλέψουμε τη ζήτηση φυσικού αερίου, όπως φαίνεται στο σχεδιάγραμμα 3.2.1.2. Μετά, το πρόγραμμά μας διαχωρίζει το διάνυσμα αυτό σε δύο ομάδες, τα δεδομένα εκπαίδευσης και τα δεδομένα επαλήθευσης. Αφότου τελειώσει η προ επεξεργασία των δεδομένων αρχικοποιείται το αντίστοιχο νευρωνικό δίκτυο χρησιμοποιώντας τη βιβλιοθήκη Keras και ξεκινάει η διαδικασία εκπαίδευσης. Κατά την διάρκεια της εκπαίδευσης υπολογίζεται το σφάλμα με δύο μεθόδους, χρησιμοποιώντας το MSE και το MAPE. Μετά το τέλος της εκπαίδευσης χρησιμοποιείται το MAPE για να υπολογιστεί και να παρουσιαστεί η ποσοστιαία επιτυχία και το MSE για να παρουσιαστεί το σφάλμα του δικτύου μας. Τέλος, το πρόγραμμα εξάγει κάποιες πληροφορίες σχετικές με τη δομή του νευρωνικού δικτύου και εκτελεί μία πρόβλεψη για να επαληθεύσει τα αποτελέσματά μας.



Σχεδιάγραμμα 3.2.1.1 Αφαιρετική δομή των νευρωνικών μας δικτύων

```
[[0.352 0.129 0.262 0.   ]
 [0.32  0.129 0.381 0.   ]
 [0.274 0.129 0.381 0.   ]
 [0.218 0.129 0.476 0.   ]
 [0.216 0.129 0.476 0.   ]]
```

Σχεδιάγραμμα 3.2.1.2 Παράδειγμα διανύσματος δεδομένων εισόδου με timestep ίσο με 5

3.3 Python modules

Βασιζόμενοι στην αρχιτεκτονική λογική της Python χρησιμοποιήσαμε διάφορες βιβλιοθήκες για να μοντελοποιήσουμε τα δίκτυά μας. Στο παράρτημα Γ, βρίσκονται δύο scripts τα οποία εγκαθιστούν αυτόματα όλες τις βιβλιοθήκες που χρειάζεται το σύστημά μας σε περιβάλλον UNIX και Windows.

Pandas
Η βιβλιοθήκη Pandas είναι μια βιβλιοθήκη ανοιχτού κώδικα, η οποία χρησιμοποιείται για ανάλυση δεδομένων στην Python. Στη δική μας υλοποίηση χρησιμοποιούμε την Pandas για να φορτώσουμε και να διαχειριστούμε τα δεδομένα εκπαίδευσης μας.

NumPy
Η βιβλιοθήκη NumPy είναι μια βιβλιοθήκη που υποστηρίζει τη διαχείριση πολυδιάστατων πινάκων καθώς επίσης προσφέρει περίπλοκες μαθηματικές συναρτήσεις για την επεξεργασία των εν λόγω πινάκων. Στη δική μας υλοποίηση χρησιμοποιούμε τη NumPy για να μετατρέψουμε τα δεδομένα μας από ένα απλό panda dataframe σε ένα λειτουργικό πίνακα που θα χρησιμοποιηθεί ως είσοδος στο νευρωνικό μας δίκτυο.

TensorFlow
Το TensorFlow είναι μια ολοκληρωμένη πλατφόρμα ανοιχτού κώδικα η οποία προσφέρει εργαλεία και βιβλιοθήκες σχετικά με τη μηχανική μάθηση. Στη δική μας υλοποίηση χρησιμοποιούμε τη βιβλιοθήκη Keras.

Keras
Το Keras είναι ένα API βαθιάς μάθησης το οποίο βασίζεται στο TensorFlow και προσφέρει στους χρήστες διάφορα νευρωνικά δίκτυα και εργαλεία μοντελοποίησης, όπως οι συναρτήσεις ενεργοποίησης. Στη δική μας υλοποίηση χρησιμοποιούμε το Keras για να μοντελοποιήσουμε τα δύο αναδρομικά δίκτυα LSMT και GRU, παράλληλα χρησιμοποιούμε τα επίπεδα Dropout και Dense που προσφέρει το Keras, καθώς και τον βελτιστοποιητή Adam[17].

Scikit-learn
Το Scikit-learn είναι μια βιβλιοθήκη μηχανικής μάθησης που προσφέρει διάφορους αλγόριθμους μάθησης όπως και διάφορα εργαλεία σχετικά με την εκπαίδευση δικτύων. Παράλληλα, η Scikit-learn είναι σχεδιασμένη για να συνεργάζεται αρμονικά με επιστημονικές βιβλιοθήκες όπως η NumPy. Στη δική μας υλοποίηση χρησιμοποιούμε ένα εργαλείο διαμοιρασμού των δεδομένων εκμάθησης και ελέγχου.

Matplotlib
Η βιβλιοθήκη Matplotlib είναι μια βιβλιοθήκη οπτικοποίησης δεδομένων, που παρέχει στατικά και δυναμικά γραφήματα. Στη δική μας υλοποίηση χρησιμοποιούμε την Matplotlib για να δημιουργήσουμε κάποιες γραφικές παραστάσεις.

Seaborn
Η Seaborn είναι μια βιβλιοθήκη οπτικοποίησης βασισμένη στην Matplotlib. Στη δική μας υλοποίηση χρησιμοποιούμε την Seaborn για να δημιουργήσουμε κάποιες γραφικές παραστάσεις.

Math
Η βιβλιοθήκη math μας δίνει πρόσβαση σε κάποιες μαθηματικές συναρτήσεις οι οποίες ακολουθούν τα στάνταρ της γλώσσας C. Στη δική μας υλοποίηση χρησιμοποιούμε την math για να εκτελέσουμε κάποιους απλούς μαθηματικούς τύπους.

Sys
Η βιβλιοθήκη sys μας δίνει πρόσβαση σε κάποιες μεταβλητές του μεταγλωττιστή της Python. Στη δική μας υλοποίηση χρησιμοποιούμε την sys για να περνάμε παραμέτρους στο πρόγραμμά μας.

Time
Η βιβλιοθήκη time μας δίνει πρόσβαση στην ώρα του συστήματος. Στη δική μας υλοποίηση χρησιμοποιούμε την time για να μπορούμε να μετρήσουμε τον χρόνο εκτέλεσης του προγράμματός μας.

OS
Η βιβλιοθήκη os μας δίνει πρόσβαση σε συναρτήσεις σχετικές με το λειτουργικό μας σύστημα. Στη δική μας υλοποίηση χρησιμοποιούμε την os για να δημιουργήσουμε τους φακέλους αποτελεσμάτων.

Κεφάλαιο 4

Αποτελέσματα

Αφότου υλοποιήθηκαν τα νευρωνικά μας δίκτυα, μελετήσαμε τις μεταβλητές που επηρεάζουν την εκμάθηση τους αλλά και τις σταθερές που καθορίζουν τη σωστή τους λειτουργία. Κατά τη διάρκεια της πειραματικής διαδικασίας εκτελέσαμε χιλιάδες διαφορετικούς συνδυασμούς μεταβλητών εισόδου, οι οποίοι έδωσαν πληθώρα διαφορετικών αποτελεσμάτων. Η διαδικασία εκτέλεσης τόσων διαφορετικών συνδυασμών είναι μια διαδικασία, η οποία απαιτεί αρκετό χρόνο και αρκετή υπολογιστική δύναμη. Ο λόγος είναι ότι για να ολοκληρωθούν οι εποχές εκπαίδευσης απαιτείται πολύς χρόνος, εξαιτίας του τεράστιου όγκου πληροφοριών που χρησιμοποιούμε, για να εκπαιδεύσουμε τα δίκτυά μας.

Παράλληλα, τα δίκτυά μας είναι ευαίσθητα σε αλλαγές μεταβλητών όπως ο ρυθμός μάθησης, το πλήθος των εποχών αλλά και το μέγεθος δέσμης (batch size), τόσο στο χρόνο εκτέλεσης όσο και στα αποτελέσματα. Από όλα τα αποτελέσματα που πήραμε, θα παρουσιάσουμε ένα μικρό αλλά περιεκτικό φάσμα, όπου φαίνονται ξεκάθαρα τα αποτελέσματα, η επιτυχία πρόβλεψης, οι τιμές των λαθών και η επίδραση που έχουν οι διάφορες παράμετροι πάνω στα τελικά αυτά αποτελέσματα.

4.1 Πειραματική προσέγγιση και μεταβλητές

Για την εξαγωγή των αποτελεσμάτων δημιουργήσαμε ένα σύνολο από bash scripts τα οποία αυτοματοποίησαν τη διαδικασία εκτέλεσης. Οι μεταβλητές που εξετάσαμε είναι το πλήθος των εποχών εκπαίδευσης, το μέγεθος της χρονοσειράς εισόδου, το πλήθος των μονάδων LSTM & GRU, το Dropout Rate του επιπέδου Dropout, το μέγεθος δέσμης και τον ρυθμό μάθησης του δικτύου. Στον πιο κάτω πίνακα υπάρχουν τα πεδία τιμών που χρησιμοποιήσαμε για κάθε μεταβλητή.

Μεταβλητή	Πεδίο Τιμών
Μέγεθος χρονοσειράς	12, 24, 48, 72
Πλήθος εποχών	10, 30, 50
Πλήθος λογικών μονάδων LSTM & GRU	64, 128, 256
Dropout rate	0.1, 0.5, 0.9
Batch size	16, 32, 64
Ρυθμός μάθησης	0.001, 0.005, 0.1

Σχεδιάγραμμα 4.1.1 Πίνακας μεταβλητών υπό εξέταση

Οι τιμές τις οποίες μετρήσαμε αντικατοπτρίζουν τόσο την ορθότητα του δικτύου, όσο και την ευχρηστία του. Πιο συγκεκριμένα, μετρήσαμε το σφάλμα, την ποσοστιαία επιτυχία, τον χρόνο εκτέλεσης αλλά και μια δική μας μετρική απόκλισης-επιτυχίας.

4.2 Συνθήκες Πειραμάτων

Όλες οι εκτελέσεις των πειραμάτων μας έγιναν σε υπολογιστές με τα παρακάτω χαρακτηριστικά:

```
Architecture:          x86_64
CPU op-mode(s):        32-bit, 64-bit
Byte Order:            Little Endian
CPU(s):                4
On-line CPU(s) list:   0-3
Thread(s) per core:    1
Core(s) per socket:    4
Socket(s):             1
NUMA node(s):          1
Vendor ID:             Genuine Intel
CPU family:            6
Model:                 60
Model name:            Intel(R) Core(TM) i5-4590 CPU @ 3.30GHz
Stepping:              3
CPU MHz:               2000.061
CPU max MHz:           3700.0000
CPU min MHz:           800.0000
BogoMIPS:              6585.07
Virtualization:        VT-x
L1d cache:             32K
L1i cache:             32K
L2 cache:              256K
L3 cache:              6144K
NUMA node0 CPU(s):     0-3
```

Παράλληλα, οι υπολογιστές αυτοί είναι εφοδιασμένοι με κάρτες γραφικών NVIDIA GeForce GTX 750 Ti, με χαρακτηριστικά:

id:	`display`
description:	VGA compatible controller
product:	GM107 [GeForce GTX 750 Ti]
vendor:	NVIDIA Corporation
physical id:	`0`
bus info:	`pci@0000:01:00.0`
version:	a2
width:	64 bits
clock:	33MHz
capabilities:	*vga_controller bus_master cap_list rom*
configuration:	driver = nvidia latency = 0
resources:	irq : 30 memory : f6000000-f6ffffff memory : e0000000-efffffff memory : f0000000-f1ffffff ioport : e000(size=128) memory : f7000000-f707ffff

4.3 Αποτελέσματα LSTM

Τα αποτελέσματα που προέκυψαν από το δίκτυο LSTM ήταν αρκετά ικανοποιητικά καθώς, ενώ ξεκινήσαμε από έξι μεταβλητές, καταφέραμε να βρούμε κάποια βέλτιστα μοντέλα εκτέλεσης. Για να κάνουμε τις μετρήσεις του σφάλματος βασιστήκαμε κυρίως στη βιβλιοθήκη Keras και στις έτοιμες μετρικές της. Για τον υπολογισμό του σφάλματος χρησιμοποιήσαμε την μέθοδο του τετραγωνικού σφάλματος και για την επιτυχία το απόλυτο ποσοστιαίο σφάλμα.
Στα υποκεφάλαια που ακολουθούν θα αναλύσουμε κάποια ενδεικτικά αποτελέσματα που προέκυψαν από την πειραματική διαδικασία. Παράλληλα θα αναλύσουμε την επίδραση της αλλαγής των τιμών των μεταβλητών.

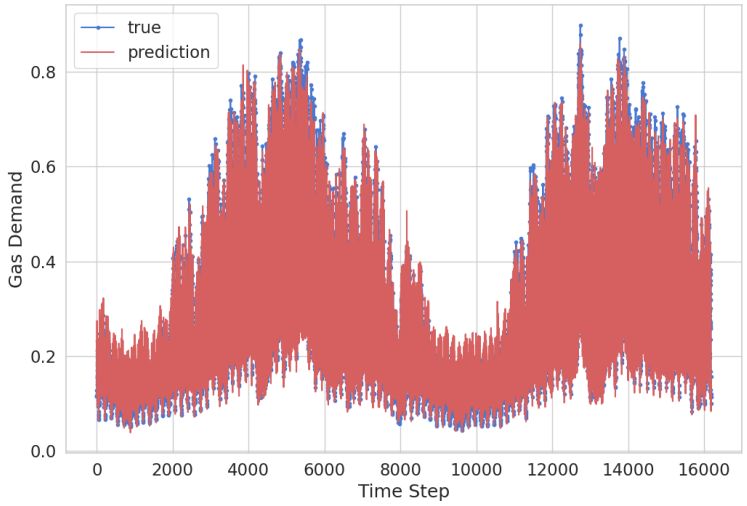
4.3.1 Αρχικά συμπεράσματα

Από τα πρώτα συμπεράσματα στα οποία καταλήξαμε, ήταν ότι ο ρυθμός μάθησης επηρεάζει καθοριστικά το σφάλμα και την επιτυχία μας, όπως φαίνεται και στο διάγραμμα 4.3.1.1 και 4.3.1.2. Για τα αποτελέσματα που παρουσιάζονται χρησιμοποιήσαμε ενδεικτικές τιμές για τις υπόλοιπες μεταβλητές, εκτός από τον ρυθμό μάθησης. Πιο συγκεκριμένα, το πλήθος των μονάδων LSTM ήταν ίσο με 128, το μέγεθος της χρονοσειράς ήταν ίσο με 72, ο ρυθμός του Dropout ήταν ίσος με 0.1, το μέγεθος δέσμης ίσο με 64 και οι εποχές ίσες με 10.

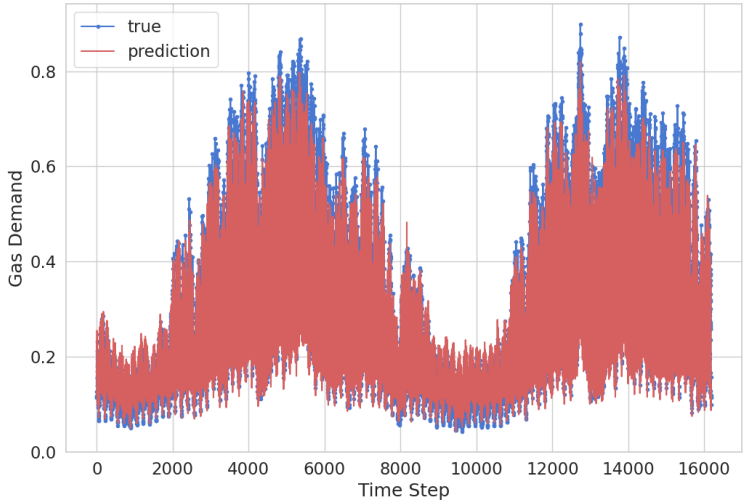
Σχεδιάγραμμα 4.3.1.1 Σύγκριση ποσοστού επιτυχίας πρόβλεψης με μεταβλητό ρυθμό μάθησης

Σχεδιάγραμμα 4.3.1.2 Σύγκριση σφάλματος πρόβλεψης με μεταβλητό ρυθμό μάθησης

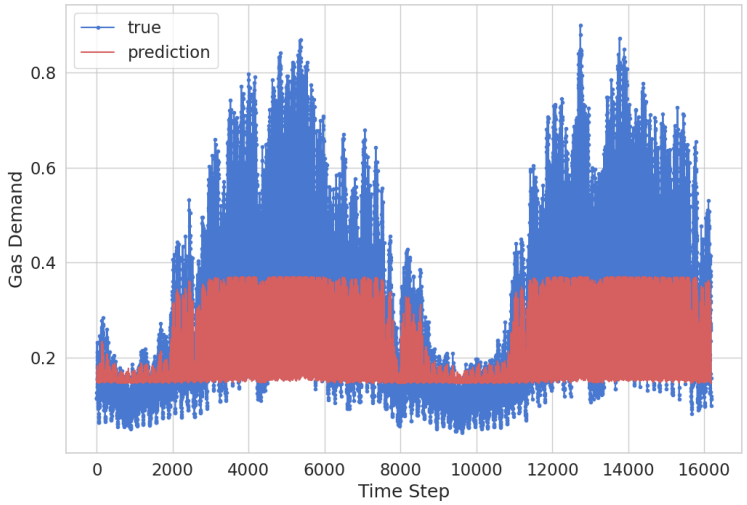
Παρατηρούμε ότι ο ρυθμός μάθησης πρέπει να έχει αρκετά χαμηλή τιμή για να μας δώσει τα βέλτιστα αποτελέσματα. Κάτι που αξίζει να σημειωθεί, είναι ότι η προεπιλεγμένη τιμή του βελτιστοποιητή Adam για τον ρυθμό μάθησης είναι 0.001 κάτι που επιβεβαιώνει τα αποτελέσματά μας. Παράλληλα, η δική μας φάση ελέγχου επιβεβαιώνει και αυτή τα αποτελέσματά μας. Πιο κάτω μπορούμε να δούμε τα αποτελέσματα πρόβλεψης των παραπάνω εκτελέσεων σε σύγκριση με τις ***πραγματικές τιμές***.



Σχεδιάγραμμα 4.3.1.3 Σύγκριση πρόβλεψης με πραγματική τιμή για ρυθμό μάθησης ίσο με 0.001



Σχεδιάγραμμα 4.3.1.4 Σύγκριση πρόβλεψης με πραγματική τιμή για ρυθμό μάθησης ίσο με 0.005

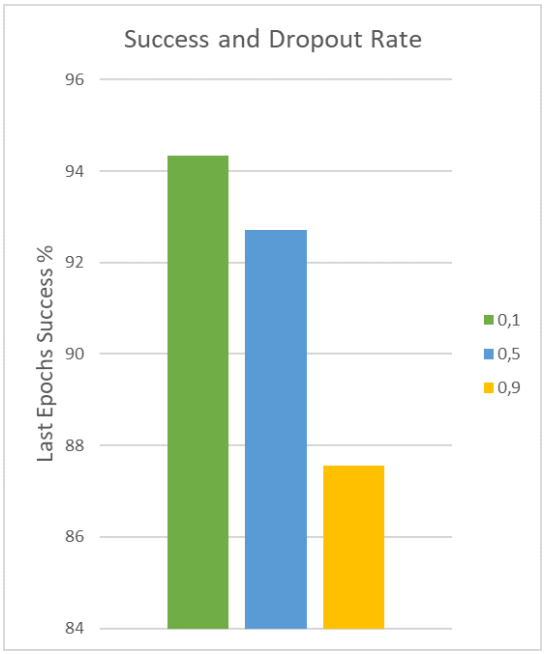


Σχεδιάγραμμα 4.3.1.3 Σύγκριση πρόβλεψης με πραγματική τιμή για ρυθμό μάθησης ίσο με 0.1

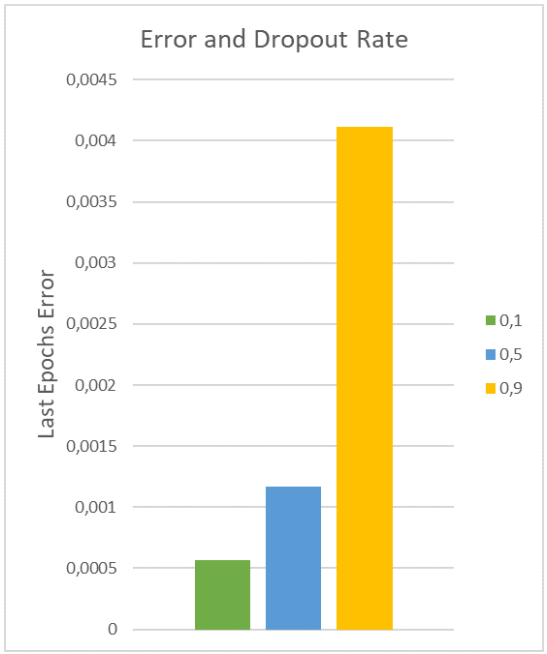
Τα παραπάνω αποτελέσματα μας δίνουν μια πιο καθαρή εικόνα για το πώς επηρεάζει ο ρυθμός μάθησης την εκμάθηση του δικτύου καθώς στις πρώτες δύο γραφικές παραστάσεις βλέπουμε σχεδόν πλήρη ταύτιση πρόβλεψης και πραγματικής τιμής, ενώ στην τελευταία με ρυθμό μάθησης ίσο με 0.1 τα αποτελέσματα είναι αποθαρρυντικά. Για την συνέχεια των πειραμάτων μας θα επικεντρωθούμε στις εκτελέσεις με ρυθμό μάθησης ίσο με 0.001.

4.3.2 Αλλαγές στον ρυθμό Dropout

Στη συνέχεια θα μελετήσουμε την επίδραση του ρυθμού Dropout, ο οποίος όπως αναφέραμε καθορίζει την πιθανότητα να αγνοηθούν κάποιες τιμές από το επίπεδο εξόδου. Για τα αποτελέσματα που παρουσιάζονται χρησιμοποιήσαμε ενδεικτικές τιμές για τις υπόλοιπες μεταβλητές, εκτός από τον ρυθμό μάθησης. Πιο συγκεκριμένα, το πλήθος των μονάδων LSTM ήταν ίσο με 128, το μέγεθος της χρονοσειράς ήταν ίσο με 72, το μέγεθος δέσμης ήταν ίσο με 64 και οι εποχές ίσες με 10.

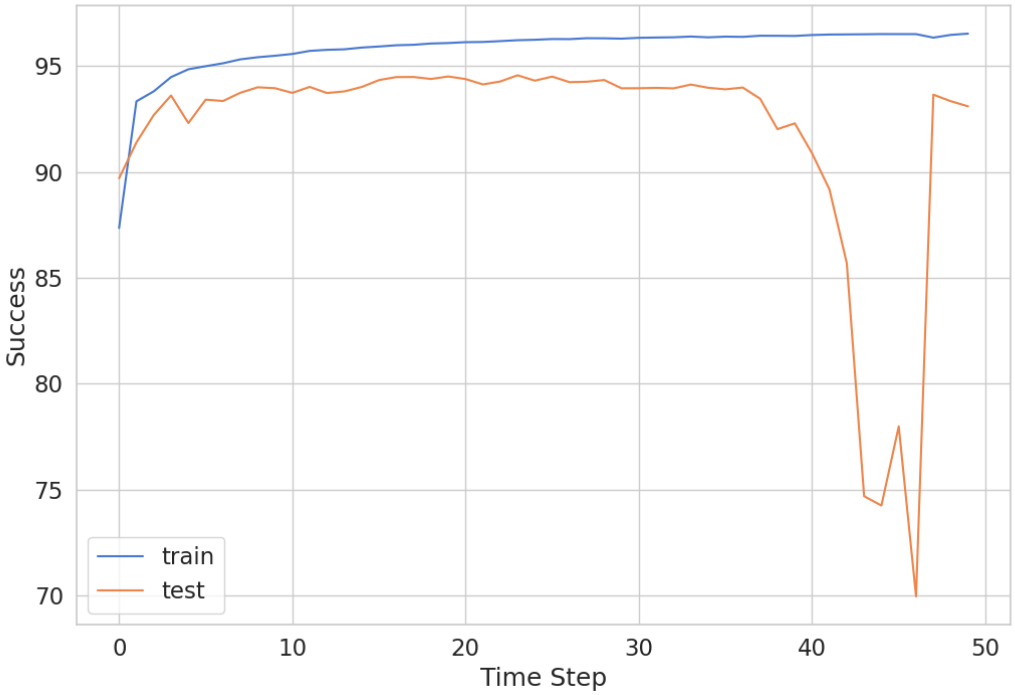


Σχεδιάγραμμα 4.3.2.1 Σύγκριση ποσοστού επιτυχίας πρόβλεψης με μεταβλητό ρυθμό Dropout



Σχεδιάγραμμα 4.3.2.2 Σύγκριση σφάλματος πρόβλεψης με μεταβλητό ρυθμό Dropout

Παρατηρούμε ότι το δίκτυό μας αποδίδει καλύτερα όσο μικρότερος είναι ο ρυθμός του Dropout, κάτι που συμβαδίζει με τη θεωρία, καθώς χρησιμοποιούμε περισσότερο όγκο πληροφοριών για να βγάλουμε το τελικό μας αποτέλεσμα. Βάσει αυτής της παραδοχής κάποιος θα μπορούσε να υποστηρίξει ότι θα έπρεπε να μειώναμε και άλλο το ρυθμό Dropout. Παρόλα αυτά το επίπεδο Dropout μας βοηθάει να καθαρίσουμε τον θόρυβο στα αποτελέσματά μας, σε εκτελέσεις με μηδενικό ρυθμό Dropout παρατηρήθηκε υπερπροσαρμογή από έναν αριθμό εποχών και πάνω, όπως φαίνεται και στο σχεδιάγραμμα 4.3.2.3.



Σχεδιάγραμμα 4.3.2.3 Γραφική παράσταση επιτυχίας πρόβλεψης με Dropout ίσο με μηδέν, πλήθος λογικών μονάδων LSTM ίσο με 512, μέγεθος χρονοσειράς ίσο με72, πλήθος εποχών ίσο με 50, μέγεθος δέσμης ίσο με 64 και ρυθμό μάθησης ίσο με 0.001

Τα παραπάνω αποτελέσματα μας δίνουν μια πιο καθαρή εικόνα για το πώς επηρεάζει ο ρυθμός Dropout την εκμάθηση του δικτύου. Για την συνέχεια των πειραμάτων μας θα επικεντρωθούμε στις εκτελέσεις με ρυθμό Dropout ίσο με 0.1.

4.3.3 Τελικά αποτελέσματα

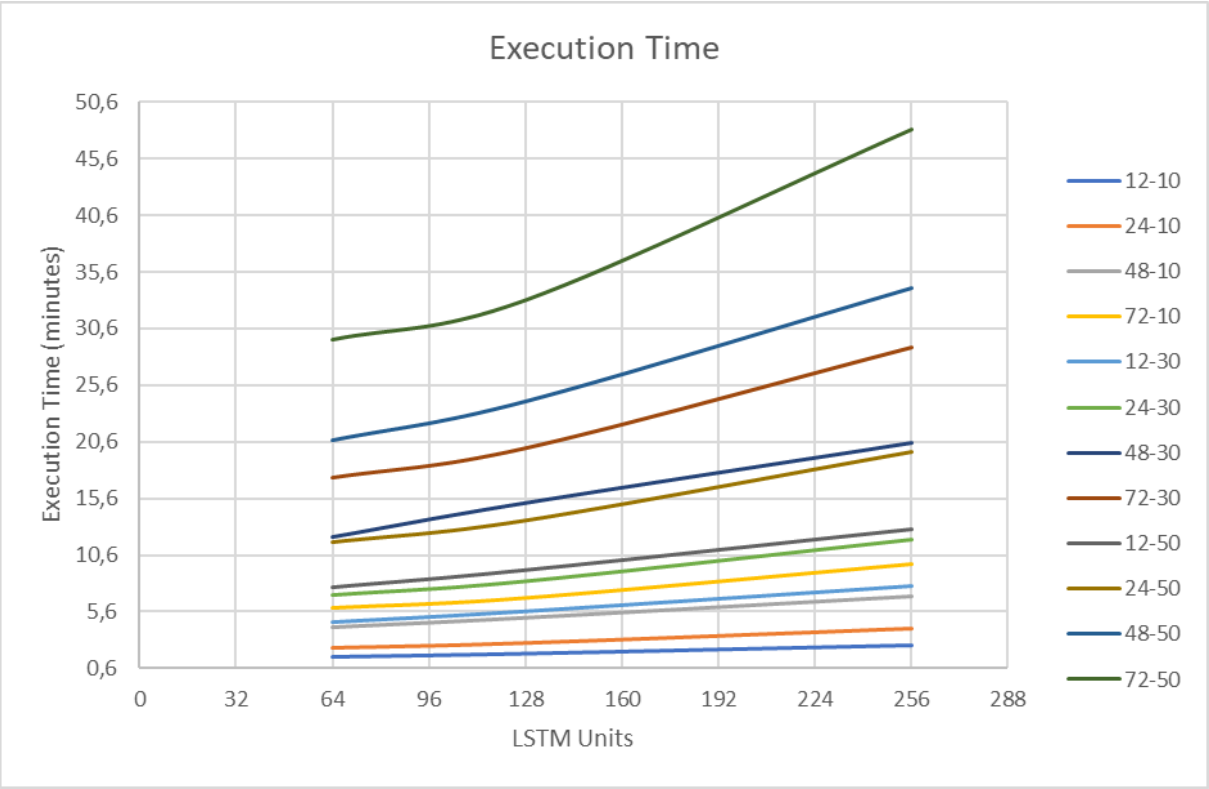
Για τις υπόλοιπες μεταβλητές θα παρουσιάσουμε μια συγκεντρωτική σύγκριση καθώς οι διαφορές μεταξύ των αποτελεσμάτων δεν είναι τόσο ευδιάκριτες. Για κάθε μια από τις δυνατές τιμές του μεγέθους δέσμης θα παρουσιάσουμε τα αποτελέσματα όλων των πιθανών συνδυασμών.

4.3.3.1 Μέγεθος δέσμης 16

Στις πιο κάτω γραφικές παραστάσεις βλέπουμε τα αποτελέσματα για εκτελέσεις με σταθερό ρυθμό μάθησης, ίσο με 0.001, σταθερό μέγεθος δέσμης, ίσο με 16 και σταθερό ρυθμό Dropout ίσο με 0.1. Παράλληλα έχουμε μεταβλητές τιμές το μέγεθος χρονοσειράς, τον αριθμό των εποχών και το πλήθος των λογικών μονάδων. Οι στατιστικές που θα αναλύσουμε είναι ο χρόνος εκτέλεσης, το σφάλμα και το ποσοστό επιτυχίας.

Στις επόμενες γραφικές παραστάσεις ο άξονας x εκφράζει το πλήθος των μονάδων LSTM, αντίστοιχα κάθε γραμμή ενώνει τα αποτελέσματα που προκύπτουν μεταβάλλοντας διακριτά το πλήθος των λογικών μονάδων LSTM, αλλά κρατώντας σταθερό το πλήθος των εποχών και το μέγεθος χρονοσειράς. Επομένως, για παράδειγμα η γραμμή 12-10, εκφράζει τη μεταβολή της υπό έλεγχο τιμής, για διάφορα διακριτά πλήθη λογικών μονάδων LSTM με σταθερό μέγεθος χρονοσειράς ίσο με 12 και πλήθος εποχών ίσο με 10.

Χρόνος εκτέλεσης

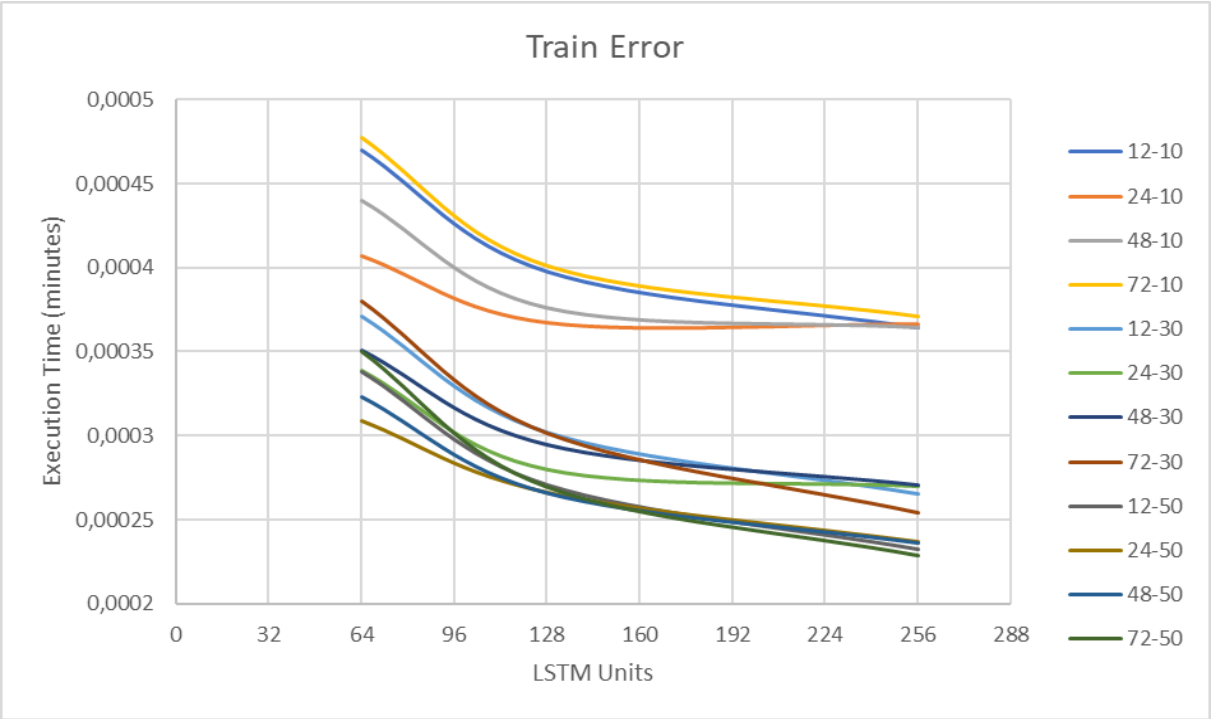


Σχεδιάγραμμα 4.3.3.1.1 Χρόνος εκτέλεσης συναρτήσει του πλήθους λογικών μονάδων LSTM, του μεγέθους χρονοσειράς και του πλήθους εποχών, με σταθερό μέγεθος δέσμης ίσο με 16

Όσον αφορά τον χρόνο εκτέλεσης, τα αποτελέσματα μας είναι απολύτως αναμενόμενα, όσο αυξάνονται οι εποχές και το μέγεθος χρονοσειράς, ο χρόνος αυξάνεται. Αντίστοιχα, η αύξηση στις λογικές μονάδες LSTM επιφέρει και αυτή αύξηση στο χρόνο εκτέλεσης, παρόλα αυτά η επίδρασή της δεν φαίνεται να είναι σημαντική.

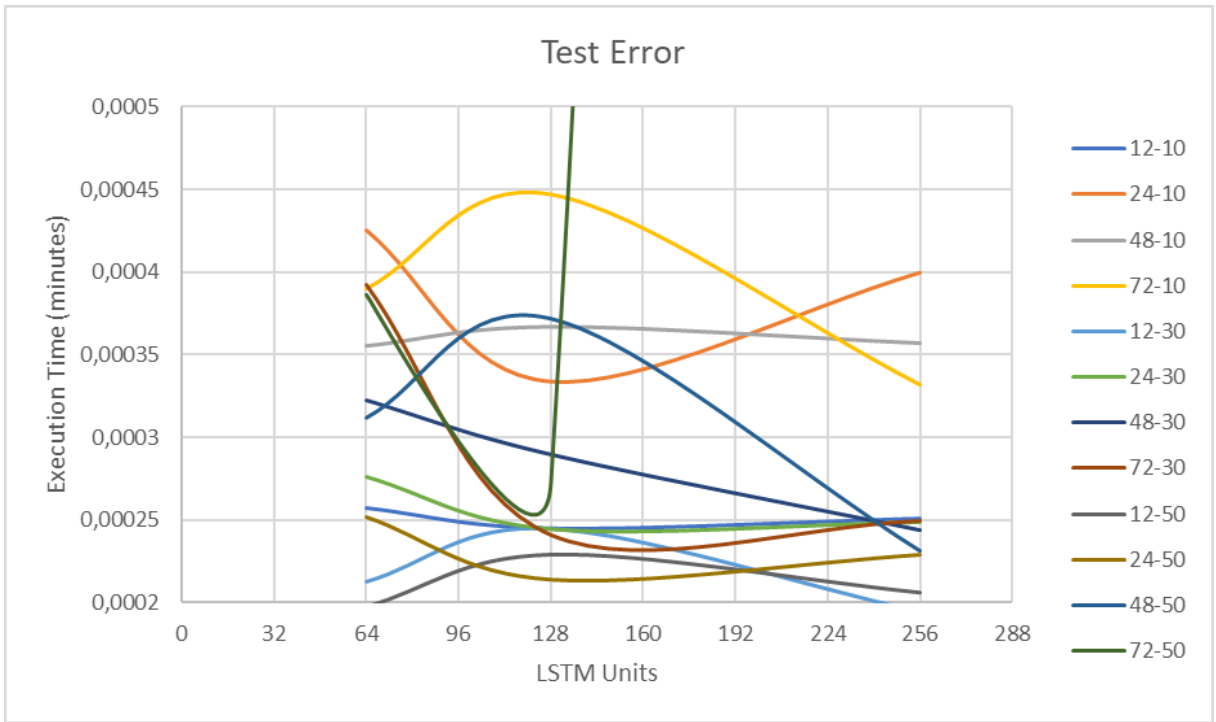
Σφάλμα

Εκπαίδευση



Σχεδιάγραμμα 4.3.3.1.2 Τελική τιμή σφάλματος εκπαίδευσης συναρτήσει του πλήθους λογικών μονάδων LSTM, του μεγέθους χρονοσειράς και του πλήθους εποχών, με σταθερό

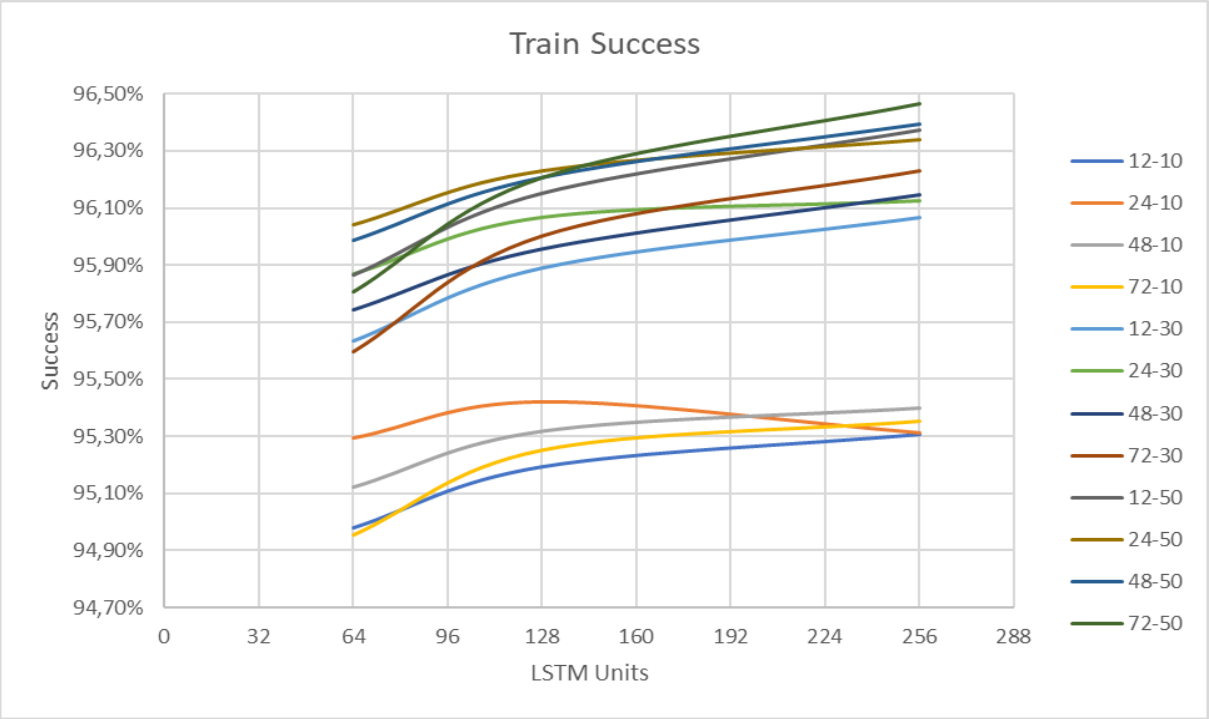
Επαλήθευση



Σχεδιάγραμμα 4.3.3.1.3 Τελική τιμή σφάλματος επαλήθευσης συναρτήσει του πλήθους λογικών μονάδων LSTM, του μεγέθους χρονοσειράς και του πλήθους εποχών, με σταθερό

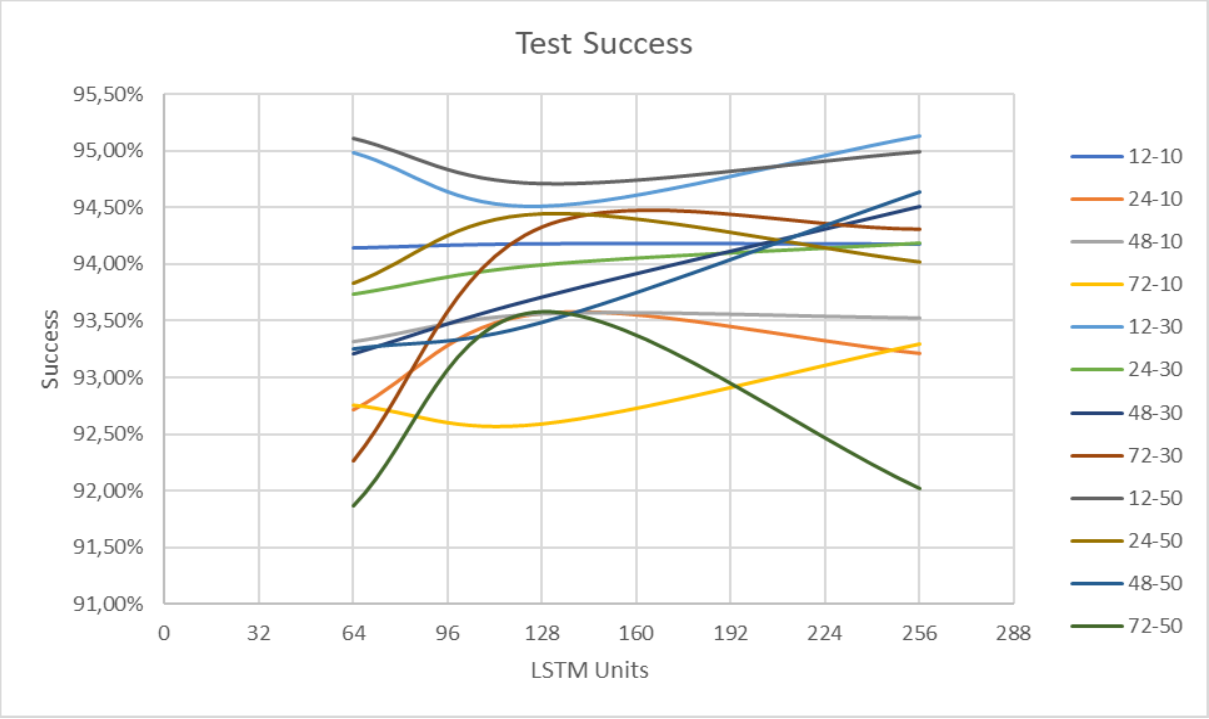
Επιτυχία

Εκπαίδευση



Σχεδιάγραμμα 4.3.3.1.4 Τελική τιμή ποσοστού επιτυχίας εκπαίδευσης συναρτήσει του πλήθους λογικών μονάδων LSTM, του μεγέθους χρονοσειράς και του πλήθους εποχών, με σταθερό μέγεθος δέσμης ίσο με 16

Επαλήθευση



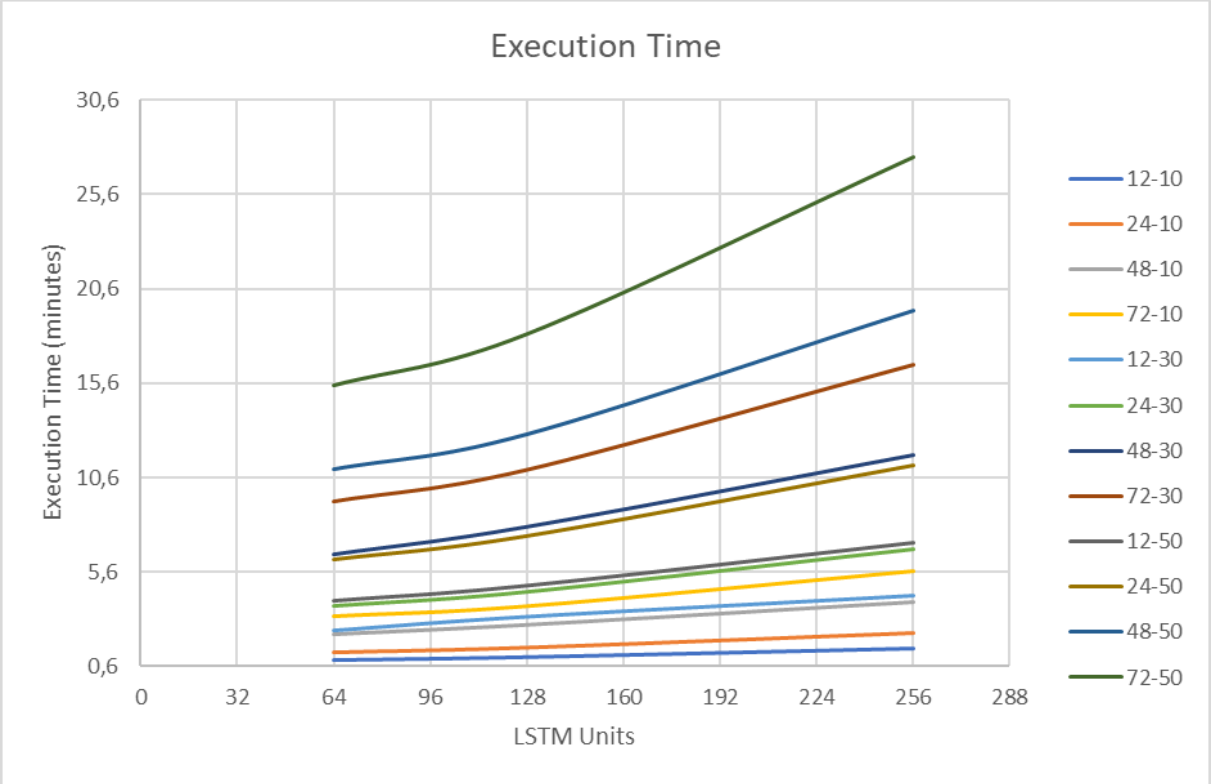
Σχεδιάγραμμα 4.3.3.1.5 Τελική τιμή ποσοστού επιτυχίας επαλήθευσης συναρτήσει του πλήθους λογικών μονάδων LSTM, του μεγέθους χρονοσειράς και του πλήθους εποχών, με σταθερό μέγεθος δέσμης ίσο με 16

4.3.3.2 Μέγεθος δέσμης 32

Στις πιο κάτω γραφικές παραστάσεις βλέπουμε τα αποτελέσματα για εκτελέσεις με σταθερό ρυθμό μάθησης, ίσο με 0.001, σταθερό μέγεθος δέσμης, ίσο με 32 και σταθερό ρυθμό Dropout ίσο με 0.1. Παράλληλα έχουμε μεταβλητές τιμές το μέγεθος χρονοσειράς, τον αριθμό των εποχών και το πλήθος των λογικών μονάδων LSTM. Οι στατιστικές που θα αναλύσουμε είναι ο χρόνος εκτέλεσης, το σφάλμα και το ποσοστό επιτυχίας.

Στις επόμενες γραφικές παραστάσεις ο άξονας x εκφράζει το πλήθος των μονάδων LSTM, αντίστοιχα κάθε γραμμή ενώνει τα αποτελέσματα που προκύπτουν μεταβάλλοντας διακριτά το πλήθος των λογικών μονάδων LSTM, αλλά κρατώντας σταθερό το πλήθος των εποχών και το μέγεθος χρονοσειράς. Επομένως, για παράδειγμα η γραμμή 12-10, εκφράζει τη μεταβολή της υπό έλεγχο τιμής, για διάφορα διακριτά πλήθη λογικών μονάδων LSTM με σταθερό μέγεθος χρονοσειράς ίσο με 12 και πλήθος εποχών ίσο με 10.

Χρόνος εκτέλεσης

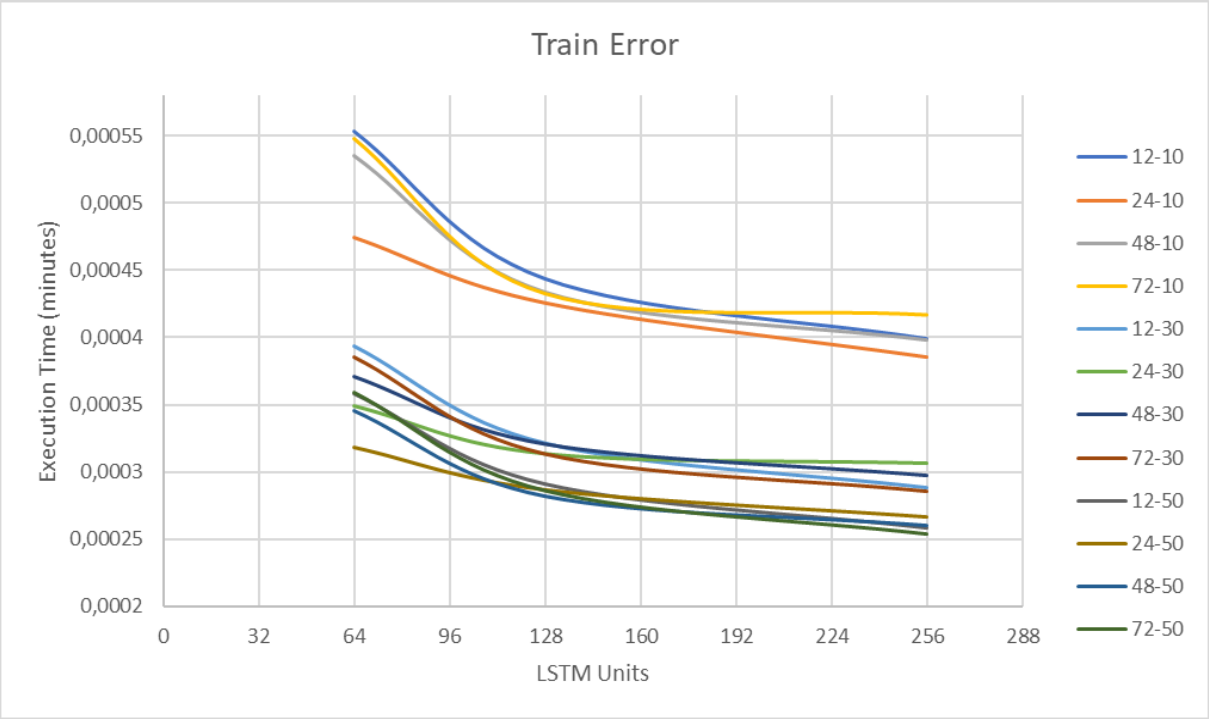


Σχεδιάγραμμα 4.3.3.2.1 Χρόνος εκτέλεσης συναρτήσει του πλήθους λογικών μονάδων LSTM, του μεγέθους χρονοσειράς και του πλήθους εποχών, με σταθερό μέγεθος δέσμης ίσο με 32

Όσον αφορά τον χρόνο εκτέλεσης, τα αποτελέσματα μας είναι απολύτως αναμενόμενα, όσο αυξάνονται οι εποχές και το μέγεθος χρονοσειράς, ο χρόνος αυξάνεται. Αντίστοιχα, η αύξηση στις λογικές μονάδες LSTM επιφέρει και αυτή αύξηση στον χρόνο εκτέλεσης, παρόλα αυτά η επίδρασή της δεν φαίνεται να είναι σημαντική.

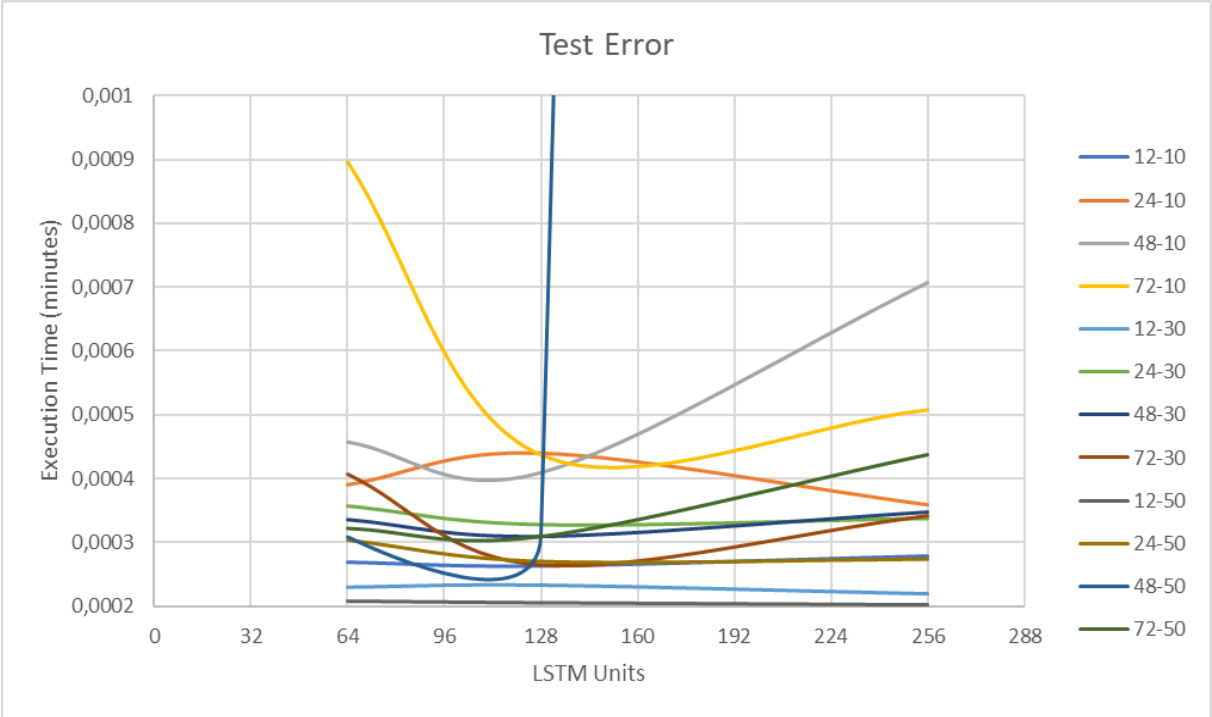
Σφάλμα

Εκπαίδευση



Σχεδιάγραμμα 4.3.3.2.2 Τελική τιμή σφάλματος εκπαίδευσης συναρτήσει του πλήθους λογικών μονάδων LSTM, του μεγέθους χρονοσειράς και του πλήθους εποχών, με σταθερό μέγεθος δέσμης ίσο με 32

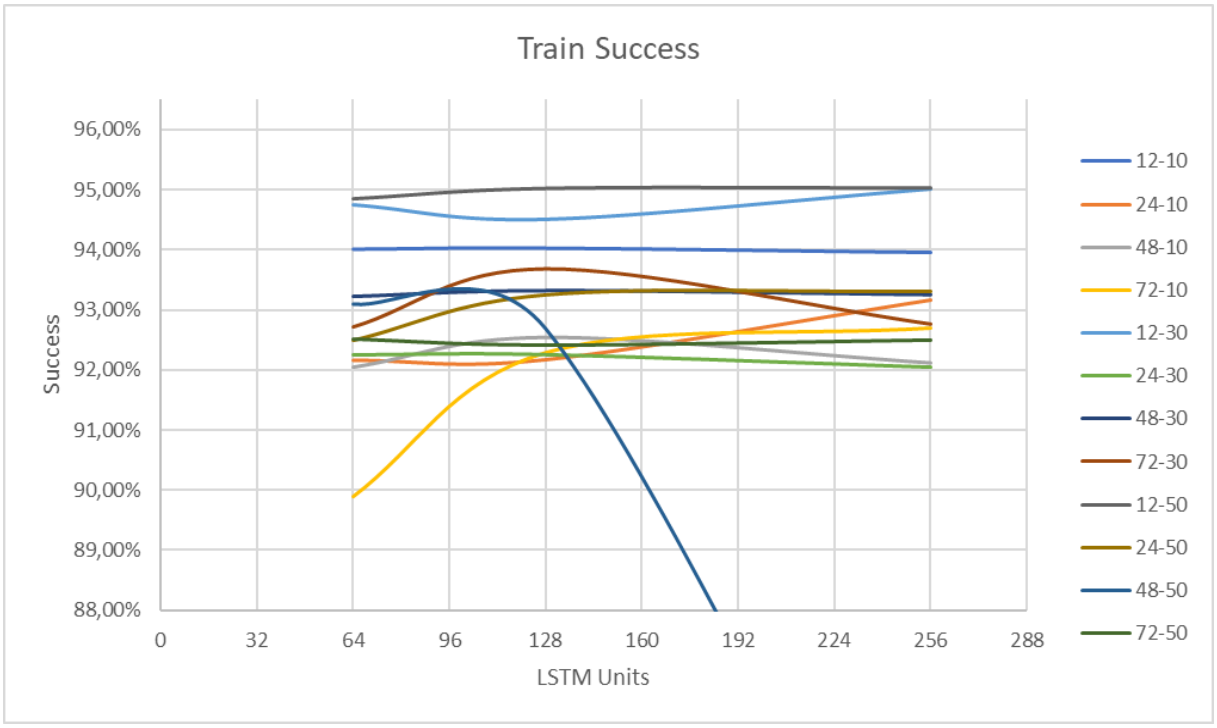
Επαλήθευση



Σχεδιάγραμμα 4.3.3.2.3 Τελική τιμή σφάλματος επαλήθευσης συναρτήσει του πλήθους λογικών μονάδων LSTM, του μεγέθους χρονοσειράς και του πλήθους εποχών, με σταθερό μέγεθος δέσμης ίσο με 32

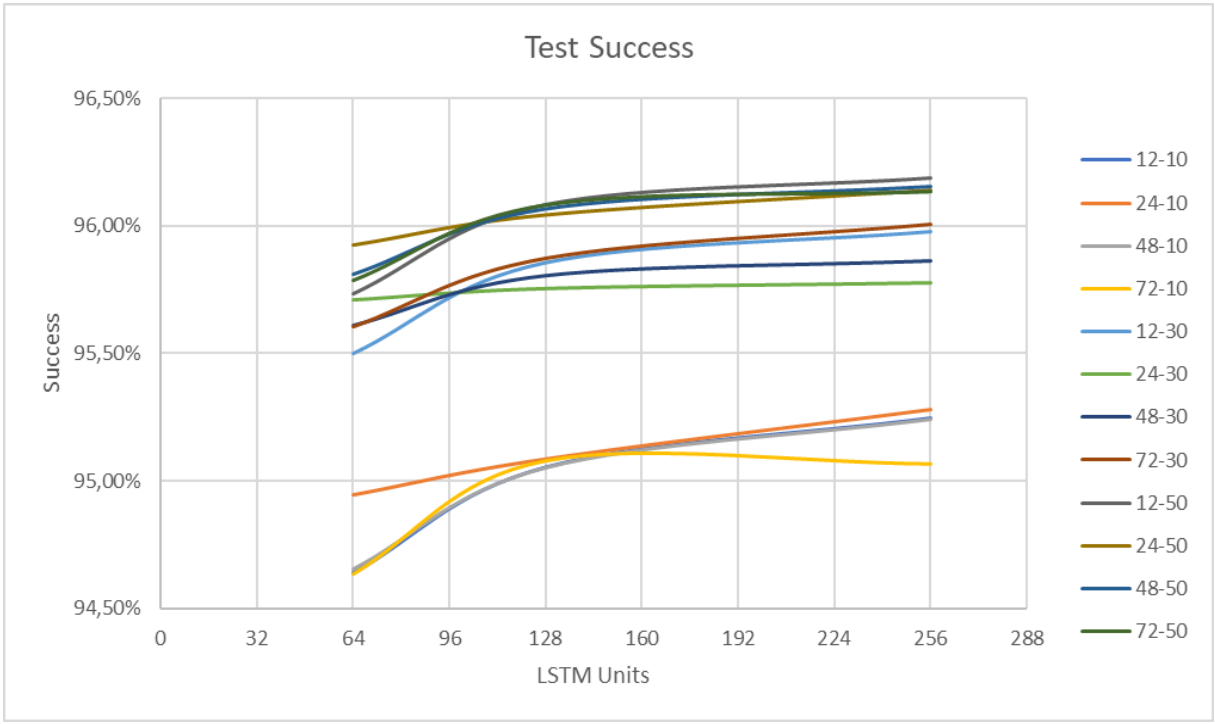
Επιτυχία

Εκπαίδευση



Σχεδιάγραμμα 4.3.3.2.4 Τελική τιμή ποσοστού επιτυχίας εκπαίδευσης συναρτήσει του πλήθους λογικών μονάδων LSTM, του μεγέθους χρονοσειράς και του πλήθους εποχών, με σταθερό μέγεθος δέσμης ίσο με 32

Επαλήθευση



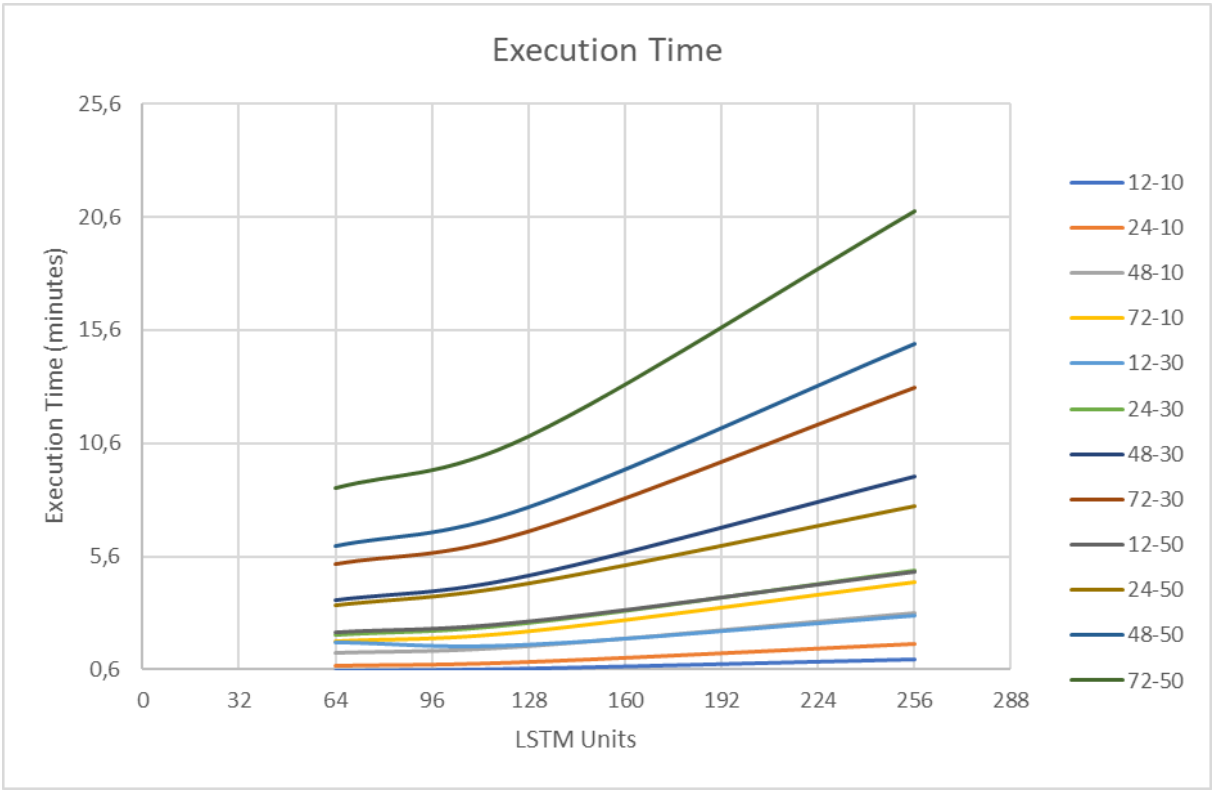
Σχεδιάγραμμα 4.3.3.2.5 Τελική τιμή ποσοστού επιτυχίας επαλήθευσης συναρτήσει του πλήθους λογικών μονάδων LSTM, του μεγέθους χρονοσειράς και του πλήθους εποχών, με σταθερό μέγεθος δέσμης ίσο με 32

4.3.3.3 Μέγεθος δέσμης 64

Στις πιο κάτω γραφικές παραστάσεις βλέπουμε τα αποτελέσματα για εκτελέσεις με σταθερό ρυθμό μάθησης, ίσο με 0.001, σταθερό μέγεθος δέσμης, ίσο με 64 και σταθερό ρυθμό Dropout ίσο με 0.1. Παράλληλα έχουμε μεταβλητές τιμές το μέγεθος χρονοσειράς, τον αριθμό των εποχών και το πλήθος των λογικών μονάδων LSTM. Οι στατιστικές που θα αναλύσουμε είναι ο χρόνος εκτέλεσης, το σφάλμα και το ποσοστό επιτυχίας.

Στις επόμενες γραφικές παραστάσεις ο άξονας x εκφράζει το πλήθος των μονάδων LSTM, αντίστοιχα κάθε γραμμή ενώνει τα αποτελέσματα που προκύπτουν μεταβάλλοντας διακριτά το πλήθος των λογικών μονάδων LSTM, αλλά κρατώντας σταθερό το πλήθος των εποχών και το μέγεθος χρονοσειράς. Επομένως, για παράδειγμα η γραμμή 12-10, εκφράζει τη μεταβολή της υπό έλεγχο τιμής, για διάφορα διακριτά πλήθη λογικών μονάδων LSTM με σταθερό μέγεθος χρονοσειράς ίσο με 12 και πλήθος εποχών ίσο με 10.

Χρόνος εκτέλεσης

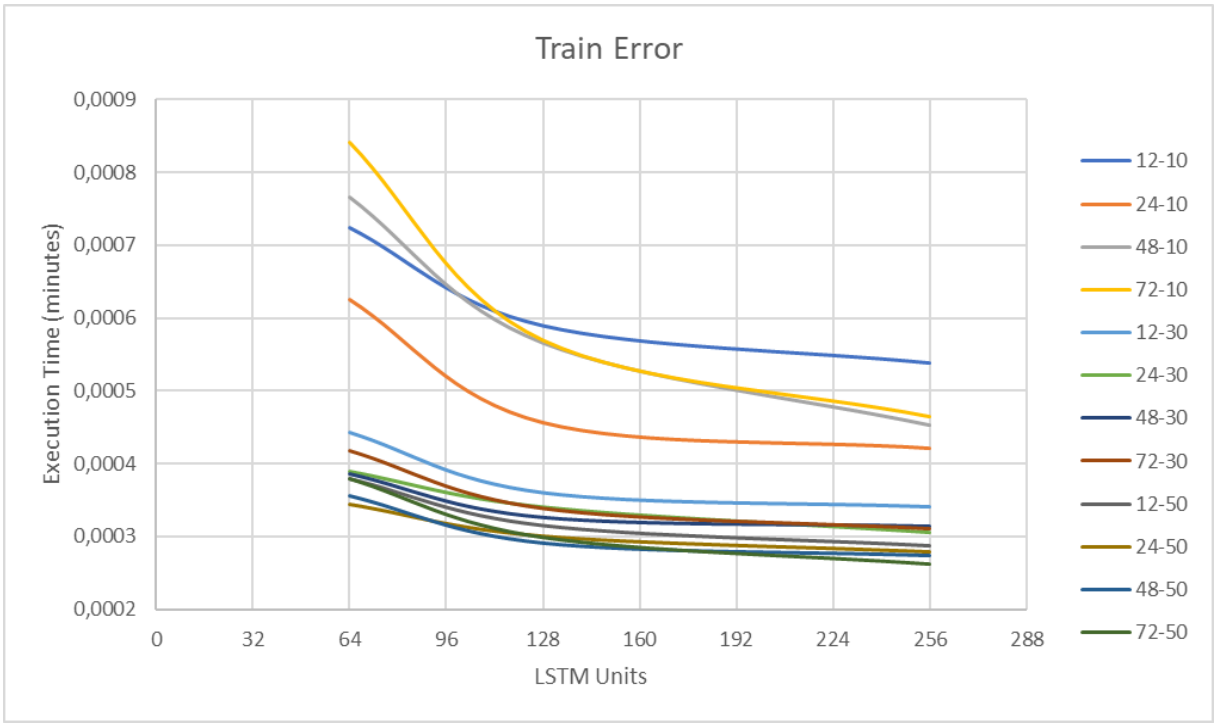


Σχεδιάγραμμα 4.3.3.3.1 Χρόνος εκτέλεσης συναρτήσει του πλήθους λογικών μονάδων LSTM, του μεγέθους χρονοσειράς και του πλήθους εποχών, με σταθερό μέγεθος δέσμης ίσο με 64

Όσον αφορά τον χρόνο εκτέλεσης, τα αποτελέσματα μας είναι απολύτως αναμενόμενα, όσο αυξάνονται οι εποχές και το μέγεθος χρονοσειράς, ο χρόνος αυξάνεται. Αντίστοιχα, η αύξηση στις λογικές μονάδες LSTM επιφέρει και αυτή αύξηση στον χρόνο εκτέλεσης, παρόλα αυτά η επίδρασή της δεν φαίνεται να είναι σημαντική.

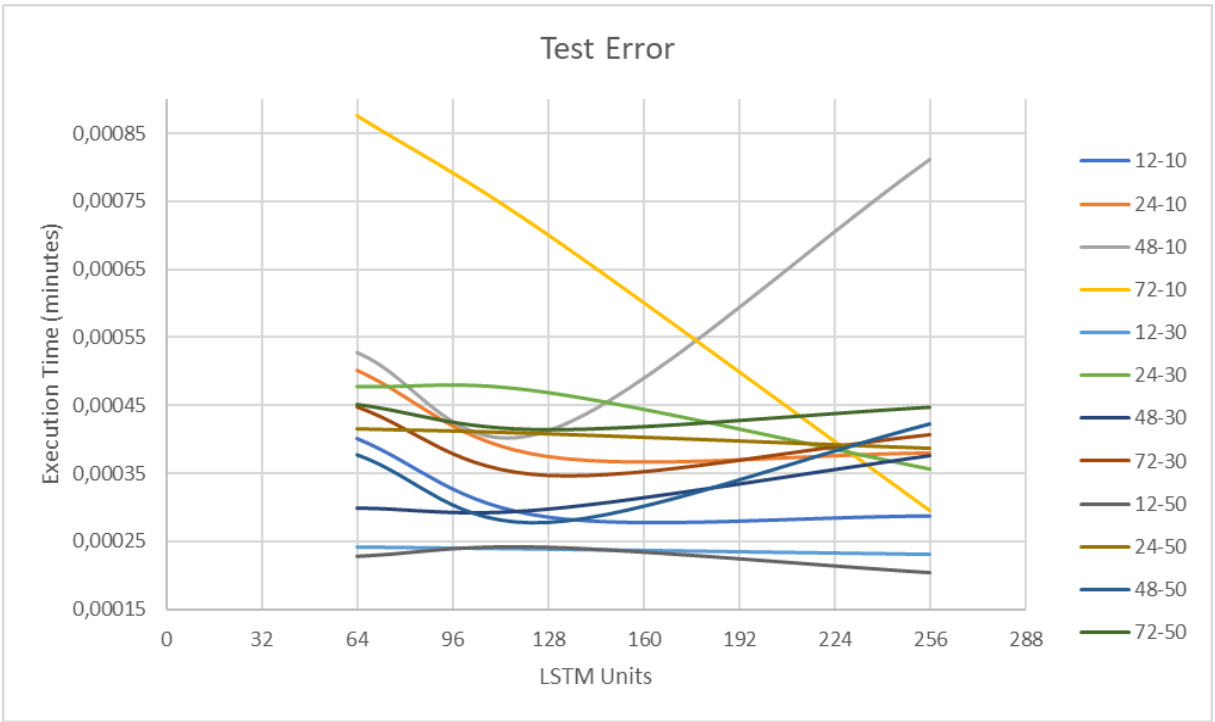
Σφάλμα

Εκπαίδευση



Σχεδιάγραμμα 4.3.3.3.2 Τελική τιμή σφάλματος εκπαίδευσης συναρτήσει του πλήθους λογικών μονάδων LSTM, του μεγέθους χρονοσειράς και του πλήθους εποχών, με σταθερό μέγεθος δέσμης ίσο με 64

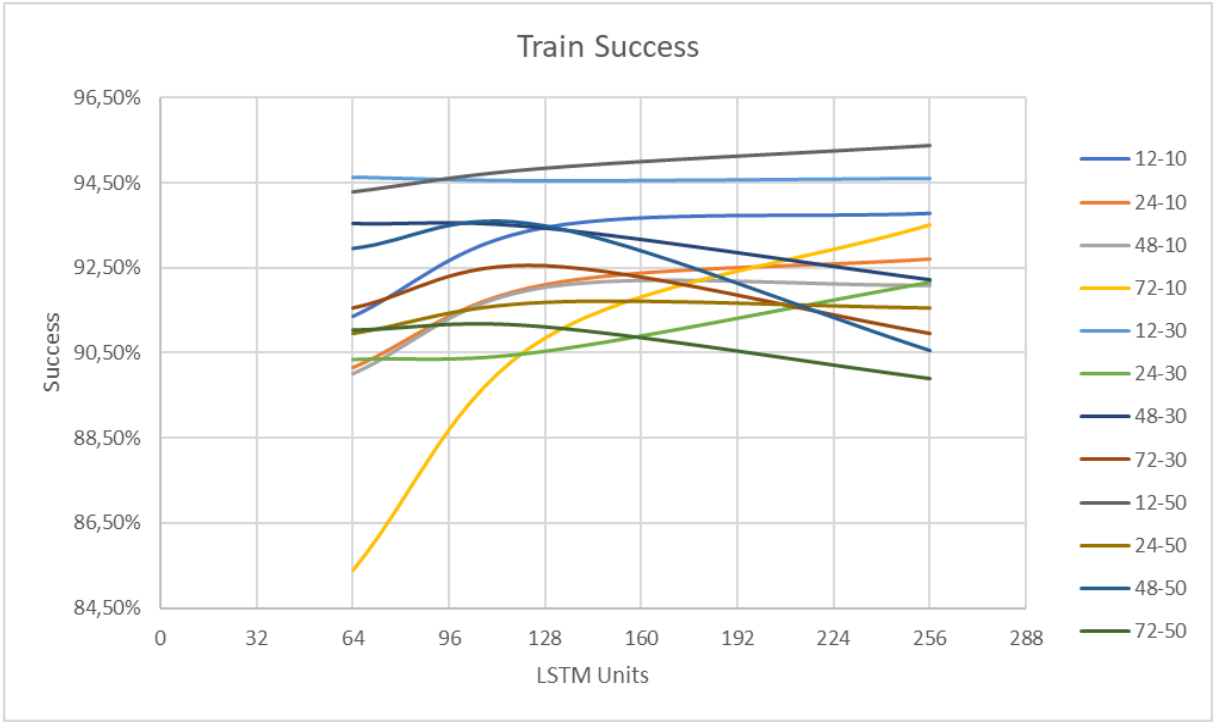
Επαλήθευση



Σχεδιάγραμμα 4.3.3.3.3 Τελική τιμή σφάλματος επαλήθευσης συναρτήσει του πλήθους λογικών μονάδων LSTM, του μεγέθους χρονοσειράς και του πλήθους εποχών, με σταθερό μέγεθος δέσμης ίσο με 64

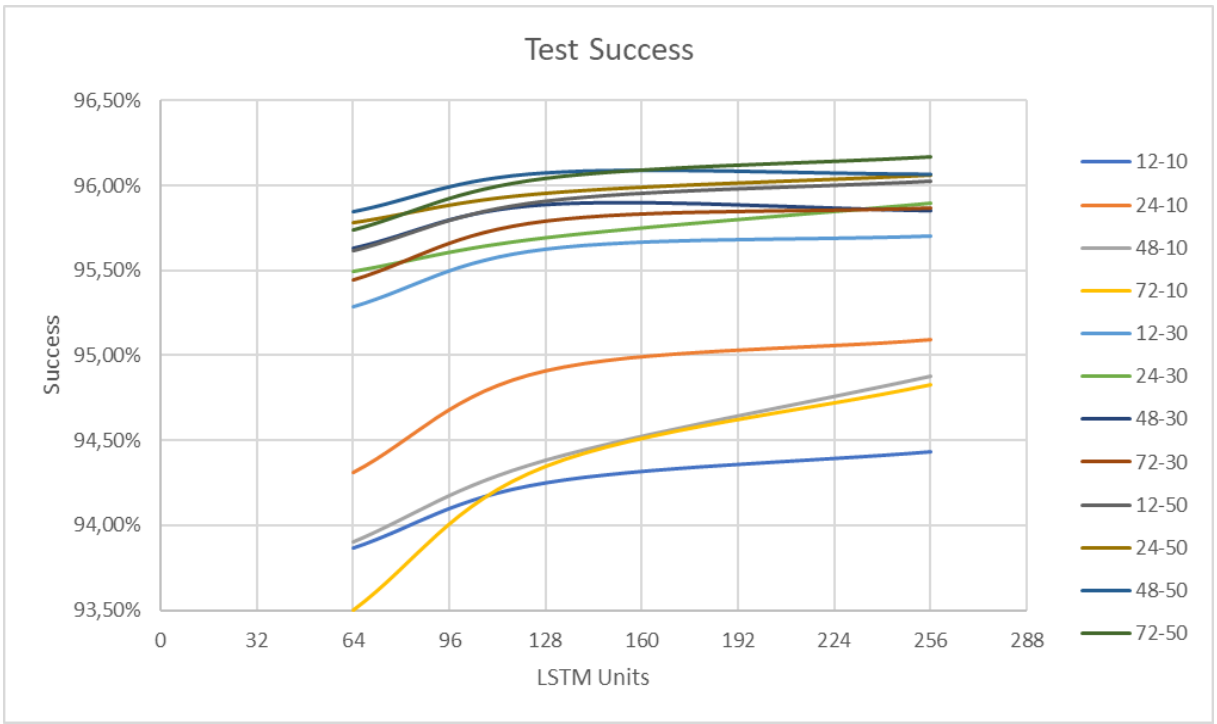
Επιτυχία

Εκπαίδευση



Σχεδιάγραμμα 4.3.3.3.4 Τελική τιμή ποσοστού επιτυχίας εκπαίδευσης συναρτήσει του πλήθους λογικών μονάδων LSTM, του μεγέθους χρονοσειράς και του πλήθους εποχών, με σταθερό μέγεθος δέσμης ίσο με 64

Επαλήθευση



Σχεδιάγραμμα 4.3.3.3.5 Τελική τιμή σφάλματος επαλήθευσης συναρτήσει του πλήθους λογικών μονάδων LSTM, του μεγέθους χρονοσειράς και του πλήθους εποχών, με σταθερό μέγεθος δέσμης ίσο με 32

Στον παρακάτω πίνακα εξετάζουμε συγκεντρωτικά τις καλύτερες εκτελέσεις από τα προηγούμενα πειράματα.

Timestep	Epochs	LSTM units	Batch size	Train error	Test error	Train success	Test success	Execution time (minutes)
12	30	256	16	0,00026	0,00019	96,07%	95,13%	7,847962983
12	50	256	32	0.00026	0.00020	96.20%	95.03%	7.127492706
12	50	256	64	0.00029	0.00020	96.02%	95.37%	4.902709373

Σχεδιάγραμμα 4.3.3.1.1 Συγκεντρωτικός πίνακας καλύτερων εκτελέσεων με μεταβλητό μέγεθος δέσμης

Παρατηρούμε ότι όσο αυξάνεται το μέγεθος δέσμης τα αποτελέσματα βελτιώνονται σχετικά και ο χρόνος εκτέλεσης μειώνεται. Εξαιτίας αυτής της παρατήρησης θα διερευνήσουμε το εύρος τιμών που δύναται να λάβει το μέγεθος δέσμης, χρησιμοποιώντας τον βέλτιστο συνδυασμό μεταβλητών.

Timestep	Epochs	LSTM units	Batch size	Train error	Test error	Train success	Test success	Execution time (minutes)
12	50	256	128	0.00046	0.00026	95.20%	94.31%	3.418876280
12	50	256	256	0.00070	0.00036	93.65%	91.36%	2.826796957
12	50	256	512	0.00069	0.00035	93.44%	92.47%	2.590876091
12	50	256	1024	0.00079	0.00030	93.56%	94.19%	2.346769370

Σχεδιάγραμμα 4.3.3.1.2 Εξερεύνηση ορίων μεγέθους δέσμης

Όπως είναι εμφανές, οι επιδόσεις του δικτύου μας χειροτερεύουν με την αύξηση του μεγέθους δέσμης πάνω από την τιμή 64.

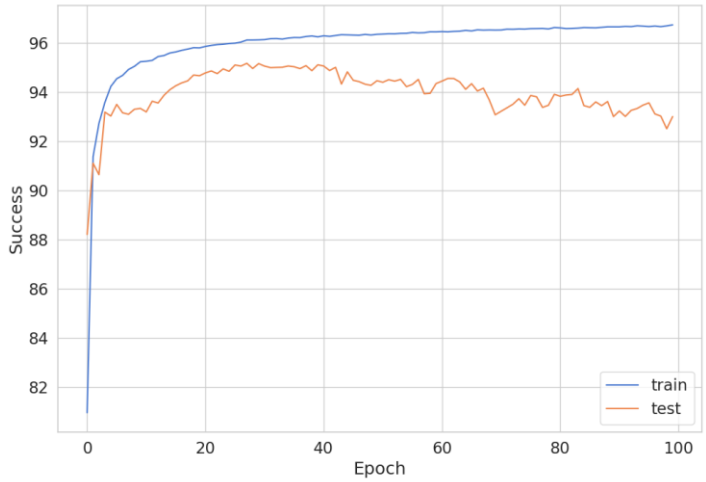
4.3.4 Επέκταση εποχών

Για να ερευνήσουμε πιο ενδελεχώς την επίδραση των εποχών στην εκπαίδευση του δικτύου μας θα δοκιμάσουμε τη βέλτιστη σύνθεση που έχουμε χτίσει μέχρι τώρα (χρονικό βήμα ίσο με 12, λογικές μονάδες ίσες με 256, μέγεθος δέσμης ίσο με 64, ρυθμό Dropout ίσο με 0.1 και ρυθμό μάθησης ίσο με 0.001) αλλά θα αυξήσουμε το πλήθος των εποχών

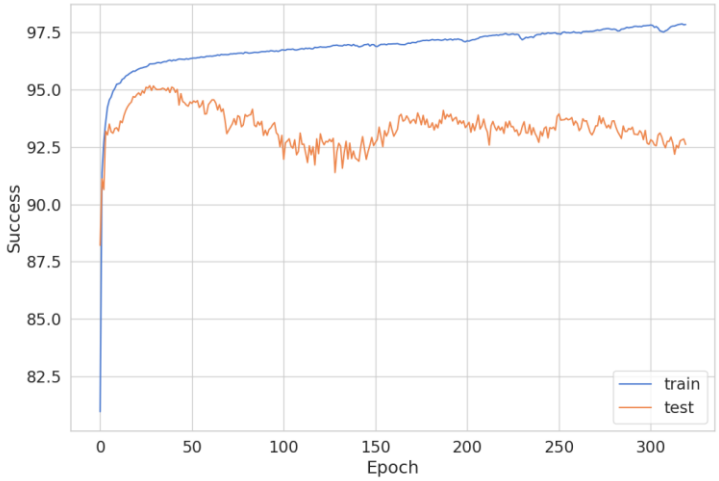
Epochs	Train Error	Test Error	Train Success	Test Success	Execution Time (minutes)
100	0.00018	0.00041	96.73%	93.00%	9.49215
320	0.00010	0.00030	97.83%	92.61%	28.910002
500	7.89e-05	0.00025	98.034%	94.21%	45.119824

Σχεδιάγραμμα 4.3.4.1 Εξερεύνηση ορίων πλήθους εποχών

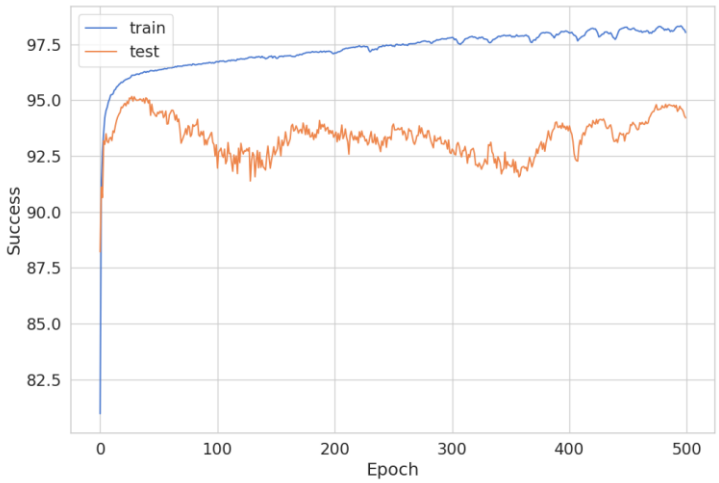
Σε πρώτη ανάγνωση τα αποτελέσματα φαίνονται πολύ καλύτερα, ωστόσο αν κοιτάξουμε τη γενική συμπεριφορά των γραφικών παραστάσεων της ποσοστιαίας επιτυχίας θα δούμε ότι υπάρχουν σκαμπανεβάσματα που παραπέμπουν σε υπερεκπαίδευση, φαινόμενο που επιβεβαιώνει και το μειωμένο ποσοστό επιτυχίας στη φάση επαλήθευσης.



Σχεδιάγραμμα 4.3.4.2 Εκτέλεση με πλήθος εποχών 100



Σχεδιάγραμμα 4.3.4.3 Εκτέλεση με πλήθος εποχών 320



Σχεδιάγραμμα 4.3.4.4 Εκτέλεση με πλήθος εποχών 500

Από τις παραπάνω εκτελέσεις καταλήγουμε στο συμπέρασμα ότι ο ιδανικός αριθμός εποχών, για το δίκτυο LSTM κυμαίνεται μεταξύ του πενήντα και του εκατό.

4.3.5 Συζήτηση αποτελεσμάτων

Από τις παραπάνω εκτελέσεις μπορούμε εύκολα να συμπεράνουμε πως το δίκτυό μας είναι αρκετά ευαίσθητο στις αλλαγές κάποιων μεταβλητών, όπως ο ρυθμός μάθησης και ο ρυθμός του Dropout. Αυτό βασίζεται σε σχεδιαγράμματα όπως το 4.3.1.5 και το 4.3.2.3 που παρουσιάζουν ανώμαλες συμπεριφορές των δικτύων σε εκτελέσεις που αυτές οι μεταβλητές έχουν αλλάξει ελαφρώς. Στο σχεδιάγραμμα 4.3.1.5 παρατηρούμε πως για ρυθμό μάθησης ίσο με 0.1 οι προβλέψεις μας είναι πάρα πολύ λανθασμένες ειδικά σε σύγκριση με τα σχεδιαγράμματα 4.3.1.4 και 4.3.13 όπου οι προβλέψεις μας είναι πολύ κοντά στην πραγματικότητα. Αντίστοιχα, στο σχεδιάγραμμα 4.3.2.3 παρατηρούμε πως για ρυθμό Dropout ίσο με μηδέν έχουμε μια απότομη πτώση του ποσοστού επιτυχίας στις εποχές 40 με 45. Το παράδοξο αυτής της παρατήρησης αυτής είναι ότι στις επόμενες εποχές το ποσοστό επιτυχίας επανέρχεται σε κανονικές τιμές, ωστόσο κάτι τέτοιο δεν πρέπει να μας καθησυχάζει καθώς μια τέτοια συμπεριφορά κρίνεται αρκετά απρόβλεπτη με αποτέλεσμα να μην μπορούμε να εμπιστευθούμε ένα τέτοιο δίκτυο. Για τις υπόλοιπες μεταβλητές δεν παρατηρήσαμε κάποια συστηματικά περίεργη συμπεριφορά. Υπήρχαν κάποιες εκτελέσεις όπου παρατηρήσαμε το φαινόμενο της υπερμάθησης αλλά κάτι τέτοιο είναι λογικό και νοουμένου ότι δεν εμφανίζεται με κάποιο συγκεκριμένο μοτίβο δεν πρέπει να μας ανησυχεί. Κλείνοντας, με βάση τις παρατηρήσεις μας η βέλτιστη σύνθεση του δικτύου LSTM στο πρόβλημα αυτό είναι αυτή που βλέπουμε στο σχεδιάγραμμα 4.3.5.1.

Μεταβλητή	Βέλτιστη Τιμή
Μέγεθος χρονοσειράς	12
Πλήθος εποχών	50
Πλήθος λογικών μονάδων LSTM	256
Dropout rate	0.1
Batch size	32
Ρυθμός μάθησης	0.001

Σχεδιάγραμμα 4.3.5.1 Πίνακας βέλτιστης σύνθεσης δικτύου LSTM

4.4 Αποτελέσματα GRU

Τα αποτελέσματα που προέκυψαν από το δίκτυο GRU ήταν αρκετά ικανοποιητικά καθώς, ενώ ξεκινήσαμε από έξι μεταβλητές, καταφέραμε να βρούμε κάποια βέλτιστα μοντέλα εκτέλεσης. Για να κάνουμε τις μετρήσεις του σφάλματος βασιστήκαμε κυρίως στη βιβλιοθήκη Keras και στις έτοιμες μετρικές της. Για τον υπολογισμό του σφάλματος χρησιμοποιήσαμε την μέθοδο του τετραγωνικού σφάλματος και για την επιτυχία το απόλυτο ποσοστιαίο σφάλμα. Στα υποκεφάλαια που ακολουθούν θα αναλύσουμε κάποια ενδεικτικά αποτελέσματα που προέκυψαν από την πειραματική διαδικασία. Παράλληλα θα αναλύσουμε την επίδραση της αλλαγής των τιμών των μεταβλητών.

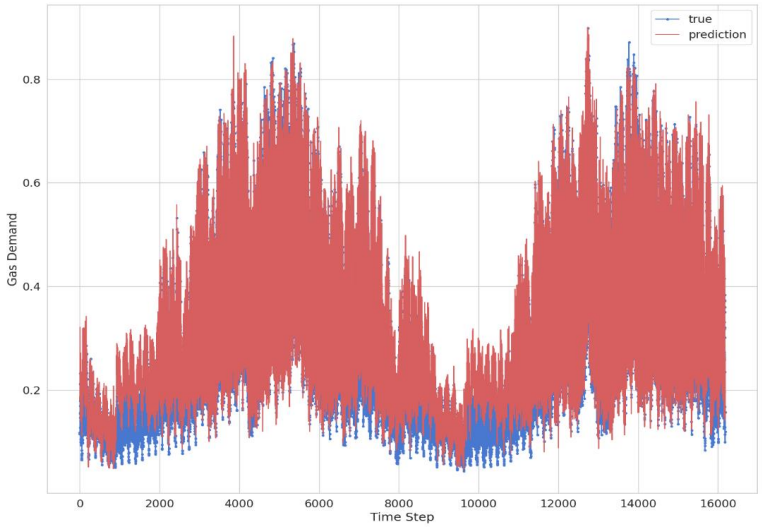
4.4.1 Αρχικά συμπεράσματα

Από τα πρώτα συμπεράσματα στα οποία καταλήξαμε, ήταν ότι ο ρυθμός μάθησης επηρεάζει καθοριστικά το σφάλμα και την επιτυχία μας, όπως φαίνεται και στο διάγραμμα 4.4.1.1 και 4.4.1.2. Για τα αποτελέσματα που παρουσιάζονται χρησιμοποιήσαμε ενδεικτικές τιμές για τις υπόλοιπες μεταβλητές, εκτός από τον ρυθμό μάθησης. Πιο συγκεκριμένα, το πλήθος των μονάδων GRU ήταν ίσο με 128, το μέγεθος της χρονοσειράς ήταν ίσο με 72, ο ρυθμός του Dropout ήταν ίσος με 0.1, το μέγεθος δέσμης ίσο με 64 και οι εποχές ίσες με 10.

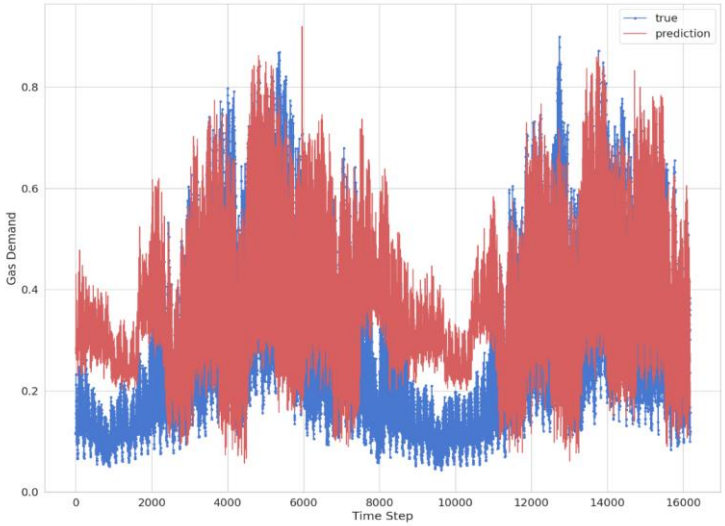
Σχεδιάγραμμα 4.4.1.1 Σύγκριση ποσοστού επιτυχίας πρόβλεψης με μεταβλητό ρυθμό μάθησης

Σχεδιάγραμμα 4.4.1.2 Σύγκριση σφάλματος πρόβλεψης με μεταβλητό ρυθμό μάθησης

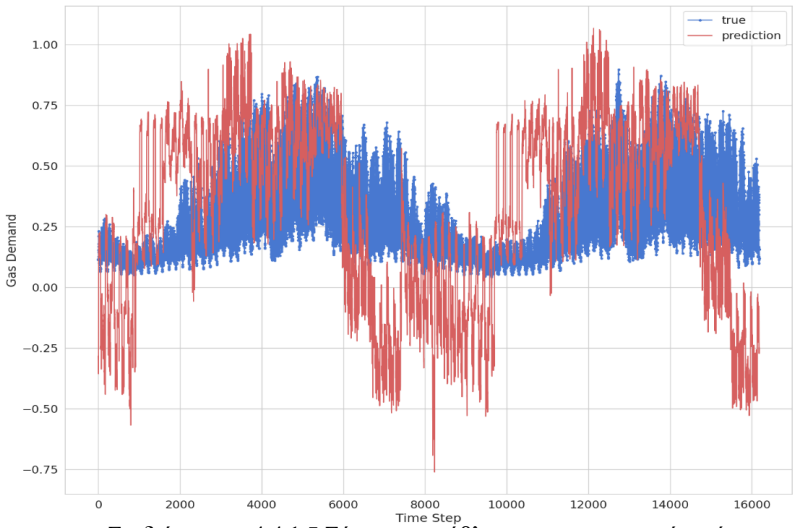
Παρατηρούμε ότι η τιμή του ρυθμού μάθησης πρέπει να έχει αρκετά χαμηλή τιμή για να μας δώσει τα βέλτιστα αποτελέσματα, αντίστοιχα για τιμή ίση με 0.9 το δίκτυό μας είναι πλήρως δυσλειτουργικό. Κάτι που αξίζει να σημειωθεί, είναι ότι η προεπιλεγμένη τιμή του βελτιστοποιητή Adam για τον ρυθμό μάθησης είναι 0.001 κάτι που επιβεβαιώνει τα αποτελέσματά μας. Παράλληλα, η δική μας φάση ελέγχου επιβεβαιώνει και αυτή τα αποτελέσματά μας. Πιο κάτω μπορούμε να δούμε τα αποτελέσματα πρόβλεψης των παραπάνω εκτελέσεων σε σύγκριση με τις πραγματικές τιμές.



Σχεδιάγραμμα 4.4.1.3 Σύγκριση πρόβλεψης με πραγματική τιμή για ρυθμό μάθησης ίσο με 0.001



Σχεδιάγραμμα 4.4.1.4 Σύγκριση πρόβλεψης με πραγματική τιμή για ρυθμό μάθησης ίσο με 0.005

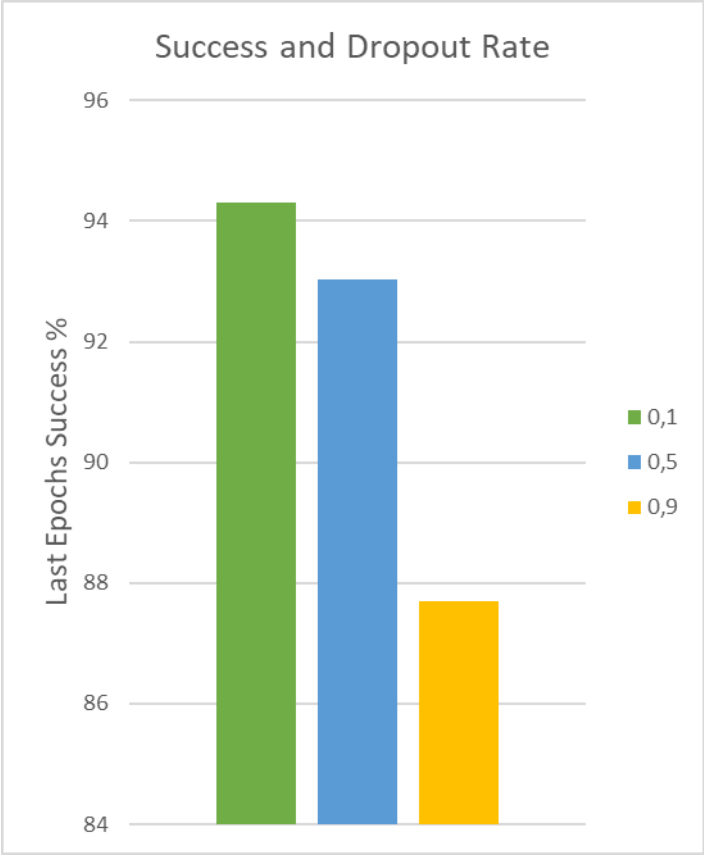


Σχεδιάγραμμα 4.4.1.5 Σύγκριση πρόβλεψης με πραγματική τιμή για ρυθμό μάθησης ίσο με 0.1

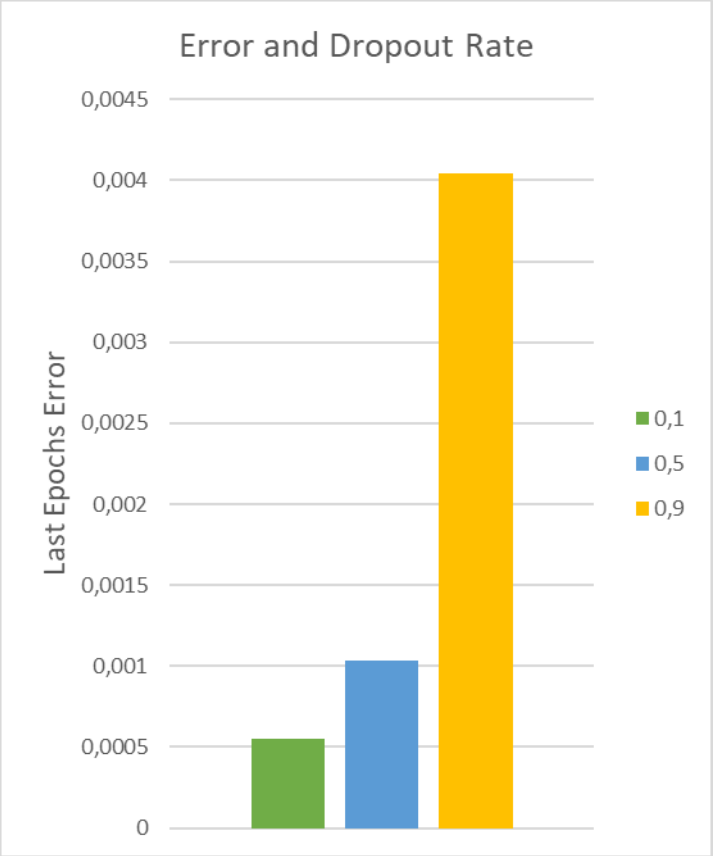
Τα παραπάνω αποτελέσματα μας δίνουν μια πιο καθαρή εικόνα για το πώς επηρεάζει ο ρυθμός μάθησης την εκμάθηση του δικτύου. Για την συνέχεια των πειραμάτων μας θα επικεντρωθούμε στις εκτελέσεις με ρυθμό μάθησης ίσο με 0.001.

4.4.2 Αλλαγές στον ρυθμό Dropout

Στη συνέχεια θα μελετήσουμε την επίδραση του ρυθμού Dropout, ο οποίος όπως αναφέραμε καθορίζει την πιθανότητα να αγνοηθούν κάποιες τιμές από το επίπεδο εξόδου. Για τα αποτελέσματα που παρουσιάζονται χρησιμοποιήσαμε ενδεικτικές τιμές για τις υπόλοιπες μεταβλητές, εκτός από τον ρυθμό μάθησης. Πιο συγκεκριμένα, το πλήθος των μονάδων GRU ήταν ίσο με 128, το μέγεθος της χρονοσειράς ήταν ίσο με 72, το μέγεθος δέσμης ήταν ίσο με 64 και οι εποχές ίσες με 10.

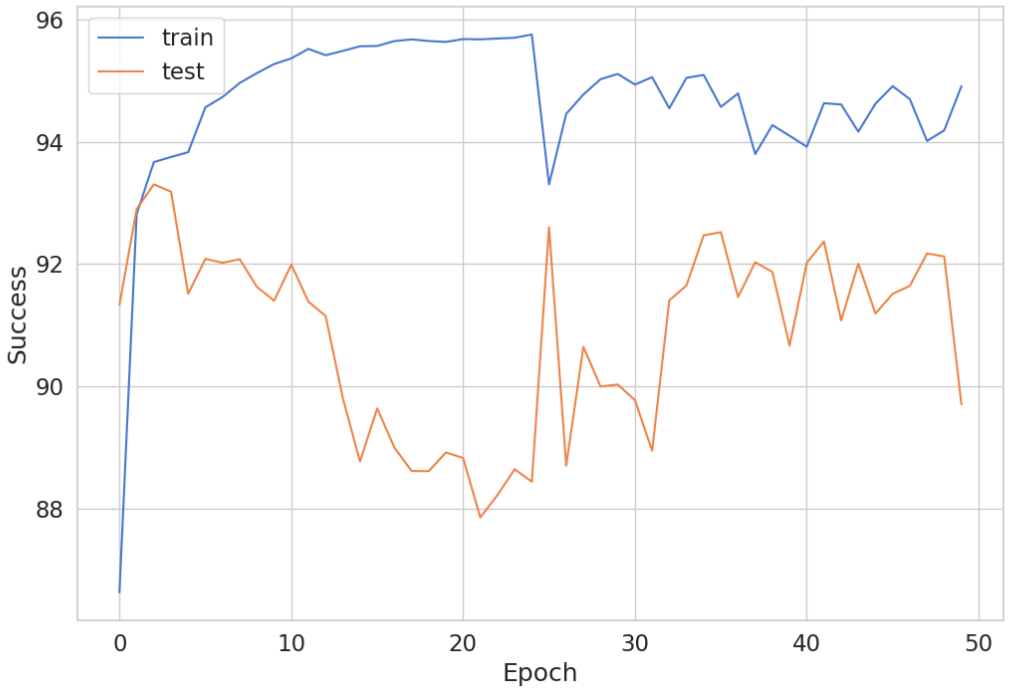


Σχεδιάγραμμα 4.4.2.1 Σύγκριση ποσοστού επιτυχίας πρόβλεψης με μεταβλητό ρυθμό Dropout



Σχεδιάγραμμα 4.4.2.2 Σύγκριση σφάλματος πρόβλεψης με μεταβλητό ρυθμό Dropout

Παρατηρούμε ότι το δίκτυό μας αποδίδει καλύτερα όσο μικρότερος είναι ο ρυθμός του Dropout, κάτι που συμβαδίζει με τη θεωρία, καθώς χρησιμοποιούμε περισσότερο όγκο πληροφοριών για να βγάλουμε το τελικό μας αποτέλεσμα. Βάσει αυτής της παραδοχής κάποιος θα μπορούσε να υποστηρίξει ότι θα έπρεπε να μειώναμε και άλλο το ρυθμό Dropout. Παρόλα αυτά το επίπεδο Dropout μας βοηθάει να καθαρίσουμε τον θόρυβο στα αποτελέσματά μας, σε εκτελέσεις με μηδενικό ρυθμό Dropout παρατηρήθηκε υπερπροσαρμογή από έναν αριθμό εποχών και πάνω, όπως φαίνεται και στο σχεδιάγραμμα 4.4.2.3.



Σχεδιάγραμμα 4.4.2.3 Γραφική παράσταση επιτυχίας πρόβλεψης με Dropout ίσο με μηδέν, πλήθος λογικών μονάδων GRU ίσο με 512, μέγεθος χρονοσειράς ίσο με72, πλήθος εποχών ίσο με 50, μέγεθος δέσμης ίσο με 64 και ρυθμό μάθησης ίσο με 0.001

Τα παραπάνω αποτελέσματα μας δίνουν μια πιο καθαρή εικόνα για το πώς επηρεάζει ο ρυθμός Dropout την εκμάθηση του δικτύου. Για την συνέχεια των πειραμάτων μας θα επικεντρωθούμε στις εκτελέσεις με ρυθμό Dropout ίσο με 0.1.

4.4.3 Τελικά αποτελέσματα

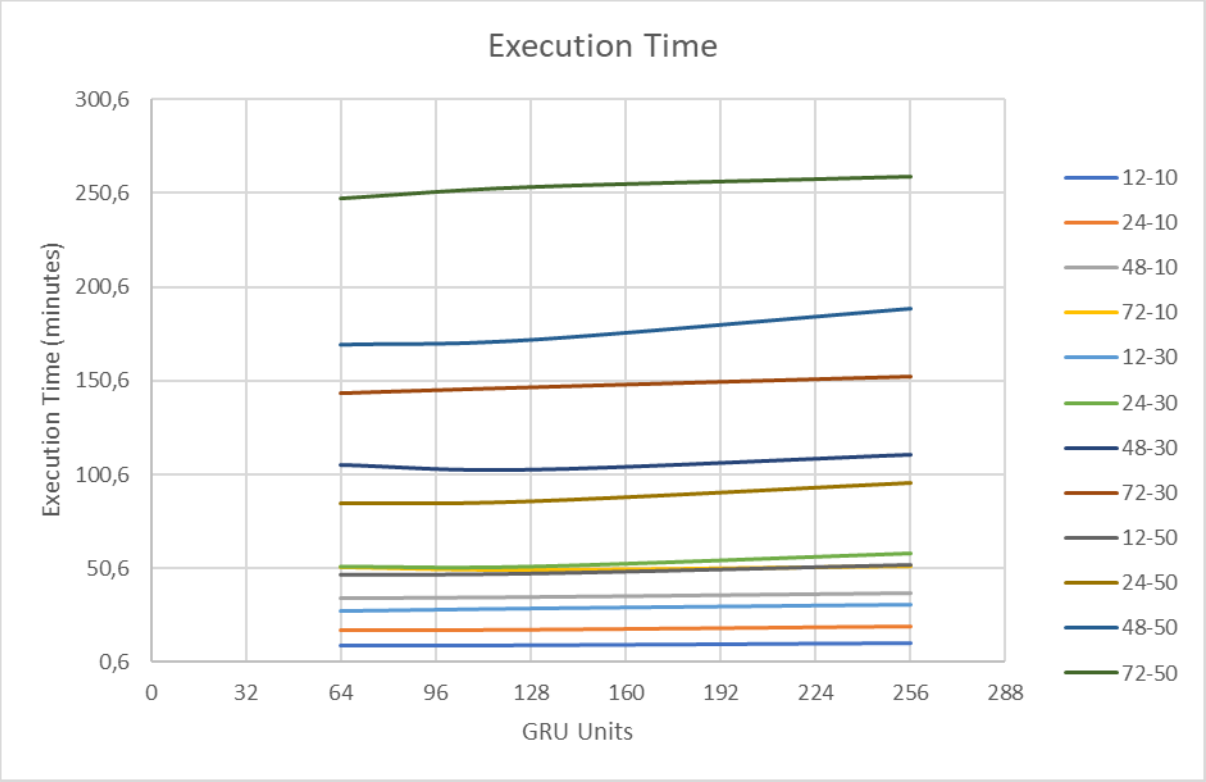
Για τις υπόλοιπες μεταβλητές θα παρουσιάσουμε μια συγκεντρωτική σύγκριση καθώς οι διαφορές μεταξύ των αποτελεσμάτων δεν είναι τόσο ευδιάκριτες. Για κάθε μια από τις δυνατές τιμές του μεγέθους δέσμης θα παρουσιάσουμε τα αποτελέσματα όλων των πιθανών συνδυασμών.

4.4.3.1 Μέγεθος δέσμης 16

Στις πιο κάτω γραφικές παραστάσεις βλέπουμε τα αποτελέσματα για εκτελέσεις με σταθερό ρυθμό μάθησης, ίσο με 0.001, σταθερό μέγεθος δέσμης, ίσο με 16 και σταθερό ρυθμό Dropout ίσο με 0.1. Παράλληλα έχουμε μεταβλητές τιμές το μέγεθος χρονοσειράς, τον αριθμό των εποχών και το πλήθος των λογικών μονάδων. Οι στατιστικές που θα αναλύσουμε είναι ο χρόνος εκτέλεσης, το σφάλμα και το ποσοστό επιτυχίας.

Στις επόμενες γραφικές παραστάσεις ο άξονας x εκφράζει το πλήθος των μονάδων GRU, αντίστοιχα κάθε γραμμή ενώνει τα αποτελέσματα που προκύπτουν μεταβάλλοντας διακριτά το πλήθος των λογικών μονάδων GRU, αλλά κρατώντας σταθερό το πλήθος των εποχών και το μέγεθος χρονοσειράς. Επομένως, για παράδειγμα η γραμμή 12-10, εκφράζει τη μεταβολή της υπό έλεγχο τιμής, για διάφορα διακριτά πλήθη λογικών μονάδων GRU με σταθερό μέγεθος χρονοσειράς ίσο με 12 και πλήθος εποχών ίσο με 10.

Χρόνος εκτέλεσης

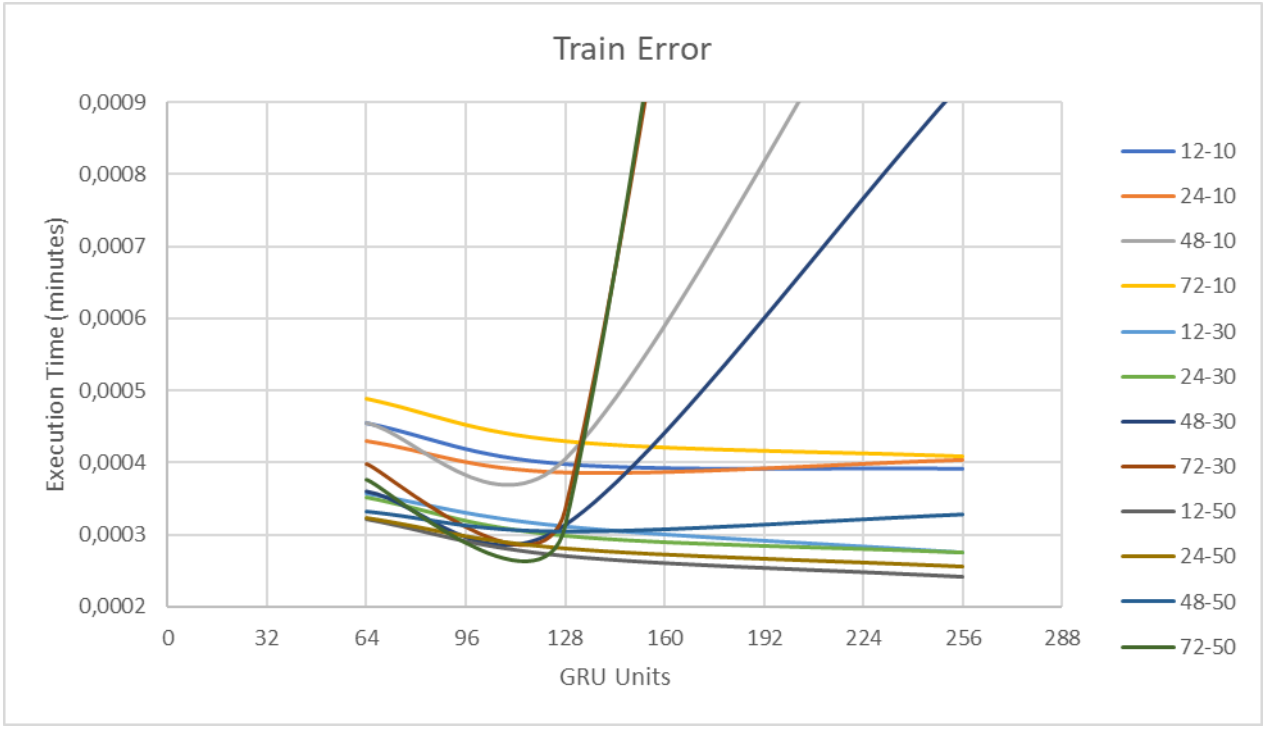


Σχεδιάγραμμα 4.4.3.1.1 Χρόνος εκτέλεσης συναρτήσει του πλήθους λογικών μονάδων GRU, του μεγέθους χρονοσειράς και του πλήθους εποχών, με σταθερό μέγεθος δέσμης ίσο με 16

Όσον αφορά τον χρόνο εκτέλεσης, τα αποτελέσματα μας είναι απολύτως αναμενόμενα, όσο αυξάνονται οι εποχές και το μέγεθος χρονοσειράς, ο χρόνος αυξάνεται. Αντίστοιχα, η αύξηση στις λογικές μονάδες GRU επιφέρει και αυτή αύξηση στο χρόνο εκτέλεσης, παρόλα αυτά η επίδρασή της δεν φαίνεται να είναι σημαντική.

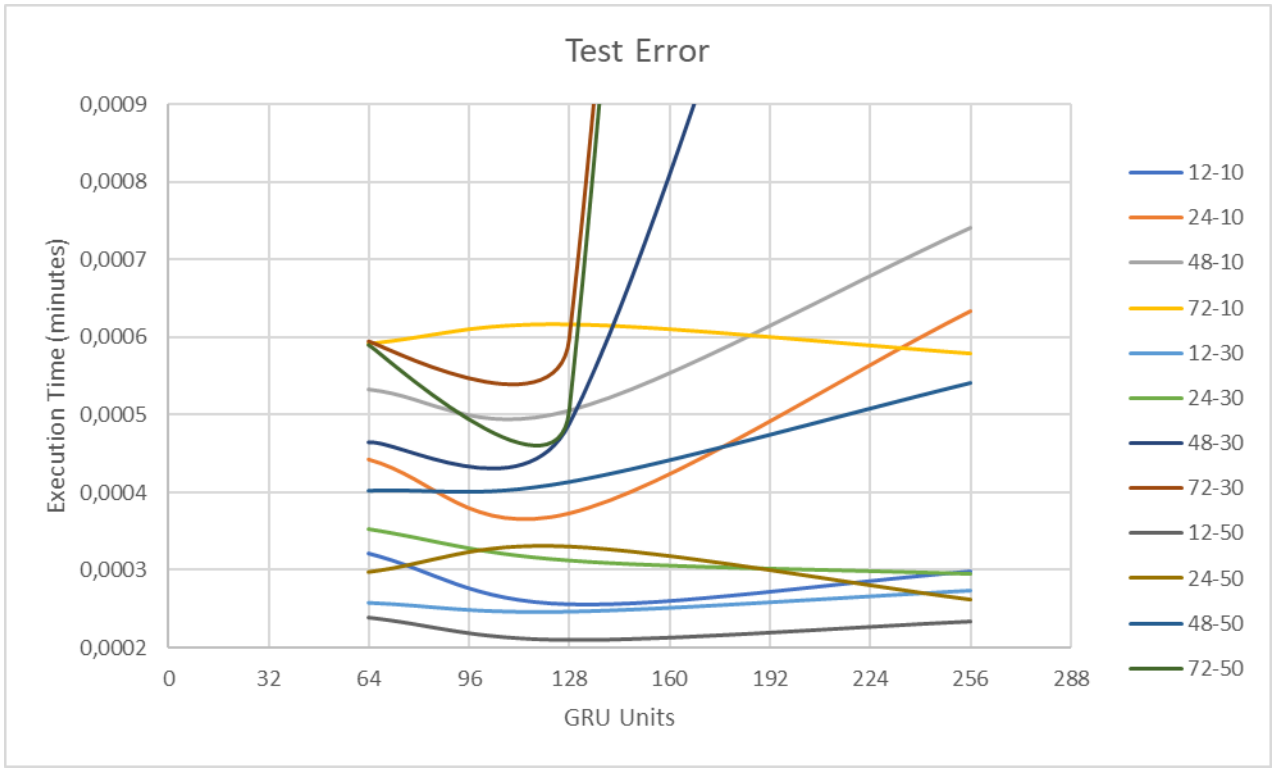
Σφάλμα

Εκπαίδευση



Σχεδιάγραμμα 4.4.3.1.2 Τελική τιμή σφάλματος εκπαίδευσης συναρτήσει του πλήθους λογικών μονάδων GRU, του μεγέθους χρονοσειράς και του πλήθους εποχών, με σταθερό μέγεθος δέσμης ίσο με 16

Επαλήθευση



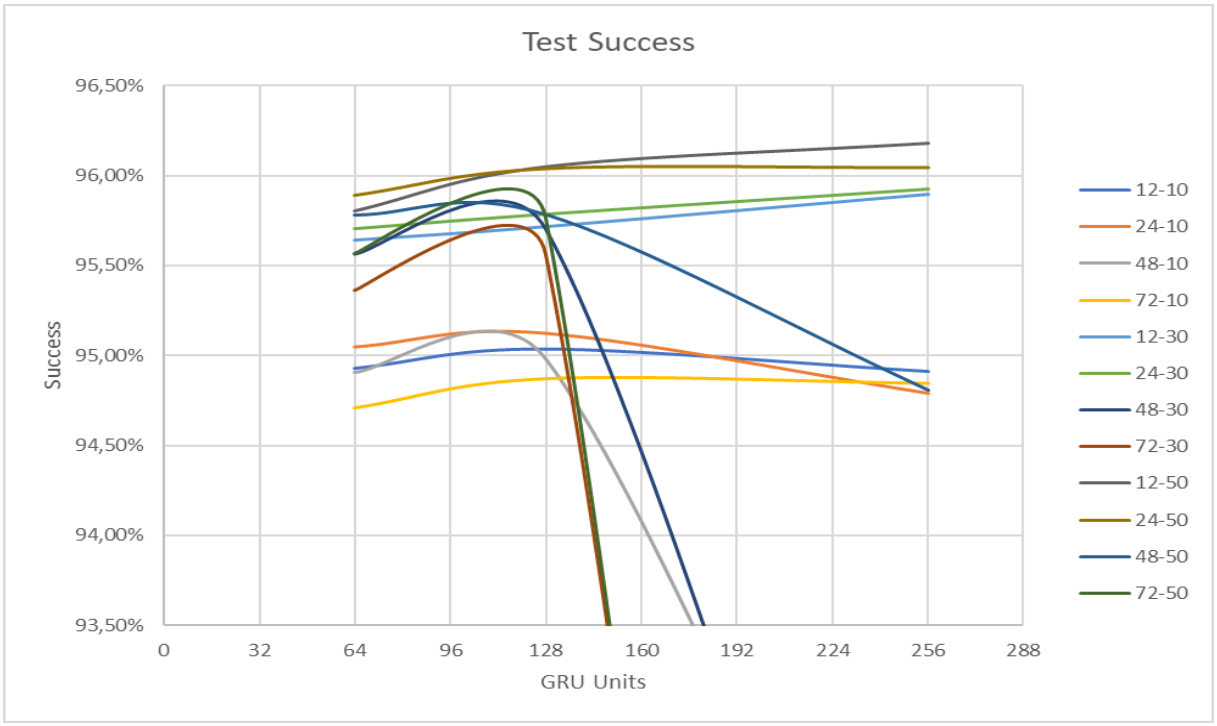
Σχεδιάγραμμα 4.4.3.1.3 Τελική τιμή σφάλματος επαλήθευσης συναρτήσει του πλήθους λογικών μονάδων GRU, του μεγέθους χρονοσειράς και του πλήθους εποχών, με σταθερό μέγεθος δέσμης ίσο με 64

Επιτυχία

Εκπαίδευση

Σχεδιάγραμμα 4.4.3.1.4 Τελική τιμή ποσοστού επιτυχίας εκπαίδευσης συναρτήσει του πλήθους λογικών μονάδων GRU, του μεγέθους χρονοσειράς και του πλήθους εποχών, με σταθερό μέγεθος δέσμης ίσο με 16

Επαλήθευση



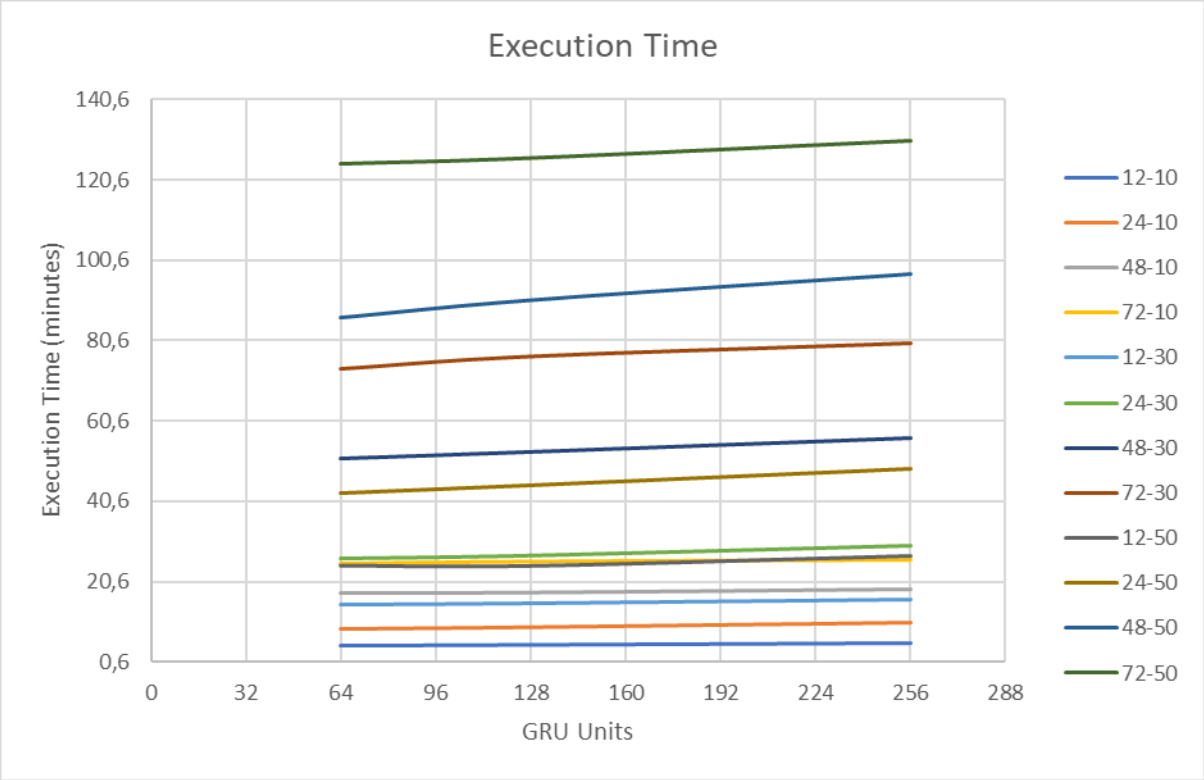
Σχεδιάγραμμα 4.4.3.1.5 Τελική τιμή ποσοστού επιτυχίας επαλήθευσης συναρτήσει του πλήθους λογικών μονάδων GRU, του μεγέθους χρονοσειράς και του πλήθους εποχών, με σταθερό μέγεθος δέσμης ίσο με 16

4.4.3.2 Μέγεθος δέσμης 32

Στις πιο κάτω γραφικές παραστάσεις βλέπουμε τα αποτελέσματα για εκτελέσεις με σταθερό ρυθμό μάθησης, ίσο με 0.001, σταθερό μέγεθος δέσμης, ίσο με 32 και σταθερό ρυθμό Dropout ίσο με 0.1. Παράλληλα έχουμε μεταβλητές τιμές το μέγεθος χρονοσειράς, τον αριθμό των εποχών και το πλήθος των λογικών μονάδων GRU. Οι στατιστικές που θα αναλύσουμε είναι ο χρόνος εκτέλεσης, το σφάλμα και το ποσοστό επιτυχίας.

Στις επόμενες γραφικές παραστάσεις ο άξονας x εκφράζει το πλήθος των μονάδων GRU, αντίστοιχα κάθε γραμμή ενώνει τα αποτελέσματα που προκύπτουν μεταβάλλοντας διακριτά το πλήθος των λογικών μονάδων GRU, αλλά κρατώντας σταθερό το πλήθος των εποχών και το μέγεθος χρονοσειράς. Επομένως, για παράδειγμα η γραμμή 12-10, εκφράζει τη μεταβολή της υπό έλεγχο τιμής, για διάφορα διακριτά πλήθη λογικών μονάδων GRU με σταθερό μέγεθος χρονοσειράς ίσο με 12 και πλήθος εποχών ίσο με 10.

Χρόνος εκτέλεσης

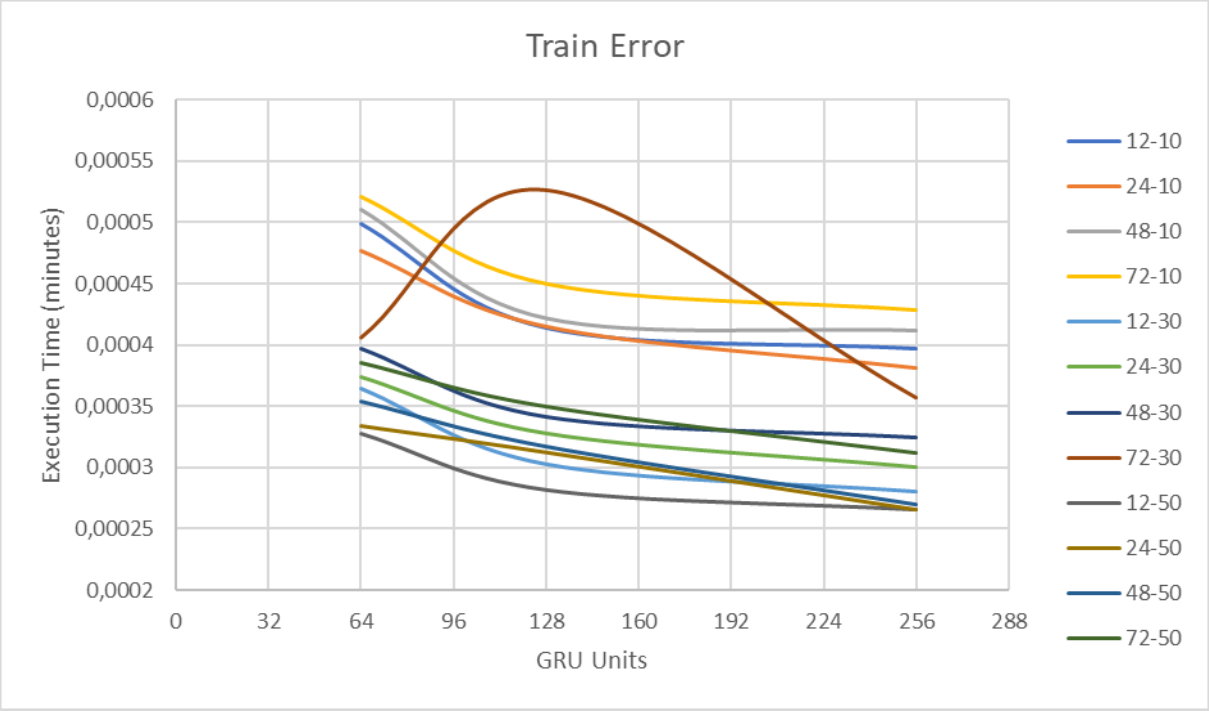


Σχεδιάγραμμα 4.4.3.2.1 Χρόνος εκτέλεσης συναρτήσει του πλήθους λογικών μονάδων GRU, του μεγέθους χρονοσειράς και του πλήθους εποχών, με σταθερό μέγεθος δέσμης ίσο με 32

Όσον αφορά τον χρόνο εκτέλεσης, τα αποτελέσματα μας είναι απολύτως αναμενόμενα, όσο αυξάνονται οι εποχές και το μέγεθος χρονοσειράς, ο χρόνος αυξάνεται. Αντίστοιχα, η αύξηση στις λογικές μονάδες GRU επιφέρει και αυτή αύξηση στον χρόνο εκτέλεσης, παρόλα αυτά η επίδρασή της δεν φαίνεται να είναι σημαντική.

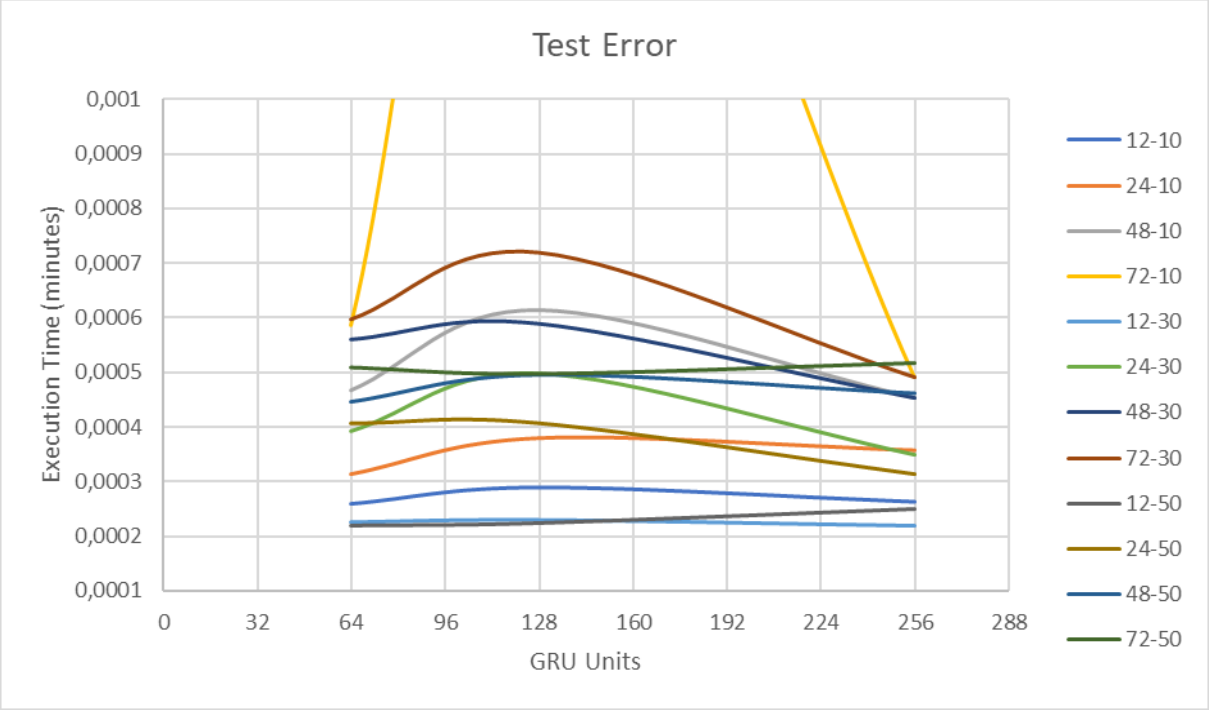
Σφάλμα

Εκπαίδευση



Σχεδιάγραμμα 4.4.3.2.2 Τελική τιμή σφάλματος εκπαίδευσης συναρτήσει του πλήθους λογικών μονάδων GRU, του μεγέθους χρονοσειράς και του πλήθους εποχών, με σταθερό μέγεθος δέσμης ίσο με 32

Επαλήθευση



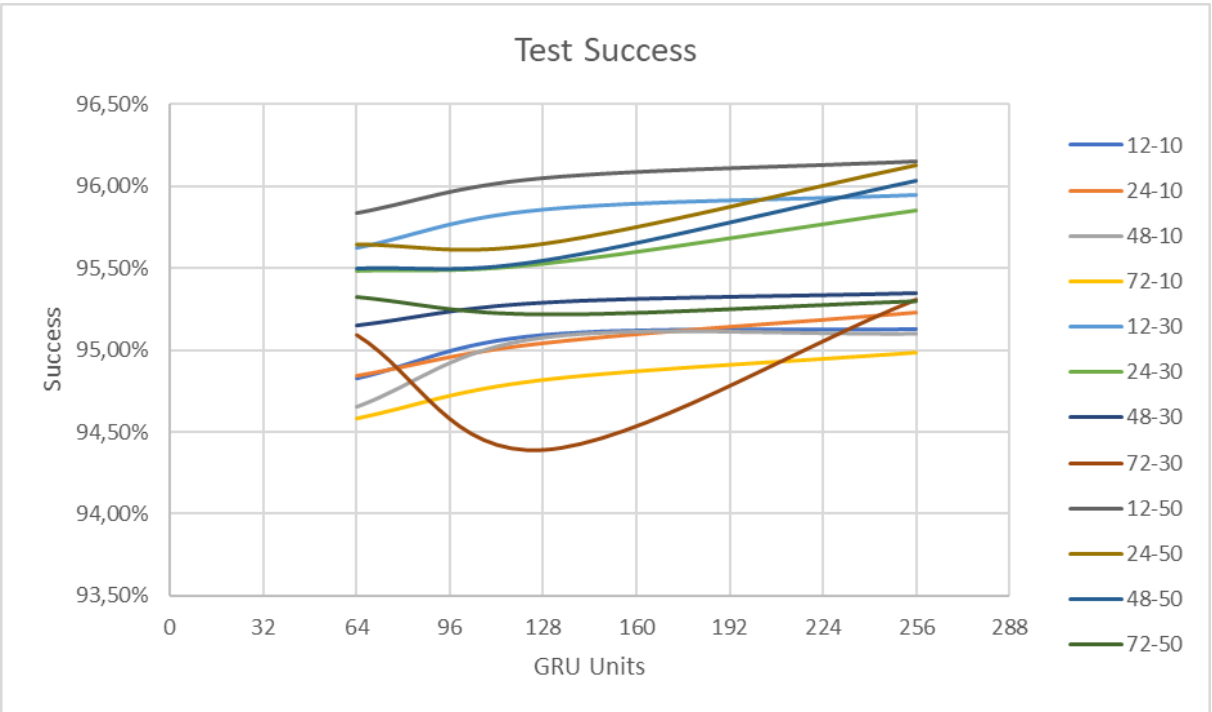
Σχεδιάγραμμα 4.4.3.2.3 Τελική τιμή σφάλματος επαλήθευσης συναρτήσει του πλήθους λογικών μονάδων GRU, του μεγέθους χρονοσειράς και του πλήθους εποχών, με σταθερό μέγεθος δέσμης ίσο με 32

Επιτυχία

Εκπαίδευση

Σχεδιάγραμμα 4.4.3.2.4 Τελική τιμή ποσοστού επιτυχίας εκπαίδευσης συναρτήσει του πλήθους λογικών μονάδων GRU, του μεγέθους χρονοσειράς και του πλήθους εποχών, με σταθερό μέγεθος δέσμης ίσο με 32

Επαλήθευση



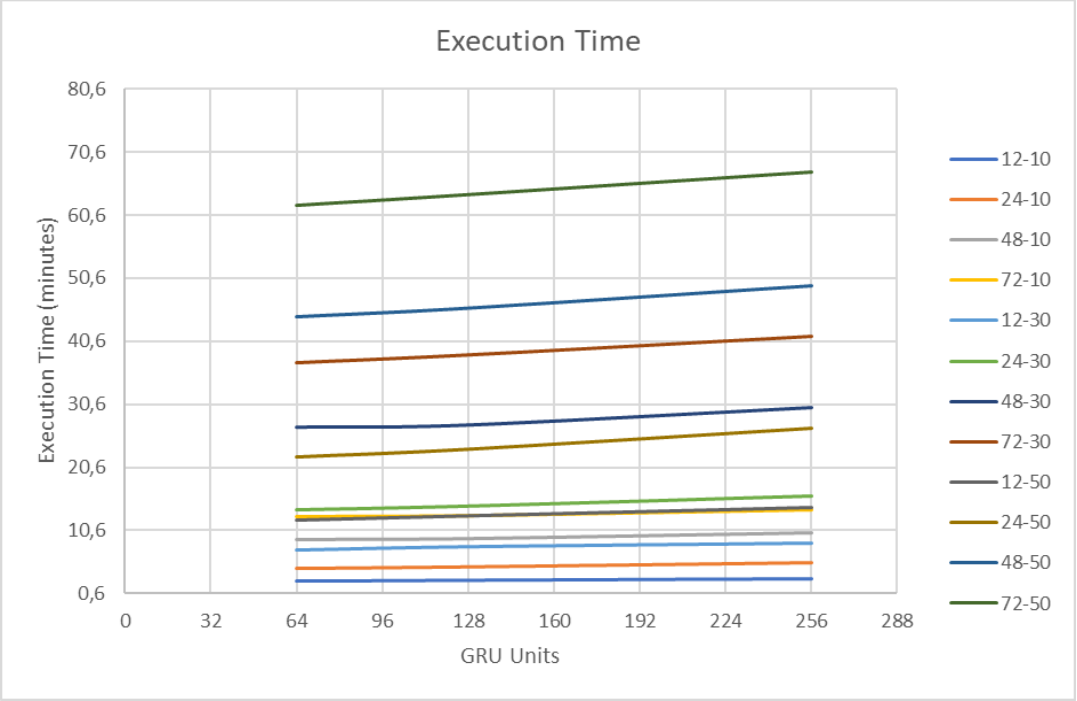
Σχεδιάγραμμα 4.4.3.2.5 Τελική τιμή ποσοστού επιτυχίας επαλήθευσης συναρτήσει του πλήθους λογικών μονάδων GRU, του μεγέθους χρονοσειράς και του πλήθους εποχών, με σταθερό μέγεθος δέσμης ίσο με 32

4.4.3.3 Μέγεθος δέσμης 64

Στις πιο κάτω γραφικές παραστάσεις βλέπουμε τα αποτελέσματα για εκτελέσεις με σταθερό ρυθμό μάθησης, ίσο με 0.001, σταθερό μέγεθος δέσμης, ίσο με 64 και σταθερό ρυθμό Dropout ίσο με 0.1. Παράλληλα έχουμε μεταβλητές τιμές το μέγεθος χρονοσειράς, τον αριθμό των εποχών και το πλήθος των λογικών μονάδων GRU. Οι στατιστικές που θα αναλύσουμε είναι ο χρόνος εκτέλεσης, το σφάλμα και το ποσοστό επιτυχίας.

Στις επόμενες γραφικές παραστάσεις ο άξονας x εκφράζει το πλήθος των μονάδων GRU, αντίστοιχα κάθε γραμμή ενώνει τα αποτελέσματα που προκύπτουν μεταβάλλοντας διακριτά το πλήθος των λογικών μονάδων GRU, αλλά κρατώντας σταθερό το πλήθος των εποχών και το μέγεθος χρονοσειράς. Επομένως, για παράδειγμα η γραμμή 12-10, εκφράζει τη μεταβολή της υπό έλεγχο τιμής, για διάφορα διακριτά πλήθη λογικών μονάδων GRU με σταθερό μέγεθος χρονοσειράς ίσο με 12 και πλήθος εποχών ίσο με 10.

Χρόνος εκτέλεσης

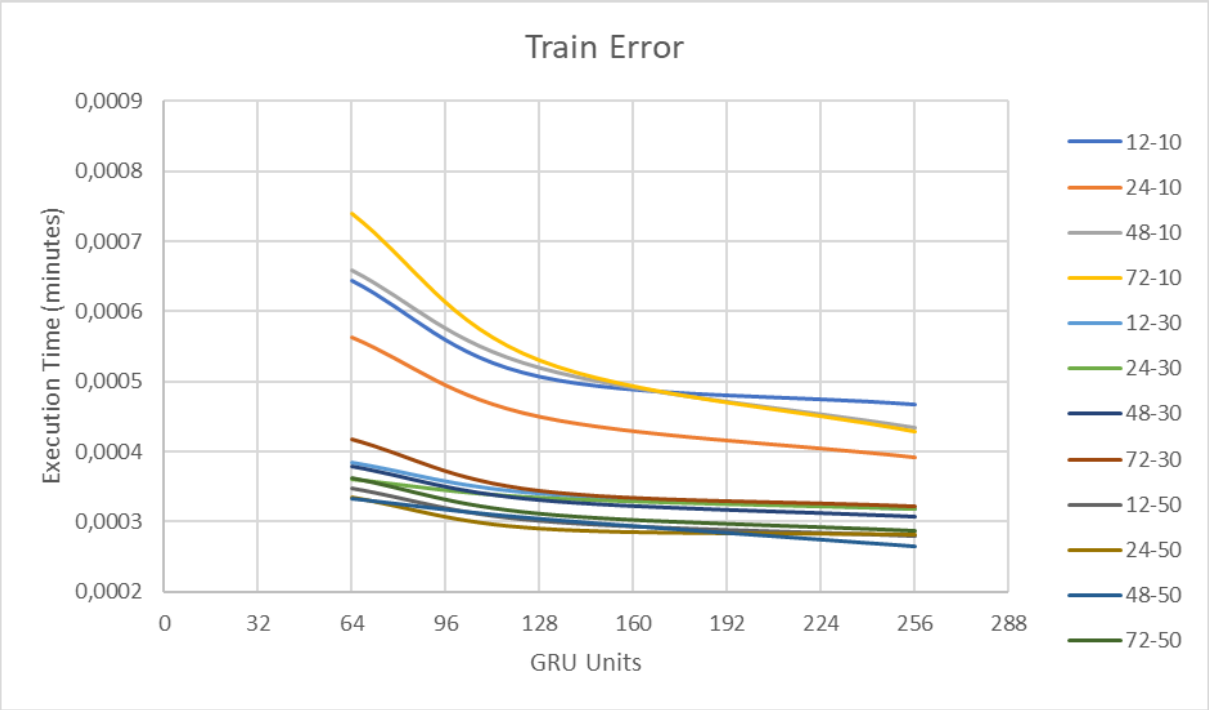


Σχεδιάγραμμα 4.4.3.3.1 Χρόνος εκτέλεσης συναρτήσει του πλήθους λογικών μονάδων GRU, του μεγέθους χρονοσειράς και του πλήθους εποχών, με σταθερό μέγεθος δέσμης ίσο με 64

Όσον αφορά τον χρόνο εκτέλεσης, τα αποτελέσματα μας είναι απολύτως αναμενόμενα, όσο αυξάνονται οι εποχές και το μέγεθος χρονοσειράς, ο χρόνος αυξάνεται. Αντίστοιχα, η αύξηση στις λογικές μονάδες GRU επιφέρει και αυτή αύξηση στον χρόνο εκτέλεσης, παρόλα αυτά η επίδρασή της δεν φαίνεται να είναι σημαντική.

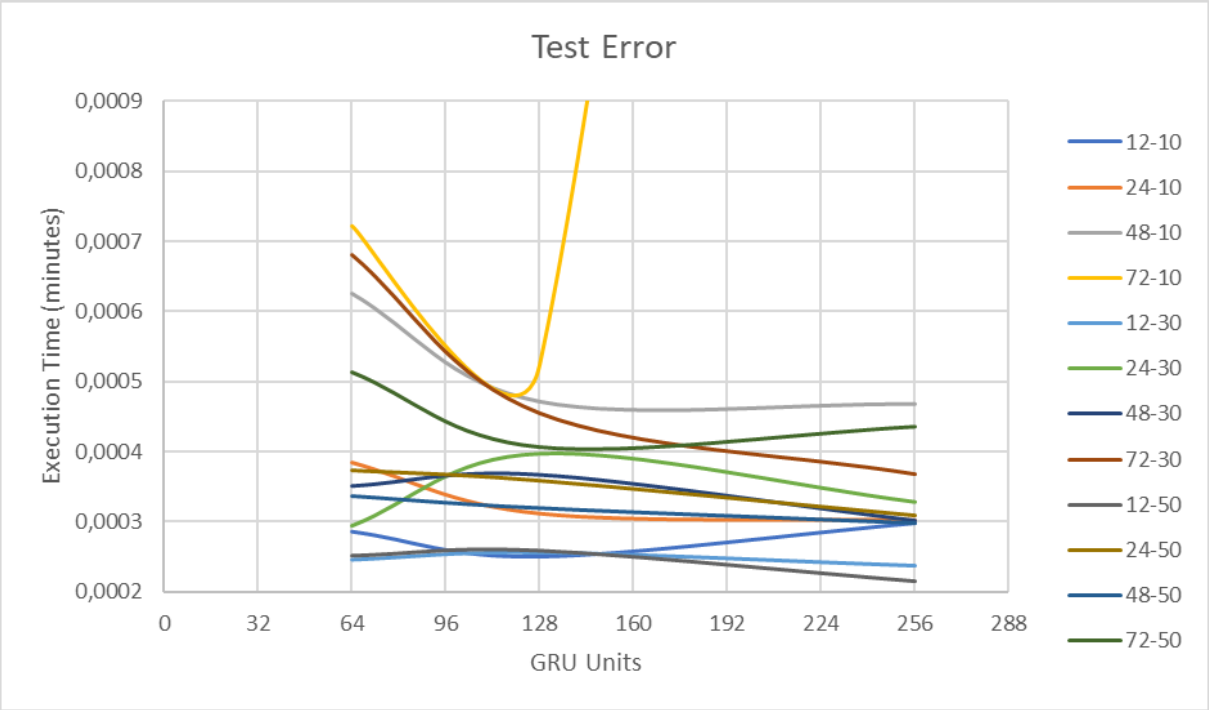
Σφάλμα

Εκπαίδευση



Σχεδιάγραμμα 4.4.3.3.2 Τελική τιμή σφάλματος εκπαίδευσης συναρτήσει του πλήθους λογικών μονάδων GRU, του μεγέθους χρονοσειράς και του πλήθους εποχών, με σταθερό μέγεθος δέσμης ίσο με 64

Επαλήθευση



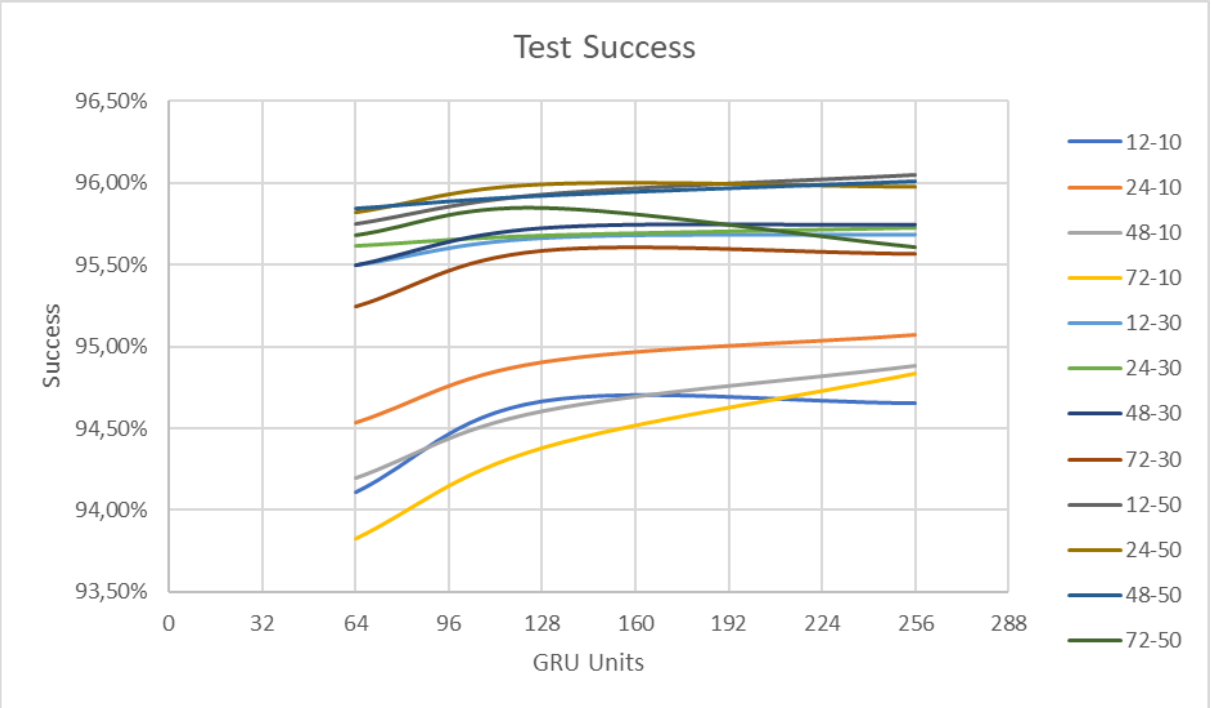
Σχεδιάγραμμα 4.4.3.3.3 Τελική τιμή σφάλματος επαλήθευσης συναρτήσει του πλήθους λογικών μονάδων GRU, του μεγέθους χρονοσειράς και του πλήθους εποχών, με σταθερό μέγεθος δέσμης ίσο με 64

Επιτυχία

Εκπαίδευση

Σχεδιάγραμμα 4.4.3.3.4 Τελική τιμή ποσοστού επιτυχίας εκπαίδευσης συναρτήσει του πλήθους λογικών μονάδων GRU, του μεγέθους χρονοσειράς και του πλήθους εποχών, με σταθερό μέγεθος δέσμης ίσο με 64

Επαλήθευση



Σχεδιάγραμμα 4.4.3.3.5 Τελική τιμή ποσοστού επιτυχίας επαλήθευσης συναρτήσει του πλήθους λογικών μονάδων GRU, του μεγέθους χρονοσειράς και του πλήθους εποχών, με σταθερό μέγεθος δέσμης ίσο με 64

Στον παρακάτω πίνακα εξετάζουμε συγκεντρωτικά τις καλύτερες εκτελέσεις από τα προηγούμενα πειράματα.

Timestep	Epochs	GRU units	Batch size	Train error	Test error	Train success	Test success	Execution time (minutes)
12	50	128	16	0.00027	0.00021	96.05%	94.81%	47.87378229
12	50	64	32	0.00033	0.00022	95.84%	94.93%	24.68941966
12	50	64	64	0.00035	0.00025	95.75%	94.79%	12.1778396

Σχεδιάγραμμα 4.4.3.1 Συγκεντρωτικός πίνακας καλύτερων εκτελέσεων με μεταβλητό μέγεθος δέσμης

Παρατηρούμε ότι για δύο από τις τέσσερις μεταβλητές οι τιμές είναι σταθερές. Παράλληλα, βλέπουμε πως όσο αυξάνεται το μέγεθος δέσμης τα αποτελέσματα δεν αλλάζουν ιδιαίτερα, παρόλα αυτά, ο χρόνος εκτέλεσης μειώνεται. Εξαιτίας αυτής της παρατήρησης θα διερευνήσουμε το εύρος τιμών που δύναται να λάβει το μέγεθος δέσμης, χρησιμοποιώντας τον βέλτιστο συνδυασμό μεταβλητών.

Timestep	Epochs	GRU units	Batch size	Train error	Test error	Train success	Test success	Execution time (minutes)
12	50	64	128	0.00050	0.00033	94.93%	92.52%	6.443715
12	50	64	256	0.00087	0.00042	93.35%	90.88%	3.539067
12	50	64	512	0.00073	0.00026	93.88%	93.86%	1.975014
12	50	64	1024	0.0018	0.00052	90.45%	92.39%	1.171726

Σχεδιάγραμμα 4.4.3.2 Εξερεύνηση ορίων μεγέθους δέσμης

Όπως είναι εμφανές, οι επιδόσεις του δικτύου μας χειροτερεύουν με την αύξηση του μεγέθους δέσμης πάνω από την τιμή 64.

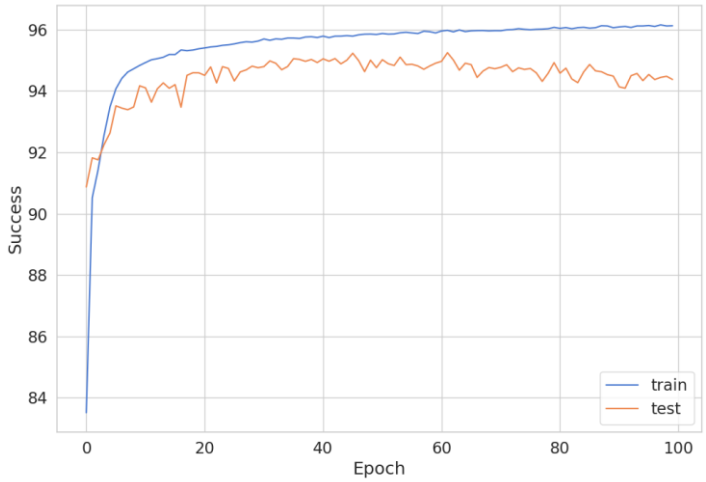
4.4.4 Επέκταση εποχών

Για να ερευνήσουμε πιο ενδελεχώς την επίδραση των εποχών στην εκπαίδευση του δικτύου μας θα δοκιμάσουμε τη βέλτιστη σύνθεση που έχουμε χτίσει μέχρι τώρα (χρονικό βήμα ίσο με 12, λογικές μονάδες ίσες με 64, μέγεθος δέσμης ίσο με 32, ρυθμό Dropout ίσο με 0.1 και ρυθμό μάθησης ίσο με 0.001) αλλά θα αυξήσουμε το πλήθος των εποχών

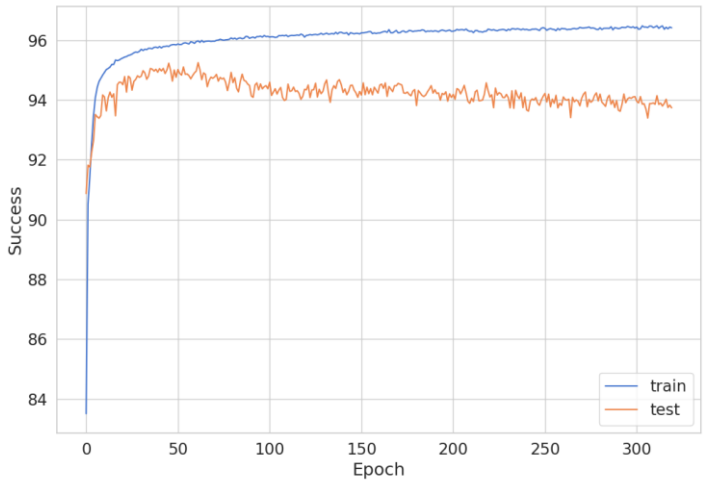
Epochs	Train Error	Test Error	Train Success	Test Success	Execution Time (minutes)
100	0.00029	0.00025	96.12%	94.37%	47.39069
320	0.00025	0.00035	96.41%	93.73%	154.122713
500	0.00024	0.00028	96.55%	94.06%	238.835782

Σχεδιάγραμμα 4.4.4.1 Εξερεύνηση ορίων πλήθους εποχών

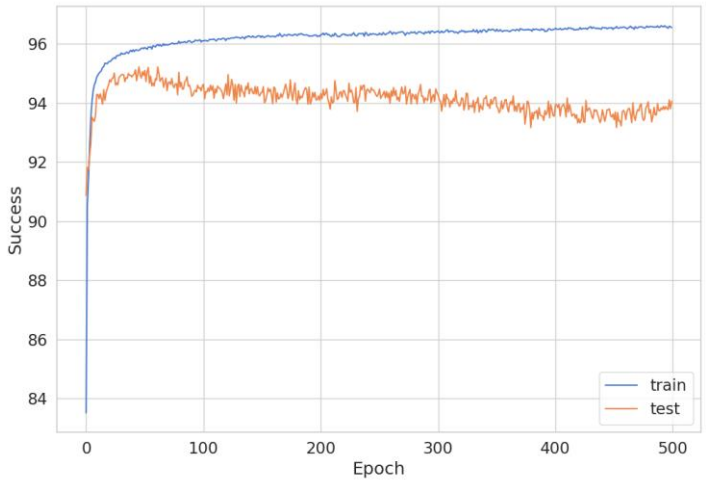
Σε πρώτη ανάγνωση τα αποτελέσματα φαίνονται πολύ καλύτερα, ωστόσο αν κοιτάξουμε τη γενική συμπεριφορά των γραφικών παραστάσεων της ποσοστιαίας επιτυχίας θα δούμε ότι υπάρχουν σκαμπανεβάσματα που παραπέμπουν σε υπερεκπαίδευση, φαινόμενο που επιβεβαιώνει και το μειωμένο ποσοστό επιτυχίας στη φάση επαλήθευσης.



Σχεδιάγραμμα 4.4.4.2 Εκτέλεση με πλήθος εποχών 100



Σχεδιάγραμμα 4.4.4.3 Εκτέλεση με πλήθος εποχών 320



Σχεδιάγραμμα 4.4.4.4 Εκτέλεση με πλήθος εποχών 500

Από τις παραπάνω εκτελέσεις καταλήγουμε στο συμπέρασμα ότι ο ιδανικός αριθμός εποχών, για το δίκτυο GRU κυμαίνεται μεταξύ του πενήντα και του εκατό.

4.4.5 Συζήτηση αποτελεσμάτων

Από τις παραπάνω εκτελέσεις μπορούμε εύκολα να συμπεράνουμε πως το δίκτυό μας είναι αρκετά ευαίσθητο στις αλλαγές κάποιων μεταβλητών, όπως ο ρυθμός μάθησης και ο ρυθμός του Dropout. Αυτό βασίζεται σε σχεδιαγράμματα όπως το 4.4.1.5 και το 4.4.2.3 που παρουσιάζουν ανώμαλες συμπεριφορές των δικτύων σε εκτελέσεις που αυτές οι μεταβλητές έχουν αλλάξει ελαφρώς. Στο σχεδιάγραμμα 4.4.1.5 παρατηρούμε πως για ρυθμό μάθησης ίσο με 0.1 οι προβλέψεις μας είναι πάρα πολύ λανθασμένες ειδικά σε σύγκριση με τα σχεδιαγράμματα 4.4.1.4 και 4.4.13 όπου οι προβλέψεις μας είναι πολύ κοντά στην πραγματικότητα. Αντίστοιχα, στο σχεδιάγραμμα 4.4.2.3 παρατηρούμε πως για ρυθμό Dropout ίσο με μηδέν έχουμε πολύ χαμηλό ποσοστό επιτυχίας. Για τις υπόλοιπες μεταβλητές δεν παρατηρήσαμε κάποια συστηματικά περίεργη συμπεριφορά. Υπήρχαν κάποιες εκτελέσεις όπου παρατηρήσαμε το φαινόμενο της υπερμάθησης αλλά κάτι τέτοιο είναι λογικό και νοουμένου ότι δεν εμφανίζεται με κάποιο συγκεκριμένο μοτίβο δεν πρέπει να μας ανησυχεί. Συμπερασματικά, με βάση τις παρατηρήσεις μας η βέλτιστη σύνθεση του δικτύου LSTM στο πρόβλημα αυτό είναι αυτή που βλέπουμε στο σχεδιάγραμμα 4.4.5.1.

Μεταβλητή	Βέλτιστη Τιμή
Μέγεθος χρονοσειράς	12
Πλήθος εποχών	50
Πλήθος λογικών μονάδων GRU	64
Dropout rate	0.1
Batch size	32
Ρυθμός μάθησης	0.001

Σχεδιάγραμμα 4.4.5.1 Πίνακας βέλτιστης σύνθεσης δικτύου GRU

4.5 Πολυεπίπεδα δίκτυα

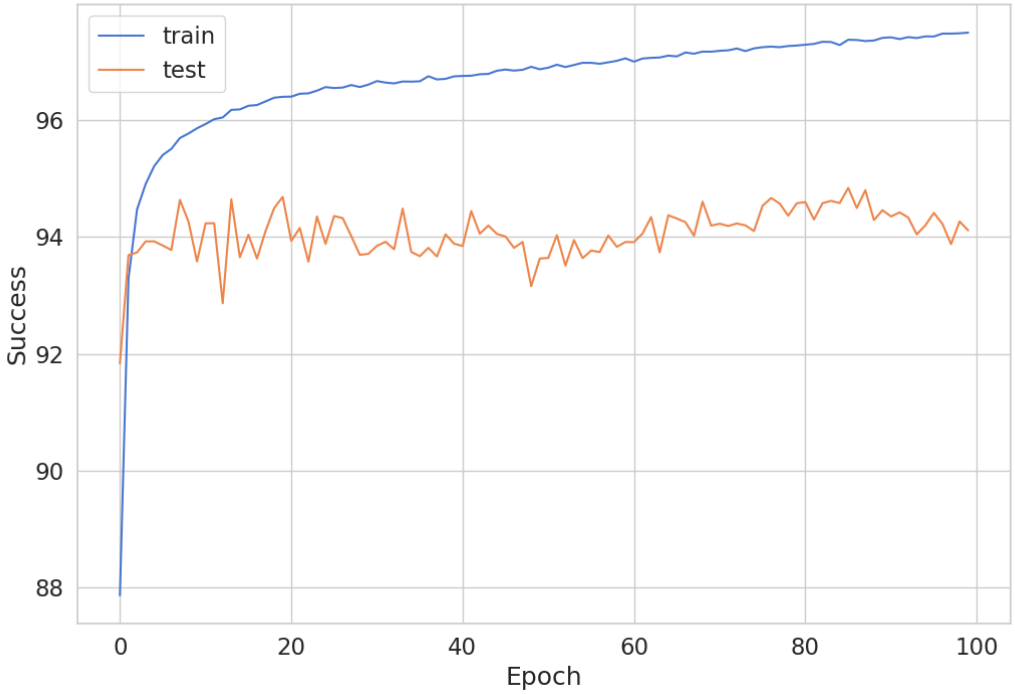
Κατά τη διάρκεια των πειραμάτων μας αποφασίσαμε να προσθέσουμε ένα ακόμα επίπεδο στα δίκτυά μας. Κάτι τέτοιο ενδείκνυται σε προβλήματα τα οποία είναι σύνθετα στη φύση τους όπως η παραγωγή κειμένου, όπως φαίνεται από τις έρευνες των Sutskever et al [18] και των Irsoy and Cardie [19]. Αυτό που θέλαμε να διερευνήσουμε είναι κατά πόσο ένα κρυφό επίπεδο LSTM και GRU θα βελτίωνε τα αποτελέσματά μας. Τα αποτελέσματα που έχουμε δεν δείχνουν κάτι τέτοιο και επιβεβαιώνουν τη θεωρία, καθώς η πολυπλοκότητα των δεδομένων μας δεν είναι ιδιαίτερα υψηλή.

4.5.1 Πολυεπίπεδο LSTM

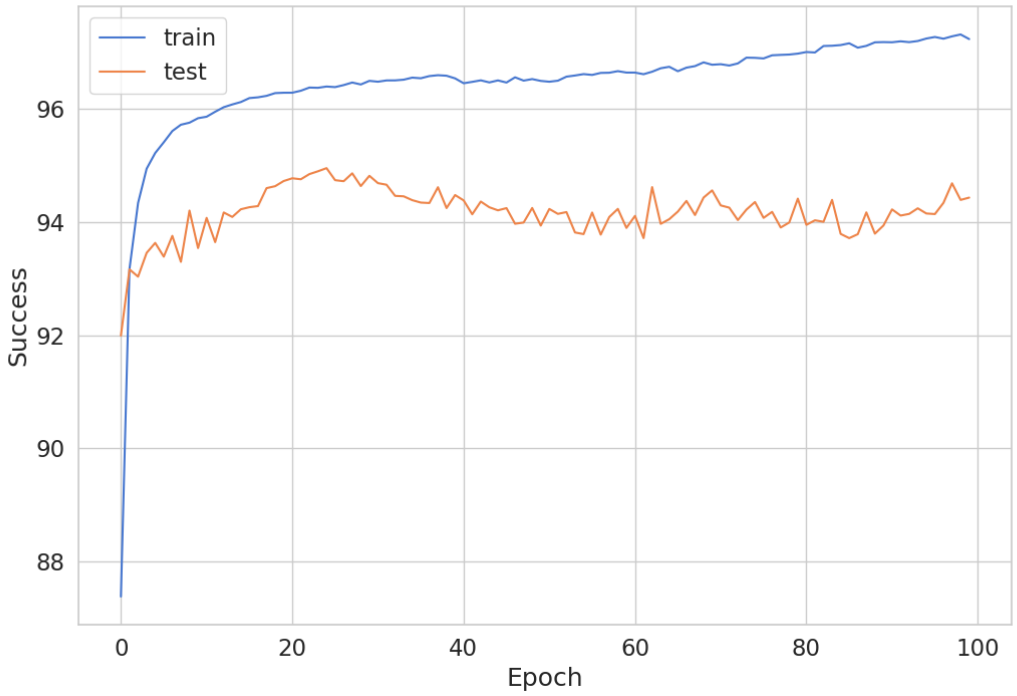
Για να ελέγξουμε την απόδοση του πολυεπίπεδου δικτύου LSTM, δοκιμάσαμε τις καλύτερες συνθέσεις στις οποίες καταλήξαμε προηγουμένως. Τα αποτελέσματα και παραπάνω λεπτομέρειες φαίνονται στον παρακάτω πίνακα.

Timestep	Epochs	LSTM units	Batch size	Train error	Test error	Train success	Test success	Execution time (minutes)
12	100	256	16	0.00011	0.00026	97.49%	94.11%	65.174499
12	100	256	32	0.00013	0.00022	97.24%	94.43%	37.103917
12	100	256	64	0.00015	0.00035	97.03%	92.91%	24.968737

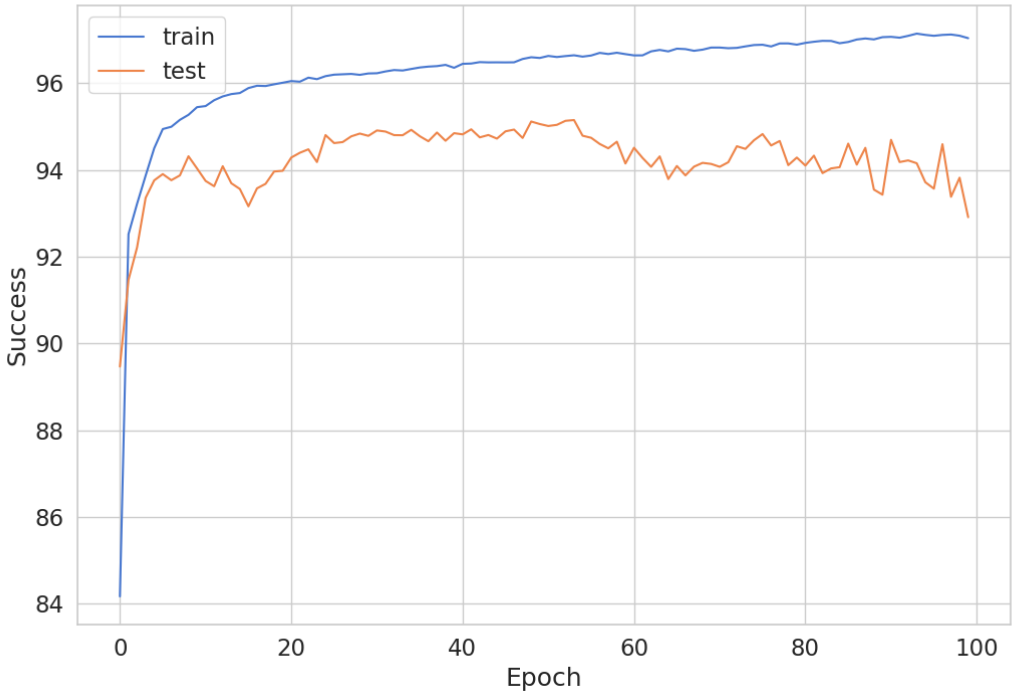
Σχεδιάγραμμα 4.5.1.1 Πίνακας αποτελεσμάτων πολυεπίπεδου δικτύου LSTM



Σχεδιάγραμμα 4.5.1.2 Ποσοστό επιτυχίας πολυεπίπεδου δικτύου LSTM με μέγεθος δέσμης 16



Σχεδιάγραμμα 4.5.1.3 Ποσοστό επιτυχίας πολυεπίπεδου δικτύου LSTM με μέγεθος δέσμης 32



Σχεδιάγραμμα 4.5.1.4 Ποσοστό επιτυχίας πολυεπίπεδου δικτύου LSTM με μέγεθος δέσμης 64

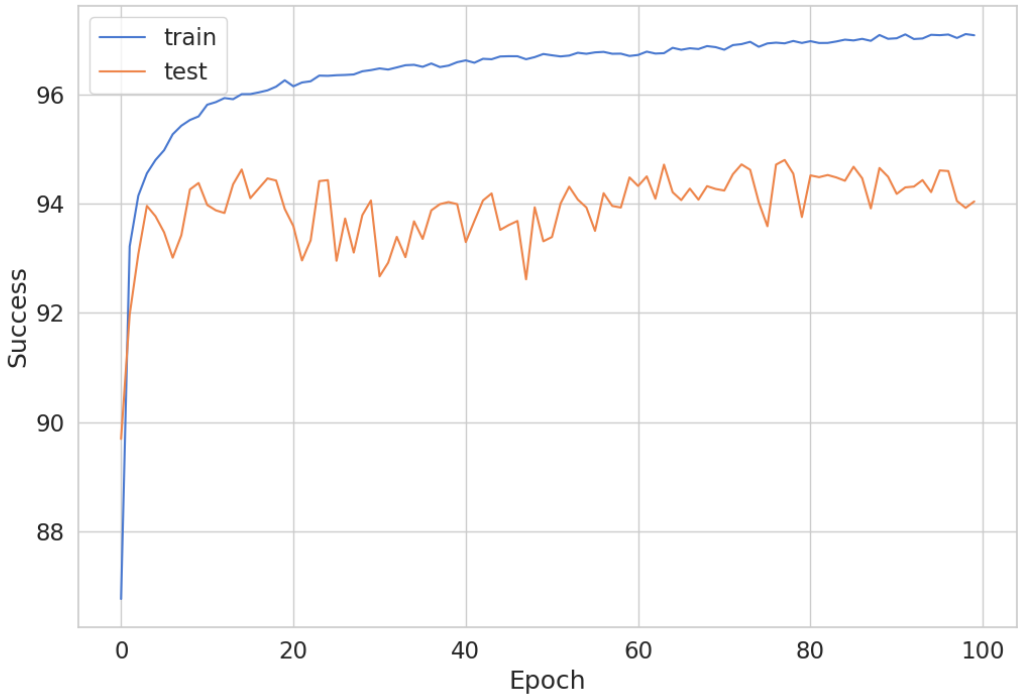
Από τις παραπάνω γραφικές παρατηρούμε πως ναι μεν το ποσοστό επιτυχίας εκπαίδευσης ανεβαίνει αρκετά, αλλά το δε ποσοστό επιτυχίας επαλήθευσης είναι πολύ χαμηλότερο από το αντίστοιχο στο μονοεπίπεδο δίκτυο.

4.5.2 Πολυεπίπεδο GRU

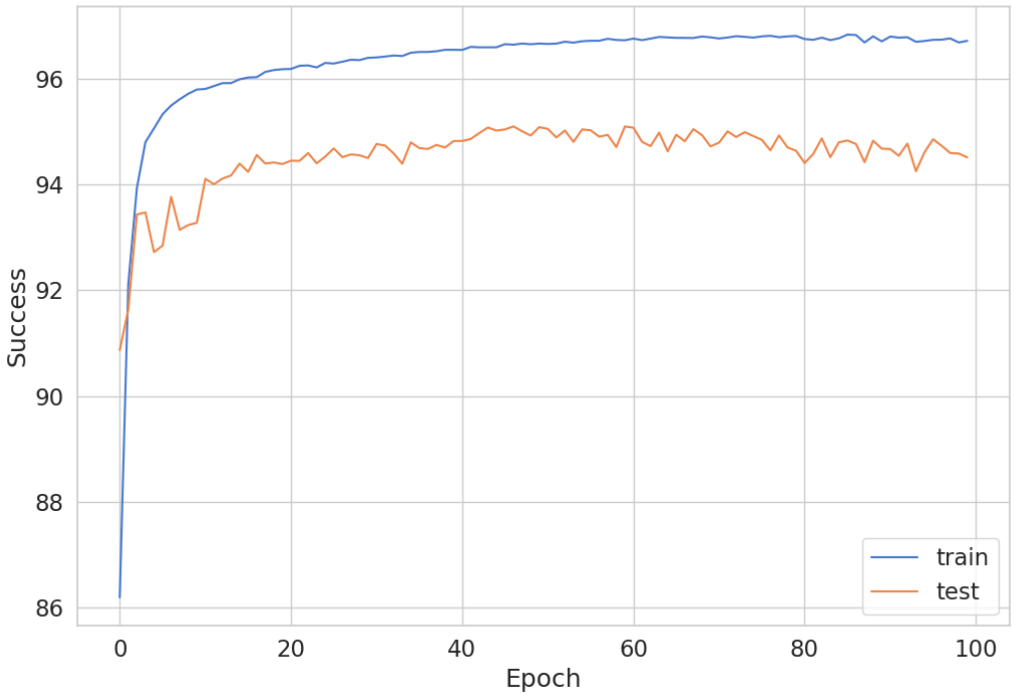
Για να ελέγξουμε την απόδοση του πολυεπίπεδου δικτύου GRU, δοκιμάσαμε τις καλύτερες συνθέσεις στις οποίες καταλήξαμε προηγουμένως. Τα αποτελέσματα και παραπάνω λεπτομέρειες φαίνονται στον παρακάτω πίνακα.

Timestep	Epochs	GRU units	Batch size	Train error	Test error	Train success	Test success	Execution time (minutes)
12	100	128	16	0.00014	0.00032	97.09%	94.05%	196.540841
12	100	64	32	0.00017	0.00023	96.72%	94.51%	95.795591
12	100	64	64	0.00018	0.00028	96.77%	94.04%	48.392464

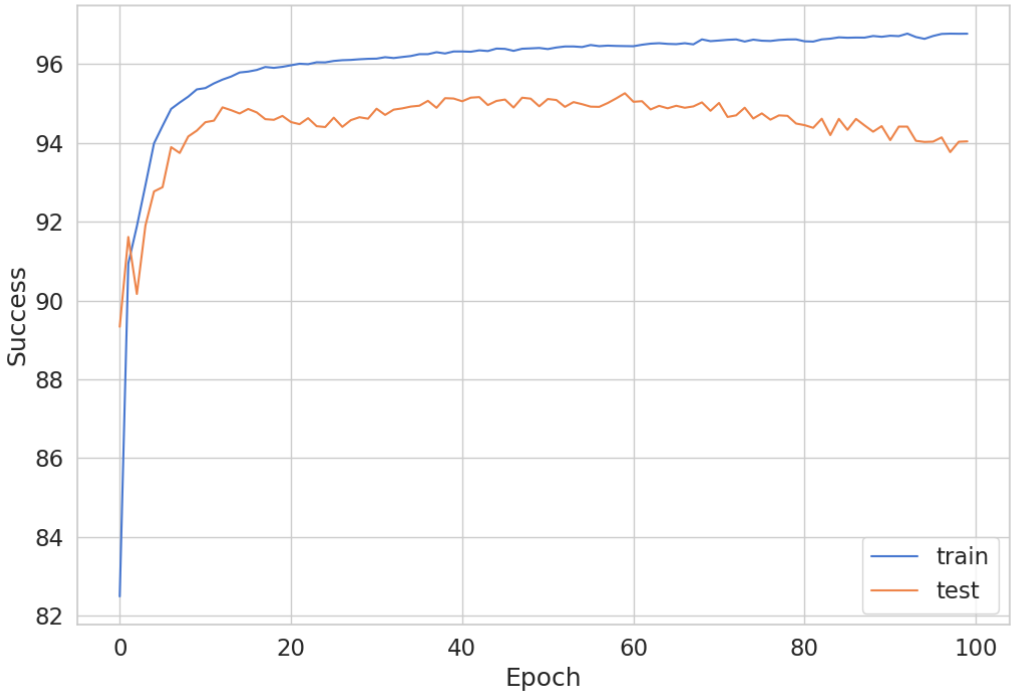
Σχεδιάγραμμα 4.5.2.1 Πίνακας αποτελεσμάτων πολυεπίπεδου δικτύου GRU



Σχεδιάγραμμα 4.5.2.2 Ποσοστό επιτυχίας πολυεπίπεδου δικτύου GRU με μέγεθος δέσμης 16



Σχεδιάγραμμα 4.5.2.3 Ποσοστό επιτυχίας πολυεπίπεδου δικτύου GRU με μέγεθος δέσμης 32



Σχεδιάγραμμα 4.5.2.4 Ποσοστό επιτυχίας πολυεπίπεδου δικτύου GRU με μέγεθος δέσμης 64

Από τις παραπάνω γραφικές παρατηρούμε πως ναι μεν το ποσοστό επιτυχίας εκπαίδευσης ανεβαίνει αρκετά, αλλά το δε ποσοστό επιτυχίας επαλήθευσης είναι πολύ χαμηλότερο από το αντίστοιχο στο μονοεπίπεδο δίκτυο.

Κεφάλαιο 5

Συμπεράσματα

5.1 Συμπεράσματα

Η αξία των φυσικών πόρων και ακόμα περισσότερο των μη-ανανεώσιμων είναι αδιαπραγμάτευτη, ιδιαίτερα στην παρούσα κοινωνικοπολιτική κατάσταση η αξία των ορυκτών καυσίμων, όπως το φυσικό αέριο, έχει αυξηθεί δραματικά. Επομένως, για να μπορέσουμε να διαχειριστούμε πιο αποδοτικά τους πόρους μας επιβάλλεται να μπορέσουμε να προβλέψουμε τη ζήτησή τους. Για να καταφέρουμε ωστόσο να κάνουμε προβλέψεις σε ένα αποδεκτό επίπεδο ακρίβειας, θα πρέπει να απομονώσουμε τους παράγοντες που επηρεάζουν τη ζήτηση. Πιο συγκεκριμένα, στη μελέτη μας χρησιμοποιήσαμε δεδομένα όπως η ταχύτητα του αέρα, η θερμοκρασία και το αν μια μέρα είναι αργία ή όχι. Επιπρόσθετα, κάποιοι άλλοι παράγοντες οι οποίοι θα μπορούσαν να μας βοηθήσουν είναι η υγρασία και κάποιοι πιο πολιτισμικοί παράγοντες, παραδείγματος χάριν το πώς αλλάζει ο πληθυσμός της υπό εξέταση περιοχής.

Στόχος αυτής της διπλωματικής εργασίας ήταν η υλοποίηση ενός συστήματος βασισμένο σε νευρωνικά δίκτυα με ανάδραση, τα οποία μπορούν να μας παρέχουν ακριβείς προβλέψεις. Για να το επιτύχουμε αυτό χρησιμοποιήσαμε δεδομένα από μια Βρετανική εταιρεία παροχής φυσικού αερίου. Τα δεδομένα αυτά περιέχουν την πραγματική ζήτηση φυσικού αερίου ανά ώρα καλύπτοντας ένα φάσμα επτά ετών. Επιπρόσθετα, χρησιμοποιήσαμε την ταχύτητα των ανέμων ανά τέσσερις ώρες και τη θερμοκρασία ανά δύο ώρες. Τα νευρωνικά δίκτυα με ανάδραση που χρησιμοποιήσαμε ήταν τα δίκτυα LSTM και GRU. Τα δύο αυτά δίκτυα χρησιμοποιούν τις λογικές πύλες για να επιλέξουν ποια κομμάτια πληροφορίας θα αξιοποιήσουν και ποια θα απορρίψουν.

Στον χώρο της μηχανικής μάθησης δεν υπάρχει πανάκεια ή ιδανικές λύσεις. Για το λόγο αυτό χρειάστηκε να διερευνήσουμε διάφορους συνδυασμούς που καθορίζουν την τοπολογία και την λειτουργία των δικτύων μας. Επομένως, κατά την διάρκεια της έρευνάς μας πειραματιστήκαμε αρκετά με μεταβλητές όπως ο αριθμός των εποχών, ο ρυθμός μάθησης, το πλήθος λογικών μονάδων και άλλα. Σύμφωνα με τα αποτελέσματα, που συλλέξαμε από τα εν λόγω πειράματα, τα δύο αυτά νευρωνικά δίκτυα κατάφεραν να προβλέψουν με ικανοποιητική ακρίβεια την ωριαία ζήτηση φυσικού αερίου. Το δίκτυο LSTM πέτυχαμε ακρίβεια πρόβλεψης της τάξεως του 95.39% ενώ με το δίκτυο GRU 94.93%.

Με την εκπλήρωση αυτής της διπλωματικής εργασίας καταλήγουμε στο συμπέρασμα ότι ναι μεν τα προχωρημένα νευρωνικά δίκτυα LSTM και GRU μπορούν να κάνουν αποτελεσματικές προβλέψεις, αλλά σε σύγκριση με πιο απλά δίκτυα με ανάδραση όπως το δίκτυο Jordan ή το πολυεπίπεδο δίκτυο Perceptron με την τεχνική του κινούμενου παραθύρου, η διαφορά δεν είναι τόσο μεγάλη. Μια πολύ σημαντική διαφορά μεταξύ των δύο αυτών ομάδων είναι ότι τα LSTM & GRU αξιοποιούν πιο αποδοτικά δεδομένα με χρονική

εξάρτηση μακράς διαρκείας, ενώ τα Jordan και MLP με back propagation through time αξιοποιούν δεδομένα με μικρή χρονική εξάρτηση.

5.2 Μελλοντική εργασία

Στην παρούσα χρονική στιγμή, δεν υπάρχει κάποιο άλλο νευρωνικό δίκτυο με ανάδραση του οποίου θα ήταν δόκιμο να αναλύσουμε την επίδοση στο συγκεκριμένο πρόβλημα. Παρόλα αυτά, αν κάποιο νευρωνικό δίκτυο αναπτυχθεί θα είχε ενδιαφέρον να ερευνήσουμε την απόδοσή του στις προβλέψεις ζήτησης φυσικού αερίου ανά ώρα. Αντίστοιχα, θα μπορούσαμε να συντάξουμε ένα συνδυαστικό νευρωνικό δίκτυο[20][21][22][23] (ensemble neural network) το οποίο θα συνδυάσει όλες τις υπάρχουσες υλοποιήσεις δικτύων με ανάδραση που έχουμε υλοποιήσει και θα διατηρεί την πιο επιτυχημένη πρόβλεψη.

Επιπρόσθετα, θα μπορούσαμε να επεξεργαστούμε τα δεδομένα μας με τεχνικές εξόρυξης δεδομένων και να επαναλάβουμε τα πειράματά μας. Πιο συγκεκριμένα, μπορούμε να καθαρίσουμε το σύνολο δεδομένων μας από τιμές που είναι είτε πολύ χαμηλές, είτε πολύ υψηλές. Επίσης θα ήταν ενδιαφέρον να κατηγοριοποιήσουμε τα είδη των αργιών μέσα στο σύνολο δεδομένων μας, καθώς κάθε είδος αργίας καθορίζει με διαφορετικό τρόπο την τιμή της ζήτησης. Ωστόσο, κάτι που οφείλουμε να προσέξουμε είναι ότι οποιαδήποτε αλλαγή γίνει στο σύνολο δεδομένων επηρεάζει όλα τα δίκτυα, επομένως για να είμαστε ακριβείς πρέπει να επαναλάβουμε τα πειράματά μας με όλες τις μεθόδους.

Τέλος, θα μπορούσαμε να σπάσουμε το σύστημά μας σε δύο ή και παραπάνω μέρη. Τα μέρη αυτά θα αναλάβουν μια συγκεκριμένη κατηγορία δεδομένων, όπως οι αργίες, οι καλοκαιρινοί και οι χειμερινοί μήνες με αποτέλεσμα να απορροφήσουμε τυχούσες ανωμαλίες στα δεδομένα οι οποίες προκύπτουν είτε από αργίες, είτε από διαφορές στα καιρικά φαινόμενα.

Βιβλιογραφία

1. Aga.org. 2022. How Does the Natural Gas Delivery System Work? | American Gas Association. [online] Available at: <https://www.aga.org/natural-gas/delivery/how-does-the-natural-gas-delivery-system-work-/> [Accessed 3 January 2022].

2. C. Prastitis, “Πρόβλεψη ζήτησης φυσικού αερίου”, BSc Thesis, Dept of Computer Science, University of Cyprus, 2009.

3. K. Hadjidemetriou, “Πρόβλεψη ζήτησης φυσικού αερίου με τη χρήση νευρωνικών δικτύων”, BSc Thesis, Dept of Computer Science, University of Cyprus, 2010.

4. D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning Representations by Back-propagating Errors,” Nature, vol. 323, no. 6088, 533–536, 1986.

5. P. J. Werbos,"Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences." Doctoral Dissertation, Harvard University, Cambridge, MA, 1974.

6. D. S. Broomhead and D. Lowe, “Multivariable Functional Interpolation and Adaptive Networks,” Complex Systems, vol. 2, 321–355, 1988.

7. M. Jordan, “Serial order: A parallel distributed processing approach”, Univ. California San Diego, Inst. Cognitive Sci., ICS Rep. 8604, 1986.

8. T. Kohonen, "Self-organized formation of topologically correct feature maps." Biological Cybernetics, 43, 59–69, 1982.

9. P. J. Werbos, Backpropagation through time: What it does and how to do it, Proceedings of the IEEE, 78(10), 1550–1560, 1990.

10. S. Hochreiter, J. Schmidhuber, "Long short-term memory". Neural Computation. 9(8), 1735–1780, 1997.

11. O. Laib, M. T. Khadir, L. Mihaylova, “Toward efficient energy systems based on natural gas consumption prediction with LSTM Recurrent Neural Networks”, Energy, 117, 530-542, 2019.

12. A. Anagnostis, E. Papageorgiou, V. Dafopoulos and D. Bochtis, "Applying Long Short-Term Memory Networks for natural gas demand prediction," 2019 10th International Conference on Information, Intelligence, Systems and Applications (IISA),1-7, doi: 10.1109/IISA.2019.8900746, 2019.

13. J. Chung, C. Gulcehre, K. Cho, Y. Bengio, “Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling”, arXiv: 1412.3555, 2014.

14. S. Hochreiter, “The Vanishing Gradient Problem During Learning Recurrent Neural Nets and Problem Solutions”, International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems. 6. 107-116. 10.1142/S0218488598000094 1998.

15. J. L. Elman, "Finding Structure in Time". Cognitive Science 14, 179–211, 1990.

16. X. Glorot and Y. Bengio, “Understanding the difficulty of training deep feedforward neural networks”, Proceedings of the thirteenth international conference on artificial intelligence and statistics, 249-256, 2010.

17. D. P. Kingma and J. Ba, “Adam: A Method for Stochastic Optimization”, arXiv: 1412.6980, 2014.

18. I. Sutskever and O. VInyals, “Sequence to Sequence Learning with Neural Networks”, arXiv: 1409.3215, 2014.

19. O. Irsoy and C. Cardie, “Opinion Mining with Deep Recurrent Neural Networks”, "Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing ({EMNLP})", 720-728, doi: 10.3115/v1/D14-1080, 2014.

20. Z.-H. Zhou, J. Wu and W. Tang, "Ensembling neural networks: Many could be better than all", Artif. Intell., vol. 137, no. 1, pp. 239-263, 2002.

21. Z.-H. Zhou, J. Wu and W. Tang, "Corrigendum to ‘ensembling neural networks: Many could be better than all’ [Artificial intelligence 137 (1–2) (2002) 239–263]", Artif. Intell., vol. 174, no. 18, pp. 1570, Dec. 2010.

22. H. Li, X. Wang and S. Ding, "Research and development of neural network ensembles: A survey", Artif. Intell. Rev., vol. 49, no. 4, pp. 455-479, Apr. 2018.

23. P. M. Granitto, P. F. Verdes and H. A. Ceccatto, "Neural network ensembles: Evaluation of aggregation algorithms", Artif. Intell., vol. 163, no. 2, pp. 139-162, Apr. 2005.

24. E. Zvornicanin Differences Between Bidirectional and Unidirectional LSTM | Baeldung. [online] Available at: < https://www.baeldung.com/cs/bidirectional-vs-unidirectional-lstm> [Accessed 28 May 2022].

Παράρτημα Α

Υλοποίηση μονοεπίπεδου LSTM *(filename: gaspredictionlstm.py)*

```
import sys
import time
import os

def main(units,timestep,epochs,batch_size,data_set,export,dir_name):
  import numpy as np
  import pandas as pd
  import seaborn as sns
  from pylab import rcParams
  import matplotlib.pyplot as plt
  from matplotlib import rc
  from sklearn.model_selection import train_test_split
  from pandas.plotting import register_matplotlib_converters
  import tensorflow as tf
  from tensorflow import keras

  # Setting variable values
  Units,Timestep,Epochs,Data_set,Export =
units,timestep,epochs,data_set,export

  # Batch Size
  Batch = batch_size

  directory = dir_name

  # Doesn't change in any expirement
  Split = 0.1

  # Dropout Rate
  Rate = 0

  # Learning Rate
  Learning_Rate = 0.001

  # Setting variables for plots
```

```
  register_matplotlib_converters()
  sns.set(style='whitegrid', palette='muted', font_scale=1.5)
  rcParams['figure.figsize'] = 12, 8
  RANDOM_SEED = 42
  np.random.seed(RANDOM_SEED)
  tf.random.set_seed(RANDOM_SEED)

  # Reading Dataset and creating the panda dataframe
  df = pd.read_csv(
    Data_set,
    parse_dates=['timestamp'],
    index_col="timestamp"
  )

  # Machine Learning part
  TRAINNTEST = 0.75
  train_size = int(len(df) * TRAINNTEST)
  test_size = len(df) - train_size
  train, test = df.iloc[0:train_size], df.iloc[train_size:len(df)]

  # We select the values we want in our timeline
  f_columns = ['demand']
  f_columns.append('temp')
  f_columns.append('wind')
  f_columns.append('holiday')

  # We break our dataset to the learning set and the testing set
  def create_dataset(X, y, time_steps=1):
      Xs, ys = [], []
      for i in range(len(X) - time_steps):
          v = X.iloc[i:(i + time_steps)].values
          Xs.append(v)
          ys.append(y.iloc[i + time_steps])
      return np.array(Xs), np.array(ys)


  # reshape to [samples, time_steps, n_features]

  X_train, y_train = create_dataset(train[f_columns], train.demand,
Timestep)
  X_test, y_test = create_dataset(test[f_columns], test.demand,
Timestep)

  # We set up our RNN
  model = keras.Sequential()
  model.add(
    keras.layers.Bidirectional(
      keras.layers.LSTM(
        units=Units,
```

```
        input_shape=(X_train.shape[1], X_train.shape[2])
      )
    )
  )
  # We start the clock
  start_time = time.time()

  # We end the RNN configuration
  model.add(keras.layers.Dropout(rate=Rate))
  model.add(keras.layers.Dense(units=1))
  opt = keras.optimizers.Adam(learning_rate=Learning_Rate)

  # Learning starts
  model.compile(loss='mean_absolute_percentage_error', optimizer=opt,
metrics=['mean_squared_error','mean_absolute_percentage_error'])

  history = model.fit(
      X_train, y_train,
      epochs=Epochs,
      batch_size=Batch,
      validation_split=Split,
      shuffle=False
  )

  # Learning done


  if(Export):
    model.save_weights(directory + '/info/lstm_weights.h5')
    # Network visualization and modeliazation
    from tensorflow.keras.utils import plot_model
    plot_model(model, to_file=str(directory
+'/info/model_plot_lstm.png'), show_shapes=True, show_layer_names=True)

    model.summary()
    print(model.optimizer.get_config())


  # Accuracy metrics
  list = []
  list_val = []
  for x in history.history['mean_absolute_percentage_error']:
    list.append(100 - x)

  for x in history.history['val_mean_absolute_percentage_error']:
    list_val.append(100 - x)

  # Accuracy Graph (Using 100 - MAPE)
  plt.clf()
```

```
plt.plot(list, label='train')
plt.plot(list_val, label='test')
plt.ylabel('Success')
plt.xlabel('Epoch')
plt.legend(loc='best')
plt.savefig(directory +'/results/success.png')
plt.show()

# Error Graph (Using MSE)
plt.clf()
plt.plot(history.history['mean_squared_error'], label='train')
plt.plot(history.history['val_mean_squared_error'], label='test')
plt.ylabel('Error')
plt.xlabel('Epoch')
plt.legend()
plt.savefig(directory +'/results/error.png')
plt.show()

# Our custom testing phase
y_pred = model.predict(X_test)

# Prediction Graph
plt.clf()
plt.plot(y_test.flatten(), marker='.', label="true")
plt.plot(y_pred.flatten(), 'r', label="prediction")
plt.ylabel('Gas Demand')
plt.xlabel('Time Step')
plt.legend()
plt.savefig(directory +'/results/results.png')


# Custom deviation and success metric
import math
def deviation(prediction,real):
  temp = 0
  for i in range(len(prediction)):
    temp+= ((prediction[i] - real[i]) / real[i]) ** 2
  return math.sqrt(1/len(prediction) * temp)

f = open(directory + "/info/details.txt","w")
f.write("success:")
f.write(str((1 - deviation(y_pred,y_test)) * 100))
f.write("%")
f.write("\n")
f.write("deviation:")
f.write(str(deviation(y_pred,y_test)))
f.write("\n")
f.write("time:")
f.write(str(time.time() - start_time))
```

```
  f.write("\n")
  f.close()


  # Error and success file created
  file = open( str(directory + "/info/error_success.txt"), "w")

  # Saving the array in a text file
  file.write("train_error")
  file.write("\t")
  file.write("test_error")
  file.write("\t")
  file.write("train_success")
  file.write("\t")
  file.write("test_success")
  file.write("\n")

  for x in range(len(history.history['mean_squared_error'])):
    file.write(str(history.history['mean_squared_error'][x]))
    file.write("\t")
    file.write(str(history.history['val_mean_squared_error'][x]))
    file.write("\t")
    file.write(str(list[x]))
    file.write("\t")
    file.write(str(list_val[x]))
    file.write("\n")

  file.close()



if __name__ == "__main__":
  if(len(sys.argv) != 7):
    print("Correct usage: LSTM Units,Timestep,Epochs,Batch Size,Data
Set,Export (T/F)")
    exit()

  dir_name = str("LSTM_" +
sys.argv[1]+"_"+sys.argv[2]+"_"+sys.argv[3]+"_"+sys.argv[4])
  try:
    os.mkdir("./"+dir_name)
  except OSError as error:
    print(error)


  try:
    os.mkdir("./" + dir_name + "/info")
  except OSError as error:
    print(error)
  try:
```

```
    os.mkdir("./" + dir_name + "/results")
  except OSError as error:
    print(error)

  f = open(dir_name + "/info/topology.txt","w+")
  f.write("============================================================"
)
  f.write("\n")
  f.write("\t\tNumber of units (LSTM): ")
  f.write(sys.argv[1])
  f.write("\n")

  f.write("\t\tTimestep (Hours): ")
  f.write(sys.argv[2])
  f.write("\n")

  f.write("\t\tEpochs: ")
  f.write(sys.argv[3])
  f.write("\n")

  f.write("\t\tBatch Size: ")
  f.write(sys.argv[4])
  f.write("\n")

  f.write("\t\tData Set: ")
  f.write(sys.argv[5])
  f.write("\n")

  f.write("\t\tExport (T/F): ")
  f.write(sys.argv[6])
  f.write("\n")

  f.write("============================================================"
)
  f.write("\n")
  f.close()
  main(int(sys.argv[1]),int(sys.argv[2]),int(sys.argv[3]),int(sys.argv[
4]),sys.argv[5],bool(sys.argv[6]),dir_name)
```

Υλοποίηση πολυεπίπεδου LSTM *(filename: gaspredictionmultilstm.py)*

```
import sys
import time
import os

def main(units,timestep,epochs,batch_size,data_set,export,dir_name):
  import numpy as np
  import pandas as pd
  import seaborn as sns
  from pylab import rcParams
  import matplotlib.pyplot as plt
  from matplotlib import rc
  from sklearn.model_selection import train_test_split
  from pandas.plotting import register_matplotlib_converters
  import tensorflow as tf
  from tensorflow import keras

  # Setting variable values
  Units,Timestep,Epochs,Data_set,Export =
units,timestep,epochs,data_set,export

  # Batch Size
  Batch = batch_size

  directory = dir_name

  # Doesn't change in any expirement
  Split = 0.1

  # Dropout Rate
  Rate = 0

  # Learning Rate
  Learning_Rate = 0.001

  # Setting variables for plots
  register_matplotlib_converters()
  sns.set(style='whitegrid', palette='muted', font_scale=1.5)
  rcParams['figure.figsize'] = 12, 8
  RANDOM_SEED = 42
  np.random.seed(RANDOM_SEED)
  tf.random.set_seed(RANDOM_SEED)

  # Reading Dataset and creating the panda dataframe
  df = pd.read_csv(
    Data_set,
    parse_dates=['timestamp'],
```

```
    index_col="timestamp"
  )

  # Machine Learning part
  TRAINNTEST = 0.75
  train_size = int(len(df) * TRAINNTEST)
  test_size = len(df) - train_size
  train, test = df.iloc[0:train_size], df.iloc[train_size:len(df)]

  # We select the values we want in our timeline
  f_columns = ['demand']
  f_columns.append('wind')
  f_columns.append('temp')
  f_columns.append('holiday')

  # We break our dataset to the learning set and the testing set
  def create_dataset(X, y, time_steps=1):
      Xs, ys = [], []
      for i in range(len(X) - time_steps):
          v = X.iloc[i:(i + time_steps)].values
          Xs.append(v)
          ys.append(y.iloc[i + time_steps])
      return np.array(Xs), np.array(ys)


  # reshape to [samples, time_steps, n_features]

  X_train, y_train = create_dataset(train[f_columns], train.demand,
Timestep)
  X_test, y_test = create_dataset(test[f_columns], test.demand,
Timestep)

  # We set up our RNN
  model = keras.Sequential()
  model.add(
    keras.layers.Bidirectional(
      keras.layers.LSTM(
        units=Units,
        input_shape=(X_train.shape[1], X_train.shape[2]),
        return_sequences=True
      )
    )
  )
  model.add(
    keras.layers.Bidirectional(
      keras.layers.LSTM(
        units=Units,
        input_shape=(X_train.shape[1], X_train.shape[2]),
      )
```

```
    )
  )
  # We start the clock
  start_time = time.time()

  # We end the RNN configuration
  model.add(keras.layers.Dropout(rate=Rate))
  model.add(keras.layers.Dense(units=1))
  opt = keras.optimizers.Adam(learning_rate=Learning_Rate)

  # Learning starts
  model.compile(loss='mean_absolute_percentage_error', optimizer=opt,
metrics=['mean_squared_error','mean_absolute_percentage_error'])

  history = model.fit(
      X_train, y_train,
      epochs=Epochs,
      batch_size=Batch,
      validation_split=Split,
      shuffle=False
  )

  # Learning done


  if(Export):
    model.save_weights(directory + '/info/lstm_weights.h5')
    # Network visualization and modeliazation
    from tensorflow.keras.utils import plot_model
    plot_model(model, to_file=str(directory
+'/info/model_plot_lstm.png'), show_shapes=True, show_layer_names=True)

    model.summary()
    print(model.optimizer.get_config())


  # Accuracy metrics
  list = []
  list_val = []
  for x in history.history['mean_absolute_percentage_error']:
    list.append(100 - x)

  for x in history.history['val_mean_absolute_percentage_error']:
    list_val.append(100 - x)

  # Accuracy Graph (Using 100 - MAPE)
  plt.clf()
  plt.plot(list, label='train')
  plt.plot(list_val, label='test')
```

```
plt.ylabel('Success')
plt.xlabel('Epoch')
plt.legend(loc='best')
plt.savefig(directory +'/results/success.png')
plt.show()

# Error Graph (Using MSE)
plt.clf()
plt.plot(history.history['mean_squared_error'], label='train')
plt.plot(history.history['val_mean_squared_error'], label='test')
plt.ylabel('Error')
plt.xlabel('Epoch')
plt.legend()
plt.savefig(directory +'/results/error.png')
plt.show()

# Our custom testing phase
y_pred = model.predict(X_test)

# Prediction Graph
plt.clf()
plt.plot(y_test.flatten(), marker='.', label="true")
plt.plot(y_pred.flatten(), 'r', label="prediction")
plt.ylabel('Gas Demand')
plt.xlabel('Time Step')
plt.legend()
plt.savefig(directory +'/results/results.png')


# Custom deviation and success metric
import math
def deviation(prediction,real):
  temp = 0
  for i in range(len(prediction)):
    temp+= ((prediction[i] - real[i]) / real[i]) ** 2
  return math.sqrt(1/len(prediction) * temp)

f = open(directory + "/info/details.txt","w")
f.write("success:")
f.write(str((1 - deviation(y_pred,y_test)) * 100))
f.write("%")
f.write("\n")
f.write("deviation:")
f.write(str(deviation(y_pred,y_test)))
f.write("\n")
f.write("time:")
f.write(str(time.time() - start_time))
f.write("\n")
f.close()
```

```
  # Error and success file created
  file = open( str(directory + "/info/error_success.txt"), "w")

  # Saving the array in a text file
  file.write("train_error")
  file.write("\t")
  file.write("test_error")
  file.write("\t")
  file.write("train_success")
  file.write("\t")
  file.write("test_success")
  file.write("\n")

  for x in range(len(history.history['mean_squared_error'])):
    file.write(str(history.history['mean_squared_error'][x]))
    file.write("\t")
    file.write(str(history.history['val_mean_squared_error'][x]))
    file.write("\t")
    file.write(str(list[x]))
    file.write("\t")
    file.write(str(list_val[x]))
    file.write("\n")

  file.close()



if __name__ == "__main__":
  if(len(sys.argv) != 7):
    print("Correct usage: LSTM Units,Timestep,Epochs,Batch Size,Data
Set,Export (T/F)")
    exit()

  dir_name = str("LSTM_" +
sys.argv[1]+"_"+sys.argv[2]+"_"+sys.argv[3]+"_"+sys.argv[4])
  try:
    os.mkdir("./"+dir_name)
  except OSError as error:
    print(error)


  try:
    os.mkdir("./" + dir_name + "/info")
  except OSError as error:
    print(error)
  try:
    os.mkdir("./" + dir_name + "/results")
  except OSError as error:
```

```
    print(error)

  f = open(dir_name + "/info/topology.txt","w+")
  f.write("============================================================"
)
  f.write("\n")
  f.write("\t\tNumber of units (LSTM): ")
  f.write(sys.argv[1])
  f.write("\n")

  f.write("\t\tTimestep (Hours): ")
  f.write(sys.argv[2])
  f.write("\n")

  f.write("\t\tEpochs: ")
  f.write(sys.argv[3])
  f.write("\n")

  f.write("\t\tBatch Size: ")
  f.write(sys.argv[4])
  f.write("\n")

  f.write("\t\tData Set: ")
  f.write(sys.argv[5])
  f.write("\n")

  f.write("\t\tExport (T/F): ")
  f.write(sys.argv[6])
  f.write("\n")

  f.write("============================================================"
)
  f.write("\n")
  f.close()
  main(int(sys.argv[1]),int(sys.argv[2]),int(sys.argv[3]),int(sys.argv[
4]),sys.argv[5],bool(sys.argv[6]),dir_name)
```

Παράρτημα Β

Υλοποίηση μονοεπίπεδου GRU *(filename: gaspredictiongru.py)*

```
import sys
import time
import os

def main(units,timestep,epochs,batch_size,data_set,export,dir_name):
  import numpy as np
  import pandas as pd
  import seaborn as sns
  from pylab import rcParams
  import matplotlib.pyplot as plt
  from matplotlib import rc
  from sklearn.model_selection import train_test_split
  from pandas.plotting import register_matplotlib_converters
  import tensorflow as tf
  from tensorflow import keras

  # Setting variable values
  Units,Timestep,Epochs,Data_set,Export =
units,timestep,epochs,data_set,export

  # Batch Size
  Batch = batch_size

  directory = dir_name

  # Doesn't change in any expirement
  Split = 0.1

  # Dropout Rate
  Rate = 0.1

  # Learning Rate
  Learning_Rate = 0.001
```

```
  # Setting variables for plots
  register_matplotlib_converters()
  sns.set(style='whitegrid', palette='muted', font_scale=1.5)
  rcParams['figure.figsize'] = 12, 8
  RANDOM_SEED = 42
  np.random.seed(RANDOM_SEED)
  tf.random.set_seed(RANDOM_SEED)

  # Reading Dataset and creating the panda dataframe
  df = pd.read_csv(
    Data_set,
    parse_dates=['timestamp'],
    index_col="timestamp"
  )

  # Machine Learning part
  TRAINNTEST = 0.75
  train_size = int(len(df) * TRAINNTEST)
  test_size = len(df) - train_size
  train, test = df.iloc[0:train_size], df.iloc[train_size:len(df)]

  # We select the values we want in our timeline
  f_columns = ['demand']
  f_columns.append('temp')
  f_columns.append('wind')
  f_columns.append('holiday')

  # We break our dataset to the learning set and the testing set
  def create_dataset(X, y, time_steps=1):
      Xs, ys = [], []
      for i in range(len(X) - time_steps):
          v = X.iloc[i:(i + time_steps)].values
          Xs.append(v)
          ys.append(y.iloc[i + time_steps])
      return np.array(Xs), np.array(ys)


  # reshape to [samples, time_steps, n_features]

  X_train, y_train = create_dataset(train[f_columns], train.demand,
Timestep)
  X_test, y_test = create_dataset(test[f_columns], test.demand,
Timestep)

  # We set up our RNN
  model = keras.Sequential()
  model.add(
    keras.layers.Bidirectional(
      keras.layers.GRU(
```

```
        units=Units,
        input_shape=(X_train.shape[1], X_train.shape[2])
      )
    )
  )
  # We start the clock
  start_time = time.time()

  # We end the RNN configuration
  model.add(keras.layers.Dropout(rate=Rate))
  model.add(keras.layers.Dense(units=1))
  opt = keras.optimizers.Adam(learning_rate=Learning_Rate)

  # Learning starts
  model.compile(loss='mean_absolute_percentage_error', optimizer=opt,
metrics=['mean_squared_error','mean_absolute_percentage_error'])

  history = model.fit(
      X_train, y_train,
      epochs=Epochs,
      batch_size=Batch,
      validation_split=Split,
      shuffle=False
  )

  # Learning done


  if(Export):
    model.save_weights(directory + '/info/gru_weights.h5')
    # Network visualization and modeliazation
    from tensorflow.keras.utils import plot_model
    plot_model(model, to_file=str(directory
+'/info/model_plot_gru.png'), show_shapes=True, show_layer_names=True)

    model.summary()
    print(model.optimizer.get_config())


  # Accuracy metrics
  list = []
  list_val = []
  for x in history.history['mean_absolute_percentage_error']:
    list.append(100 - x)

  for x in history.history['val_mean_absolute_percentage_error']:
    list_val.append(100 - x)

  # Accuracy Graph (Using 100 - MAPE)
```

```
plt.clf()
plt.plot(list, label='train')
plt.plot(list_val, label='test')
plt.ylabel('Success')
plt.xlabel('Epoch')
plt.legend(loc='best')
plt.savefig(directory +'/results/success.png')
plt.show()

# Error Graph (Using MSE)
plt.clf()
plt.plot(history.history['mean_squared_error'], label='train')
plt.plot(history.history['val_mean_squared_error'], label='test')
plt.ylabel('Error')
plt.xlabel('Epoch')
plt.legend()
plt.savefig(directory +'/results/error.png')
plt.show()

# Our custom testing phase
y_pred = model.predict(X_test)

# Prediction Graph
plt.clf()
plt.plot(y_test.flatten(), marker='.', label="true")
plt.plot(y_pred.flatten(), 'r', label="prediction")
plt.ylabel('Gas Demand')
plt.xlabel('Time Step')
plt.legend()
plt.savefig(directory +'/results/results.png')


# Custom deviation and success metric
import math
def deviation(prediction,real):
  temp = 0
  for i in range(len(prediction)):
    temp+= ((prediction[i] - real[i]) / real[i]) ** 2
  return math.sqrt(1/len(prediction) * temp)

f = open(directory + "/info/details.txt","w")
f.write("success:")
f.write(str((1 - deviation(y_pred,y_test)) * 100))
f.write("%")
f.write("\n")
f.write("deviation:")
f.write(str(deviation(y_pred,y_test)))
f.write("\n")
f.write("time:")
```

```
  f.write(str(time.time() - start_time))
  f.write("\n")
  f.close()


  # Error and success file created
  file = open( str(directory + "/info/error_success.txt"), "w")

  # Saving the array in a text file
  file.write("train_error")
  file.write("\t")
  file.write("test_error")
  file.write("\t")
  file.write("train_success")
  file.write("\t")
  file.write("test_success")
  file.write("\n")

  for x in range(len(history.history['mean_squared_error'])):
    file.write(str(history.history['mean_squared_error'][x]))
    file.write("\t")
    file.write(str(history.history['val_mean_squared_error'][x]))
    file.write("\t")
    file.write(str(list[x]))
    file.write("\t")
    file.write(str(list_val[x]))
    file.write("\n")

  file.close()



if __name__ == "__main__":
  if(len(sys.argv) != 7):
    print("Correct usage: GRU Units,Timestep,Epochs,Batch Size,Data
Set,Export (T/F)")
    exit()

  dir_name = str("GRU_" +
sys.argv[1]+"_"+sys.argv[2]+"_"+sys.argv[3]+"_"+sys.argv[4])
  try:
    os.mkdir("./"+dir_name)
  except OSError as error:
    print(error)


  try:
    os.mkdir("./" + dir_name + "/info")
  except OSError as error:
    print(error)
```

```
  try:
    os.mkdir("./" + dir_name + "/results")
  except OSError as error:
    print(error)

  f = open(dir_name + "/info/topology.txt","w+")
  f.write("============================================================"
)
  f.write("\n")
  f.write("\t\tNumber of units (GRU): ")
  f.write(sys.argv[1])
  f.write("\n")

  f.write("\t\tTimestep (Hours): ")
  f.write(sys.argv[2])
  f.write("\n")

  f.write("\t\tEpochs: ")
  f.write(sys.argv[3])
  f.write("\n")

  f.write("\t\tBatch Size: ")
  f.write(sys.argv[4])
  f.write("\n")

  f.write("\t\tData Set: ")
  f.write(sys.argv[5])
  f.write("\n")

  f.write("\t\tExport (T/F): ")
  f.write(sys.argv[6])
  f.write("\n")

  f.write("============================================================"
)
  f.write("\n")
  f.close()
  main(int(sys.argv[1]),int(sys.argv[2]),int(sys.argv[3]),int(sys.argv[
4]),sys.argv[5],bool(sys.argv[6]),dir_name)
```

Υλοποίηση πολυεπίπεδου GRU *(filename: gaspredictionmultigru.py)*

```
import sys
import time
import os

def main(units,timestep,epochs,batch_size,data_set,export,dir_name):
  import numpy as np
  import pandas as pd
  import seaborn as sns
  from pylab import rcParams
  import matplotlib.pyplot as plt
  from matplotlib import rc
  from sklearn.model_selection import train_test_split
  from pandas.plotting import register_matplotlib_converters
  import tensorflow as tf
  from tensorflow import keras

  # Setting variable values
  Units,Timestep,Epochs,Data_set,Export =
units,timestep,epochs,data_set,export

  # Batch Size
  Batch = batch_size

  directory = dir_name

  # Doesn't change in any expirement
  Split = 0.1

  # Dropout Rate
  Rate = 0

  # Learning Rate
  Learning_Rate = 0.001

  # Setting variables for plots
  register_matplotlib_converters()
  sns.set(style='whitegrid', palette='muted', font_scale=1.5)
  rcParams['figure.figsize'] = 12, 8
  RANDOM_SEED = 42
  np.random.seed(RANDOM_SEED)
  tf.random.set_seed(RANDOM_SEED)

  # Reading Dataset and creating the panda dataframe
  df = pd.read_csv(
    Data_set,
    parse_dates=['timestamp'],
    index_col="timestamp"
```

```
  )

  # Machine Learning part
  TRAINNTEST = 0.75
  train_size = int(len(df) * TRAINNTEST)
  test_size = len(df) - train_size
  train, test = df.iloc[0:train_size], df.iloc[train_size:len(df)]

  # We select the values we want in our timeline
  f_columns = ['demand']
  f_columns.append('wind')
  f_columns.append('temp')
  f_columns.append('holiday')

  # We break our dataset to the learning set and the testing set
  def create_dataset(X, y, time_steps=1):
      Xs, ys = [], []
      for i in range(len(X) - time_steps):
          v = X.iloc[i:(i + time_steps)].values
          Xs.append(v)
          ys.append(y.iloc[i + time_steps])
      return np.array(Xs), np.array(ys)


  # reshape to [samples, time_steps, n_features]

  X_train, y_train = create_dataset(train[f_columns], train.demand,
Timestep)
  X_test, y_test = create_dataset(test[f_columns], test.demand,
Timestep)

  # We set up our RNN
  model = keras.Sequential()
  model.add(
    keras.layers.Bidirectional(
      keras.layers.GRU(
        units=Units,
        input_shape=(X_train.shape[1], X_train.shape[2]),
        return_sequences=True
      )
    )
  )
  model.add(
    keras.layers.Bidirectional(
      keras.layers.GRU(
        units=Units,
        input_shape=(X_train.shape[1], X_train.shape[2]),
      )
    )
```

```
  )
  # We start the clock
  start_time = time.time()

  # We end the RNN configuration
  model.add(keras.layers.Dropout(rate=Rate))
  model.add(keras.layers.Dense(units=1))
  opt = keras.optimizers.Adam(learning_rate=Learning_Rate)

  # Learning starts
  model.compile(loss='mean_absolute_percentage_error', optimizer=opt,
metrics=['mean_squared_error','mean_absolute_percentage_error'])

  history = model.fit(
      X_train, y_train,
      epochs=Epochs,
      batch_size=Batch,
      validation_split=Split,
      shuffle=False
  )

  # Learning done


  if(Export):
    model.save_weights(directory + '/info/gru_weights.h5')
    # Network visualization and modeliazation
    from tensorflow.keras.utils import plot_model
    plot_model(model, to_file=str(directory
+'/info/model_plot_gru.png'), show_shapes=True, show_layer_names=True)

    model.summary()
    print(model.optimizer.get_config())


  # Accuracy metrics
  list = []
  list_val = []
  for x in history.history['mean_absolute_percentage_error']:
    list.append(100 - x)

  for x in history.history['val_mean_absolute_percentage_error']:
    list_val.append(100 - x)

  # Accuracy Graph (Using 100 - MAPE)
  plt.clf()
  plt.plot(list, label='train')
  plt.plot(list_val, label='test')
  plt.ylabel('Success')
```

```
plt.xlabel('Epoch')
plt.legend(loc='best')
plt.savefig(directory +'/results/success.png')
plt.show()

# Error Graph (Using MSE)
plt.clf()
plt.plot(history.history['mean_squared_error'], label='train')
plt.plot(history.history['val_mean_squared_error'], label='test')
plt.ylabel('Error')
plt.xlabel('Epoch')
plt.legend()
plt.savefig(directory +'/results/error.png')
plt.show()

# Our custom testing phase
y_pred = model.predict(X_test)

# Prediction Graph
plt.clf()
plt.plot(y_test.flatten(), marker='.', label="true")
plt.plot(y_pred.flatten(), 'r', label="prediction")
plt.ylabel('Gas Demand')
plt.xlabel('Time Step')
plt.legend()
plt.savefig(directory +'/results/results.png')


# Custom deviation and success metric
import math
def deviation(prediction,real):
  temp = 0
  for i in range(len(prediction)):
    temp+= ((prediction[i] - real[i]) / real[i]) ** 2
  return math.sqrt(1/len(prediction) * temp)

f = open(directory + "/info/details.txt","w")
f.write("success:")
f.write(str((1 - deviation(y_pred,y_test)) * 100))
f.write("%")
f.write("\n")
f.write("deviation:")
f.write(str(deviation(y_pred,y_test)))
f.write("\n")
f.write("time:")
f.write(str(time.time() - start_time))
f.write("\n")
f.close()
```

```
  # Error and success file created
  file = open( str(directory + "/info/error_success.txt"), "w")

  # Saving the array in a text file
  file.write("train_error")
  file.write("\t")
  file.write("test_error")
  file.write("\t")
  file.write("train_success")
  file.write("\t")
  file.write("test_success")
  file.write("\n")

  for x in range(len(history.history['mean_squared_error'])):
    file.write(str(history.history['mean_squared_error'][x]))
    file.write("\t")
    file.write(str(history.history['val_mean_squared_error'][x]))
    file.write("\t")
    file.write(str(list[x]))
    file.write("\t")
    file.write(str(list_val[x]))
    file.write("\n")

  file.close()



if __name__ == "__main__":
  if(len(sys.argv) != 7):
    print("Correct usage: GRU Units,Timestep,Epochs,Batch Size,Data
Set,Export (T/F)")
    exit()

  dir_name = str("GRU_" +
sys.argv[1]+"_"+sys.argv[2]+"_"+sys.argv[3]+"_"+sys.argv[4])
  try:
    os.mkdir("./"+dir_name)
  except OSError as error:
    print(error)


  try:
    os.mkdir("./" + dir_name + "/info")
  except OSError as error:
    print(error)
  try:
    os.mkdir("./" + dir_name + "/results")
  except OSError as error:
    print(error)
```

```
  f = open(dir_name + "/info/topology.txt","w+")
  f.write("============================================================"
)
  f.write("\n")
  f.write("\t\tNumber of units (GRU): ")
  f.write(sys.argv[1])
  f.write("\n")

  f.write("\t\tTimestep (Hours): ")
  f.write(sys.argv[2])
  f.write("\n")

  f.write("\t\tEpochs: ")
  f.write(sys.argv[3])
  f.write("\n")

  f.write("\t\tBatch Size: ")
  f.write(sys.argv[4])
  f.write("\n")

  f.write("\t\tData Set: ")
  f.write(sys.argv[5])
  f.write("\n")

  f.write("\t\tExport (T/F): ")
  f.write(sys.argv[6])
  f.write("\n")

  f.write("============================================================"
)
  f.write("\n")
  f.close()
  main(int(sys.argv[1]),int(sys.argv[2]),int(sys.argv[3]),int(sys.argv[
4]),sys.argv[5],bool(sys.argv[6]),dir_name)
```

Παράρτημα Γ

Εγκατάσταση σε περιβάλλον Unix *(filename: installMeUnix.sh)*

```
# !/bin/bash

while read p; do
  pip3 install "$p"
done <modules
```

Εγκατάσταση σε περιβάλλον Windows *(filename: installMeWindows.ps1)*

```
foreach($line in [System.IO.File]::ReadLines("modules"))
{
        pip install $line
}
```

Module list *(filename: modules)*

tk
numpy
matplotlib
pandas
keras
sklearn
tensorflow
seaborn