ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ, ΗΛΕΚΤΡΟΝΙΚΗΣ
ΚΑΙ ΣΥΣΤΗΜΑΤΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ

Αυτοματοποίηση της Εκπαίδευσης Νευρωνικών Δικτύων μέσω Εξελικτικών Αλγορίθμων

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Στυλιανός Τσανάκας

Επιβλέπουσα : Θεοδώρα Βαρβαρίγου

Καθηγήτρια Ε.Μ.Π.

Αθήνα, Απρίλιος 2021

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ, ΗΛΕΚΤΡΟΝΙΚΗΣ
ΚΑΙ ΣΥΣΤΗΜΑΤΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ

Αυτοματοποίηση της Εκπαίδευσης Νευρωνικών Δικτύων μέσω Εξελικτικών Αλγορίθμων

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Στυλιανός Τσανάκας

Επιβλέπουσα : Θεοδώρα Βαρβαρίγου

Καθηγήτρια Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 5η Απριλίου 2021.

.....	.....	.....
Θ. Βαρβαρίγου	Σ. Παπαβασιλείου	Δ. Ασκούνης
Καθ. Ε.Μ.Π	Καθ. Ε.Μ.Π.	Καθ. Ε.Μ.Π.

Αθήνα, Απρίλιος 2021

.....
Στυλιανός Τσανάκας

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Περίληψη

Στην παρούσα διπλωματική παρουσιάζεται το μοντέλο των νευρωνικών δικτύων, οι σχεδιαστικές του επιλογές, οι μέθοδοι εκπαίδευσης καθώς και οι ευριστικές μέθοδοι που προσεγγίζουν με αυτοματοποιημένο τρόπο τις βέλτιστες τοπολογίες νευρωνικών δικτύων. Η διπλωματική εργασία περιλαμβάνει θεωρητικό και πρακτικό μέρος. Στο θεωρητικό μέρος αναλύονται τα δομικά χαρακτηριστικά των νευρωνικών δικτύων, των συναρτήσεων και τον αλγορίθμων. Το πρακτικό μέρος περιλαμβάνει την υλοποίηση στην γλώσσα προγραμματισμού python3 και βιβλιοθήκες μηχανικής μάθησης και βαθιάς μάθησης. Μέρος του κώδικα βρίσκεται στο τέλος της διπλωματικής στο παράρτημα.

Η παρούσα διπλωματική πραγματοποιήθηκε στα πλαίσια δραστηριοτήτων του ερευνητικού έργου Accordion που έχει λάβει χρηματοδότηση από το πρόγραμμα έρευνας και καινοτομίας «Ορίζοντας 2020» της Ευρωπαϊκής Ένωσης βάσει συμφωνίας επιχορήγησης No 871793”. Μέρος της διπλωματικής έχει δημοσιευθεί σε πέντε επιστημονικά συνέδρια με κριτές. Οι τίτλοι των καινοτόμων δημοσιεύσεων είναι (α) Πρόβλεψη κατανομής επισκεπτών για μεγάλες εκδηλώσεις σε έξυπνες πόλεις, (β) Πρόβλεψη επόμενης θέσης για πλοία που χρησιμοποιούν LSTM Νευρωνικά δίκτυα και δεδομένα τροχιάς, (γ) Χρησιμοποιώντας τα νευρωνικά δίκτυα LSTM ως προγνωστικούς παράγοντες αξιοποίησης πόρων: Η περίπτωση της εκπαίδευσης μοντέλων βαθιάς μάθησης σε υπολογιστική των άκρων, (δ) Πρόβλεψη χρήσης πόρων σε υπολογιστικές υποδομές των άκρων με CNN και ένα μοντέλο υβριδικής Μπευζιανής βελτιστοποίησης υπερπαραμέτρων με την μέθοδο σμήνους σωματιδίων και (ε) Υπερτονισμός GRU νευρωνικών δικτύων με την υβριδική Μπευζιανή-εξελικτική στρατηγική για πρόβλεψη χρήσης πόρων στην υπολογιστική των άκρων.

Λέξεις Κλειδιά

Νευρωνικά Δίκτυα, Βαθιά Μάθηση, Μέθοδοι Βελτιστοποίησης, Ανατροφοδοτούμενες Μονάδες με Πύλη, Εξελικτική Στρατηγική, Γενετικοί Αλγόριθμοι, Μπεϋζιανή Βελτιστοποίηση, Υπερτονισμός, Πρόβλεψη τροχιάς, Υπολογιστική Πόροι στα Άκρα

Abstract

In this dissertation is presented the model of neural networks, its design options, training methods as well as heuristic methods that approach in an automated way the optimal neural network topologies. The dissertation includes theoretical and practical parts. The theoretical part analyzes the structural characteristics of neural networks, functions and algorithms. The practical part includes the implementation in the python3 programming language and libraries of machine learning and deep learning. Part of the code is at the end of the diploma in the appendix.

This diploma was carried out in the framework of the activities of the research project Accordion, which has received funding from the research and innovation program "Horizon 2020" of the European Union under a grant agreement No. 871793 ". Part of the dissertation has been published in five peer-reviewed scientific conferences. The titles of the innovative publications are: (a) Predicting visitor distribution for large events in smart cities, (b) Next position prediction for vessels using LSTM neural networks and trajectory data, (c) Using LSTM neural networks as resource utilization predictors: The case of training deep learning models on the edge, (d) Predicting resource usage in edge computing infrastructures with CNN and a hybrid Bayesian particle swarm hyper-parameter optimization model, and (e) Hypertuning GRU neural networks with a hybrid Bayesian-evolutionary strategy for edge resource usage prediction

Keywords

Neural Networks, Deep Learning, Optimization Methods, Gated Recurrent Units, Evolutionary Strategy, Genetic Algorithms, Bayesian Optimisation, Hypertuning, Trajectory prediction, Edge Resources

Ευχαριστίες

Ολοκληρώνοντας τις προπτυχιακές σπουδές μου στο Εθνικό Μετσόβιο Πολυτεχνείο, θα ήθελα να εκφράσω τις ευχαριστίες μου προς την κ. Βαρβαρίγου για τη δυνατότητα που μου έδωσε να εκπονήσω τη διπλωματική μου εργασία σε ένα ιδιαίτερα ενδιαφέρον και σημαντικό τομέα, όπως είναι αυτός της βαθιάς μηχανικής μάθησης. Ένα τεράστιο ευχαριστώ οφείλω στον δρ Γιάννη Βιόλο για την στήριξη, την έμπνευση και την καθοδήγηση των προσπαθειών μου κατά τη διάρκεια εκπόνησής της εργασίας αυτής. Θα ήθελα ακόμη να ευχαριστήσω τους φίλους και συναδέλφους Τίτα Παγουλάτου και Θεόδωρο Θεοδωρόπουλο για την συνεργασία μας, μέσα από την οποία εξελίχθηκαν οι γνώσεις μου πάνω στο αντικείμενο της μηχανικής μάθησης. Επιπροσθέτως, θα ήθελα να ευχαριστήσω τους καθηγητές κ. Ασκούνη και κ. Παπαβασιλείου που ήταν παρόντες στην παρουσίαση της διπλωματικής μου εργασίας και συνέθεσαν την τριμελή επιτροπή.

Κλείνοντας, θέλω να ευχαριστήσω ιδιαιτέρως τους γονείς μου και το οικογενειακό μου περιβάλλον για την ουσιαστική και αμέριστη στήριξη που μου παρείχαν καθ'όλη την διάρκεια των σπουδών μου.

Στυλιανός Τσανάκας,
Αθήνα, Απρίλιος 2021

Περιεχόμενα

Ευρετήριο Σχημάτων

Ευρετήριο Πινάκων

1. Εισαγωγή

Ο μεγάλος όγκος δεδομένων που χαρακτηρίζει την εποχή μας είναι μια πολύτιμη πηγή πληροφορίας στο πλαίσιο πολλών εφαρμογών. Η βαθιά μάθηση [1] παρέχει χρήσιμες μεθόδους για τη μοντελοποίηση και την εξαγωγή γνώσεων από τα δεδομένα. Στην παρούσα διπλωματική εργασία θα παρουσιάσουμε το μοντέλο των νευρωνικών δικτύων, τις συναρτήσεις ενεργοποίησης του διάφορους τύπους επιπέδων, τους αλγορίθμους εκπαίδευσης των νευρωνικών δικτύων, την κανονικοποίηση. Θα παρουσιάσαμε το Tensorflow [2] που είναι η πιο διαδεδομένη βιβλιοθήκη για βαθιά μάθηση. Θα συζητήσουμε το πρόβλημα εύρεσης των βέλτιστων υπερπαραμέτρων του νευρωνικού δικτύου και δύο μεθόδους για την ευρεση των υπερπαραμέτρων. Η πρώτη είναι μια ευρέως διαδεδομένη μέθοδος με γενετικούς αλγορίθμους. Η δεύτερη είναι μια καινοτόμος υβριδική μέθοδος που συνδυάζει την εξελικτική στρατηγική με την μπεϋζιανή βελτιστοποίηση. Έπειτα θα δούμε δύο πειραματικές αξιολογήσεις.

Στην πρώτη πειραματική αξιολόγηση προτείνουμε τη χρήση τεχνητών νευρωνικών δικτύων (ANN) με στρώματα LSTM για την επόμενη θέση πρόβλεψης κινούμενων αντικειμένων χρησιμοποιώντας έναν γενετικό αλγόριθμο και μια μέθοδο μεταφοράς γνώσης. Ο γενετικός αλγόριθμος κάνει μια έξυπνη αναζήτηση στον χώρο των τοπολογιών για να εκτιμήσει μια κοντά στην βέλτιστη αρχιτεκτονική ANN. Έπειτα το μοντέλο θα αξιοποιήσει μια μέθοδο μεταφοράς γνώσης για να εκμεταλλευτεί ένα αποθετήριο καλά εκπαιδευμένων ANN μοντέλων προκειμένου να επιταχύνει τη διαδικασία εκπαίδευσης. Αξιολογήσαμε το προτεινόμενο μοντέλο μας με πραγματικά δεδομένα από την πορεία διαφόρων πλειών και κάναμε την σύγκριση με μια μέθοδο πρόβλεψης τροχιάς με δύο γνωστά AutoML μοντέλα. Τα πειραματικά αποτελέσματα έδειξαν ότι το προτεινόμενο μοντέλο έχει καλύτερη ακρίβεια από άλλα μοντέλα και η διαδικασία εκμάθησης μεταφοράς μειώνει δραματικά τον χρόνο εκπαίδευσης. Τα αποτελέσματα μας ενθαρρύνουν για την εφαρμογή της μεθόδου μας.

Στην δεύτερη πειραματική αξιολόγηση ασχολούμαστε με το πως η πρόβλεψη χρήσης πόρων μέσω ANN μπορεί να προσφέρει μια σημαντική πηγή πληροφορίας για μια προσαρμοστική κατανομή πόρων, την εκφόρτωση εργασιών και μια προληπτική ανοχή σφαλμάτων σε μια υπολογιστική υποδομή των άκρων. Συγκεκριμένα σε αυτή την διπλωματική καταλήξαμε ότι ένα μοντέλο πολλαπλών εξόδων Gated Recurrent Neural Network ενδείκνυνται στην πρόβλεψη χρήσης πόρων. Επίσης, προτείνουμε τον καινοτόμο αλγόριθμο Hybrid Bayesian Evolutionary Strategy (HBES) για την ακριβή προσαρμογή των μοντέλων χρήσης πόρων με έναν αυτόματο τρόπο για να εξασφαλίσουμε την γενικότητα των λύσεων. Οι προτεινόμενος μηχανισμός πρόβλεψης με ANN έχουν αξιολογηθεί πειραματικά και συγκριθεί με άλλες μεθόδους διαθέσιμες στην βιβλιογραφία με σημαντικές βελτιώσεις στην ακρίβεια προβλέψεων.

Δομή Διπλωματικής Εργασίας

Στο 1ο Κεφάλαιο, γίνεται εισαγωγή του θέματος, επεξηγείται η δομή της διπλωματική εργασίας, αναφέρεται το επιστημονικό πεδίο που ανήκει καθώς και οι δύο πειραματικές αξιολογήσεις που υλοποιήθηκαν. Στο 2ο Κεφάλαιο, περιγράφονται τα νευρωνικά δίκτυα και η βαθιά μάθηση. Στο 3ο Κεφάλαιο, περιγράφεται η βιβλιοθήκη βαθιάς μάθησης TensorFlow που χρησιμοποιήθηκε για την προγραμματιστική υλοποίηση. Στο 4ο Κεφάλαιο, αναλύεται ο γενετικός αλγόριθμος βελτιστοποίησης για την εύρεση των υπερπαραμέτρων σε νευρωνικά δίκτυα. Στο 5ο κεφάλαιο προτείνεται και αναλύετε η καινοτόμος μέθοδος εύρεσης υπερπαραμέτρων που συνδυάζει την εξελικτική στρατηγική με την Μπεϋζιανή βελτιστοποίηση. Στο 6ο Κεφάλαιο αξιολογείται το προτεινόμενο μοντέλο και περιγράφονται τα πειραματικά αποτελέσματα στην περίπτωση πρόβλεψης της τροχαίας πλοίων. Στο 6ο Κεφάλαιο αξιολογείται το προτεινόμενο μοντέλο και περιγράφονται τα πειραματικά αποτελέσματα

στην περίπτωση πρόβλεψης χρησης πόρων σε υποδομές των άκρων. Στο κεφάλαιο 8 ολοκληρώνεται η διπλωματική αναφέροντας συμπεράσματα και θέματα που μπορούν να μελετηθούν στο μέλλον.

2. Νευρωνικά Δίκτυα και Βαθιά Μάθηση

Τα τεχνητά νευρωνικά δίκτυα Artificial Neural Networks (ANNs) [1] είναι δίκτυα απλών στοιχείων επεξεργασίας, που ονομάζονται νευρώνες. Οι νευρώνες λειτουργούν πάνω σε δεδομένα και συνδέονται αναμεταξύ τους. Οι ερευνητές των ANN εμπνεύστηκαν από τη δομή των πραγματικών νευρώνων του εγκεφάλου, αλλά τα στοιχεία επεξεργασίας και οι αρχιτεκτονικές που χρησιμοποιούνται στα ANN απέχουν πολύ από τη βιολογική τους έμπνευση.
Υπάρχουν πολλοί τύποι νευρωνικών δικτύων, αλλά οι βασικές αρχές τους είναι πολύ παρόμοιες. Κάθε νευρώνας στο δίκτυο μπορεί να λαμβάνει σήματα εισόδου, να τα επεξεργάζεται και να στέλνει σήματα εξόδου. Κάθε νευρώνας συνδέεται τουλάχιστον με έναν νευρώνα και κάθε σύνδεση βαθμονομείται από έναν πραγματικό αριθμό, που ονομάζεται συντελεστής βάρους. Ο συντελεστής βάρους αντικατοπτρίζει τον βαθμό σπουδαιότητας της δεδομένης σύνδεσης στο νευρωνικό δίκτυο.
Ένα νευρωνικό δίκτυο μπορεί να λειτουργήσει ως μία συναρτησιακή προσέγγιση (function approximator), δηλαδή μπορεί να πραγματοποιήσει μια απεικόνιση ενός διανύσματος σε έναν διανυσματικό χώρο σε ένα διάνυσμα του ίδιου ή ενός άλλου διανυσματικού χώρου. Τα νευρωνικά δίκτυα είναι σε θέση να χρησιμοποιήσουν άδηλη πληροφορία που βρίσκεται στα δεδομένα. Η διαδικασία «σύλληψης» και μοντελοποίησης της άγνωστης πληροφορίας ονομάζεται «εκμάθηση νευρωνικού δικτύου» ή «εκπαίδευση νευρωνικού δικτύου».
Ακολουθώντας έναν μαθηματικό φορμαλισμό, η διαδικασία της μάθησης σημαίνει να γίνει μια προσαρμογή των συντελεστών βάρους του ANN με τέτοιο τρόπο ώστε να πληρούνται ορισμένες προϋποθέσεις. Υπάρχουν δύο βασικοί τύποι εκπαιδευτικής διαδικασίας: εποπτευόμενη (supervised) και μη εποπτευόμενη (unsupervised) εκπαίδευση. Στην εποπτευόμενη εκπαίδευση (π.χ. με ένα νευρωνικό δίκτυο πολλαπλών επιπέδων feed-forward (MLF)) το νευρωνικό δίκτυο γνωρίζει την επιθυμητή έξοδο και η προσαρμογή των συντελεστών βάρους γίνεται με τέτοιο τρόπο, ώστε οι υπολογισμένες και οι επιθυμητές έξοδοι να είναι όσο το δυνατόν πιο κοντά. Στην μη επιτηρούμενη εκπαίδευση, όπως στην συσταδοποίηση [3] η επιθυμητή έξοδος δεν είναι γνωστή, το σύστημα βασίζεται σε μια σειρά από μοτίβα και στη συνέχεια συγκλίνει (ή όχι) σε μία σταθερή κατάσταση έπειτα από έναν αριθμό επαναλήψεων.

2.1. Συναρτήσεις Ενεργοποίησης

Οι συναρτήσεις ενεργοποίησης (activation functions) [4] είναι οι συναρτήσεις που εφαρμόζονται στο αποτέλεσμα των γραμμικών υπολογισμών κάθε νευρώνα και δίνουν την τελική έξοδο που προωθείται στους νευρώνες του επόμενου επιπέδου. Η χρήση τους είναι απαραίτητη για να επιτύχουμε καλής ακρίβειας αποτελέσματα. Εάν κάθε νευρώνας απλώς πολλαπλασιάζει τα βάρη του με τις εισόδους του και προσθέτει προκατάληψη (bias), το αποτέλεσμα θα ήταν γραμμικό ανεξαρτήτως του αριθμού των κρυμμένων επιπέδων (hidden layer) που διαθέτει το νευρωνικό μας δίκτυο.

Στη βαθιά μάθηση χρησιμοποιούνται πολλές μη γραμμικές συναρτήσεις ενεργοποίησης, δίνοντας τη δυνατότητα καθορισμού των παραμέτρων για καλύτερο έλεγχο αποτελεσμάτων. Οι σημαντικότερες των των συναρτήσεων αυτών είναι οι εξής:

2.1.1. Sigmoid

Η σιγμοειδής συνάρτηση $f(x) = \frac{1}{1+e^{-\beta x}}$, είναι από τις πρώτες συναρτήσεις που χρησιμοποιήθηκαν ως συναρτήσεις ενεργοποίησης, επιστρέφει τιμές μεταξύ του 0 και του 1. Αν και παραμένει από τις

πλέον σημαντικές συναρτήσεις στον τομέα των νευρωνικών δικτύων, παρουσιάζει δύο ελαττώματα που μειώνουν τη δημοτικότητα της. Αρχικά, σε περιπτώσεις που η ενεργοποίηση κάποιου νευρώνα πλησιάζει τις τιμές 0 ή 1, η κλίση της τείνει στο 0. Αυτό έχει ως αποτέλεσμα κατα το back propagation να τείνει στο μηδέν και η συνολική κλίση και εν τέλει μηδενικά σήματα να φεύγουν από τους νευρώνες στα βάρη. Ακόμη, η έξοδος της σιγμοειδούς δεν ισορροπεί γύρω από το 0, καθώς παίρνει τιμές μεταξύ μηδέν και ένα. Αυτό προκαλεί δυσκολίες στην βελτιστοποίηση, επειδή τα βάρη θα γίνονται ταυτοχρόνως είτε όλα θετικά είτε όλα αρνητικά, κάτι που κάνει την κλίση να παίρνει ακραίες τιμές σε διαφορετικές κατευθύνσεις.

Εκτός από την σιγμοειδή, στα ANN χρησιμοποιείται και μια υλοποίηση της hard sigmoid, που είναι μια προσέγγιση της σιγμοειδούς ευκολότερη στον υπολογισμό. Η hard sigmoid ορίζεται ως εξής:
$f(x) =$

$$f(x)=\begin{cases} 0, & x<-2.5 \\ 0.2\,x+0.5, & -2.5\le x\le 2.5 \\ 1, & x>2.5 \end{cases}$$

2.1.2. Softplus

Με το όνομα αυτό καλούμε τη συνάρτηση $f(x)=\ln(1+e^x)$. Η βασική διαφορά της με την σιγμοειδή είναι πως δεν είναι άνω φραγμένη, και επομένως το πρόβλημα του κορεσμού στις ακραίες τιμές εμφανίζεται μόνο στις χαμηλές τιμές (κοντά στο 0). Ενδιαφέρον προκαλεί ότι παραγωγίζοντας την Softplus καταλήγουμε στην Σιγμοειδή.

2.1.3. Tanh

Η συνάρτηση υπερβολικής εφαπτομένης (Hyperbolic Tangent Function) , με τύπο $f(x) = \frac{1}{1+e^{-2x}} - 1$παρουσιάζει πολλές ομοιότητες με τη σιγμοειδή, ωστόσο περιορίζει το αποτέλεσμά της μεταξύ του -1 και του 1, επομένως είναι κεντραρισμένη γύρω από το 0 με συνέπεια να μην εμφανίζει τα προβλήματα της σιγμοειδούς στην βελτιστοποίηση (optimization). Ωστόσο και αυτή τείνει να μηδενίζει την κλίση της όταν η έξοδος της πλησιάζει πολύ στις ακραίες τιμές της.

2.1.4. Softsign

Η Softsign είναι μια εναλλακτική της Tanh, που συγκλίνει πολυωνυμικά, σε αντίθεση με την εκθετική σύγκλιση της Υπερβολικής εφαπτομένης. Ορίζεται ως $f(x) = \frac{x}{1+|x|}$ και η έξοδος της προσεγγίζει κατα πολύ αυτή της tanh.

2.1.5. ReLU

Η συνάρτηση Διορθωμένης Γραμμικής Μονάδας (Rectified Linear Unit) έχει ιδιαίτερη σημασία, και γενικευμένη χρήση ακόμα και σήμερα στον τομέα της μηχανικής μάθησης. Με τύπο $f(x)=\max(0,x)$ αποφεύγει το πρόβλημα του μηδενισμού κλίσης που έχουν οι παραπάνω συναρτήσεις, και είναι ταχύτερη λόγω των απλούστερων υπολογισμών που απαιτεί. Αρνητικές είσοδοι προκαλούν μηδενική έξοδο, κάτι που σημαίνει “αποσύνδεση” του νευρώνα (δεν ενημερώνεται). Αυτή η αποσύνδεση αφενός βοηθάει το νευρωνικό δίκτυο στο να μην υπερπροσαρμοστεί στα υπο εκπαίδευση δεδομένα, περιορίζει όμως την εκπαίδευση του σε ορισμένες περιπτώσεις. Ετσι ένα πρόβλημα που παρουσιάζεται ειδικά όταν ο ρυθμός εκπαιδευσης είναι υψηλός είναι πως μεγάλες κλίσεις μπορούν να προκαλέσουν ενημέρωση βαρών με τέτοιον τρόπο ωστε ο νευρώνας να μην μπορεί να ενεργοποιηθεί

ξανα στο μέλλον ανεξάρτητα των νέων δεδομένων. Διάφορες παραλλαγές της ReLU χρησιμοποιούνται λόγω της μεγάλης της αποδοχής, και το Keras ενσωματώνει τόσο την Exponential Linear Unit (ELU):

$$f(x) = \begin{cases} \alpha(exp(x) - 1) & \text{for} \quad x \leq 0 \\ x & \text{for} \quad x > 0 \end{cases}$$

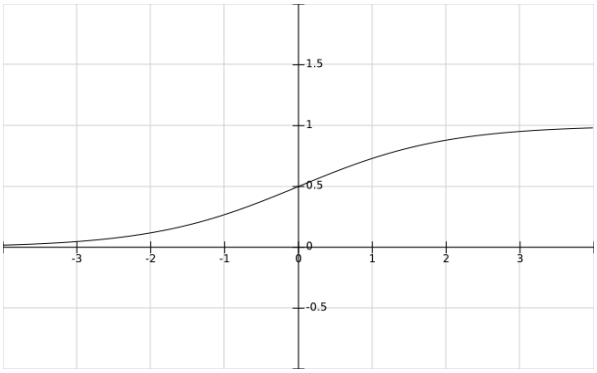
όσο και την Scaled Exponential Linear Unit (SELU), που πολλαπλασιάζει το αποτέλεσμα της ELU με μία δοθείσα τιμή. Τα πλεονεκτήματα της Εκθετικής Γραμμικής Μονάδας είναι ότι αρνητικές τιμές κοντά στο 0 έχουν πιο φυσική κλίση και επιτρέπουν ταχύτερη εκπαίδευση από τους απενεργοποιημένους νευρώνες της RELU, αλλά και από τις διάφορες άλλες παραλλαγές της που προσφέρουν γραμμική εκπαίδευση στις αρνητικές τιμές.
Μια τέτοια παραλλαγή (που δεν έχει άμεση υλοποίηση στο Keras) είναι η Leaky ReLU, που αντιμετωπίζει το πρόβλημα των νεκρών νευρώνων της RELU χρησιμοποιώντας την ακόλουθη συνάρτηση προσφέροντας έτσι δυνατότητα σε αρνητικές κλίσεις να επανέλθουν στο θετικό τμήμα.

$$f(x)=\begin{cases} 0.05\,x & for\,x<0 \\ x & for\,x\geq 0 \end{cases}$$

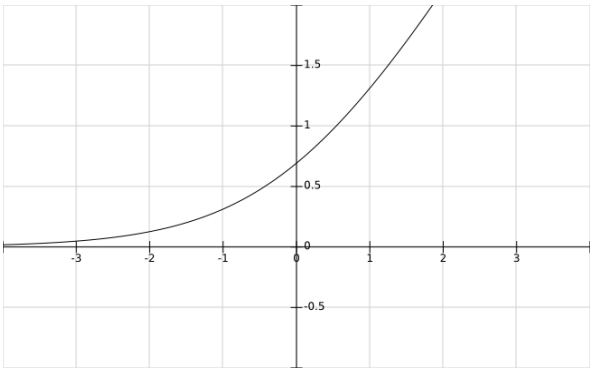
2.1.6. Softmax

Η Softmax $f(x) = \frac{e^{z_i}}{\sum_{\kappa=1}^{K} e^{z_\kappa}}$είναι μία πολύ σημαντική συνάρτηση ενεργοποίησης και έχει πολύ μεγάλη χρήση στο επίπεδο νευρώνων εξόδου των δικτύων. Κανονικοποιεί τις τιμές στο διάστημα [0,1] και φροντίζει να αθροίζουν στην μονάδα, κάτι που την καθιστά ιδανική για την έξοδο όταν αναμένουμε πιθανότητες ως τελικό αποτέλεσμα (πχ κατηγοριοποίηση εικόνων).

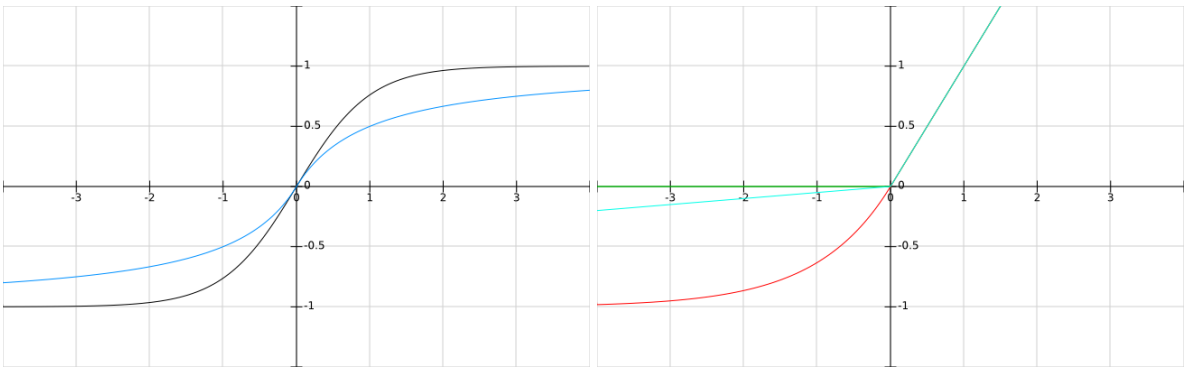
Στα ANN επίσης υπάρχει η γραμμική συνάρτηση f(x)=x, η οποία βρίσκει χρήση στο επίπεδο εξόδου σε προβλήματα οπισθοδρόμησης (regression). Το σχήμα 2.1 απεικονίζει τις πιο δημοφιλείς συναρτήσεις ενεργοποίησης.



Η σιγμοειδής συνάρτηση



Η συνάρτηση softplus



Με μαύρο χρώμα η συνάρτηση tanh,
ενώ με μπλε χρώμα η softsign

Πρασινο χρώμα: ReLU
Γαλάζιο χρώμα: LReLU
Κόκκινο χρώμα: ELU

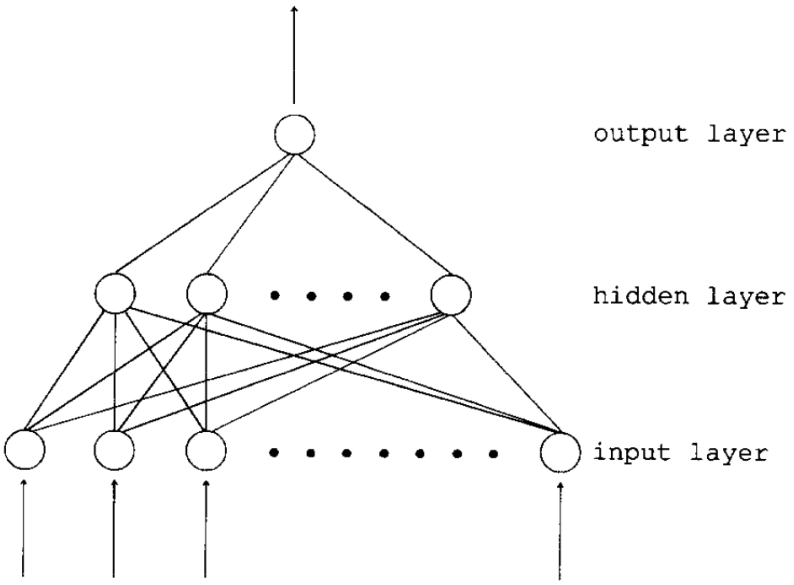
Σχήμα 2.1Συναρτήσεις Ενεργοποίησης

2.2. Επίπεδα

Τα επίπεδα ή αλλιώς στρώματα (layers) είναι το υψηλότερου-επιπέδου δομικό στοιχείο στη βαθιά μάθηση. Ένα επίπεδο λειτουργεί ως ένα κοντέινερ που συνήθως λαμβάνει μια σταθμισμένη είσοδο, τη μετατρέπει κυρίως μέσω ενός συνόλου μη γραμμικών συναρτήσεων σε μια καινούρια απεικόνιση και στη συνέχεια μεταβιβάζει αυτές τις τιμές ως έξοδο στο επόμενο επίπεδο. Στη συνέχεια θα δούμε τα βασικά είδη επιπέδων που χρησιμοποιούμε στα ANN.

2.2.1. Τροφοδοτούμενα προς τα εμπρός

Τα Πολυστρωματικά Τροφοδοτούμενα προς τα Εμπρός νευρωνικά, Multi Layer Feedforward (MLF) [5] δίκτυα που τροφοδοτούνται προς τα εμπρός, εκπαιδεύονται με την διαδικασία εκμάθησης back-propagation, είναι τα πιο απλά και δημοφιλή νευρωνικά δίκτυα. Ένα νευρωνικό δίκτυο MLF αποτελείται από νευρώνες, οι οποίοι ταξινομούνται σε στρώματα σχήμα 2.2
Το πρώτο στρώμα ονομάζεται στρώμα εισόδου, το τελευταίο στρώμα ονομάζεται στρώμα εξόδου και τα στρώματα μεταξύ αυτών είναι τα κρυφά στρώματα Για την επίσημη περιγραφή των νευρώνων μπορούμε να χρησιμοποιήσουμε τη λεγόμενη συνάρτηση απεικόνισης Γ, η οποία εκχωρεί για κάθε νευρώνα i ένα υποσύνολο $\Gamma(i) \subseteq VT$ που αποτελείται από όλους τους προγόνους του δεδομένου νευρώνα. Ένα υποσύνολο $\Gamma^{-1}(i) \subseteq VT$ από ό, τι αποτελείται από όλους τους προκατόχους του δεδομένου νευρώνα i. Κάθε νευρώνας σε ένα συγκεκριμένο στρώμα συνδέεται με όλους τους νευρώνες στο επόμενο στρώμα. Η σύνδεση μεταξύ του νευρώνα ith και jth χαρακτηρίζεται από τον συντελεστή βάρους w_{ij} και τον νευρώνα i από τον συντελεστή κατωφλίου θ_i σχήμα 2.2. Ο συντελεστής βάρους αντανακλά το βαθμό σπουδαιότητας της δεδομένης σύνδεσης στο νευρωνικό δίκτυο. Η τιμή εξόδου (δραστηριότητα) του i νευρώνα x_i καθορίζεται από τις ακόλουθες εξισώσεις και θεωρεί ότι:

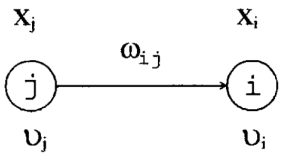


Σχήμα.2.2 Τροφοδοτούμενο προς τα εμπρός ANN

$$x_i = f(\xi_i)$$

$$\xi_i = \theta_i + \sum_{j \in \Gamma_i^{-1}} \omega_{ij} x_j$$

όπου ξ_i είναι η πιθανότητα του i νευρώνα και η συνάρτηση $f(\xi_i)$ είναι η συνάρτηση ενεργοποίησης. Το άθροισμα στην παραπάνω εξίσωση πραγματοποιείται σε όλους τους νευρώνες j που μεταφέρουν το σήμα στον νευρώνα i όπως βλέπουμε στο σχήμα 2.3. Ο συντελεστής κατωφλίου μπορεί να γίνει κατανοητός ως συντελεστής βάρους της σύνδεσης με τον τυπικά προστιθέμενο νευρώνα j, όπου $x_j = 1$ (το λεγόμενο bias)



Σχήμα.2.3 Σύναψη νευρώνων

Έστω η συνάρτηση ενεργοποίησης:

$$f(\xi) = \frac{1}{1+exp(-\xi)}$$

Η εποπτευόμενη διαδικασία προσαρμοζει τους συντελεστές κατωφλίου θ_i και τους συντελεστές βάρους w_{ij} με σκοπό να ελαχιστοποιηθεί το άθροισμα των τετραγωνικών διαφορών μεταξύ των υπολογισμένων και των απαιτούμενων τιμών εξόδου. Αυτό επιτυγχάνεται με την ελαχιστοποίηση της αντικειμενικής συνάρτησης E:

$$E = \sum_{o} \frac{1}{2} (x_o - \hat{x}_o)^2$$

όπου x_o, και $\hat{x}_o$, είναι διανύσματα που αποτελούνται από τις συμπιεσμένες και απαιτούμενες δραστηριότητες των εξερχόμενων νευρώνων και το άθροισμα τρέχει σε όλους τους εξερχόμενους νευρώνες ο.

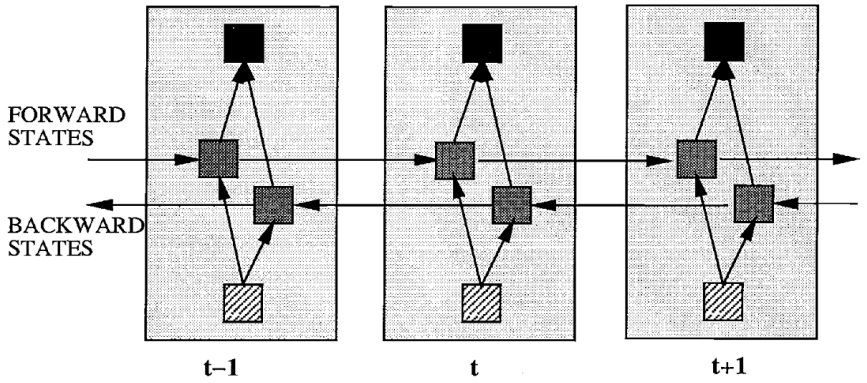
2.2.2. Επαναλαμβανόμενο Νευρωνικά Δίκτυα

Τα Επαναλαμβανόμενα Νευρωνικά Δίκτυα (recurrent neural networks) (RNN) [6] είναι ένας τύπος ANN, που διευκολύνει τη δυναμική και χρονική συσχέτιση των δεδομένων. Τα RNN διαχειρίζονται αλληλουχίες δεδομένων και διατηρούν την κατάσταση των προηγούμενων εισόδων. Το αρχιτεκτονικό παράδειγμα RNN βασίζεται σε κόμβους οργανωμένους σε διαδοχικά επίπεδα, όπου κάθε κόμβος συνδέεται με κόμβους του επόμενου διαδοχικού επιπέδου και έχει επίσης επαναλαμβανόμενες συνδέσεις. Οι επαναλαμβανόμενες συνδέσεις επιτρέπουν την πληροφορία σχετικά με τις προηγούμενες εισόδους δεδομένων να επηρεάσουν τις μελλοντικές εξόδους, καθιστώντας έτσι το RNN μια καλή επιλογή μοντελοποίησης χρονοσειρών.
Μέσω των RNN μπορούμε να προβλέπουμε τις μελλοντικές τιμές μιας μετρικής που μας ενδιαφέρει με βάση τις τρέχουσες αλλά και τις προηγούμενες τιμές τους. Είναι σημαντικό να αναφέρουμε ότι το RNN τείνει να συγκλίνει συχνά σε υποβέλτιστες λύσεις λόγω της εκπαίδευσης του με την μέθοδο εκμάθησης με κλίση (gradient descent) και backpropagation. Κατά τη διάρκεια αυτών των μεθόδων εκπαίδευσης, κάθε ένα από τα βάρη του δικτύου λαμβάνει μια ενημέρωση ανάλογη με τη μερική παράγωγο της συνάρτησης σφάλματος σε σχέση με το τρέχον βάρος. Αυτό το συμβάν αναφέρεται ως (vanishing gradient) και περιγράφει την περίπτωση η βαθμίδα (gradient) να είναι τόσο μικρή ώστε τα βάρη του δικτύου να μπορούν να ενημερώνονται για μεγαλύτερο χρονικό διάστημα με τρόπο που είναι επωφελής για τη διαδικασία εκμάθησης. Το αποτέλεσμα της βαθμίδας μειωνεται κατα πολυ στα πρώτα στρώματα του RNN, υποβαθμίζοντας την ικανότητα χρήσης μοτίβων σε μεγαλύτερες ακολουθίες. Στην περίπτωση των μετρήσεων χρονοσειρών, είναι σημαντικό να είναι σε θέση να δημιουργούνται μακροπρόθεσμες συσχετίσεις σε μεγαλύτερες ακολουθίες.

2.2.3. Αμφίδρομο Επαναλαμβανόμενο Νευρωνικό Δίκτυο

Για να ξεπεραστούν οι περιορισμοί του κανονικού RNN που περιγράφονται στην προηγούμενη ενότητα, προτείνουμε ένα αμφίδρομο επαναλαμβανόμενο νευρωνικό δίκτυο (Bidirectional Recurrent Neural Network) (BRNN) [7] που μπορεί να εκπαιδευτεί χρησιμοποιώντας όλες τις διαθέσιμες πληροφορίες εισαγωγής από το παρελθόν και το μέλλον ενός συγκεκριμένου χρονικού πλαισίου.
Η δομή των BRNN βασίζεται στην ιδέα ότι για να χωριστούν οι νευρώνες κατάστασης ενός κανονικού RNN σε ένα μέρος που είναι υπεύθυνο για τη θετική κατεύθυνση του χρόνου (προς τα εμπρός καταστάσεις) και ένα μέρος που είναι υπεύθυνο για την αρνητική κατεύθυνση του χρόνου (προς τα πίσω καταστάσεις). Οι έξοδοι από τις προς τα εμπρός καταστάσεις δεν συνδέονται με τις εισόδους των προς τα πίσω καταστάσεων και αντίστροφα Αυτό οδηγεί στη γενική δομή που φαίνεται στο σχήμα 2.4, όπου ξετυλίγεται σε τρία χρονικά βήματα. Εάν οι εμπρόσθιες καταστάσεις αφαιρεθούν, προκύπτει ένα κανονικό RNN με αντίστροφο άξονα χρόνου. Με τις δύο κατευθύνσεις στο ίδιο δίκτυο, οι πληροφορίες εισαγωγής στο παρελθόν και το μέλλον του τρέχοντος

αξιολογούμενου χρονικού πλαισίου μπορούν να χρησιμοποιηθούν άμεσα για την ελαχιστοποίηση της συνάρτησης σφάλματος χωρίς την ανάγκη καθυστερήσεων για να συμπεριληφθούν μελλοντικές πληροφορίες.



Σχήμα 2.4 Η γενική δομή του αμφίδρομου επαναλαμβανόμενου νευρωνικού δικτύου για τρία χρονικά βήματα.

Το BRNN μπορεί κυρίως να εκπαιδευτεί με τους ίδιους αλγόριθμους με έναν κανονικό μονόδρομο RNN, επειδή δεν υπάρχουν αλληλεπιδράσεις μεταξύ των δύο τύπων καταστάσεων των νευρώνων και μπορεί να ξεδιπλωθεί σε ένα γενικό feedforward ANN. Ωστόσο, εάν, για παράδειγμα, χρησιμοποιηθεί οποιαδήποτε μορφή backward μέσω του χρόνου, η διαδικασία προώθησης προς τα εμπρός και προς τα πίσω είναι ελαφρώς πιο περίπλοκη, επειδή η ενημέρωση των νευρώνων κατάστασης και εξόδου δεν μπορεί πλέον να γίνει μία κάθε φορά. Εάν χρησιμοποιείται το BPTT, τα εμπρός και πίσω περάσματα πάνω από το ξεδιπλωμένο BRNN με την πάροδο του χρόνου γίνονται σχεδόν με τον ίδιο τρόπο όπως για ένα κανονικό MLP. Κάποια ειδική μεταχείριση είναι απαραίτητη μόνο στην αρχή και στο τέλος των δεδομένων εκπαίδευσης. Η εμπρόσθια κατάσταση που εισέρχεται την χρονική στιγμή t=1 και στην backward state την χρονική στιγμή t=T δεν είναι γνωστές. Ο καθορισμός αυτών θα μπορούσε να γίνει μέρος της διαδικασίας εκμάθησης, αλλά εδώ, ορίζεται αυθαίρετα σε μια σταθερή τιμή (0,5). Επιπλέον, οι παράγωγοι τοπικής κατάστασης στο t=T για την εμπρόσθια κατάσταση και t=1 για την backward κατάσταση δεν είναι γνωστές και τίθενται εδώ στο μηδέν, υποθέτοντας ότι οι πληροφορίες πέρα από αυτό το σημείο δεν είναι σημαντικές για την τρέχουσα ενημέρωση, που είναι για συγκεκριμένα όρια. Η διαδικασία εκπαίδευσης για το ξεδιπλωμένο αμφίδρομο δίκτυο με την πάροδο του χρόνου μπορεί να συνοψιστεί ως εξής:

1) Εμπρόσθιο πέρασμα

Εκτελούνται όλα τα δεδομένα εισόδου για $1 \leq t \leq T$ μέσω του BRNN και προσδιορίζονται όλες οι προβλεπόμενες έξοδοι.

α) Γίνεται το εμπρόσθιο πέρασμα μόνο για τις εμπρόσθιες καταστάσεις (από t=1 έως t=T) και το backward περασμα για τις καταστάσεις από t = T έως t = 1.

β) Γίνεται το εμπρόσθιο πέρασμα για την έξοδο των νευρώνων.

2) Backward πέρασμα

Υπολογίζεται η παράγωγος της αντικειμενικής συνάρτησης για το χρονικό διάστημα $1 \leq t \leq T$που χρησιμοποιείται στο μπροστινό πέρασμα.

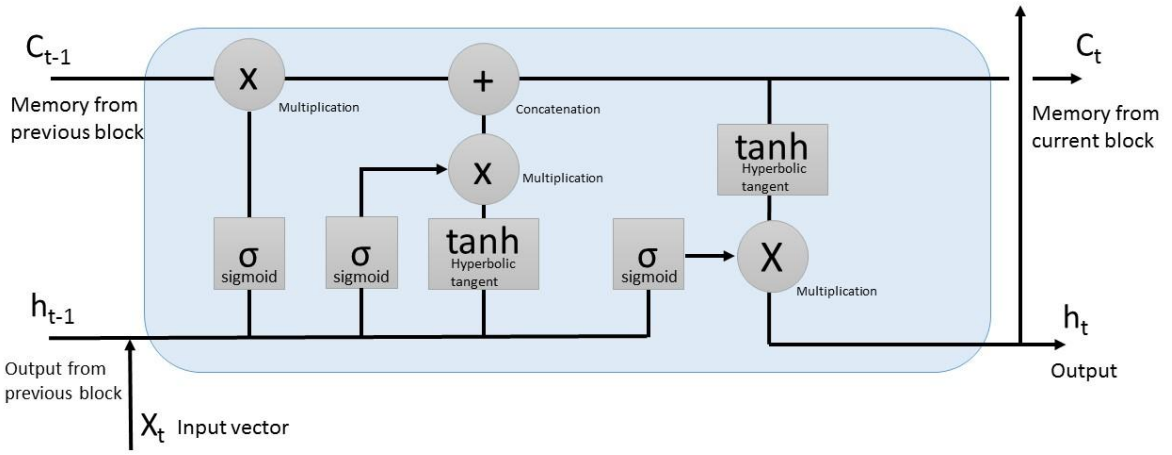
α) Γίνεται το backward πέρασμα για τους νευρώνες εξόδου.

β) Γίνεται το backward πέρασμα για τις εμπρόσθιες καταστάσεις από t = T έως t =1 και backward για τις καταστάσεις από t = 2 έως t = T.

3) Ενημέρωση των βαρών

2.2.4. Μακράς-Βραχείας Μνήμης LSTM

Προκειμένου να αντιμετωπιστούν οι περιορισμοί που περιγράφονται στα RNN προτάθηκαν οι νευρώνες Μακράς-Βραχείας Μνήμης (Long Short Term Memory) (LSTM) [8]. Τα δίκτυα LSTM μοιάζουν με τα RNN, διευκολύνουν τον μηχανισμό κρυφής κατάστασης που συνδέει έναν κόμβο του δικτύου με τον άλλο επιτρέποντας έτσι την εκδήλωση δυναμικής χρονικής συμπεριφοράς με παρόμοιο τρόπο. Αυτό που διαφοροποιεί τα δίκτυα LSTM είναι η χρησιμοποίηση της δομής cell state. Η αρχιτεκτονική LSTM επιτρέπει την τροποποίηση των πληροφοριών της cell state, μόνο μέσω ορισμένων ειδικών ρυθμιστικών μηχανισμών, που ονομάζονται πυλες (gates). Κάθε LSTM κόμβος όπως φαίνεται στο σχήμα 2.5 περιέχει τρεις ξεχωριστές δομές που σχετίζονται με την πύλη. Όλοι οι κόμβοι περιεχουν sigmoid που παρέχουν ομαλές καμπύλες στη ζώνη 0 έως 1, διασφαλίζοντας έτσι ότι το μοντέλο θα παραμείνει διαφοροποιησιμο. Συμπιέζοντας τις τιμές μεταξύ 0 και 1, η σιγμοειδής (sigmoid) activation βοηθά το δίκτυο να μάθει ποια δεδομένα είναι σημαντικά ή όχι και, στη συνέχεια, να τα διατηρήσει ή να τα ξεχάσει.



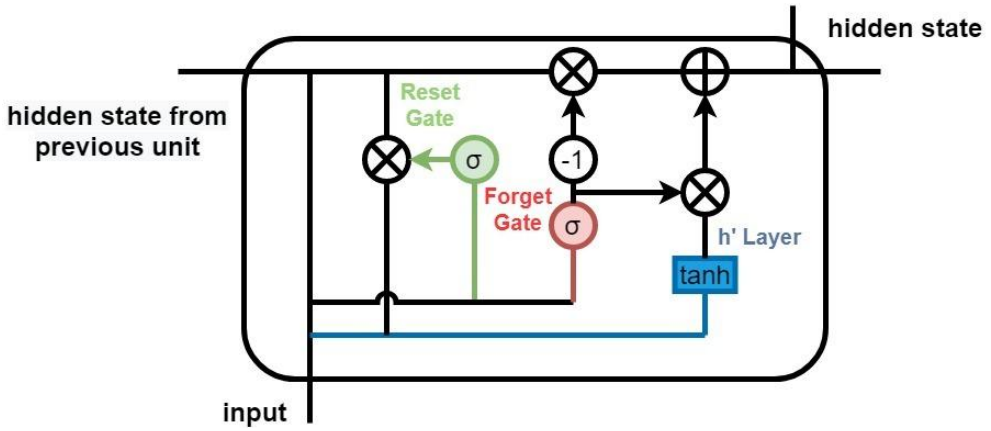
Σχήμα 2.5 Νευρώνας Μακράς-Βραχείας Μνήμης

Η Forget Gate αποφασίζει τι κομμάτι πληροφοριών θα αφαιρεθεί από την κατάσταση κελιών. Περιέχει ένα sigmoid activation που δημιουργεί ένα συνδυασμό της κρυφής κατάστασης και του διανύσματος εισόδου και στη συνέχεια παράγει μια έξοδο για κάθε τιμή στην cell state.
Η δομή πύλης εισόδου αποτελείται από δύο μέρη. Το πρώτο είναι το Layer Gate Layer που αποφασίζει ποιες πληροφορίες θα ενημερωθούν. Το δεύτερο είναι ένα tanh που ελαχιστοποιεί τις επιπτώσεις του φαινομένου gradient descent. Εκτελεί αυτή την εργασία διανέμοντας τις κλίσεις με επαρκή τρόπο, με μηδενικό κέντρο. Αυτό επιτρέπει στις πληροφορίες του κελιού να επιπλέουν περισσότερο χωρίς εξαφάνιση ή εκτίναξη. Αυτός ο φορέας δημιουργίας επιπέδου περιέχει τιμές που θα μπορούσαν ενδεχομένως να προστεθούν στην κατάσταση κελιών. Για να ενημερώνουν σωστά οι πληροφορίες της κατάστασης κελιού, πρέπει να χρησιμοποιηθούν αυτά τα επίπεδα. Η τρέχουσα

κατάσταση πολλαπλασιάζεται με την έξοδο του Forget Gate Structure, εκτελώντας έτσι τη διαδικασία «Forget».
Το Output Gate αποφασίζει ποια θα είναι η επόμενη κρυφή κατάσταση. Η έξοδος είναι ουσιαστικά μια φιλτραρισμένη έκδοση του Cell State. Για να γίνει αυτό, πρέπει πρώτα να δημιουργηθεί ένα φίλτρο χρησιμοποιώντας ένα σιγμοειδές στρώμα. Στη συνέχεια, βάζουμε την cell state μέσω του επιπέδου tanh και πολλαπλασιάζουμε την έξοδο του σιγμοειδούς στρώματος. Η συνολική έξοδος είναι η κρυφή κατάσταση. Η νέα κατάσταση κυψέλης καθώς και η νέα κρυφή κατάσταση μεταφέρονται στη συνέχεια στον επόμενο κόμβο LSTM που αντιστοιχεί στο επόμενο χρονικό βήμα.

2.2.5. Ανατροφοδοτουμενη μονάδα με πύλη GRU

Η Ανατροφοδοτούμενη Μονάδα με Πύλη (Gated Recurrent Unit) (GRU) [9] είναι παρόμοια με το LSTM καθώς και οι δύο καταφέρνουν να αποτρέψουν το πρόβλημα της εξαφάνισης της κλίσης χρησιμοποιώντας τις δομές πύλης. Αυτό που τους ξεχωρίζει είναι ότι η GRU συνδυάζει την πύλη ξεκλειδώματος και την πύλη εισόδου για να σχηματίσει μία πύλη ενημέρωσης. Με τη μείωση του αριθμού των εμπλεκόμενων πυλών, η GRU είναι σε θέση να παρέχει λιγότερο περίπλοκες δομές και συνεπώς να είναι πιο υπολογιστικά αποτελεσματική σε σύγκριση με την LSTM, ενώ ταυτόχρονα καταφέρνει να αποδώση εξίσου καλή.
Τα δίκτυα GRU διευκολύνουν πολύ τον μηχανισμό κρυφής κατάστασης που συνδέει μια μονάδα του δικτύου με την επόμενη επιτρέποντας έτσι την εκδήλωση δυναμικής χρονικής συμπεριφοράς με παρόμοιο τρόπο. Κάθε μονάδα GRU όπως φαίνεται στο σχήμα 2.6 είναι ενδεικτική ενός συγκεκριμένου χρονικού βήματος που διευκολύνει τη μετάδοση σημαντικών πληροφοριών μέσω του συνεχούς χρόνου. Επιπλέον περιέχει δύο ξεχωριστές δομές πυλών οι οποίες θα εξηγηθούν λεπτομερώς παρακάτω. Η πρώτη αναφέρεται ως πύλη επαναφοράς, ενώ η δεύτερη αναφέρεται ως πύλη ενημέρωσης. Και οι δύο φέρουν σιγμοειδή στρώματα που παρέχουν ομαλές καμπύλες στη ζώνη 0 έως 1, διασφαλίζοντας έτσι ότι το μοντέλο θα παραμείνει διαφορίσιμο. Με τη συμπίεση των τιμών μεταξύ 0 και 1, η ενεργοποίηση της σιγμοειδούς βοηθά το δίκτυο να μάθει ποια δεδομένα είναι σημαντικά και ποια όχι. Στη συνέχεια, να τα διατηρήσει ή να τα ξεχάσει.
Η πύλη επαναφοράς που είναι υπεύθυνη για να αποφασίσει πόσες από τις προηγούμενες πληροφορίες θα ξεχαστούν. Όπως και πριν, είναι απαραίτητο να πολλαπλασιαστεί η είσοδος και η κρυφή κατάσταση με τα αντίστοιχα βάρη τους και στη συνέχεια να τοποθετηθεί το άθροισμα των αποτελεσμάτων πολλαπλασιασμού μέσω ενός σιγμοειδούς layer.
Για άλλη μια φορά χρειάζεται να πολλαπλασιαστεί η κρυφή κατάσταση με τα προοπτικά βάρη της. Το αποτέλεσμα αυτής της λειτουργίας χρησιμοποιείται για τον υπολογισμό του γινομένου Hadamard μεταξύ της εισόδου και της εξόδου της πύλης επαναφοράς. Αυτή η διαδικασία είναι απαραίτητη για να αποφασιστεί πόση από τις πληροφορίες που συλλέχθηκαν κατά τα προηγούμενα χρονικά βήματα θα αφαιρεθεί. Τότε μοιάζει πολύ πριν η είσοδος πολλαπλασιαστεί με τα βάρη της και το αποτέλεσμα προστίθεται μαζί με το γινόμενο Hadamard.



Σχήμα 2.6 Ανατροφοδοτούμενη μονάδα με πύλη

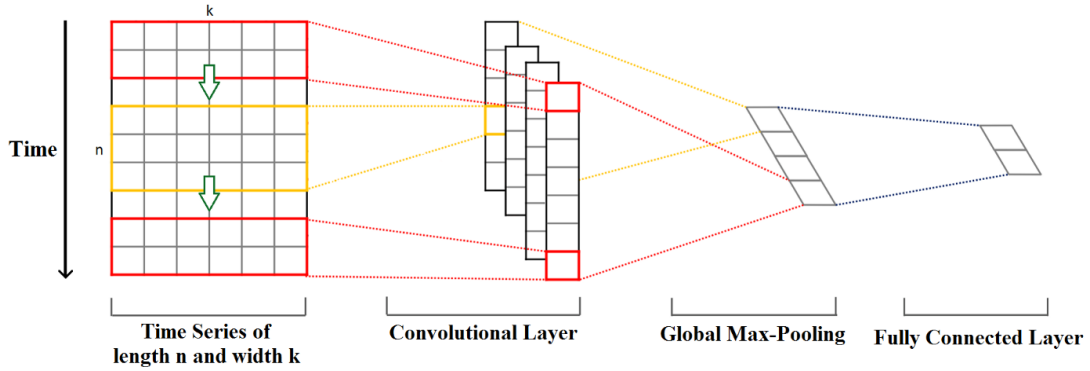
Τέλος, το αποτέλεσμα αυτών των διεργασιών τοποθετείται μέσω ενός στρώματος tanh που ελαχιστοποιεί τις επιπτώσεις του φαινομένου της εξαφάνισης κλίσης. Εκτελείται αυτή η εργασία διανέμοντας τις κλίσεις, γύρω από ένα μηδενικό κέντρο. Αυτό επιτρέπει στις πληροφορίες να παραμένουν περισσότερο χωρίς να εξαφανίζονται. Το γινόμενο αυτό μέχρι στιγμής θα αναφέρεται ως h′.

Η πύλη ενημέρωσης που είναι υπεύθυνη για τον προσδιορισμό του ποσού των πληροφοριών που συγκεντρώθηκαν κατά τα προηγούμενα χρονικά βήματα πρέπει να περάσει για μελλοντική χρήση. Από αυτή την άποψη, η συμπεριφορά του είναι αρκετά παρόμοια με αυτήν της πύλης επαναφοράς. Το πρώτο βήμα απαιτεί τον πολλαπλασιασμό της εισόδου και της κρυφής κατάστασης με τα προοπτικά βάρη τους. Η κρυφή κατάσταση περιλαμβάνει πληροφορίες που προέρχονται από τις προηγούμενες t - 1 μονάδες. Στη συνέχεια, τα αποτελέσματα πολλαπλασιασμού προστίθενται μεταξύ τους και τοποθετούνται μέσα σε ένα σιγμοειδές στρώμα.

Για να είναι ενημερωμένη η κρυφή κατάσταση, το πρώτο βήμα είναι να εκτελεστεί ένας πολλαπλασιασμός με στοιχεία για την έξοδο της πύλης ενημέρωσης και την κρυφή κατάσταση. Το δεύτερο είναι να εκτελεστεί ένας πολλαπλασιασμός με το h′. Η ενημερωμένη κρυφή κατάσταση είναι το άθροισμα των δύο γινομένων. Η ενημερωμένη κρυφή κατάσταση μεταφέρεται στη συνέχεια στην επόμενη μονάδα GRU η οποία αντιστοιχεί στο επόμενο χρονικό βήμα.

2.2.6. Συνελικτικά Νευρωνικά Δίκτυα Convolutional

Τα Συνελικτικά Νευρωνικά Δίκτυα (Convolutional Neural Network) (CNN) [10] είναι ένας τύπος νευρωνικού δικτύου που είναι κατάλληλος για την επεξεργασία δεδομένων που έχουν δομή πλέγματος. Η αρχιτεκτονική του αποτελείται συνήθως από ένα ή περισσότερα συνελικτικά στρώματα με στάδια δειγματοληψίας και από ένα ή περισσότερα πλήρως συνδεδεμένα στρώματα όπως συμβαίνει σε κοινά νευρωνικά δίκτυα πολλαπλών επιπέδων. Τα CNN, όπως φαίνεται στο σχήμα 2.7 [11], χρησιμοποιούν ένα βελτιστοποιημένο σύνολο χαρακτηριστικών που ονομάζεται πυρήνας (kernel). Ο Πυρήνας επιτρέπει στα CNN να απομνημονεύουν εύκολα τη χωρική σειρά χαρακτηριστικών. Αυτό βοηθά τα CNN να λειτουργούν με σημαντική ακρίβεια και αποτελεσματικότητα. Πραγματοποιείται μια συνέλιξη μεταξύ της μήτρας δεδομένων και της μήτρας του πυρήνα, με το παράθυρο της συνέλιξης να κινεί n στοιχεία για κάθε πολλαπλασιασμό. Η παράμετρος n ονομάζεται strides.



Σχήμα 2.7 Συνελικτικά Νευρωνικά Δίκτυα

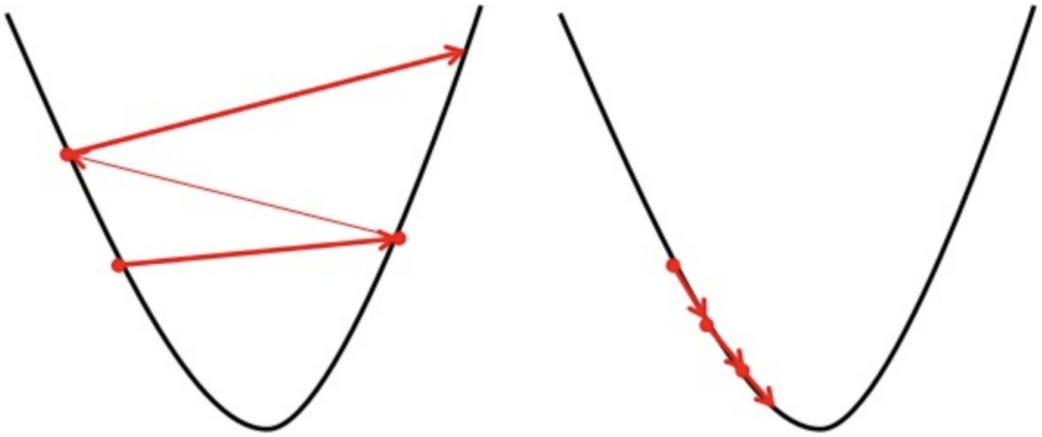
Μετά το συνελικτικό επίπεδο, μια συνήθης πρακτική είναι η προσθήκη ενός στρώματος συγκέντρωσης (pooling layer), είτε ενός μέγιστου είτε ενός μέσου όρου. Ο σκοπός του στρώματος συγκέντρωσης είναι να μειώσει τη διάσταση και, συνεπώς, την υπολογιστική ισχύ που απαιτείται για την επεξεργασία των δεδομένων, καθώς και για την καταστολή του θορύβου. Μετά τα στρώματα της συνέλιξης και της συγκέντρωσης, στα δεδομένα γίνεται μια πράξη flatten, οπότε μπορούν να χρησιμοποιηθούν ως είσοδος σε ένα πλήρως συνδεδεμένο δίκτυο προώθησης τροφοδοσίας, το οποίο μπορεί να βοηθήσει στην εκμάθηση μη γραμμικών μοτίβων και χαρακτηριστικών.

2.3. Βελτιστοποιητες

Οι βελτιστοποιητές (optimizers) [12] είναι οι μέθοδοι που χρησιμοποιούμε για να τροποποιήσουμε τις παραμέτρους του δικτύου μας. Παίρνοντας τα αποτελέσματα της συνάρτησης λάθους σε κάθε εκπαίδευση, αλλάζουμε κατάλληλα τα βάρη ωστε να επιτύχουμε ελαχιστοποίηση των λαθών. Κάποιο από τους πιο διαδεδομένους βελτιστοποιητες περιγράφονται παρακάτω.

2.3.1. Gradient Descent

Ο σημαντικότερος βελτιστοποιητής είναι η κλίση (gradient descent). Κατα τη χρήση του, υπολογίζουμε τη μεταβολή που θα είχε στην συνάρτηση λάθους κάθε μεμονωμένη αλλαγή βαρών (κλίση), και προσαρμόζουμε τα βάρη μας με βάση τη συνάρτηση αυτή, επαναλαμβάνοντας τη διαδικασία αυτήν ώστε να ελαχιστοποιήσουμε την τιμή της συνάρτησης λάθους. Μία σημαντική παράμετρος που βοηθάει σε αυτήν την διαδικασία είναι ο ρυθμός εκμάθησης (learning rate). Μεγάλος ρυθμος εκμάθησης συνεπάγεται μεγάλες μεταβολές προς την φαινομενικά κάθε φορά σωστή κατεύθυνση, κατι που μπορει να έχει ως αποτέλεσμα αδυναμία σύγκλισης στο ζητούμενο ελάχιστο. Μικρός ρυθμός εκμάθησης θα οδηγήσει σίγουρα σε κάποιο τοπικό ελάχιστο, αλλά πιθανότατα θα εγκλωβιστεί στο πλησιέστερο χωρίς να γνωρίζουμε εάν το ελάχιστο αυτό είναι και το ολικό. Θα πρέπει επομένως να βρίσκεται μια ισορροπία στην χρήση του learning rate.
Σημαντικό ρόλο στο κομμάτι της βελτιστοποίησης παίζει και η κανονικοποίηση. Στόχος ενός νευρωνικού δικτύου είναι να μπορεί να γενικοποιεί τη χρήση του και να μην υπερπροσαρμόζεται στα παραδείγματα εκπαίδευσης του, κάτι που μπορεί να συμβεί αν πχ μια παράμετρος έχει υπερβολικά υψηλό βάρος, με συνέπεια να κυριαρχεί στα αποτελέσματα. Κατα την κανονικοποίηση εφαρμόζουμε τεχνικές που αποτρέπουν τέτοια συμπεριφορά, επιβάλλοντας ποινές στις υψηλές τιμές βαρών, ακόμα και όταν λειτουργούν σύμφωνα με τα αναμενόμενα. Με αυτόν τον τρόπο διατηρούμε χαμηλά τα βάρη και γενικευουμε καλύτερα το δίκτυό μας σε νέα δεδομένα.

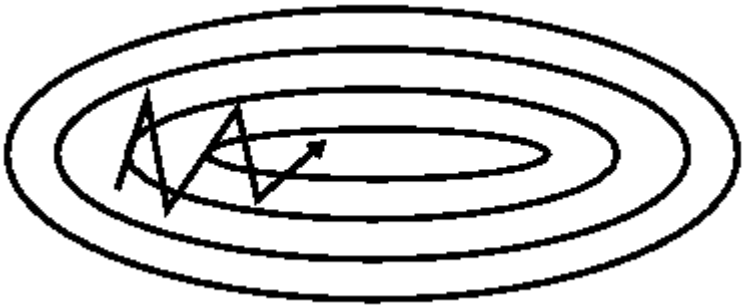


Σχήμα 2.8 Συμπεριφορά της κλίσης όταν το learning rate είναι: α) πολύ υψηλό β) πολύ χαμηλό

2.3.2. Stochastic Gradient Descent

Η στοχαστική κλίση (SGD) που παρέχεται από το περιβάλλον Keras, αντί να υπολογίζει την κλίση για όλα τα παραδείγματα εκπαίδευσης σε κάθε πέρασμα του βελτιστοποιητή, χρησιμοποιεί πακέτα παραδειγμάτων ή τυχαία παραδείγματα κάθε φορά. Εισάγοντας την έννοια της ορμής (momentum), επιχειρούμε να επιταχύνουμε την εκμάθηση σε διαστάσεις που οι κλίσεις τους διατηρούν την ίδια κατεύθυνση, και να μειώσουμε τις αλλαγές αντίστοιχα σε αυτές των οποίων οι κλίσεις αλλάζουν κατεύθυνση. Με αυτόν τον τρόπο επιτυγχάνουμε ταχύτερη σύγκλιση στο ελάχιστο μας, και λιγότερες ταλαντώσεις γύρω από αυτό. Ωστόσο, συσσωρευμένη ορμή καθώς πλησιάζουμε στο στόχο μας, θα μπορούσε να μας οδηγήσει στο να ξεπεράσουμε και σε μερικές περιπτώσεις να ξεφύγουμε εντελώς από το ζητούμενο ελάχιστο. Με τη χρήση Nesterov Momentum σχήμα 2.9, επιχειρούνται διορθώσεις της κίνησης με βάση την νέα κλίση, ώστε να επιτύχουμε την σύγκλιση.

SGD χωρίς ορμή



SGD με ορμή

Σχήμα 2.9 SGD με και χωρίς ορμή

2.3.3. Adagrad

Η ιδέα γύρω από την Adagrad είναι η προσαρμογή του ρυθμού εκμάθησης σε κάθε παράμετρο του δικτύου ξεχωριστά. Επομένως ορισμένα βάρη τροποποιούνται ταχύτερα από άλλα. Η adagrad βρίσκει χρήση σε αραιά datasets με λίγα δεδομένα εκπαίδευσης, όμως έχει το μειονέκτημα ότι ο ρυθμός εκμάθησης μειώνεται συνεχώς με το χρόνο, με συνέπεια να φτάνει σε σημείο όπου το δίκτυο δεν μπορεί να εκπαιδευτεί πλέον.

2.3.4. Adadelta

Η Adadelta αποτελεί μία συνάρτηση βελτιστοποίησης που επιχειρεί να διορθώσει το πρόβλημα της Adagrad, βασίζοντας το ρυθμό εκμάθησής της όχι σε όλες τις προηγούμενες κλίσεις, αλλά σε ένα "κινούμενο παράθυρο", επιτυγχάνοντας τη συνέχεια της εκπαιδευσης ακόμα και μετά από πολλές ενημερώσεις.

2.3.5. RMSProp

Άλλη μία βελτίωση της Adagrad, χρησιμοποιεί τη διαίρεση του ρυθμου εκμάθησης για κάθε βάρος με έναν κινητό μέσο όρο (σταθερού μεγέθους) των πρόσφατων κλίσεων του βάρους αυτού.

2.3.6. Adam

Ο βελτιστοποιητής Adam εκτός από τα ξεχωριστά learning rates για τα διαφορετικά βάρη, υπολογίζει ξεχωριστές ορμές για αυτά. Χρησιμοποιεί τις δύο πρώτες ροπές (μέσο όρο και διασπορά) της κλίσης για την ενημέρωση των παραμέτρων με παρόμοιο τρόπο με την Adadelta, και έχει ευρεία χρήση λόγω των καλών αποτελεσμάτων του.

2.3.7. Adamax

Ο Adamax αποτελεί μια παραλλαγή του Adam, που χρησιμοποιεί την νόρμα απείρου αντί για την δευτερη ροπή, και λειτουργεί καλύτερα σε δεδομένα με αραιές ενημερώσεις παραμέτρων (όπως πχ οι ενσωματώσεις), καθώς είναι πιο ανθεκτικός στις μικροαλλαγές της κλίσης.

2.3.8. Nadam

Η τελευταία συνάρτηση βελτιστοποίησης που παρέχεται από το Keras, εκμεταλλεύεται τα χαρακτηριστικά της RMSProp όπως και η Adam, όμως χρησιμοποιεί Nesterov Momentum το οποίο φέρνει καλύτερα αποτελέσματα από την κλασική ορμή που χρησιμοποιεί η Adam.

2.4. Κανονικοποίηση

Το μοντέλο DL πρέπει να είναι κανονικοποιημένο για να μπορεί να γενικευει σε νέα δεδομένα. Μία κοινή πρακτική για την επίτευξη αυτού είναι η τεχνική dropout, στην οποία ένα ποσοστό των επαναλαμβανόμενων συνδέσεων αποκλείεται από την ενεργοποίηση κατά τη διάρκεια κάθε εκμάθησης. Η πρόωρη διακοπή (early stopping) είναι μια ακόμη τεχνική κανονικοποίησης στην οποία αξιολογούμε το μοντέλο σε κάθε επανάληψη της εκπαίδευσης και όταν η ακρίβεια μειώνεται στο σύνολο δεδομένων δοκιμών, η εκπαίδευση σταματά.

2.4.1. L1και L2 Κανονικοποίηση

Η χρήση των L1 και L2 [13] είναι η πιο δημοφιλής μέθοδος κανονικοποίησης. Μέσω των L1 ή L2 γίνεται η ενημέρωση της συνάρτηση γενικού κόστους προσθέτοντας έναν ακόμη όρο τον όρο κανονικοποίησης.

Συνάρτηση κόστους = Απώλεια + Όρος κανονικοποίησης

Λόγω της προσθήκης αυτού του όρου κανονικοποίησης, οι τιμές των πινάκων με βάρη μειώνονται κάνοντας την υπόθεση ότι ένα νευρωνικό δίκτυο με μικρότερους πίνακες βάρους οδηγεί σε απλούστερα μοντέλα. Επομένως, μειώνονται τα βάρη σε μεγάλο βαθμό.

Στην εξίσωση το lambda είναι η παράμετρος κανονικοποίησης. Είναι μια υπερπαραμέτρος της οποίας η τιμή βελτιστοποιείται για καλύτερα αποτελέσματα. Η κανονικοποίηση L2 είναι επίσης γνωστή ως μείωση του βάρους καθώς αναγκάζει τα βάρη να συγκλίνουν προς το μηδέν (αλλά όχι ακριβώς το μηδέν).

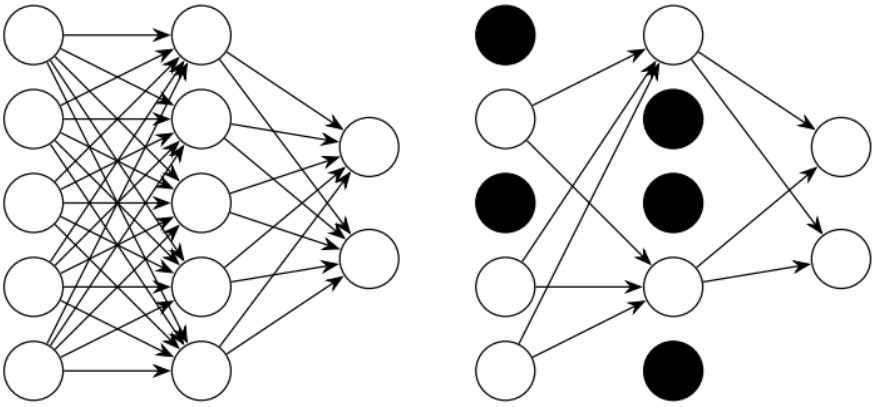
$$Cost\ function = Loss + \frac{\lambda}{2m} * \sum ||w||^2$$

$$Cost\ function = Loss + \frac{\lambda}{2m} * \sum ||w||$$

Στην L1, τιμωρούμε την απόλυτη αξία των βαρών. Σε αντίθεση με το L2, τα βάρη μπορεί να μειωθούν εδώ στο μηδέν. Ως εκ τούτου, είναι πολύ χρήσιμο όταν προσπαθούμε να συμπιέσουμε το νευρωνικό δίκτυο. Διαφορετικά, προτιμάται συνήθως το L2 από αυτό.

2.4.2. Εξομάλυνση Dropout

Η μέθοδος εξομάλυνσης (Dropout) [14], που εισήχθη από τους Hinton το 2012, παρέχει μια απλή τεχνική για την αποφυγή του overfitting στα feedforward νευρωνικά δίκτυα. Κατά τη διάρκεια κάθε επανάληψης στην εκπαίδευση, κάθε νευρώνας παραλείπεται από το δίκτυο με πιθανότητα p. Μόλις ολοκληρωθεί η εκπαίδευση του ANN, χρησιμοποιείται το πλήρες δίκτυο, αν και οι έξοδοι νευρώνων πολλαπλασιάζονται με την πιθανότητα p ότι ο νευρώνας παραλείφθηκε. Αυτό αντισταθμίζει το μεγαλύτερο μέγεθος του δικτύου τώρα που δεν παραλείπονται οι νευρώνες και μπορεί να ερμηνευθεί ως μέσος όρος στα πιθανά δίκτυα κατά τη διάρκεια της εκπαίδευσης. Η πιθανότητα μπορεί να διαφέρει για κάθε στρώμα, με την πρωτότυπη δημοσίευση του Hinton να προτείνει p = 0,2 για το επίπεδο εισόδου και p = 0,5 για κρυφά layers. Οι νευρώνες στο επίπεδο εξόδου δεν εξομαλύνονται. Αυτή η μέθοδος απεικονίζεται στο Σχήμα 2.10.



Σχήμα 2.10 Εξομάλυνση ANN

Στο Σχήμα 2.10 βλέπουμε ότι το αριστερό δίκτυο είναι πλήρως συνδεδεμένο και το δεξί είχε απορρίψει νευρώνες με πιθανότητα 0,5. Η εξομάλυνση δεν εφαρμόζεται στο επίπεδο εξόδου.
Μαθηματικά, δίνεται η συμπεριφορά της τυπικής εξομάλυνσης κατά τη διάρκεια της εκπαίδευσης για ένα επίπεδο νευρωνικού δικτύου με:

$$y = f(Wx) \circ m,\ m_i \sim Bernoulli(1 - p)$$

όπου y είναι η έξοδος στρώματος, f (·) είναι η συνάρτηση ενεργοποίησης, W είναι η μήτρα βάρους στρώσης, x είναι το επίπεδο εισόδου, και m είναι η μάσκα εγκατάλειψης στρώσης, με κάθε στοιχείο mi να έχει πιθανότητα από 0 έως p. Απο την στιγμή που το ANN είναι εκπαιδευμένο, η έξοδος στρώματος δίνεται από τον τύπο

$$y = (1 - p)\, f(Wx)$$

Η εξομάλυνση ισοδυναμεί με την προσθήκη ενός επιπλέον στρώματος μετά από ένα στρώμα νευρώνων που απλώς ρυθμίζει τιμές στο μηδέν με κάποια πιθανότητα κατά τη διάρκεια της εκπαίδευσης και πολλαπλασιάζονται με 1 - p κατά τη διάρκεια της αξιολόγησης.

Άλλες τυποποιήσεις τυπικής εξομάλυνσης μπορεί να κλιμακώσουν τα βάρη και όχι τις εξόδους κατά τη διάρκεια της δοκιμής, ή να κλιμακώνουν τις εξόδους κατά 1 / (1 - p) κατά τη διάρκεια της εκπαίδευσης αντί να τις κλιμακώσουν κατά τη διάρκεια της δοκιμής, αλλά και οι δύο προσεγγίσεις έχουν το ίδιο τελικά αποτέλεσμα.
Αυτή η μέθοδος αποδείχθηκε αποτελεσματική για τη ρύθμιση των ANN, επιτρέποντάς τους να εκπαιδεύονται για περισσότερες περιόδους χωρίς να επέλθει υπερβολική προσαρμογή στα δεδομένα εκπαίδευσης (overfitting) και με αποτέλεσμα βελτιωμένη ακρίβεια στα δεδομένα δοκιμής [1, 27]. Η εξομάλυνση χρησιμοποιείται ευρέως στην πράξη

2.4.3. Πρόωρο Σταμάτημα Early Stopping

Το πρόωρο σταμάτημα [15] είναι η πιο συχνά χρησιμοποιούμενη μέθοδος για την αποφυγή του overfitting. Το πρόωρο σταμάτημα βασίζεται στο να διαιρέσουμε τα δεδομένα σε δύο σύνολα, την εκπαίδευση και την αξιολόγηση, και να υπολογίσουμε περιοδικά το σφάλμα αξιολόγησης κατά τη διάρκεια της εκπαίδευσης. Η προπόνηση διακόπτεται όταν το ποσοστό σφάλματος αξιολογησης αρχίζει να αυξάνει. Είναι σημαντικό να αντιληφθούμε ότι το σφάλμα αξιολόγησης δεν αποτελεί καλή εκτίμηση του σφάλματος γενίκευσης. Μία μέθοδος για την εκτίμηση του σφάλματος γενίκευσης είναι η δοκιμή του δικτύου σε ένα τρίτο σύνολο δεδομένων, το σύνολο δοκιμών, το οποίο δεν χρησιμοποιείται καθόλου κατά τη διάρκεια της εκπαιδευτικής διαδικασίας. Το μειονέκτημα του χωρισμού σε σύνολο δοκιμών (split sample) είναι ότι μειώνει το ποσό των διαθέσιμων δεδομένων τόσο για εκπαίδευση όσο και για επικύρωση.
Μια διαφορετική δυνατότητα για τον υπολογισμό της γενίκευσης είναι η χρήση της λεγόμενης διασταυρούμενης επικύρωσης. Η διασταυρούμενη επικύρωση είναι μια βελτίωση στην επικύρωση split-sample που επιτρέπει να χρησιμοποιείτε όλα τα δεδομένα για εκπαίδευση. Στη διασταυρούμενη επικύρωση με k-αναδιπλώσεις, διαιρούνται τα δεδομένα σε k υποσύνολα ίσου μεγέθους. Εκπαιδεύουμε τους καθαρούς χρόνους k, κάθε φορά αφήνοντας ένα από τα υποσύνολα εκτός, αλλά χρησιμοποιώντας μόνο το υπολειπόμενο υποσύνολο για να υπολογίσουμε το κριτήριο σφάλματος. Εάν το k ισούται με το μέγεθος του δείγματος, αυτό ονομάζεται διασταυρούμενη επικύρωση. Ενώ έχει προταθεί η εφαρμογή της διασταυρούμενης επικύρωσης στην πρόωρη διακοπή, ο σωστός τρόπος για να γίνει αυτό δεν είναι προφανής. Το μειονέκτημα της πολλαπλής επικύρωσης είναι ότι πρέπει να επανεκπαιδευτεί το ANN πολλές φορές. Αλλά στην περίπτωση των νευρωνικών δικτύων MLF, η μεταβλητότητα μεταξύ των αποτελεσμάτων που λαμβάνονται σε διαφορετικές δοκιμές προκαλείται συχνά με το γεγονός, ότι η μάθηση ολοκληρώθηκε σε πολλά διαφορετικά τοπικά ελάχιστα. Συνεπώς, η μέθοδος διασταυρούμενης επικύρωσης είναι πιο κατάλληλη για νευρωνικά δίκτυα χωρίς τον κίνδυνο να βρεθεί στα τοπικά ελάχιστα. Υπάρχει μια μέθοδος επίσης παρόμοια με τη διασταυρούμενη επικύρωση, η λεγόμενη boot-strapping που φαίνεται να λειτουργεί καλύτερα από την πολλαπλή επικύρωση σε πολλές περιπτώσεις.
Το πρόωρο σταμάτημα έχει τα πλεονεκτήματά του ότι είναι γρήγορο και απαιτεί μόνο μία σημαντική απόφαση του χρήστη, το ποσοστό των δεδομένων που θα χρησιμοποιηθεί για την επικύρωση. Τα μειονεκτήματα που έχει είναι ότι δεν είναι γνωστο απο πριν το πόσα δεδομένα χρησιμοποιούνται για την εκπαίδευση και για το σύνολο επικύρωσης, καθώς επίσης και το πώς να χωριστούν τα δεδομένα σε ένα σετ εκπαίδευσης και σε ένα σετ δοκιμών.

2.5. Automated Deep Learning

Οι αποφάσεις επιλογής υπερπαραμέτρων περιλαμβάνουν [16] κυρίως τον αριθμό των επιπέδων ενός ANN και τον αριθμό των μονάδων που έχει κάθε επίπεδο. Επιπλέον, μπορεί επίσης να περιλαμβάνει τον τύπο της συνάρτησης ενεργοποίησης και τον αλγόριθμο δεδομένων εκπαίδευσης. Σε πολλές περιπτώσεις, οι υπερπαράμετροι ορίζονται με βάση την εξειδίκευση σε έναν τομέα προβλημάτων και την χειροκίνητη προσαρμογή.

Ένας μεγάλος αριθμός επιπέδων και κόμβων μπορεί να αυξήσει την χωρητικότητα και την απομνημόνευση του μοντέλου, αλλά μπορεί να μειώσει τη δυνατότητα γενίκευσης. Από την άλλη πλευρά, ένας μικρός αριθμός επιπέδων και μονάδων μειώνει τις μαθησιακές ικανότητες του μοντέλου. Η απόφαση μεταξύ του μεγάλου ANN που υπερκαλύπτει τα δεδομένα και του μικρού ANN που κανει underfit είναι γνωστό ως αντιστάθμιση διακύμανσης - απόκλισης. Σε κάθε περίπτωση δεν είναι δυνατή η εξαντλητική αναζήτηση όλων των πιθανών τοπολογιών των ANN, οπότε καταλήγουμε σε μια ευριστική μέθοδο.

Οι ευριστικές τεχνικές βελτιστοποίησης μπορούν να κάνουν αυτόματη και συστηματική αναζήτηση στον πιθανό χώρο αρχιτεκτονικών με χαμηλό χρόνο εκτέλεσης. Η βιβλιογραφία περιλαμβάνει πολλές τεχνικές που χωρίζονται σε δύο κύριες κατηγορίες, τους αλγόριθμους εξέλιξης και τη βελτιστοποίηση Bayesian.

3. TensorFlow

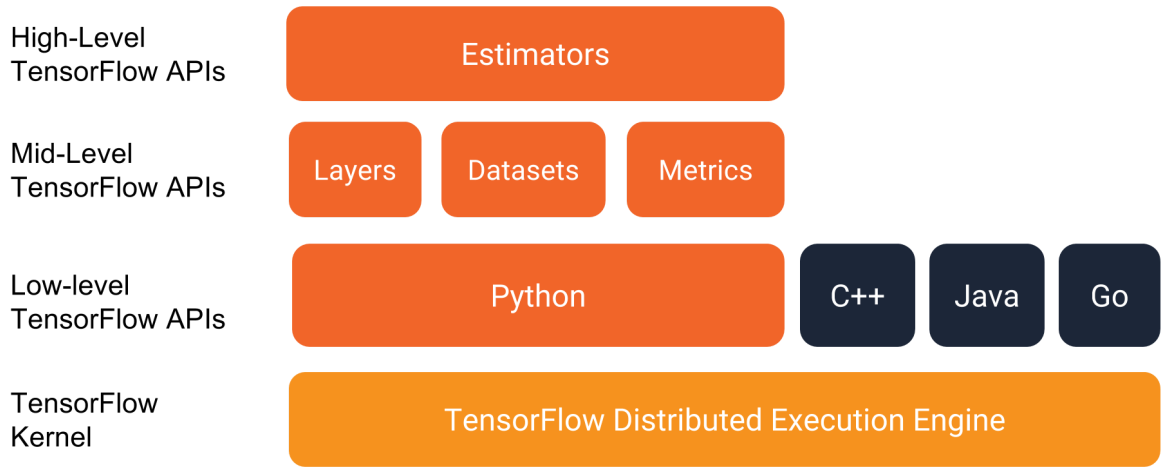
Το TensorFlow [1] είναι μία βιβλιοθήκη ανοικτού λογισμικού για προγραμματισμό ροής δεδομένων (dataflow programming) που χρησιμοποιείται ευρέως στον τομέα της μηχανικής μάθησης. Αναπτύσσεται από την Google Brain, και έχει γραφτεί κατα κύριο λόγο σε Python και C++.

3.1. Keras

Το TensorFlow προσφέρεται για προγραμματισμό τόσο σε χαμηλό όσο και σε υψηλό επίπεδο μέσω των API που διαθέτει. Ένα τέτοιο υποστηριζόμενο API είναι το Keras, το οποίο επιτρέπει την μοντελοποίηση δικτύων σε υψηλό επίπεδο, μέσα σε ελάχιστες γραμμές κώδικα, προσφέροντας παραμετροποίηση μεταξύ άλλων στο επίπεδο της συνάρτησης ενεργοποίησης, της αρχικοποίησης προκατάληψης (bias), και του αριθμού νευρώνων κάθε επιπέδου. Αναλαμβάνει επίσης τόσο την εκπαίδευση του δικτύου, μέσω συναρτήσεων βελτιστοποίησης και απώλειας όσο και την αξιολόγησή του μέσω κάποιας μετρικής. Ταυτόχρονα, με μεγαλύτερη παραμετροποίηση σε χαμηλότερο επίπεδο μπορεί κανείς μέσω του Keras να δημιουργήσει προσαρμοσμένα στις ανάγκες του μοντέλα δικτύων με ξεχωριστά χαρακτηριστικά ανά επίπεδο νευρώνων. Δίνεται η δυνατότητα αποθήκευσης και επαναφοράς τόσο ολόκληρου του μοντέλου, όσο και ξεχωριστά είτε της μορφολογίας του είτε των βαρών του, ενώ μπορεί να "τρέξει" σε πολλαπλές GPU και CPU για την παραλληλοποίηση και ταχύτερη εκτέλεση του κώδικα.

3.2. Estimator

Ένα άλλο high level API που προσφέρει το TensorFlow για την απλοποίηση προγραμματισμού μηχανικής μάθησης είναι το Estimator. Οι estimators αναλαμβάνουν να εκτελέσουν εκπαίδευση, εκτίμηση, και πρόβλεψη στο δίκτυό μας, και έχουν δημιουργηθεί πάνω στο Keras, κάτι που απλοποιεί κατα πολύ την παραμετροποίησή τους.

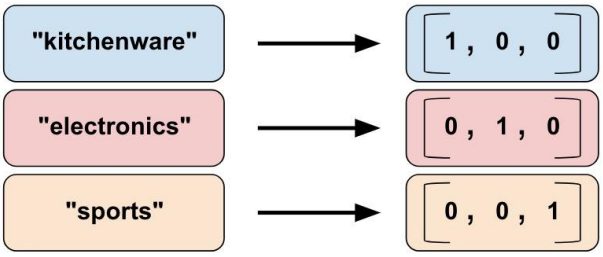


Σχήμα 3.1 Επίπεδα των API του TensorFlow

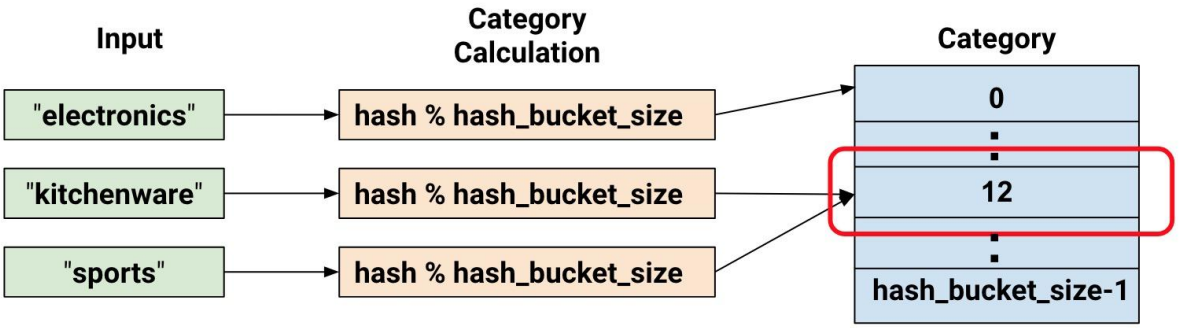
Τα πλεονεκτήματα χρήσης των εκτιμητών είναι η φορητότητά τους (μπορούν να χρησιμοποιηθούν τοπικά ή και σε περιβάλλον απομακρυσμένων σέρβερ, και να τρέξουν σε CPU, GPU ή και TPU (Tensor Processive Unit) χωρίς να χρειαστεί να αλλάξει ή να επαναπρογραμματιστεί το μοντέλο), η ευκολία χρήσης τους λόγω της εργασίας σε υψηλό επίπεδο, καθώς και το ότι οι estimators δημιουργούν αυτόματα τον γράφο του δικτύου. Η χρήση των προκαθορισμένων estimators περιλαμβάνει:

-Συγγραφή συνάρτησης εισαγωγής δεδομένων
-Καθορισμό των στηλών χαρακτηριστικών (feature columns)
-Ενεργοποίηση του estimator της επιλογής μας
-Κλήση συνάρτησης εκπαίδευσης/εκτίμησης

Η εισαγωγή δεδομένων για τη δημιουργία των Datasets γίνεται εύκολα μέσω CSV αρχείων, ενώ όσον αφορά τις στήλες χαρακτηριστικών, το TensorFlow υποστηρίζει δύο βασικές κατηγορίες:
Dense και Categorical Columns. Στις Dense στήλες συναντάμε αριθμητικά συνήθως χαρακτηριστικά, ενω για τις περιπτώσεις όπου θέλουμε να χρησιμοποιήσουμε κατηγορίες που δεν απάγονται άμεσα σε αριθμούς, χρησιμοποιούμε Categorical Columns, όπου κάνουμε αντιστοίχιση (mapping) των κατηγοριών μας σε λίστες αριθμών (είτε με άμεση αντιστοίχιση, είτε μέσω hashing)



Σχήμα 3.2 Παράδειγμα αντιστοίχισης Categorical Column



Σχήμα 3.3 Παράδειγμα Hashed Column

Μία ειδική κατηγορία στηλών είναι οι ενσωματώσεις (embeddings). Οι ενσωματώσεις είναι η αντιστοίχιση διακριτών αντικειμένων (πχ λέξεις) σε διανύσματα πραγματικών αριθμών. Ακολουθεί ως παράδειγμα μια ενσωμάτωση 300 διαστάσεων ορισμένων λέξεων.

```
blue:  (0.01359, 0.00075997, 0.24608, ..., -0.2524, 1.0048, 0.06259)
blues:  (0.01396, 0.11887, -0.48963, ..., 0.033483, -0.10007, 0.1158)
orange:  (-0.24776, -0.12359, 0.20986, ..., 0.079717, 0.23865, -0.014213)
oranges:  (-0.35609, 0.21854, 0.080944, ..., -0.35413, 0.38511, -0.070976)
```

Σχήμα 3.4 Διανυσματικές Αναπαραστάσεις

Οι μεμονωμένες διαστάσεις δεν εχουν κάποιο συγκεκριμένο νόημα στο παραπάνω σχήμα. Το συνολικό ωστόσο πρότυπο της θέσης και της απόστασης μεταξύ των διανυσμάτων αυτών, είναι που τις κατατάσσει ως περισσότερο ή λιγότερο σχετικές με άλλες λέξεις του λεξικού αυτού. Η μηχανική μάθηση λειτουργεί αποδοτικά σε πυκνά διανύσματα. Επομένως η ενσωμάτωση αποτελεί έναν πολύ καλό τρόπο απόδοσης τέτοιων εννοιών που δεν μπορούν να προσδιοριστούν αριθμητικά. Ακόμα και ως έξοδος όμως αποτελούν σημαντικό τρόπο αναπαράστασης, καθώς μπορούν μέσω διανυσματικών μετρικών (απόσταση, γωνία) να αποδίδουν ομοιότητα ή μη μεταξύ των εννοιών. Οι κοντινότερες έννοιες στο παραπάνω παράδειγμα, με τις αντίστοιχες γωνίες που σχηματίζουν δίνονται στο σχήμα 3.5.

```
blue:  (red, 47.6°), (yellow, 51.9°), (purple, 52.4°)
blues:  (jazz, 53.3°), (folk, 59.1°), (bluegrass, 60.6°)
orange:  (yellow, 53.5°), (colored, 58.0°), (bright, 59.9°)
oranges:  (apples, 45.3°), (lemons, 48.3°), (mangoes, 50.4°)
```

Σχήμα 3.5 Αναπαράσταση εννοιών

Βλέπουμε λοιπόν ότι τα πορτοκάλια έχουν μεγαλύτερη συσχέτιση με τα μήλα από ότι με τα μάνγκο (κάτι που αντικατοπτρίζεται από τη διαφορά στις γωνίες που σχηματίζονται). Το TensorFlow διαθέτει συναρτήσεις που βοηθούν τόσο στην δημιουργία, όσο και στην απεικόνιση των embeddings.
Οι estimators αυτόματα αποθηκεύουν checkpoints σε τακτά χρονικά διαστήματα, και μετά από κάθε εκπαίδευση, ενώ διατηρούν ανα πάσα στιγμή μόνο τα 5 πλέον πρόσφατα checkpoint διαθέσιμα. Τα checkpoints είναι εκδόσεις του υπο εκπαίδευση μοντέλου, και είναι ωστόσο εξαρτώμενα από τον κώδικα που δημιούργησε το μοντέλο αυτό. Οποιαδήποτε αλλαγή στη δομή του δικτύου θα προκαλέσει ασυμβατότητα με τα παλαιότερα δημιουργημένα checkpoint. Παράλληλα αποθηκεύονται και αρχεία γεγονότων (event files) τα οποία περιλαμβάνουν πληροφορία για χρήση από το TensorBoard, τη μηχανή γραφικής αναπαραστασης του TensorFlow. Τόσο η συχνότητα και η διεύθυνση αποθήκευσης, όσο και ο αριθμός των αποθηκευμένων checkpoints που βρίσκονται ανα πάσα στιγμή στη μνήμη είναι εύκολα παραμετροποιήσιμα από τον χρήστη.
Πέρα από τους προκαθορισμένους estimators που παρέχονται στο API, παρέχεται η δυνατότητα να δημιουργήσει ο χρήστης τον δικό του estimator, ωστε να παραμετροποιήσει σε ακόμα μεγαλύτερο βαθμό το δίκτυό του. Σε αυτήν την περίπτωση, καλείται να δημιουργήσει τη συνάρτηση μοντέλου, να καθορίσει δηλαδή την αρχιτεκτονική του μοντέλου (επίπεδο εισόδου, κρυφά επίπεδα, επίπεδο εξόδου), και να προσδιορίσει τις απαραίτητες συναρτήσεις για την πρόβλεψη, την εκτίμηση και την εκπαίδευσή του.
Ένα βασικό χαρακτηριστικό του TensorFlow είναι πως μπορεί να λειτουργεί παράλληλα σε επεξεργαστικές μονάδες GPU, CPU, και Cloud TPU, δίνοντας στον χρήστη τη δυνατότητα παραμετροποίησης και παραλληλοποίησης του κώδικα του. Η παραμετροποίηση αφορά την επιλογή συγκεκριμένης συσκευής για την εκτέλεση μεμονωμένων τμημάτων κώδικα, καθώς και τον ποσοστιαίο έλεγχο δεσμευμένης μνήμης για την εκτέλεση του.

3.3. Χαμηλού επιπέδου APIs

Εκτός από τα high level APIs, το TensorFlow επιτρέπει στον χρήστη να δουλέψει σε χαμηλό επίπεδο (TensorFlow Core), δίνοντας του μεγαλύτερο έλεγχο σε πιο θεμελιώδεις τομείς, όπως το πρόγραμμα(Graph), και η εκτέλεση (runtime session). Η εργασία σε χαμηλότερο επίπεδο διευκολύνει το debugging του κώδικα, και βελτιώνει την κατανόηση του χρήστη στις εσωτερικές λειτουργίες των APIs υψηλού επιπέδου.
Ο χρήστης εδώ καλείται αρχικά να δημιουργήσει και στη συνέχεια να θέσει σε λειτουργία τον υπολογιστικό γράφο του δικτύου του. Ένας υπολογιστικός γράφος, είναι μια σειρά λειτουργιών προσαρμοσμένες σε έναν γράφο που αποτελείται από τα εξής αντικείμενα:

-Λειτουργίες: Οι κόμβοι του γράφου, αφορούν υπολογισμούς που εκτελούνται σε και παράγουν τανυστές.
-Τανυστές (Tensors): Οι ακμές του γράφου, αναπαριστούν τα δεδομένα που διατρέχουν το γράφο.

Η λειτουργία του γράφου αυτού πραγματοποιείται από τη συνεδρία (session) αφού ο χρήστης προσδιορίσει αναλυτικά το dataset, την αρχιτεκτονική του δικτύου, τις συναρτήσεις βελτιστοποίησης, λάθους και πρόβλεψης, και αρχικοποιήσει το session αυτό. Τα δεδομένα που χρησιμοποιεί το δίκτυό είναι οι τανυστές (tensors), που αποτελούν γενίκευση των διανυσμάτων και των πινάκων σε περισσότερες πιθανώς διαστάσεις. Κάθε τανυστής χαρακτηρίζεται από έναν τύπο δεδομένων και ένα "σχήμα". Εξαιρώντας τον τανυστή-μεταβλητή (μία ειδική περίπτωση τανυστών) κάθε τανυστής έχει μοναδική τιμή σε δεδομένη εκτέλεση. Ωστόσο, ο υπολογισμός δις του ίδιου τανυστή μπορεί να δώσει διαφορετικές τιμές (πχ σε περίπτωση γεννήτριας τυχαίου αριθμού).
Ο αριθμός διαστάσεων των τανυστών αποτελεί τον βαθμό (rank) τους. Βαθμός 0 αντιπροσωπεύει βαθμωτές τιμές, βαθμός 1 διανύσματα, βαθμός 2 πίνακες κοκ.
Το σχήμα των τανυστών αναφέρεται στο μέγεθος της κάθε διάστασης.

Ως παράδειγμα, ο τανυστής
[[2.4, 5.1], [3.3, 7.9], [8.5, 6.1]]
είναι τανυστής βαθμού 2 και σχήματος [3, 2].

Ο τύπος δεδομένων που περιέχεται σε έναν τανυστή μπορεί να είναι προσημασμένος ή μη ακέραιος, floating-point, boolean, μιγαδικός αριθμός μεταξύ άλλων, 8-64bit ακρίβειας.

Ο γράφος περιέχει δύο είδη πληροφοριών: την δομή του και τις συλλογές (collections). Το πρώτο αφορά όλους τους κόμβους και τις ακμές, και τον τρόπο που συνδέονται, αλλά όχι τον τρόπο χρήσης τους. Η συλλογή περιλαμβάνει την αντιστοίχιση αντικειμένων με κλειδιά, ωστε δεδομένου ενός κλειδιού να είναι δυνατή η πρόσβαση σε όλα τα συσχετιζόμενα αντικείμενα.

Η κλάση Saver αναλαμβάνει να πραγματοποιήσει την αποθήκευση και ανάκτηση πληροφοριών, προσφέροντας τη δυνατότητα αποθήκευσης συγκεκριμένων ή και όλων των μεταβλητών του δικτύου, την ανάκτησή τους και τη χρήση των checkpoints για την ανάγνωση των αποθηκευμένων στοιχείων. Δύναται επίσης ο χρήστης να αποθηκεύσει ολόκληρο το μοντέλο του δικτύου του και να το επαναφέρει σε ανακτήσιμη μορφή ανεξάρτητη γλώσσας, που περιλαμβάνει τις μεταβλητές, το γράφο και τα μεταδεδομένα του. Τα αποθηκευμένα μοντέλα είναι συμβατά και με το Estimator API για εργασία σε υψηλότερο επίπεδο, ενώ μπορούν να τρέξουν σε εξυπηρετητές (servers) απομακρυσμένα.

3.4. Βοηθητικά εργαλεία

Ακόμη, το TensorFlow διαθέτει εργαλεία για τη διευκόλυνση της εκπαίδευσης δικτύων, όπως το Eager Execution, το TensorBoard και το TensorFlow Debugger.

Eager Execution
Το Eager Execution αποτελεί ένα περιβάλλον προστακτικού προγραμματισμού που αποτιμά άμεσα τις λειτουργίες παρακάμπτοντας την δημιουργία γράφου. Διαθέτει μια φιλική διεπαφή που χρησιμοποιεί δομές της Python, ευκολότερο debugging, και φυσική ροή ελέγχου.

TensorBoard
Το TensorBoard είναι ένα εργαλείο γραφικής αναπαράστασης των μοντέλων και των αποτελεσμάτων τους μέσα στο TensorFlow. Μπορεί να παρασκευάσει διαγράμματα (plots), ιστογράμματα, ή ακόμα και τον γράφο του δικτύου και αποσκοπεί στην καλύτερη κατανόηση του μοντέλου από τον

προγραμματιστή για πιο επιτυχημένο debugging, καθώς και στην εκτύπωση των αποτελεσμάτων της εκπαίδευσης για άμεση εποπτεία.

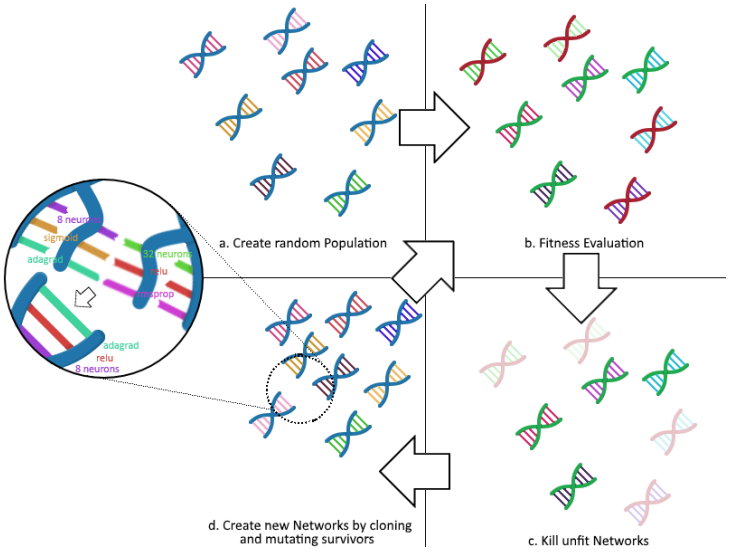
TensorFlow Debugger

Τέλος, το TensorFlow Debugger, είναι το εργαλείο που προσφέρεται για debugging, και διατίθεται τόσο σε μορφή γραμμης εντολών, όσο πλέον και σε γραφικό περιβάλλον. Είναι συμβατό τόσο σε χαμηλό επίπεδο, όσο και με τα high level APIs (Keras, Estimators).

4. Γενετικοί αλγόριθμοι για εύρεση υπερπαραμέτρων

Ένας γενετικός αλγόριθμος [17] μπορεί να χρησιμοποιηθεί για την αυτοματοποιημένη επιλογή των υπερ παραμέτρων ενός ANN (Hypertuning). Τα κύρια βήματα των γενετικών αλγορίθμων συνοψίζονται ως εξής. Πρώτον, επιλέγουμε ένα μέγεθος πληθυσμού (N) και δημιουργούμε τον εν λόγω πληθυσμό του ANN με τυχαίες υπερπαραμέτρους. Δεύτερον, επιλέγουμε πολλές γενιές για τον αλγόριθμο. Επαναλαμβάνουμε τις γενιές ακολουθώντας τα ακόλουθα βήματα: i) Εκπαιδεύουμε και αξιολογούμε ολόκληρο τον πληθυσμό και, στη συνέχεια, ταξινομούμε με κάποια μετρικη. ii) Τα (B) καλύτερα ANN προχωράνε στην επόμενη γενιά μαζί με κάποια τυχαία επιλεγμένη διαταραχή (R), ενώ προσέχουμε να αποφύγουμε τη σύγκλιση πολύ γρήγορα. iii) Δημιουργούμε τα υπόλοιπα (NRB) μέλη του πληθυσμού επιλέγοντας τις υπερ-παραμέτρους των νέων ANN επιλέγοντας από τις υπερ-παραμέτρους δύο γονικών ANN, που επιλέχθηκαν τυχαία * από (BUR) με την πιθανότητα τυχαίας αλλαγής HP σε τελείως διαφορετική τιμή (μετάλλαξη).

Οι αποφάσεις των υπερπαραμέτρων περιλαμβάνουν κυρίως τον αριθμό των επιπέδων σε ένα ANN και τον αριθμό των μονάδων σε κάθε επίπεδο. Επιπλέον, μπορεί επίσης να περιλαμβάνει ο τύπος της συνάρτησης ενεργοποίησης και το ποσοστό dropout. Σε πολλές περιπτώσεις, οι υπερ-παράμετροι καθορίζονται με βάση την εξειδίκευση σε ένα τομέα και τη χειροκίνητη προσαρμογή. Ένας μεγάλος αριθμός επιπέδων και μονάδων μπορεί να αυξήσει την χωρητικότητα και την απομνημόνευση ενός μοντέλου, αλλά μπορεί επίσης να μειώσει τη γενίκευση. Ωστόσο, ένας μικρός αριθμός επιπέδων και μονάδων μειώνει τις μαθησιακές ικανότητες του μοντέλου. Η απόφαση μεταξύ ενός μεγάλου ANN που υπερβαίνει τα δεδομένα και ενός μικρού ANN είναι γνωστή ως αντιστάθμιση διακύμανσης. Μια αυτόματη, συστηματική αναζήτηση μέσα σε έναν χώρο συνδυασμών υπερπαραμέτρων μπορεί να πραγματοποιηθεί από έναν γενετικό αλγόριθμο.



Σχήμα 4.1 Γενετικός αλγόριθμος

5. Bayesian Evolution Strategy

5.1. Εξελικτική Στρατηγική Evolution Strategy

Η Εξελικτική Στρατηγική (Evolution Strategy) (ES) [18] ανήκει στην κατηγορία των εξελικτικών αλγορίθμων που βασίζονται σε πληθυσμιακές προσεγγίσεις μετα-ευρετικής βελτιστοποίησης εμπνευσμένες από τις αρχές της βιολογικής εξέλιξης. Η διατύπωση του ES βασίζεται σε διαδοχικές επαναλήψεις μετάλλαξης και επιλογής πάνω από έναν πληθυσμό υποψηφίων λύσεων. Οι υποψήφιες λύσεις, που ονομάζονται επίσης άτομα, αρχικοποιούνται σε τυχαίες θέσεις σε ένα ν-διάστατο χώρο και κινούνται προς θέσεις που ελαχιστοποιούν μια αντικειμενική συνάρτηση. Αυτές οι διαστάσεις αντιστοιχούν στις αριθμητικές υπερπαραμέτρους GRU-RNN που πρέπει να βελτιστοποιηθούν.

Για τις ανάγκες του Hypertuning, χρησιμοποιήθηκαν οι αριθμητικές υπερπαράμετροι ως ο χώρος αναζήτησης του ES και το μέσο τετραγωνικό σφάλμα (MSE) του υποψηφίου ANN ως η συνάρτηση αξιολόγησης. Σε κάθε επανάληψη ένας αριθμός ANN εκπαιδεύεται, αξιολογείται με το MSE και οι πιο ακριβείς λύσεις μεταλλάσσονται στην επόμενη επανάληψη. Η μετάλλαξη είναι μια στοχαστική διαδικασία που βασίζεται σε μια κανονική κατανομή που εισάγει παραλλαγές στα άτομα που ταιριάζουν καλύτερα σε κάθε επανάληψη. Στην αρχή η αναζήτηση για διαφορετικές υποψήφιες λύσεις είναι εντονη, κάνοντας ισχυρότερες μεταλλάξεις προς νέες περιοχές του χώρου αναζήτησης. Σε κάθε επανάληψη η εξερεύνηση μειώνεται και αυξάνεται η εκμετάλλευση των λύσεων που ταιριάζουν καλύτερα χρησιμοποιώντας μια μεταβλητή ελέγχου αυτοπροσαρμογής. Αυτό σημαίνει ότι η μετάλλαξη εισάγει ισχυρές παραλλαγές στις πρώτες επαναλήψεις και οι παραλλαγές φθίνουν καθώς η εκτέλεση του αλγορίθμου εξελίσσεται προκειμένου να συγκλίνει σε μια σχεδόν βέλτιστη αρχιτεκτονική ANN.

Οι φαινότυποι των υπερπαραμέτρων ANN κωδικοποιούνται ως αριθμητικοί γονότυποι στο ES. Η αναπαράσταση του γονότυπου περιλαμβάνει πραγματικές τιμές κλιμακούμενες από μηδέν έως το ένα για κάθε διάσταση. Μετά την αναζήτηση στις διαστάσεις του ES, οι ενημερωμένες τιμές θα αφαιρεθούν και αντιστοιχίζονται σε τιμές υπερπαραμέτρων για την κατασκευή του μοντέλου ANN και την αξιολόγησή του. Στο στάδιο επιλογής, εκτιμώνται οι μέσοι όροι των γονότυπων για τα 40% άτομα που ταιριάζουν καλύτερα. Αυτές οι μέσες τιμές στους γονότυπους αποτελούν την αρχική θέση αναζήτησης που θα μεταλλαχθεί στην επόμενη επανάληψη. Αυτή η έξυπνη αναζήτηση του ES πραγματοποιείται έως ότου δεν σημειωθεί σημαντική βελτίωση στη λειτουργία φυσικής κατάστασης.

Τα άτομα που ταιριάζουν καλύτερα είναι υπεύθυνα για την προώθηση των ποιοτικών βελτιώσεων στη διαδικασία της εξέλιξης. Η διαδικασία επιλογής είναι ντετερμινιστική με βάση την επιλογή των ατόμων με το υψηλότερο κόστος. Η εκτίμηση της μέσης θέσης είναι επίσης ντετερμινιστική με βάση τη μέση τιμή για κάθε ξεχωριστό γονίδιο. Η μετάλλαξη είναι η λειτουργία που εισάγει μικρές, τυχαίες και αμερόληπτες αλλαγές σε κάθε άτομο. Η τυχαία διακύμανση της μετάλλαξης έχει το πλεονέκτημα να δοκιμάζει νέες περιοχές στον χώρο αναζήτησης παρέχοντας την ευκαιρία στον πληθυσμό να δραπετεύσει από τα τοπικά ελάχιστα. Αλλά η μετάλλαξη φέρνει το μειονέκτημα ότι η λειτουργία της εξέλιξης μπορεί να έχει διακυμάνσεις ή να αποκλίνει προσωρινά από καλές θέσεις. Για να αντιμετωπίσουμε αυτό το πρόβλημα διατηρούμε πάντα την καλύτερη υποψήφια λύση από τις προηγούμενες επαναλήψεις και σε περίπτωση που είναι πιο ακριβείς από το καλύτερο άτομο της τελευταίας επανάληψης, θα είναι το τελικό αποτέλεσμα.

Τα μοντέλα Hypertuning DL με ES σε αντίθεση με άλλους εξελικτικούς αλγόριθμους, όπως ο γενετικός αλγόριθμος που περιγράφηκε στην προηγούμενη ενότητα, έχει το πλεονέκτημα ότι δεν συνδυάζει διαφορετικές τοπολογίες ANN που μπορεί να έχουν σημαντικές αποκλίσεις στους

φαινοτύπους τους. Αυτό συμβαίνει επειδή το crossover του γενετικού αλγορίθμου έχει τη δυσκολία ότι οι γονείς μπορεί να έχουν διαφορετικές αρχιτεκτονικές που δεν μπορούν να ενοποιηθούν στους απογόνους τους. Ένα τυπικό παράδειγμα είναι εάν ο ένας γονέας είναι LSTM-RNN 2 στρωμάτων ακολουθούμενος με 6 feedforward layer και ο δεύτερος γονέας είναι GRU-RNN 2 layer ακολουθούμενος με 4 feedforward layer. Οι φαινότυποι των LSTM και GRU δεν μπορούν να ανασυνδυαστούν ομαλά. Από την άλλη πλευρά, το ES βασίζεται μόνο στην επιλογή και τη μετάλλαξη που οδηγούν ομαλά την εξελικτική διαδικασία. Συγκεκριμένα, οι πράξεις μετάλλαξης εισάγουν παραλλαγές στους υποψηφίους που έχουν επιβιώσει παρέχοντας την ευκαιρία να δοκιμάσουν λύσεις γειτόνων που μπορεί να οδηγήσουν σε βελτιωμένη αξία φυσικής κατάστασης.

5.2. Bayesian optimisation

Η μέθοδος της Μπευζιανής βελτιστοποίησης (Bayesian Optimization) (BO) [19] χρησιμοποιείται ευρέως για την εκτίμηση των υπερπαραμέτρων σε μοντέλα ML και DL. Για αυτό ήταν μια προφανής επιλογή για τη διαδικασία αναζήτησης στον κατηγοριοποιημένο διανυσματικό χώρο, προκειμένου να βρεθούν οι πλησιέστερες προς τις βέλτιστες ονομαστικές υπερπαράμετροι των ANN. Το BO ζητά επαναληπτικά νέες παρατηρήσεις του χώρου αναζήτησης με μια συνάρτηση απόκτησης και εκτιμά την αντικειμενική συνάρτηση με μια συνάρτηση υποκατάστασης. Η αύξηση των παρατηρήσεων της BO δίνει την γενική (global) βέλτιστη τοποθεσία με μεγαλύτερη πιθανότητα. Όμως, πρέπει να λάβουμε υπόψη ότι ο αριθμός των παρατηρήσεων είναι πεπερασμένος και υπολογιστικά ακριβός. Οπότε η διαδικασία αναζήτησης πρέπει να επιλέξει σημεία που μεγιστοποιούν την πιθανότητα εύρεσης ενός νέου βέλτιστου μετά από μια εξερεύνηση.
Η συνάρτηση αντικατάστασης προσεγγίζει την αντικειμενική συνάρτηση και ενημερώνεται κάθε φορά που η αντικειμενική συνάρτηση αξιολογείται στα νέα υποψήφια σημεία. Το BO μοντελοποιείται ως επί το πλείστον με μια παλινδρόμηση Gaussian Process που καθορίζεται από μια μέση συνάρτηση και μια συνάρτηση συνδιακύμανσης στα αξιολογημένα σημεία. Επομένως, με τις προσεγγιστικές διανομές Gauss μπορούμε να υπολογίσουμε περιθωριες συχνοτητες και όρους σε κλειστή μορφή.
Η συνάρτηση απόκτησης αποφασίζει πού θα γίνει η δειγματοληψία στη συνέχεια κατά την επαναληπτική διαδικασία του BO, βρίσκοντας τα σημεία που μεγιστοποιούν την αναμενόμενη βελτίωση. Η αναμενόμενη βελτίωση είναι συνάρτηση δύο συστατικών. Το πρώτο, εκτιμά τις περιοχές που η συνάρτηση αντικατάστασης έχει βέλτιστα σημεία και το δεύτερο εκτιμά τις περιοχές με υψηλή αβεβαιότητα πρόβλεψης που δεν έχουν διερευνηθεί ακόμα αρκετά.

5.3. Υβριδική στρατηγική εξέλιξης με τη μέθοδο βελτιστοποίησης Bayesian

Η βελτιστοποίηση υπερπαραμέτρων για ένα ANN είναι μια δύσκολη πρόκληση καθώς περιλαμβάνει τις σημαντικές αρχιτεκτονικές αποφάσεις για μια σχεδόν βέλτιστη τοπολογία. Η Υβριδική στρατηγική εξέλιξης με τη μέθοδο Baysian βελτιστοποίησης (Hybrid Bayesian Evolutionary Strategy) (HBES) αποτελεί μια καινοτόμο, ολιστική και ενοποιημένη προσέγγιση που συνδυάζει τις μεθοδολογιών ES και BO. Το ES είναι υπεύθυνο για την εξέλιξη ενός πληθυσμού ANN βάσει των αριθμητικών υπερπαραμέτρων τους και κάθε μεμονωμένο ANN εκτιμά τις ονομαστικές υπερπαραμέτρους με το BO όπως περιγράφεται στον Αλγόριθμο 1.

Αλγόριθμος 1: Αλγόριθμος Υβριδικού Μπεϋζιανού και Εξελικτικής Στρατηγικής

Βήμα 1: Αρχικοποίηση της Εξελικτικής Στρατηγικής

Θέτουμε το αρχικό σημείο αναζήτησης του αλγορίθμου. Συνήθως επιλέγουμε το [0.5, 0.5, …, 0.5], έχοντας μεταφέρει τις πιθανές τιμές υπερπαραμέτρων στο διάστημα [0,1]

Βήμα 2: Για κάθε δίκτυο που ανήκει στον πληθυσμό μας:

i) Προσθέτουμε τυχαίο θόρυβο στο σημείο αναζήτησης

ii) Αναιρούμε την κλίμακα του [0,1] που είχαμε επιβάλει στις τιμές των υπερπαραμέτρων ώστε να έχουμε τις πραγματικές τιμές που θα χρησιμοποιήσει το δίκτυο μας

iii) Μπεϋζιανή Βελτιστοποίηση με Γκαουσιανές Διαδικασίες

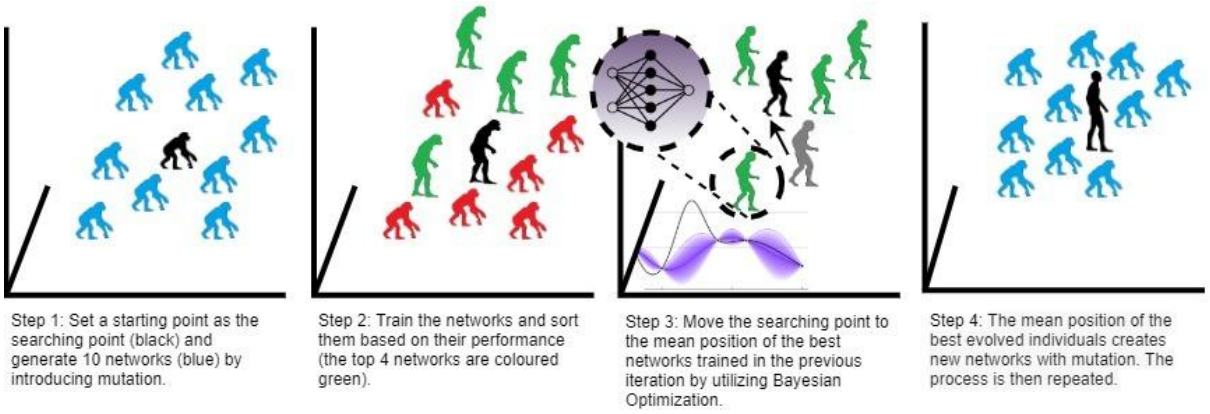
1) Εφαρμογή πρότερης Γκαουσιανής Διαδικασίας (Gaussian Process prior) στην f
2) Παρατηρούμε την f σε n_0 σημεία σύμφωνα με έναν αρχικό πειραματικό σχεδιασμό
3) Αρχικοποιούμε $n=n_0$
4) Επαναλαμβάνουμε όσο $n \leq N$:
    a) Ενημερώνουμε την μεταγενέστερη κατανομή πιθανότητας (posterior probability distribution) στην f χρησιμοποιώντας όλα τα διαθέσιμα δεδομένα
    b) Ορίζουμε το x_n ως τη μεγιστοποίηση της συνάρτησης απόκτησης (acquisition function) στο x
    c) Παρατηρούμε το $y_n=f(x_n,x_i(t+1),v_i(t+1))$
    d) Θέτουμε $n \leftarrow n+1$

Βήμα 3: Ταξινομούμε τα αποτελέσματα και την αντίστοιχη λίστα υπερπαραμέτρων

Βήμα 4: Βρίσκουμε το νέο σημείο αναζήτησης του αλγορίθμου, υπολογίζοντας το μέσο όρο των συντεταγμένων των κορυφαίων k δικτύων

Βήμα 5: Πηγαίνουμε στο βήμα 2 μέχρι να ολοκληρωθεί ο απαιτούμενος αριθμός εποχών/επαναλήψεων

Οι αριθμητικές υπερ-παράμετροι είναι ο αριθμός των επαναλαμβανόμενων layer και των feedforward επιπέδων, ο αριθμός των νευρώνων για κάθε layer, οι εποχές (epochs) εκπαίδευσης, το μέγεθος των παρτίδων (batches), το ποσοστό της εξομάλυνσης και ο ρυθμός εκμάθησης. Οι ονομαστικές υπερπαράμετροι είναι ο τύπος του RNN, οι λειτουργίες ενεργοποίησης και οι βελτιστοποιητές. Η αποκτηθείσα γνώση των ονομαστικών υπερπαραμέτρων είναι καθολική μέσω του πληθυσμού και ενημερώνεται από όλα τα άτομα κατά τη διάρκεια των γενεών. Ο απώτερος στόχος του HBES είναι μέσω της διαδικασίας εξέλιξης του Μπεηζιανού μοντέλου να καταλήξει σε ένα σχεδόν βέλτιστο και εκπαιδευμένο ANN που μπορεί να προβλέπει έγκαιρα και με ακρίβεια τη μετρική εξόδου στα επόμενα χρονικά βήματα. Τα τέσσερα κύρια βήματα HBES για μία επανάληψη της εξέλιξης των ατόμων απεικονίζονται στο Σχήμα 5.1



Σχήμα 5.1 Υβριδικός Μπευζιανός εξελικτικός αλγόριθμος

6. Πειραματική αξιολόγηση με πρόβλεψη τροχιάς σε κινούμενα πλοία

Η πρόβλεψη της επόμενης θέσης για ένα κινούμενο αντικείμενο [20] είναι ένα θέμα μεγάλου ενδιαφέροντος σε πολλούς τομείς όπως η μεταφορά, η ασφάλεια, η μετανάστευση δεδομένων και οι έξυπνες πόλεις. Μια ανάλυση του τρόπου με τον οποίο ένα αντικείμενο κινήθηκε σε προηγούμενα χρονικά βήματα μπορεί να χρησιμοποιηθεί για τη δημιουργία ενός μοντέλου πρόβλεψης για τις επόμενες θέσεις. Πολλοί τύποι μοντέλων πρόβλεψης που βασίζονται σε παραδοσιακούς αλγόριθμους μηχανικής μάθησης έχουν χρησιμοποιηθεί στο παρελθόν με επιτυχία και σε αυτή την πειραματική αξιολόγηση θα δοκιμάσουμε μια προσέγγιση με νευρωνικά δίκτυα και έναν γενετικό αλγόριθμο για hypertuning.

Η έλευση της εποχής του Deep Learning (DL) μας έκανε να ξανασκεφτούμε, να επανασχεδιάσουμε και να πραγματοποιήσουμε έρευνα σε πολλές εφαρμογές πρόβλεψης που χρησιμοποιούν παραδοσιακές τεχνικές μηχανικής μάθησης. Επιπλέον, η ακολουθία θέσεων με την πάροδο του χρόνου ενός κινούμενου αντικειμένου μπορεί να μοντελοποιηθεί ως πρόβλημα χρονοσειρών. Η ιδέα μιας προσέγγισης με τεχνητά νευρωνικά δίκτυα και η εφαρμογή των χρονοσειρών στο πλαίσιο της πρόβλεψης της επόμενης θέσης μάς έκανε να δοκιμάσουμε μια λύση με LSTM νευρωνικά δίκτυα.

Ένας νευρώνας LSTM σε αντίθεση με τους κανονικούς νευρώνες έχει μονάδες με μνήμη που αντιλαμβάνονται τη σειρά των παρατηρήσεων και μπορούν να μάθουν χρονικές εξαρτήσεις. Το LSTM μπορεί να έχει γενικευμένες και τοπικές ιδιότητες εξομάλυνσης εάν εκπαιδεύονται με προσοχή στη λεπτομέρεια. Πολλοί ειδικοί της DL καταλήγουν στο συμπέρασμα ότι η ακρίβεια ενός μοντέλου DL έχει τρεις βασικούς παράγοντες: Η ποιότητα των δεδομένων που χρησιμοποιείται, η μεθοδολογία εκπαίδευσης του και ο αρχιτεκτονικός σχεδιασμός των ANN. Για να καταλήξουμε σε ένα βέλτιστο ή κοντά στον βέλτιστο αρχιτεκτονικό σχεδιασμό για ένα δεδομένο σύνολο δεδομένων μπορούμε να χρησιμοποιήσουμε έναν γενετικό αλγόριθμο.

Ένας ακόμη τομέας DL που αξίζει να αναφέρουμε είναι η χρήση τεχνικών μεταφοράς γνώσης. Πολλές φορές έχουμε καλά εκπαιδευμένα και υψηλής ακρίβειας μοντέλα για ένα σύνολο περιπτώσεων χρήσης και θέλουμε να τα χρησιμοποιήσουμε για διαφορετικές αλλά παρόμοιες περιπτώσεις χρήσης. Σε αυτό το πλαίσιο εξετάζουμε την υπόθεση να κάνουμε μια καλή εκπαίδευση ενός μοντέλου σε ορισμένες τροχιές και να το αξιοποιήσουμε για να κάνουμε μια σύντομη αλλά υψηλής ακρίβειας εκπαίδευση σε επόμενες νέες τροχιές.

Αξιολογήσαμε το προτεινόμενο μοντέλο μας χρησιμοποιώντας πραγματικά δεδομένα από διάφορα πλοία και τροχιές. Τα αποτελέσματα μετρώνται με τη χρήση του τύπου Vincenty και της απόστασης αντί του γεωγραφικού πλάτους και μήκους. Τα αποτελέσματα δείχνουν ότι το προτεινόμενο μοντέλο είναι πρακτικό, αποτελεσματικό και ξεπερνά τις σύγχρονες μεθόδους.

Οι σημαντικότερες συνεισφορές αυτής της μελέτης είναι οι εξής:

• Σχεδιάζουμε και υλοποιούμε ένα ακριβές μοντέλο για την πρόβλεψη της επόμενης θέσης των σκαφών.

• Δείχνουμε θεωρητικά και πρακτικά την καταλληλότητα του LSTM ANN για γεω-χωρικά δεδομένα.

• Εκτελούμε πειράματα για το πώς να βρούμε σχεδόν βέλτιστες αρχιτεκτονικές με LSTM layers χρησιμοποιώντας έναν γενετικό αλγόριθμο.

• Εξετάζουμε πώς να βελτιώσουμε τη διαδικασία εκπαίδευσης χρησιμοποιώντας ένα αποθετήριο μοντέλων ANN με μια προσέγγιση μεταφοράς γνώσης.

6.1. Προσέγγιση με Χρονοσειρές και Νευρωνικά Δίκτυα

Οι χωροχρονικές παρατηρήσεις κινούμενων αντικειμένων έχουν μια δομή ακολουθίας δεδομένων που μπορεί να μοντελοποιηθεί ως χρονοσειρές. Οι συνεχείς παρατηρήσεις συσχετίζονται επειδή κάθε θέση εξαρτάται από τις προηγούμενες θέσεις. Οι διαδοχικές παρατηρήσεις σχετίζονται στενότερα από τις παρατηρήσεις που βρίσκονται μακριά. Επιπλέον, η διαδικασία κίνησης των σωμάτων ακολουθεί συγκεκριμένους νόμους της φυσικής όπως της εγγενής αδράνειας, και των εξωτερικών δυνάμεων που αλλάζουν τη μηχανική ενέργεια του κινούμενου αντικειμένου.

Η επόμενη θέση που θα βρεθεί ένα κινούμενο αντικείμενο δεν μπορεί να προβλεφθεί απλά με τους νόμους κίνησης της φυσικής και τις παρελθούσες παρατηρήσεις. Σε πραγματικές συνθήκες, τα προβλήματα είναι πιο περίπλοκα. Για παράδειγμα, ο καπετάνιος ενός σκάφους μπορεί να αλλάξει ταχύτητα ή κατεύθυνση για να αποφύγει μια σύγκρουση. Στην περίπτωση που ένας καπετάνιος πήρε την απόφαση να πλοηγηθεί σε διαφορετική κατεύθυνση, η διαδικασία πλοήγησης ακολουθεί μια ακολουθία βημάτων που μπορούν να αναγνωριστούν με τεχνικές μηχανικής εκμάθησης. Έτσι, η νέα απόφαση του καπετάνιου θα μπορούσε να έχει ως αποτέλεσμα κάποια λάθος συμπεράσματα του μοντέλου μόνο στα πρώτα λίγα βήματα.

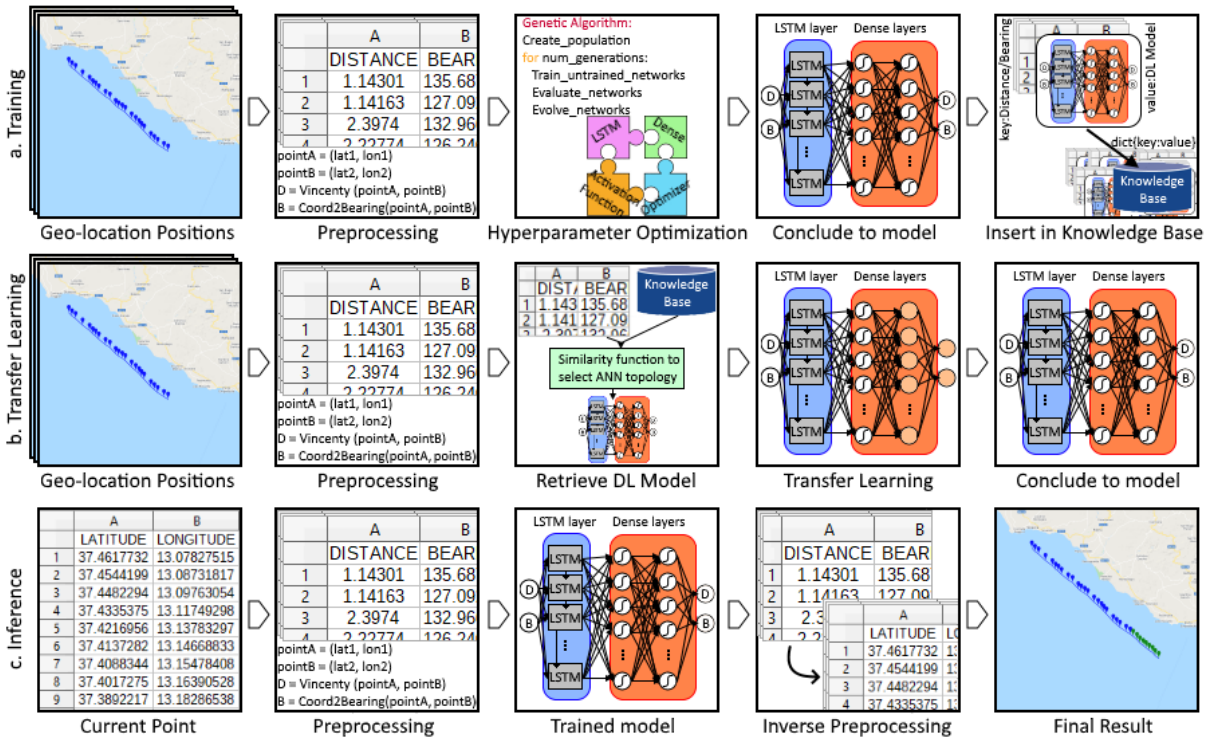
Μοντέλα που αξιοποιούν τη σειριακή εξάρτηση στις παρατηρήσεις μπορούν να ενισχύσουν την προβλεπόμενη ακρίβεια, ενώ θα πρέπει να λάβουμε υπόψη ποιες πρόσθετες τεχνικές μπορούν να συμπεράνουν τη συμπεριφορά κινησης. Οι ιστορικές παρατηρήσεις γεω-τοποθεσίας περιέχουν πολύτιμη πληροφορία για την κινηση των αντικειμένων. Η ικανότητα μάθησης και πρόβλεψης ενός μοντέλου δεν βασίζεται στην απομνημόνευση προηγούμενων περιστάσεων αλλά στη γενίκευση και την πρόβλεψη υπό νέες συνθήκες.

Σε κινούμενα αντικείμενα όπως τα πλοία οι επόμενες θέσεις γεωγραφικού πλάτους και μήκους εξαρτώνται από την προηγούμενη, την απόσταση και τα φυσικά χαρακτηριστικά. Επιπλέον, η ταχύτητα και η κατεύθυνση αλλάζουν σταδιακά με ντετερμινιστικό τρόπο που ακολουθεί κοινά μοτίβα. Τα LSTM νευρωνικά δίκτυα έχοντας μια δομή τύπου αλυσίδας, είναι σε θέση να συνδέουν πληροφορίες για μεγάλες περιόδους και να χαρτογραφούν μοτίβα κίνησης σε επόμενες θέσεις.

Λαμβάνοντας υπόψη ότι χρειαζόμαστε ένα μοντέλο ικανό να αξιοποιήσει την εξάρτηση στις παρατηρήσεις, να μάθουμε από τις κινήσεις και να γενικεύουμε σε νέες άγνωστες καταστάσεις, προτείνουμε τη χρήση νευρωνικών δικτύων με LSTM.

6.2. Προτεινόμενο Μοντέλο

Το μοντέλο πρόβλεψης της επόμενης θέσης περιλαμβάνει τρεις διαφορετικές ομοχειρίες όπως απεικονίζεται στο σχήμα 6.1 Οι δύο από αυτές, η εκπαίδευση και η μεταφορά γνώσης χρησιμοποιούνται για τη δημιουργία ενός μοντέλου πρόβλεψης, αξιοποιώντας την εμπειρία προηγούμενων μοντέλων. Η τρίτη ομοχειρία είναι η πραγματοποίηση προβλέψεων σε πραγματικό χρόνο με βάση την τρέχουσα θέση του σκάφους. Οι ομοχειρίες αποτελούνται από μια ακολουθία ξεχωριστών βημάτων επεξεργασίας. Τα βασικά στοιχεία των κόμβων επεξεργασίας περιγράφονται στις επόμενες ενότητες.



Σχήμα 6.1 Εκπαίδευση, μεταφορά εκμάθησης και ροή εργασιών συμπερασμάτων

Η ομοχειρία εκπαίδευσης (σχήμα 6.1a) λαμβάνει ως είσοδο μια δέσμη (batch) τροχιών σκαφών. Κάθε τροχιά περιέχει μια ακολουθία γεωγραφικών τοποθεσιών, δηλαδή γεωγραφικό πλάτος και μήκος όπως απεικονίζεται στον χάρτη. Οι γεωγραφικές τοποθεσίες μετατρέπονται σε ομαλοποιημένα χαρακτηριστικά απόστασης και γωνίας. Τα χαρακτηριστικά των δεδομένων εισάγονται σε έναν γενετικό αλγόριθμο που κάνει μια έξυπνη αναζήτηση σε διαφορετικές τοπολογίες ANN, προκειμένου να καταλήξει σε μια αρχιτεκτονική ANN κοντά στο βέλτιστο. Αυτή η τεχνική βελτιστοποίησης υπερπαραμέτρων είναι ένας αυτοματοποιημένος και αποτελεσματικός τρόπος δοκιμής διαφορετικών τοπολογιών ANN. Τα μοντέλα αποτελεσμάτων αποθηκεύονται σε μια βάση γνώσεων με μια δομή δεδομένων απεικόνισης η οποία έχει ως κλειδί (key) την αναπαράσταση της απόστασης και της γωνίας των τροχιών, και ως αξία (value) το αντίστοιχο εκπαιδευμένο DL μοντέλο. Αυτή η δομή δεδομένων απεικόνισης θα χρησιμοποιηθεί στην ομοχειρία εκμάθησης μεταφοράς για την εκτίμηση των εφαρμοσμένων προ-εκπαιδευμένων αρχιτεκτονικών DL για νέες τροχιές.

Στην ομοχειρία εκμάθησης μεταφοράς (σχήμα 6.1b) παίρνουμε ως είσοδο μια νέα τροχιά, εφαρμόζουμε τους ίδιους μετασχηματισμούς προεπεξεργασίας για να πάρουμε τα χαρακτηριστικά απόστασης και γωνίας. Χρησιμοποιώντας μια συνάρτηση ομοιότητας μεταξύ των χαρακτηριστικών απόστασης της νέας τροχιάς με τις τροχιές της βάσης γνώσεων, επιλέγουμε μια τοπολογία ANN. Σε αυτήν την έρευνα χρησιμοποιήσαμε την Ευκλείδεια απόσταση. Θα είχε ενδιαφερον σε μια μελλοντική εργασία να δοκιμάσουμε διαφορετικές συναρτήσεις ομοιότητας. Στο μοντέλο ANN που ανακτήθηκε εφαρμόζεται η διαδικασία μεταφοράς γνώσης που περιλαμβάνει τη διατήρηση ορισμένων επιπέδων χωρις να εκπαιδευτούν και επανεκπαίδευση των υπολειπόμενων επιπέδων με τα δεδομένα της νέας τροχιάς. Μια δεύτερη προσέγγιση είναι η διατήρηση των ίδιων υπερπαραμέτρων και επανεκπαίδευση όλων των βαρών ANN. Στην πειραματική αξιολόγηση δοκιμάζουμε και τα δύο.

Η ομοχειρία συμπερασμάτων (σχήμα 6.1c) χρησιμοποιεί το μοντέλο εξόδου του αγωγού εκμάθησης μεταφοράς. Παίρνει ως είσοδο την τρέχουσα γεωγραφική τοποθεσία των πλοίων, ακολουθεί τους

ίδιους μετασχηματισμούς προεπεξεργασίας και παρέχει μια πρόβλεψη χρησιμοποιώντας το εκπαιδευμένο μοντέλο βαθιάς μάθησης. Στην έξοδο του μοντέλου DL εφαρμόζεται ένας αντίστροφος μετασχηματισμός για να ληφθούν οι προβλέψεις γεωγραφικού πλάτους και μήκους. Αυτή η διαδικασία μπορεί να πραγματοποιηθεί σε πραγματικό χρόνο και για κάθε τρέχουσα θέση να προβλέπει την επόμενη πρόβλεψη που διαμορφώνει την τροχιά.

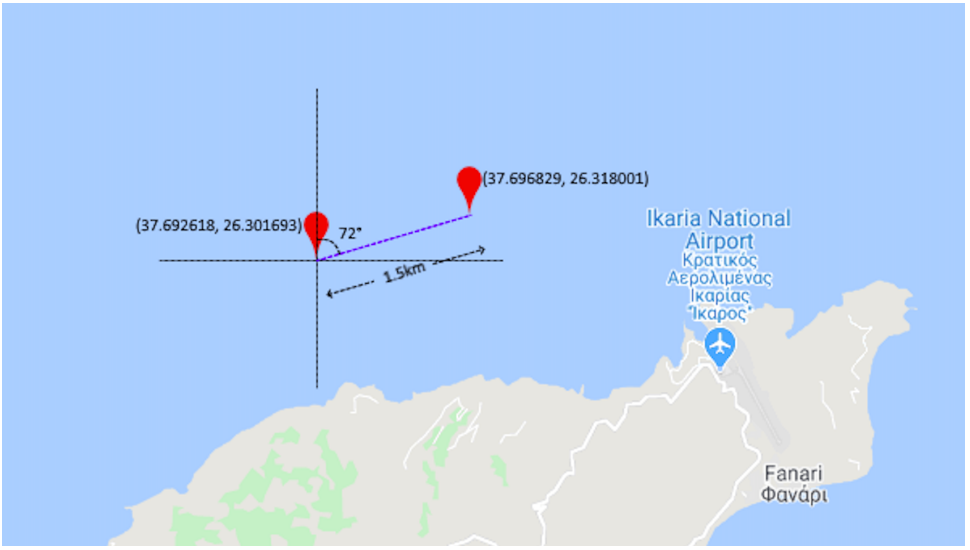
6.3. Χαρακτηριστικά δεδομένων και προεπεξεργασία

Τα χαρακτηριστικά των δεδομένων και η αναπαράσταση δεδομένων εισόδου / εξόδου παίζουν βασικό ρόλο στην ακρίβεια του μοντέλου. Δοκιμάσαμε διαφορετικούς μετασχηματισμούς δεδομένων στις δυνατότητες εισαγωγής και τις τιμές εξόδου του ANN, όπως θα δούμε στην ενότητα της πειραματικής αξιολόγησης. Η χρήση των χαρακτηριστικών της απόστασης και της γωνίας αντί γεωγραφικής θέσης με γεωγραφικό πλάτος και μήκος είναι ένας τρόπος βελτίωσης της ακρίβειας. Τα μέτρα απόστασης και γωνίας μπορούν να χρησιμοποιηθούν στην πλοήγηση αέρα, χερσαίας και θάλασσας ως διαφορετικό σύστημα από τις συντεταγμένες GPS.
Στο σχήμα 6.2 απεικονίζονται και τα δύο συστήματα. Το γεωγραφικό πλάτος και το γεωγραφικό μήκος αντιπροσωπεύουν μια απόσταση ενός σημείου από τον ισημερινό και τον πρώτο μεσημβρινό ανάλογα και μετρώνται σε μοίρες και τιμές μεταξύ (-90,90) και (-180,180). Στο σύστημα απόστασης / γωνίας, η απόσταση χρησιμοποιείται ως η απόσταση μεταξύ της τρέχουσας και της προηγούμενης γεωγραφικής θέσης του σκάφους και η γωνία είναι η απεικονιζόμενη δεξιόστροφη γωνιακή κίνηση μεταξύ των δύο θέσεων. Οι τριγωνομετρικές εξισώσεις που κάνουν τον μετασχηματισμό από το ένα σύστημα στο άλλο δίνονται στις ακόλουθες εξισώσεις.

$$a = sin^2(\Delta\varphi/2) + cos\varphi_1 \cdot cos\varphi_2 \cdot sin^2(\Delta\lambda/2)$$
$$c = 2 \cdot arctan2(\sqrt{\alpha}, \sqrt{(1-\alpha)})$$
$$Distance = R \cdot c$$

$$Bearing = arctan2(sin(\Delta\lambda) \cdot cos(\varphi_2), cos(\varphi_1) \cdot sin(\varphi_2) - sin(\varphi_1) \cdot cos(\varphi_2) \cdot cos(\Delta\lambda)$$

Όπου το R είναι η ακτίνα της γης, φ σημαίνει γεωγραφικό πλάτος και λ γεωγραφικό μήκος για τα σημεία ένα και δύο. Τα φ και λ μετατρέπονται σε ακτίνια προτού χρησιμοποιηθούν στις εξισώσεις. θ είναι το αποτέλεσμα βρίσκεται της γωνίας που βρίσκεται στο εύρος (-180,180) οπότε πρέπει να μετατραπεί σε εύρος (0,360) εκτός από τη μετατροπή σε μοίρες.



Σχ. 6.2 Μετασχηματισμός απόστασης / γωνίας

Το πλεονέκτημα της απόστασης και της γωνίας αντί των συντεταγμένων μπορεί να αιτιολογηθεί από το γεγονός ότι η πρόβλεψη της επόμενης θέσης από το νευρωνικό δίκτυο περιορίζει τον χώρο αναζήτησης με τρόπο παρόμοιο με τον τρόπο που το ανθρώπινο μυαλό αναγνωρίζει ότι υπάρχει περιορισμένη απόσταση κίνησης που μπορεί εύκολα να εκτιμηθεί είτε από τις προηγούμενες μετρήσεις είτε από την ταχύτητα και το χρονικό διάστημα μέχρι την επόμενη παρατήρηση.

Το σκάφος μπορεί να ταξιδέψει σε δεδομένο χρονικό ορίζοντα, εστιάζοντας στην εκτίμησης της επόμενης θέσης σε έναν κύκλο γύρω από την αρχική θέση. Τα χαρακτηριστικά των δεδομένων μετασχηματίζονται ξεχωριστά με κλιμάκωση μεταξύ μηδέν και ενός. Μετά την φάση συμπεράσματος, χρησιμοποιήσαμε έναν αντίστροφο μετασχηματισμό για να έχουμε τις πραγματικές τιμές απόστασης και γωνίας. Χρησιμοποιώντας την τρέχουσα θέση και την εκτιμώμενη απόσταση και γωνία μπορούμε να υπολογίσουμε την επόμενη εκτιμώμενη θέση σε όρους γεωγραφικού πλάτους, μήκους.

6.4. Διαδικασία Μάθησης

Οι παράμετροι ενός ANN υπολογίζονται χρησιμοποιώντας μια μέθοδο βελτιστοποίησης που ελαχιστοποιεί μια αντικειμενική συνάρτηση όπως περιγράψαμε στις προηγούμενες ενότητες. Η αντικειμενική συνάρτηση εκφράζει τη διαφορά μεταξύ της προβλεπόμενης εξόδου και της πραγματικής εξόδου. Οι αντικειμενικές συναρτήσεις που χρησιμοποιούνται πιο συχνά είναι το μέσο απόλυτο σφάλμα L1 και το μέσο τετράγωνο σφάλμα L2. Στην εργασία μας θέλαμε να αποφύγουμε τα μεγάλα σφάλματα, οπότε χρησιμοποιήσαμε το L2 καθώς δίνει μεγαλύτερο βάρος σε μεγάλα σφάλματα από το L1. Η βιβλιογραφία προτείνει ένα πλούσιο σύνολο τεχνικών βελτιστοποίησης. Στην εργασία μας δοκιμάσαμε το Stochastic Gradient Descent (SGD), το Root Mean Square Propagation (RMPSrop), το Adaptive Gradient Algorithm (Adagrad), την επέκτασή του Adadelta, το Adaptive Moment Estimation (Adam) και τις παραλλαγές του AdaMax και Nadam που χρησιμοποιούν την Nesterov ορμή.
Οι μέθοδοι βελτιστοποίησης συνήθως υπολογίζουν τα gradients, δηλαδή τις μερικές παραγώγους της αντικειμενικής συνάρτησης σε σχέση με τα βάρη και τα bias των παραμέτρων του ANN. Αυτές οι παράμετροι ενημερώνονται στην αντίθετη κατεύθυνση της υπολογιζόμενης κλίσης. Αυτή η διαδικασία

επαναλαμβάνεται έως ότου η αντικειμενική συνάρτηση φτάσει στο ελάχιστο. Η βιβλιογραφία αναφέρει πολλές μεθόδους ουτως ώστε να μην υπάρχει σύγκλιση σε τοπικό ελάχιστο αλλά να αναζητείται το γενικό ελάχιστο. Ένα δεύτερο σημαντικό θέμα στις τεχνικές βελτιστοποίησης είναι η ταχύτητα σύγκλισης των παραμέτρων.
Το SGD εκτελεί ενημέρωση παραμέτρων για κάθε παράδειγμα εκπαίδευσης και είναι συνήθως τεχνική γρήγορης σύγκλισης. Αλλά έχει το μειονέκτημα ότι περιπλέκει τη σύγκλιση και μπορεί να ξεπεράσει τα ελάχιστα λόγω των συχνών διακυμάνσεων. Οι RMSProp και Adam είναι προσεγγίσεις Stochastic Gradient Descent με προσαρμοστικό ποσοστό μάθησης. Το RMSProp έχει την προσέγγιση Momentum στην οποία η διαβάθμιση σε κάθε επανάληψη είναι το άθροισμα της τρέχουσας διαβάθμισης συν τις προηγούμενες διαβαθμίσης και έχει ως αποτέλεσμα να περιορίζει την ταλάντωση. Ο Adam με παρόμοιο τρόπο χρησιμοποιεί την τεχνική Momentum, αλλά βρίσκει επίσης μια αναλλοίωτη κατεύθυνση κλίσης σε αντίθεση με τις άλλες ταλαντούμενες κατευθύνσεις, με την πλοήγηση στα κρίσιμα σημεία (critical points). Το AdaMax είναι μια ειδική περίπτωση του Adam όπου η ορμή της δεύτερης τάξης αντικαθίσταται από την ορμή άπειρης τάξης.
Μια επιπλέον τεχνική στη μαθησιακή διαδικασία είναι η κανονικοποίηση. Το Dropout είναι μια επιτυχημένη τεχνική κανονικοποίησης, αλλά λόγω του ότι χρειάζεται επιπλέον επεξεργασία για να χρησιμοποιηθεί σωστά με το LSTM . Αποφασίσαμε λοιπόν να το αφήσουμε για μελλοντική δουλειά.

6.5. Αποτελέσματα και συζήτηση

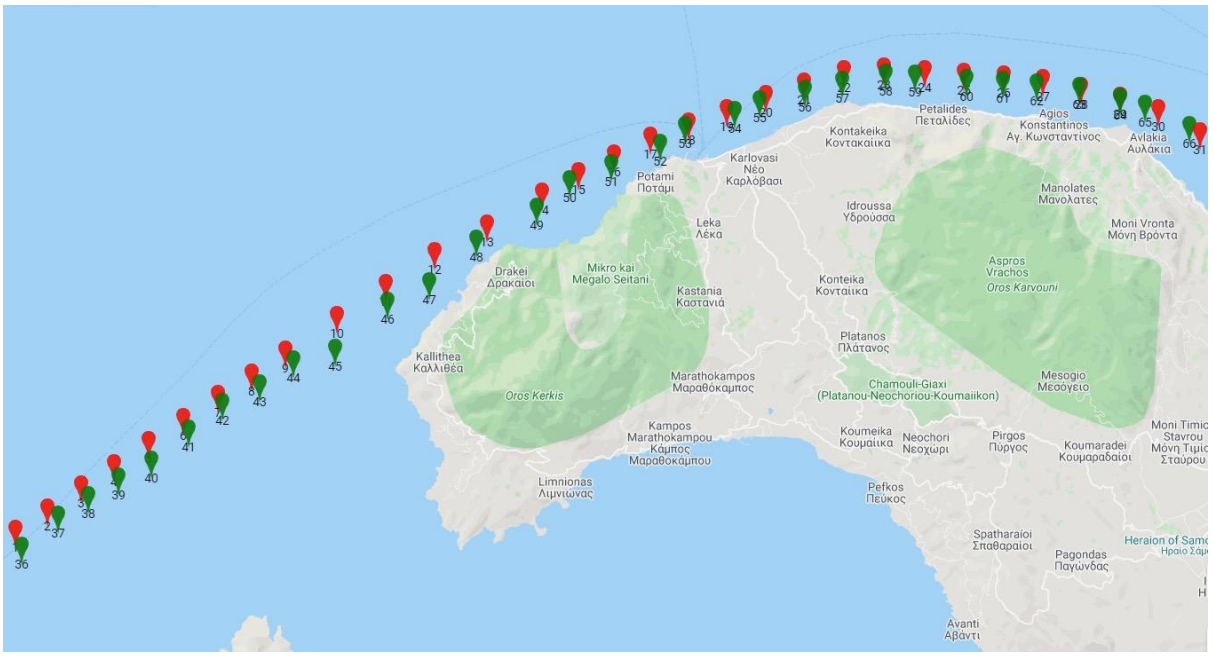
Η πρόβλεψη της επόμενης θέσης που χρησιμοποιεί το μοντέλο νευρωνικό δίκτυο με LSTM επίπεδα έχει δοκιμαστεί και αξιολογηθεί με ένα σύνολο δεδομένων χιλιάδων τροχιών από διαφορετικούς τύπους πλοίων, προκειμένου να αποδειχθεί η εφαρμοσιμότητα και η αποτελεσματικότητά του. Ακολουθώντας αυτή την προσέγγιση δοκιμάσαμε διαφορετικές παραλλαγές μέχρι να καταλήξουμε στην πιο ακριβή ρύθμιση.
Ένα παράδειγμα της πρόβλεψης τροχιάς ενός πλοίου απεικονίζεται στο σχήμα 6.3 Το μοντέλο εκπαιδεύτηκε με το πρώτο μέρος της τροχιάς και στη συνέχεια προβλέπει σε κάθε τρέχουσα θέση την επόμενη θέση που θα βρεθεί σε ένα συγκεκριμένο χρονικό βήμα. Στο σχήμα, βλέπουμε τις πραγματικές επόμενες θέσεις με πράσινο χρώμα και την πρόβλεψη με κόκκινο.
Το σύνολο δεδομένων αποτελείται από διαφορετικούς τύπους σκαφών, χρονικά βήματα και θέσεις σε μορφή γεωγραφικού πλάτους και μήκους. Για κάθε σκάφος έχουμε από τέσσερις εκατοντάδες έως έξι χιλιάδες κομμάτια θέσεων. Στο στάδιο της προεπεξεργασίας δεδομένων, αφαιρέθηκαν πολλές διπλές και μη έγκυρες τιμές λόγω των προβλημάτων επικοινωνίας του μηχανισμού θέσης. Τα σκάφη, τα μαθήματα και οι γεωγραφικές τοποθεσίες μπορούν να μοντελοποιηθούν με μέσω πινάκων NumPy τανυστών TensorFlow.

6.6. Μέτρα αξιολόγησης

Για την αξιολόγηση του προτεινόμενου μοντέλου μετρήσαμε τη διαφορά μεταξύ των πραγματικών επόμενων θέσεων από τις προβλέψεις με τον τύπο Vincenty (Vin_m) και τον χρόνο εκπαίδευσης ($Train_{sec}$) των μοντέλων. Επιπλέον, για κάθε μοντέλο μετρήσαμε την πιθανότητα να παρέχουμε τις καλύτερες προβλέψεις (Pr_{best}), το ποσοστό που έχει σφάλμα πρόβλεψης μικρότερο από 1km ($Err_{<1km}$) και λιγότερο από 0,5km ($Err_{<0.5km}$). Αυτά τα μέτρα χρησιμοποιούνται στον Πίνακα 6.1 και 6.2. Στον Πίνακα 6.2 φαίνεται η αξιολόγηση της μεταφοράς γνώσης. Η αξιολόγηση έγινε με το

Μέσο Σφάλμα Τετραγώνου (*MSE*) που μετρά τον μέσο όρο των τετραγώνων της απόστασης και τα λάθη γωνίας.

Ο τύπος Vincenty υπολογίζει την απόσταση μεταξύ δύο δεδομένων σημείων (Lat, Lon). Στο πρόβλημα πρόβλεψης της επόμενης θέσης, ενδείκνυται ως τρόπος αξιολόγησης των προβλέψεων, επειδή παρέχει μια πιο κατανοητή έκφραση του σφάλματος. Επίσης οι συχνά χρησιμοποιούμενες μετρικές αξιολόγησης (όπως το MSE) όταν η έξοδος είναι μια πλειάδα αποτυγχάνουν να διακρίνουν σωστά τη διαφορά σπουδαιότητας των στοιχείων της πλειάδας ειδικά στην περίπτωση απόστασης / γωνίας. Αυτό σημαίνει ότι ένα μικρό σφάλμα σε απόλυτη απόσταση έχει διαφορετικό αποτέλεσμα από ένα μικρό σφάλμα στη γωνίας, και αυτό δεν εμφανίζεται όταν χρησιμοποιούνται παραδοσιακές μετρήσεις αξιολόγησης παλινδρόμησης DL.



Σχήμα 6.3 Πρόβλεψη πορείας

6.7. Σύγκριση με άλλα μοντέλα

Προκειμένου να αξιολογηθεί το προτεινόμενο νευρωνικό μοντέλο πρόβλεψης τροχιάς με γενετικούς αλγόριθμους βελτιστοποίησης υπερπαραμέτρων, κάνουμε δύο διαφορετικούς τύπους συγκρίσεων. Το προτεινόμενο μοντέλο συγκρίνεται με διαφορετικά μοντέλα πρόβλεψης τροχιάς που χρησιμοποιούν παραδοσιακούς αλγόριθμους μηχανικής εκμάθησης .

Στο [21] οι συγγραφείς πειραματίζονται με πολλαπλά μοντέλα μηχανικής εκμάθησης όπως η Γραμμική Παλινδρόμηση, τα Τυχαία Δάση και οι Πολλαπλές Στρώσεις (MLP) και κατέληξαν στο συμπέρασμα ότι θα κατασκευάσουν δύο διαφορετικά MLPs ένα για την πρόβλεψη γεωγραφικού πλάτους και ένα για γεωγραφικό μήκος. Τα MLP αποτελούνται από ένα κρυφό στρώμα, feedforward συνδεδεμένους νευρώνες και την τοπολογία του ANN με χειροκίνητες δοκιμές υπερπαραμέτρων. Στο χρονικό διάστημα των 10 λεπτών έφτασαν σε μια μέση ακρίβεια 0,65 χλμ.

Επίσης έγιναν δοκιμές με μετα-μοντέλα που περιέχουν ένα σύνολο παραδοσιακών μοντέλων μηχανικής εκμάθησης, τεχνικές προεπεξεργασίας και έναν μηχανισμό βελτιστοποίησης παραμέτρων.

Αυτα τα μοντέλα καλούνται με τον όρο AutoML [22]. Ένα πιθανό μοντέλο για τη σύλληψη της σχέσης μεταξύ των συστατικών του μετα-μοντέλου, των ρυθμίσεων υπερπαραμέτρων και της ακρίβειας του μοντέλου μπορεί να κάνει μια έξυπνη αναζήτηση στον χώρο των πιθανών τοπολογιών προκειμένου να καταλήξει σε ένα κοντά στο βέλτιστο μοντέλο. Η εξερεύνηση περιλαμβανει αναζήτηση σε περιοχές όπου το μοντέλο είναι αβέβαιο καθώς και εστίαση στον χώρο όπου το μοντέλο έχει καλή απόδοση.
Τα δέντρα αποφάσεων που ενισχύονται με gradient [23] είναι επίσης μια μέθοδος συνόλου που χρησιμοποιεί μια προσέγγιση βελτιστοποίησης υπερπαραμέτρων διαβάθμισης για να κατασκευάσει ένα ακριβές μοντέλο πρόβλεψης που αποτελείται από πολλούς αδύναμους εκτιμητές. Ακολουθεί μια επαναληπτική διαδικασία στην οποία προστίθενται νέα μοντέλα για τη διόρθωση των σφαλμάτων των υφιστάμενων μοντέλων. Ο αλγόριθμος διαβάθμισης με gradients εφαρμόζεται στη συνάρτηση απώλειας για να επιλεγούν μοντέλα που είναι κοντά στα βέλτιστα για να προστεθούν.

6.8. Πειραματική διαδικασία και αποτελέσματα

Για την αξιολόγηση ακολουθήσαμε τη μέθοδο εκτός-δείγματος (out-of-sample) που διατηρεί τη χρονική σειρά των παρατηρήσεων χωρίζοντας τις θέσεις δεδομένων κίνησης κάθε τροχιάς σε δύο ακολουθίες. Η ακολουθία εκπαίδευσης και η ακολουθία δοκιμών. Η σειρά εκπαίδευσης αποτελείται από τις πρώτες 66% γεω-θέσεις που χρησιμοποιήθηκαν για να ανακτήσουν το καταλληλο μοντέλο από τη βάση γνώσης και να εκπαιδεύσουν το μοντέλο πρόβλεψης. Η ακολουθία δοκιμών αποτελείται από τις επόμενες 34% γεω-θέσεις και χρησιμοποιείται για την αξιολόγηση με τη μέτρηση Vincenty. Στο πλαίσιο των εφαρμογών χρονοσειρών δεν χρησιμοποιήθηκε ο μηχανισμός τυχαίου ανακατέματος των δεδομένων όπως κάνουμε με την παραδοσιακή αξιολόγηση μηχανικής μάθησης, καθώς θα παραμόρφωνε την ακολουθία των παρατηρήσεων.
Πειραματιστήκαμε με δύο ξεχωριστά παλινδρομικά LSTM νευρωνικά δίκτυα, ένα ενοποιημένου μοντέλου παλινδρόμησης που έχει δύο εξόδους έναντι ξεχωριστών μοντέλων για την απόσταση και για την γωνία και καταλήξαμε στο συμπέρασμα ότι το ενοποιημένο μοντέλο ξεπερνά τα δύο ξεχωριστά. Για να βρούμε τις πλησιέστερες προς τις βέλτιστες τοπολογίες των μοντέλων ANN που θα αποθηκευτούν στη βάση γνώσεων χρησιμοποιήσαμε έναν γενετικό αλγόριθμο. Η βάση γνώσεων αποτελείται από ένα λεξικό με κλειδί το προεπεξεργασμένο σύνολο δεδομένων και τιμή το μοντέλο πρόβλεψης.
Ο γενετικός αλγόριθμος ξεκινά με την εκπαίδευση ενός πληθυσμού 14 ατόμων και 7 γενεών. Αφού ολοκληρωθεί η αξιολόγηση της πρώτης, τυχαίας επιλεγμένης γενιάς, τα κορυφαία 50% άτομα των δικτύων προχωρούν στην επόμενη γενιά, μαζί με 10% πιθανότητα για κάθε άλλο δίκτυο. Τα υπόλοιπα άτομα του πληθυσμού γεμίζουν με δίκτυα που δημιουργούνται με ζευγάρωμα δύο δικτύων που έχουν ήδη προχωρήσει. Το ζευγάρωμα αποτελείται από τυχαία επιλογή υπερπαραμέτρων για το νέο δίκτυο από τη λίστα υπερπαραμέτρων των δύο γονικών δικτύων. Αυτή η διαδικασία επαναλαμβάνεται έως ότου ολοκληρωθεί η αξιολόγηση της τελευταίας γενιάς, όπου επιλέγεται το δίκτυο με την καλύτερη απόδοση.

Οι υποψήφιες τοπολογίες ANN αποτελούνται από:
• Ένα επίπεδο εισόδου
• (0-2) Επίπεδα LSTM
• (1-5) Πυκνά επίπεδα
• Επίπεδο εξόδου

Οι υπερ-παράμετροι κάθε LSTM Layer είναι:
• $lstm_{units}$: Διαστάσεις του χώρου εξόδου με τιμές: (2, 8, 16, 32, 64, 128)
• $lstm_{activation}$: Λειτουργία ενεργοποίησης με τιμές: (tanh, sigmoid, relu, linear, h*ardsigmoid*)
• $recurrent_{activation}$: Λειτουργία ενεργοποίησης που χρησιμοποιείται για τις επαναλαμβανόμενες τιμές με τιμές: (tanh, sigmoid, relu, γραμμικό, h*ardsigmoid*)
Οι υπερ-παράμετροι κάθε Dense Layer είναι:
• $number_{neurons}$: Οι διαστάσεις του χώρου εξόδου με τιμές: (2, 8, 16, 32, 64, 128)
• *activation*: Συνάρτηση ενεργοποίησης για χρήση με τιμές: (tanh, σιγμοειδές, relu, γραμμικό)

Οι υπερ-παράμετροι του δικτύου περιλαμβάνουν:
• *LSTMLayers*: Αριθμός επιπέδων LSTM στο δίκτυο με τιμές: (0,1,2)
• *DenseLayers*: Αριθμός πυκνών επιπέδων στο δίκτυο με τιμές: (1, 2, 3, 4, 5)
• *Optimizer*: Το εργαλείο βελτιστοποίησης εκπαίδευσης του δικτύου με τιμές: (rmsprop, adam, sgd, adagrad, adadelta, adamax, nadam)

Το XGBoost παίρνει επίσης ως είσοδο την απόσταση και τη γωνία μεταξύ των δύο διαδοχικών σημείων αλλά λειτουργεί με δύο διαφορετικά μοντέλα. Το ένα αντιστοιχεί στην προβλεπόμενη απόσταση και το άλλο στην προβλεπόμενη γωνία από το αρχικό σημείο. Το XGBoost έκανε μια έξυπνη αναζήτηση μέσω 20736 συνδυασμών δέντρων με υπερπαραμέτρους:
• *gamma*: με τιμές: (0, 0.2, 0.4) Απαιτείται ελάχιστη μείωση απώλειας για να γίνει ένα περαιτέρω διαμέρισμα σε έναν κόμβο φύλλων του δέντρου. Όσο μεγαλύτερο είναι το γάμμα, τόσο πιο μικρός θα είναι ο αλγόριθμος.
• $depth_{max}$: με τιμές: (3, 5, 10, 15) που υποδηλώνουν το μέγιστο βάθος ενός δέντρου. Η αύξηση αυτής της τιμής θα κάνει το μοντέλο πιο περίπλοκο και πιο πιθανό να εμφανίσει φαινόμενο overfitting.
• $childWeight_{min}$: με τιμές: (3,7) Ελάχιστο απαιτούμενο βάρος (hessian) σε ένα παιδί. Εάν το βήμα του διαμερίσματος δέντρου έχει ως αποτέλεσμα έναν κόμβο φύλλων με το άθροισμα του βάρους εμφάνισης μικρότερο από $childWeight_{min}$, τότε η διαδικασία δημιουργίας θα σταματήσει τον περαιτέρω διαχωρισμό. Στην εργασία γραμμικής παλινδρόμησης, αυτό αντιστοιχεί απλώς στον ελάχιστο αριθμό παρουσιών που πρέπει να υπάρχουν σε κάθε κόμβο. Όσο μεγαλύτερο είναι το $childWeight_{min}$, τόσο πιο συντηρητικός θα είναι ο αλγόριθμος.
• $colsample_{bytree}$: με τιμές: (0.3,0.7) είναι η αναλογία δειγματοληψίας των στηλών κατά την κατασκευή κάθε δέντρου. Η δειγματοληψία πραγματοποιείται μία φορά για κάθε δέντρο που έχει κατασκευαστεί με *learningrate*: *values*: (0.05, 0.15, 0.3)

Το Auto-sklearn παρέχει μηχανισμό μηχανικής εκμάθησης που αναζητά τον σωστό αλγόριθμο εκμάθησης και βελτιστοποιεί τις υπερπαραμέτρους του με μόνη παράμετρο τον χρόνο εκπαίδευσης. Πειραματιστήκαμε με διαφορετικούς χρόνους εκπαίδευσης για διαφορετικές τροχιές και καταλήξαμε στο συμπέρασμα ότι δεν υπήρξε βελτίωση της ακρίβειας για χρόνους εκπαίδευσης μεγαλύτερους από 12 λεπτά.

6.8.1. Μοντέλα αποτελεσμάτων

Συνοψίζουμε τα αποτελέσματα του προτεινόμενου μοντέλου και μια σύγκριση με τα υπάρχοντα μοντέλα στον Πίνακα 6.1. Η πειραματική αξιολόγηση έδειξε ότι χρησιμοποιώντας την απόσταση και την γωνία της τελευταία γνωστής θέσης για την πρόβλεψη της επόμενης θέσης φάνηκε να παράγει πολύ καλύτερα αποτελέσματα από την χρήση απλών συντεταγμένων. Περιορίσαμε έτσι τον χώρο αναζήτησης σε έναν κύκλο γύρω από την τρέχουσα θέση, όμοια με τον τρόπο με τον οποίο ένα ανθρώπινο μυαλό θα αντιμετώπιζε αυτό το πρόβλημα.

Συγκρίνοντας τα αποτελέσματα μεταξύ Gradient Boosted Trees, AutoML και DL Networks που εκπαιδεύτηκαν με τον γενετικό αλγόριθμο, ο τελευταίος είχε καλύτερη απόδοση με την πλειονότητα των αποτελεσμάτων του να είναι κάτω των 0.5 χιλιομέτρων μέσου σφάλματος. Αναφέρεται ως σημαντική βελτίωση ότι το μέσο σφάλμα στο προτεινόμενο μοντέλο για διάστημα 10 λεπτών είναι 421 m ενώ σε διαφορετικά μοντέλα [21] ήταν 650 m. Επιπλέον, το προτεινόμενο μοντέλο ξεπερνά τα άλλα AutoML μοντέλα. Η προσαρμοστικότητά του μας επιτρέπει να προσαρμόσουμε τη χρήση του σύμφωνα με τις ανάγκες μας, απλώς αυξάνοντας ή μειώνοντας τον αριθμό των γενεών ή το μέγεθος του πληθυσμού, καλύπτοντας είτε καλύτερη ακρίβεια αποτελεσμάτων είτε λιγότερο χρόνο εκπαίδευσης.

Παρακολουθήσαμε την έξοδο απώλειας του γενετικού αλγορίθμου σε κάθε επανάληψη. Συγκεκριμένα παρακολουθήσαμε τη μέση απώλεια και την ελάχιστη απώλεια του ANN σε κάθε γενιά. Τα αποτελέσματα έδειξαν ότι η ελάχιστη απώλεια είχε μια σταδιακή σύγκλιση, ενώ η μέση απώλεια δεν ήταν πάντα συγκλίνουσα. Η αιτία αυτού του φαινομένου είναι η τυχαία παράμετρος στο στάδιο της μετάλλαξης.

Μέθοδος	Μετρική Vincenty	Χρόνος Εκπαίδευσης	Πιθανότητα βέλτιστης μεθόδου	Σφάλμα $_{<1km}$	Σφάλμα $_{<0,5\ km}$
Πρόβλεψη Συντεταγμένων					
Γενετικός DL	2180	7416	12.75	60.78	27.45
Πρόβλεψη Απόστασης/Γωνίας					
AutoSKLearn	1584	720	8.82	54.58	32.35
XGBoost	578	2519	15.36	90.20	49.02
Γενετικός DL	421	3585	63.07	97.39	70.92

Πίνακας 6.1 Αξιολόγηση μοντέλων

Όσον αφορά τον χρόνο πρόβλεψης, ο χρόνος απόκρισης της πρώτης πρόβλεψης κυμαίνεται από 50 msec έως 300 msec. Ο ακριβής αριθμός εξαρτάται από τον αριθμό των επιπέδων ANN. Όλοι οι επόμενοι χρόνοι απόκρισης όπως απεικονίζονται στον Πίνακα 6.2 είναι κοντά στα 25.5 msec στην περίπτωση μιας μεμονωμένης πρόβλεψης ($TestS_{ms}$) και 27.5 msec για μια πρόβλεψη δέσμης ($TestB_{ms}$) 100 παρατηρήσεων. Αυτοί οι χρόνοι είναι καταμετρημένοι (configuration) κατά τη χρήση του μοντέλου στο περιβάλλον του Colab που τρέχει στο Google cloud και τοπικά σε υπολογιστές με

μικές διαφορές. Αυτός ο χρόνος απόκρισης είναι αρκετά χαμηλός και είναι χαμηλότερος από τον χρόνο απόκρισης που αναφέρεται στο [21], οπότε δεν συνεχίσαμε την ερευνητική μας εργασία για να τον βελτιώσουμε περισσότερο.
Από την άλλη πλευρά, ο χρόνος εκπαίδευσης στο προτεινόμενο μοντέλο μας είναι σημαντικά μεγαλύτερος από τα άλλα μοντέλα. Ο λόγος είναι οι πολλές υπερπαράμετροι και οι συνδυασμοί τους που εξετάζει ο γενετικός αλγόριθμος. Αφού εξετάσαμε τα μοντέλα εξόδου, παρατηρήσαμε ότι υπάρχουν ομάδες με κοινά μοτίβα στις τοπολογίες τους. Αυτό πρέπει να διερευνηθεί περισσότερο σε μια μελλοντική εργασία, αλλά μια πιθανή ερμηνεία είναι ότι ο ίδιος ο τύπος πλοίων ή συμπεριφορά πλοήγησης μπορεί να παράγει παρόμοιους τύπους ANN. Καθώς ο επόμενος στόχος μας ήταν να βρούμε μια βιώσιμη λύση για τη μείωση του χρόνου εκπαίδευσης, προσπαθήσαμε να αξιοποιήσουμε αυτό το φαινόμενο.
Κάναμε το επόμενο σύνολο πειραμάτων αφού δημιουργήσαμε μια ικανοποιητική ποσότητα μοντέλων. Σε αυτήν την περίπτωση, αντί να χρησιμοποιήσουμε τον γενετικό αλγόριθμο για την αρχικοποίηση του μοντέλου, επιλέξαμε τις υπερπαραμέτρους ενός καλά εκπαιδευμένου μοντέλου που εκτιμάται ότι εμφανίζει το ίδιο μοτίβο στην τοπολογία του. Αυτή η διαδικασία είναι γνωστή στη βιβλιογραφία ως μεταφορά γνώσης και η επόμενη υποενότητα περιγράφει τα αποτελέσματά μας.

6.8.2. Μεταφορά γνώσης

Η διαδικασία μεταφοράς γνώσης [24], επιτάχυνε τη διαδικασία εκπαίδευσης με μια μικρή υποχώρηση στην ακρίβεια. Αντί να αναζητήσουμε μια σχεδόν βέλτιστη τοπολογία ANN χρησιμοποιώντας τον γενετικό αλγόριθμο και να εκπαιδεύσουμε το μοντέλο από το μηδέν, ξεκινήσαμε την εκπαιδευτική διαδικασία με ένα προ-εκπαιδευμένο μοντέλο και εξετάσαμε δύο διαφορετικούς τύπους μεταφοράς γνώσης.

Μέθοδος	MSE $_{(10^{-6})}$	Μεμονωμένη πρόβλεψη $_{(ms)}$	Ομαδική πρόβλεψη $_{(ms)}$	Χρόνος εκπαίδευσης $_{(s)}$
Αρχική εκπαίδευση	22898	25.66	27.70	3585
Επανεκπαίδευση μοντέλου	23264	25.57	27.12	66
Εκπαίδευση μεταφοράς γνώσης	23341	25.40	26.99	33

Πίνακας 6.2 Χρόνος εκπαίδευσης και πρόβλεψης

Στο Model Retrain, θεωρούμε ως δεδομένες τις υπερπαραμέτρους του μοντέλου και κάνουμε μια πλήρη εκπαίδευση στις παραμέτρους, δηλαδή στα βάρη και τα bias. Στο Transfer Train, παγώνουμε ορισμένα επίπεδα και επανεκπαιδεύουμε τα υπόλοιπα επίπεδα με τα νέα δεδομένα τροχιάς. Τα επανεκπαιδευμένα επίπεδα είχαν αρχικοποιηθεί με τα βάρη του βασικού μοντέλου γνώσης. Η απόδοση αυτών των μοντέλων παρουσιάζεται στον Πίνακα 6.2. Η αρχική εκπαίδευση είναι η περίπτωση που κάνουμε μια εκπαίδευση από το μηδέν με γενετικούς αλγόριθμους.
Η διαδικασία εκμάθησης μεταφοράς κάνει την υπόθεση ότι τα πρώτα επίπεδα του ANN διατηρούν τη γενική γνώση της κινητικότητας των πλοίων, ενώ τα τελευταία στρώματα σχετίζονται με τις ιδιαιτερότητες των συγκεκριμένων τροχιών. Ένα παρόμοιο φαινόμενο λαμβάνει χώρα στην επεξεργασία εικόνας με βαθιά μάθηση στην οποία τα πρώτα επίπεδα κατανοούν τη γενική δομή των σχημάτων και των χρωμάτων, ενώ τα τελευταία επίπεδα κάνουν συγκεκριμένη αναγνώριση αντικειμένων.

Στη διαδικασία μεταφοράς γνώσης κάναμε κάποιες δοκιμές και καταλήξαμε στην επανεκπαίδευση των δύο τελευταίων επιπέδων. Αυτή είναι μια αρχική έρευνα και η επιλογή των δύο τελευταίων επιπέδων δεν αποτελεί τελική λύση. Υπάρχει ακόμη ένας μεγάλος χώρος για περαιτέρω έρευνα σχετικά με τον τρόπο εφαρμογής της μεταφοράς γνώσης στην πρόβλεψη θεσης κινούμενων αντικειμένων, αλλά σίγουρα τα πειραματικά αποτελέσματα είναι ενθαρρυντικά. Τόσο η πλήρης επανεκπαίδευση όσο και η μεταφορά γνώσης των τελικών επιπέδων επιτυγχάνουν μια παρόμοια τιμή της MSE με την αρχική μας εκπαίδευση, ενώ υπάρχει δραματική μείωση του χρόνου εκπαίδευσης από 3585 σε μόνο 33 δευτερόλεπτα.

7. Πειραματική αξιολόγηση με πρόβλεψη χρήσης πόρων σε υποδομές των άκρων

Κατά την τελευταία δεκαετία, υπήρξε μια αυξανόμενη ανάγκη να φέρουμε την επεξεργασία και τα δεδομένα πιο κοντά στις συσκευές όπου αυτά δημιουργούνται. Αυτές οι συσκευές μπορεί να περιλαμβάνουν έξυπνα αντικείμενα, κινητά τηλέφωνα, πύλες δικτύου και αισθητήρες από το Διαδίκτυο των Πραγμάτων (Internet of Things) (IoT). Αυτό το κατανεμημένο υπολογιστικό παράδειγμα, που ορίζεται ως υπολογιστική των άκρων στοχεύει στη δημιουργία αποκεντρωμένων τοπολογιών και επιτρέπει τη μετεγκατάσταση διάφορων υπολογιστικών και αποθηκευτικών πόρων πιο κοντά στα άκρα του δικτύου. Με αυτόν τον τρόπο, αναμένεται η παροχή υπηρεσιών και να προσφέρεται σε καλύτερους χρόνους απόκρισης και μεγαλύτερα ποσοστά μεταφοράς.
Όταν εξετάζουμε τη φύση και τις απαιτήσεις των σύγχρονων εφαρμογών, καθίσταται σαφές ότι η εκτέλεσή τους σε ενορχηστρωμένους κόμβους επεξεργασίας είναι μεγάλης σημασίας, προκειμένου να ανταποκρίνονται στα πρότυπα ποιότητας υπηρεσίας (Quality of Services) (QoS) που έχουν καθοριστεί από τη βιομηχανία. Πιο συγκεκριμένα, η αύξηση των συσκευών IoT στην άκρη του δικτύου έχει ως αποτέλεσμα την παραγωγή τεράστιων όγκων δεδομένων και φόρτου εργασίας. Αυτή η πρωτοφανής κατάσταση γέννησε την ανάγκη για μια έξυπνη και προσαρμοστική διαχείριση των υπολογιστικών πόρων, οι οποίοι είναι σε θέση να παρέχουν υψηλή απόδοση και χαμηλή καθυστέρηση με τον περιορισμό ότι οι κόμβοι επεξεργασίας στα άκρα είναι περιορισμένοι και πολύτιμοι.

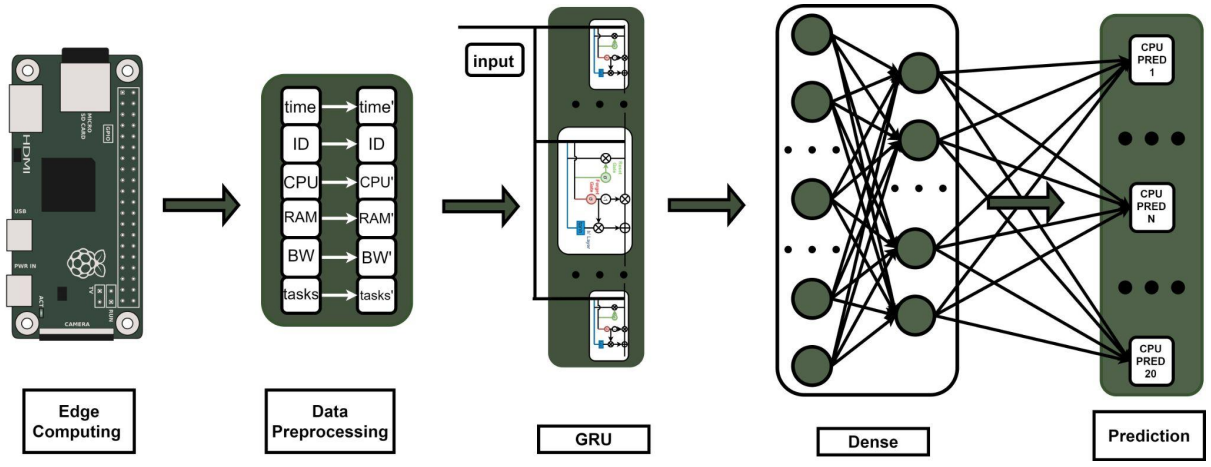
7.1. Προσέγγιση με Νευρωνικά Δίκτυα και Εύρεση Υπερπαραμέτρων

Οι λειτουργίες διαχείρισης υπολογιστικής των άκρων όπως η εκφόρτωση εργασιών, η προσαρμοστική κατανομή πόρων και η προληπτική ανοχή σφαλμάτων μπορούν να κάνουν καλύτερη χρήση των κόμβων επεξεργασίας με έναν προγνωστικό μηχανισμό χρήσης πόρων. Ο κύριος τομέας που πρέπει να εστιάσουμε είναι το ποσοστό που λειτουργεί η CPU, η RAM, το εύρος ζώνης και τα αποθηκευτικά μέσα (σκληρός δίσκος). Αυτές οι μετρήσεις έχουν μια ακολουθιακή συσχέτιση, καθιστώντας καλή επιλογή την μοντελοποίηση με μια προσέγγιση χρονοσειρών. Ένα μοντέλο παλινδρόμησης RNN που αξιοποιεί χαρακτηριστικά χρονοσειρών μέσω GRU όπως τα είδαμε στις προηγούμενες ενότητες μπορεί να είναι μια πολύ καλή λύση για την πρόβλεψη των μετρικών πόρων.
Η επόμενη πρόκληση είναι η μοντελοποίηση μιας συστηματικής μεθοδολογίας για την κατασκευή μοντέλων RNN με αυτόματο τρόπο χρησιμοποιώντας ιστορικά δεδομένα. Η κοινή προσέγγιση της μη αυτόματης δοκιμής και σφάλματος για την εύρεση μιας αποδεκτής αρχιτεκτονικής DL χρειάζεται πολλές ώρες εργασίας για τους επιστήμονες της μηχανικής μάθησης κάθε φορά που οι εφαρμογές, η συμπεριφορά του χρήστη ή η υποδομή στα άκρα αλλάζουν. Οι διαθέσιμες μέθοδοι αυτοματοποίησης DL εξακολουθούν να έχουν σημαντικά μειονεκτήματα και είναι μια ανοιχτή πρόκληση για το πώς οι εξελικτικοί αλγόριθμοι μπορούν να επεκταθούν και να συνδυαστούν με άλλα μοντέλα hypertuning.
Όλα τα παραπάνω μας έδωσαν κίνητρο να προτείνουμε ένα μοντέλο GRU-RNN παλινδρόμησης που προβλέπει με ενοποιημένο τρόπο τη χρήση πόρων των κόμβων επεξεργασίας των άκρων, αξιοποιώντας χαρακτηριστικά χρονοσειρών. Προκειμένου να καταλήξουμε στο βέλτιστο μοντέλο παλινδρόμησης RNN, προτείνουμε την εφαρμογή του αλγορίθμου βελτιστοποίησης Hybrid Bayesian Evolution Strategy (HBES) που περιγράψαμε στα προηγούμενα κεφάλαια.
Οι τρεις σημαντικές συνεισφορές της εργασίας αυτής είναι:

• Η πρόταση και η ανάλυση των θεωρητικών αρχών και της εφαρμογής του μοντέλου παλινδρόμησης πολλαπλών εξόδων GRU-RNN ως προγνωστικής χρήσης πόρων στην υπολογιστική των άκρων.
• Η πρόταση ενός καινοτόμου υβριδικού μοντέλου εύρεσης υπερπαραμέτρων που συνδυάζει την εξελικτική στρατηγική (ES) με τον Bayesian Optimization (BO) αλγόριθμο προκειμένου να συγκλίνει σε μια βέλτιστη αρχιτεκτονική RNN.
• Μια πειραματική αξιολόγηση του προτεινόμενου προγνωστικού μοντέλου χρήσης πόρων και του αλγορίθμου βελτιστοποίησης HBES. Καθώς και η σύγκριση με προηγούμενα μοντέλα μηχανικής μάθησης σε μια πραγματική υπολογιστική υποδομή των άκρων.

7.2. Πρόβλεψη χρήσης πόρων στις υπολογιστικές υποδομές των άκρων

Η διαχείριση και ενορχήστρωση των υποδομών υπολογιστικής των άκρων μπορεί να βελτιωθεί προβλέποντας τις χρονικά εξελισσόμενες μετρήσεις χρήσης πόρων με ένα μοντέλο RNN. Κυρίως αυτές οι μετρήσεις είναι CPU, RAM, εύρος ζώνης και το I/O δίσκου. Η υπολογιστική των άκρων χαρακτηρίζεται από τη δυναμική συμπεριφορά και την ετερογένεια των κόμβων επεξεργασίας σε συνδυασμό με περιορισμούς που επιφέρουν οι συμφωνίες επιπέδου εξυπηρέτησης (Service Level Agreements) (SLAs). Η λήψη αποφάσεων σε ένα δυναμικό και ετερογενές περιβάλλον είναι μια δύσκολη διαδικασία και θα πρέπει να χρησιμοποιείται κάθε διαθέσιμη πληροφορία. Η πρόβλεψη της χρήσης πόρων, αξιοποιώντας τα χαρακτηριστικά των χρονοσειρών των ιστορικών δεδομένων, αποτελεί ένα από τα πιο πολύτιμα στοιχεία. Είναι ένας ισχυρός δείκτης για τη διαθεσιμότητα των κόμβων επεξεργασίας προκειμένου ένας κόμβος να λάβει περαιτέρω φόρτο εργασίας ή να προβλεφθεί η υποβάθμιση του QoS στα επόμενα βήματα. Από αυτή την άποψη, τα διαθέσιμα δημόσια εργαλεία παρακολούθησης όπως το Prometheus, το OpenTSDB, το Nagios και το InfuxDB μπορούν να παρέχουν τις μετρήσεις πόρων σε μορφή ρευμάτων (streams) ή σε μια βάση δεδομένων χρονοσειρών όπως η PromQL. Αυτές οι βάσεις δεδομένων χρονοσειρών μπορούν να χρησιμοποιηθούν για την εξαγωγή των ιστορικών συνόλων δεδομένων για το μοντέλο RNN.



Σχήμα 7.1 Σχήμα Ενσωματωμένου GRU στην πρόβλεψη χρήσης πόρων.

Η δυναμική συμπεριφορά των κόμβων που βρίσκονται στο άκρο του δικτύου οφείλεται στη κλιμακούμενη συμπεριφορά των αιτημάτων εφαρμογής και του φόρτου εργασίας. Ο αριθμός των αιτημάτων ανά χρονικό διάστημα αλλάζει κατά τη διάρκεια των ημερών και πολλά περιοδικά

φαινόμενα μπορούν να το επηρεάσουν. Η ετερογένεια σημαίνει ότι οι κόμβοι ακρών έχουν ως επί το πλείστον διαφορετικά χαρακτηριστικά στο μέγεθος της μνήμης και στην υπολογιστική ισχύ. Μπορούμε να μελετήσουμε αυτήν την ετερογένεια λαμβάνοντας υπόψη τα διαφορετικά είδη των Raspberry Pis που είναι διαθέσιμα, συνυπάρχουν και συνεργάζονται σε μια υποδομή.
Τα SLA θέτουν τους περιορισμούς για την απόδοση των κόμβων άκρου όσον αφορά τη διαθεσιμότητα, την απόδοση και τους διαφορετικούς τύπους καθυστερήσεων. Οι πάροχοι υπηρεσιών έχουν στόχο να μην παραβιαστούν τα SLA. Με αυτόν τον τρόπο, πρέπει να λαμβάνουν έγκαιρες και βέλτιστες αποφάσεις σχετικά με την εκφόρτωση εργασιών, την κατανομή πόρων και την προληπτική ανοχή σφαλμάτων.

7.2.1. Προσαρμοστική κατανομή πόρων

Η απόδοση μια εφαρμογής εξαρτάται από τους διαθέσιμους πόρους στις συσκευές τελικού χρήστη και τους κόμβους των άκρων. Προκειμένου να ξεπεραστούν οι υπολογιστικοί περιορισμοί, είναι δυνατό να εκχωρηθούν πρόσθετοι υπολογιστικοί πόροι άκρων, οι οποίοι χειρίζονται σωστά τον φόρτο εργασίας που δημιουργείται από τις διάφορες εργασίες. Υπάρχουν δύο τρόποι με τους οποίους μπορεί να επιτευχθεί αυτή η λειτουργικότητα. Ο πρώτος αναφέρεται ως οριζόντια κλιμάκωση και είναι η διαδικασία εκχώρησης πρόσθετων υπολογιστικών κόμβων ενδεχομένως από διαφορετικούς κεντρικούς υπολογιστές. Ο δεύτερος ονομάζεται κατακόρυφη κλιμάκωση και είναι η διαδικασία αύξησης της υπολογιστικής ικανότητας των υπαρχόντων κόμβων, εκχωρώντας επιπλέον πόρους από έναν υπολογιστή.
Στην περίπτωση της οριζόντιας κλιμάκωσης είναι ζωτικής σημασίας να έχουμε μια εκ των προτέρων εκτίμηση της σχετικά βραχυπρόθεσμης αναμενόμενης χρήσης των πόρων, επειδή η δημιουργία μιας εικονικής μηχανής απαιτεί χρονικό διάστημα αρκετών λεπτών. Αυτό το χρονικό διάστημα αποτελεί έναν αρκετά σημαντικό περιορισμό στην ικανότητα του δικτύου να αντιδρά έγκαιρα σε περίπτωση ξαφνικής εκτίναξης (burst) στη χρήση του δικτύου. Επιπλέον, όταν χρησιμοποιούνται φορητές συσκευές άκρων που λειτουργούν με τη δική τους αυτόνομη τροφοδοσία μπαταρίας, η διάρκεια για την οποία μπορούν να παραμείνουν ενεργές εξαρτάται σε μεγάλο βαθμό από τη χρήση πόρων που χρειάζονται. Από την άλλη πλευρά, η κατανομή πρόσθετων υπολογιστικών πόρων από ορισμένους δημόσιους παρόχους υπολογιστικής του νέφους αυξάνει το συνολικό κόστος της λειτουργίας. Η λύση θα ήταν η εφαρμογή της αρχής που ονομάζεται «ζώνη goldilocks», η οποία διασφαλίζει ότι οι εικονικές μηχανές λειτουργούν με καθορισμένη χωρητικότητα, η οποία ταυτόχρονα παρατείνει τη διάρκεια ζωής της μπαταρίας των συσκευών, διατηρεί το συνολικό κόστος στο ελάχιστο και επιτρέπει στο δίκτυο να αντιδρά έγκαιρα σε αλλαγές στην κίνηση.
Η πειραματική αξιολόγηση πραγματοποιήθηκε σε πραγματικό άκρο υποδομής, όπου οι κόμβοι επεξεργασίας άκρων ήταν Raspberry Pi3 με τετραπύρηνο ARM Cortex-A53 64-bit στα 1.4GHz, φορτωμένο με λειτουργικό σύστημα Raspbian που είναι μια έκδοση του Debian Linux. Το σύνολο δεδομένων που κατασκευάστηκε από ένα εργαλείο παρακολούθησης που υλοποιήθηκε στην Python 3 υλοποιήθηκε με τις βιβλιοθήκες psutil και GPUtil . Παρακολουθήσαμε τη χρήση CPU, RAM, δίσκου και εύρους ζώνης σε πραγματικό χρόνο σε ένα χρονικό διάστημα ενός δευτερολέπτου.
Η εφαρμογή που χρησιμοποιήθηκε, όπως θα περιγραφεί παρακάτω, ήταν μια κατηγοριοποίηση κειμένων επεξεργασίας φυσικής γλώσσας. Η περίπτωση χρήσης ήταν να γίνει η κατηγοριοποίηση κειμένων σε ένα περιβάλλον υπολογιστών αιχμής, τοπικά, κοντά στους κατόχους κειμένου και όχι σε υποδομές υπολογιστικού νέφους λόγω ιδιωτικότητας των δεδομένων. Ο λόγος για αυτήν την επιλογή είναι ότι οι κάτοχοι κειμένου δεν συμφώνησαν να μεταφερθούν και να επεξεργαστούν τα κείμενά τους σε απομακρυσμένους διακομιστές. Προκειμένου να ελέγξουμε την εφαρμογή από απόσταση και να

πάρουμε τα σύνολα δεδομένων χρήσης πόρων χρησιμοποιήσαμε το πρωτόκολλο SSH, αλλά δεν είχαμε τα δικαιώματα πρόσβασης στα επεξεργασμένα κείμενα.
Για να αξιολογήσουμε την ακρίβεια του προτεινόμενου μοντέλου χρησιμοποιήσαμε μετρήσεις σφάλματος και μέτρηση χρόνου. Οι μετρήσεις σφαλμάτων είναι το μέσο απόλυτο σφάλμα (MAE) και η ρίζα του μέσου τετραγωνικού σφάλματος (RMSE). Το MAE εκφράζει τη μέση απόλυτη διαφορά μεταξύ των τιμών στόχων και των προβλεπόμενων τιμών. Τετραγωνίζοντας τα σφάλματα πρόβλεψης και παίρνοντας τον μέσο όρο των τετραγώνων έχουμε το RMSE που εκφράζει την τυπική απόκλιση των σφαλμάτων και δίνει έμφαση στην εξάπλωση των σφαλμάτων. Το MAE προτιμάται όταν όλα τα σφάλματα έχουν την ίδια σημασία, ενώ το RMSE όταν πρέπει να τιμωρούμε τα μεγάλα σφάλματα ακόμα κι αν δεν συμβαίνουν συχνά.

7.2.2. Έξυπνη εκφόρτωση εργασιών

Η εκφόρτωση εργασιών αναφέρεται στη διαδικασία καθορισμού συγκεκριμένων πόρων δικτύου για τη διαχείριση τους, ανάλογα με τις απαιτήσεις τους. Αυτή η διαδικασία βασίζεται στην μάλλον απλή αρχή της σωστής κατανομής του φόρτου εργασίας μεταξύ των διαθέσιμων υπολογιστικών πόρων και των πόρων αποθήκευσης, προκειμένου να επιτευχθεί καλύτερος χρόνος απόκρισης. Ο μηχανισμός λήψης αποφάσεων εκφόρτωσης εργασιών, αφού εξετάσει την τρέχουσα χρήση πόρων για κάθε τρέχοντα κόμβο, θα αποφασίσει ποιος θα λάβει την επόμενη εργασία. Ο κύριος σχεδιαστικός σκοπός της διαδικασίας επιλογής κόμβων των άκρου είναι να αποφευχθεί η επιλογή ενός κόμβου, ο οποίος ήδη λειτουργεί κοντά στα όρια των δυνατοτήτων του. Επιπλέον, κανείς δεν μπορεί παρά να παρατηρήσει ότι οι λειτουργίες της έξυπνης εκφόρτωσης εργασιών και της προσαρμοστικής κατανομής πόρων είναι στενά συνδεδεμένες. Με τη διασφάλιση ότι οι διάφοροι κόμβοι άκρων δεν λειτουργούν κοντά στη μέγιστη χωρητικότητα τους, το χρονικό πλαίσιο που το δίκτυο μπορεί να ανταποκριθεί σε μια πιθανή, εκτίναξη στην παραγωγή εργασιών αυξάνεται σημαντικά.

7.2.3. Προληπτική ανοχή σφαλμάτων

Δεδομένου ότι η υπολογιστική των άκρων αποτελείται από ένα μεγάλο αριθμό κόμβων που λειτουργούν ταυτόχρονα, είναι εξαιρετικά σημαντικό να λάβουμε υπόψη μας ότι η αποτυχία κάποιων κόμβων είναι αναπόφευκτη. Με αυτόν τον τρόπο είναι δυνατόν να διασφαλιστεί ότι το δίκτυο θα συνεχίσει να λειτουργεί χωρίς διακοπή όταν ένας ή περισσότεροι κόμβοι του δικτύου αποτύχουν. Ο κύριος τρόπος για να διασφαλιστεί ότι ένα δίκτυο θα είναι σε θέση να συνεχίσει να εκτελεί τη λειτουργία του ακόμη και μετά από μια κρίσιμη αποτυχία, είναι να χρησιμοποιεί εφεδρικούς κόμβους, οι οποίοι αντικαθιστούν αυτόματα τους αποτυχημένους, με τρόπο που δεν συμβαίνει απώλεια υπηρεσίας.
Όπως εξηγείται στην προηγούμενη υποενότητα, η δημιουργία μιας νέας εικονικής μηχανής απαιτεί μια συγκεκριμένη περίοδο αναμονής, η οποία θα είχε σοβαρές επιπτώσεις στην αποτελεσματικότητα του δικτύου. Έτσι, η ανοχή σφαλμάτων μπορεί να επωφεληθεί από προνοητικές τεχνικές. Ανά πάσα στιγμή, το δίκτυο πρέπει να περιέχει έναν συγκεκριμένο αριθμό υπολογιστικών κόμβων, οι οποίοι παραμένουν αδρανείς έως ότου ένα; από τους κόμβους που λειτουργεί παύσει να λειτουργεί σωστά. Με τη χρήση αλγορίθμων μηχανικής μάθησης είναι δυνατή η εξαγωγή πληροφοριών σχετικά με τα μοτίβα χρήσης πόρων των επιρρεπών σε αποτυχία κόμβων και το χρονικό πλαίσιο στο οποίο παρουσιάζονται τα σφάλματα. Αυτό επιτρέπει στην προνοητική ανοχή σφαλμάτων να διασφαλίσει ότι οι λειτουργίες του δικτύου θα συνεχίσουν να λειτουργούν χωρίς διακοπή και ότι το συνολικό κόστος θα διατηρηθεί στο ελάχιστο.

7.3. Δημιουργία Dataset

Στόχος μας ήταν να βρούμε ένα υπολογιστικό περιβάλλον των άκρων χαμηλού κόστους. Έτσι, επιλέξαμε το Raspberry Pi3 ως πλατφόρμα ανάπτυξης για την εφαρμογή μας. Το Raspberry Pi είναι ένας υπολογιστής μικρού μεγέθους χαμηλού κόστους που μπορεί να φιλοξενήσει εφαρμογές που εκτελούνται σε ένα λειτουργικό σύστημα Raspbian που βασίζεται στο Debian Linux. Αυτό το μοντέλο περιέχει τετραπύρηνο ARM Cortex-A53 64-bit στα 1,4GHz, έχοντας ένα υπολογιστικό περιβάλλον των άκρων που υποστηρίζει λειτουργικά συστήματα βασισμένα σε Linux και βιβλιοθήκες μηχανικής μάθησης. Ωστόσο, απαιτείται μια προσεκτική ρύθμιση για την εγκατάσταση του Tensorflow και την εκτέλεση της εφαρμογή στην περιορισμένη δυνατότητα μνήμης και υπολογισμού μιας συσκευής. Πρώτα χρειάζεται να συνδεθούμε με τη συσκευή μας. Για τον έλεγχο της εφαρμογής από απόσταση χρησιμοποιήθηκε το πρωτόκολλο SSH. Στη συνέχεια, αφού έγινε η συνδεση με επιτυχία στη συσκευή, εγκαταστήσαμε όλες τις απαιτήσεις για το Tensorflow και την εφαρμογή κατηγοριοποίησης.


```
{”timestamp”: 1584552696,
”cpu_values”: {”cpu_freq”: [1400.0, 600.0, 1400.0],
”cpu_percent”: [100.0, 0.0, 0.0, 0.0]},
”ram_values”: [971055104, 757248000, 22.0, 142651392, 396947456,
308183040, 178515968, 39874560, 391581696, 6840320, 67715072],
”disk_values”: [{”device”: ”/dev/root”, ”usage”: [27666726912,
7450722304, 18787008512, 28.4]},
{”device”: ”/dev/mmcblk0p6”, ”usage”: [264288256, 54746624,
209541632, 20.7]}],
”network_values”: [{”device”: ”wlan0”, ”bytes_sent”: 0,
”bytes_recv”: 0, ”packets_sent”: 0, ”packets_recv”: 0},
{”device”: ”eth0”, ”bytes_sent”: 2821510, ”bytes_recv”: 31041987,
”packets_sent”: 21059, ”packets_recv”: 32964},
{”device”: ”lo”, ”bytes_sent”: 0, ”bytes_recv”: 0,
”packets_sent”: 0, ”packets_recv”: 0}],
”gpu_values”: [] }
```


Σχήμα 7.2 Παράμετροι ενδιαφέροντος

Ως εργαλείο παρακολούθησης επιλέξαμε να αναπτύξουμε ένα σενάριο python που συνδυάζει τις βιβλιοθήκες psutil και GPUtil, παρέχοντας πληθώρα εργαλείων παρακολούθησης. Στο σχήμα 7.2 φαίνονται οι παράμετροι ενδιαφέροντος σχετικά με τις καταχωρήσεις των μετρήσεων της CPU, της μνήμης RAM, του δικτύου, του σκληρού δίσκου και της GPU σε πραγματικό χρόνο. Λόγω της φύσης της python, η εκτέλεση του σεναρίου είναι γρήγορη και συμβατή με τους περιορισμούς της υπολογιστικής των άκρων.

Το σενάριο μας επιτρέπει να καθορίσουμε ένα σύνολο παραμέτρων σε χρόνο εκτέλεσης, βελτιστοποιώντας τη διαδικασία παρακολούθησης των κόμβων. Αυτές οι παράμετροι περιλαμβάνουν τη συχνότητα λήψης των μετρήσεων που παρακολουθήθηκαν, το όριο μεγέθους αρχείου των αρχείων καταγραφής για την πιο εύκολη επεξεργασία των αρχείων και τρεις λίστες εξαιρούμενων συσκευών, εάν έχουμε, μία για κάθε κατηγορία συσκευών (HDD, GPU και Network). Το Σχήμα 7.3 δείχνει μια

επισκόπηση των μετρήσεων που παρακολουθούμε, διαθέτοντας για κάθε συνάρτηση το όνομά της, τον τύπο επιστροφής, την κατηγορία και μια σύντομη περιγραφή:
Όλα τα δεδομένα που εξάγονται από το εργαλείο παρακολούθησης διαχωρίζονται ανά κατηγορία και μετατρέπονται σε μορφή JSON για ομοιομορφία και ευκολότερη χρήση κατά τα επόμενα βήματα της διαδικασίας μοντελοποίησης. Ένα παράδειγμα σχήματος JSON είναι διαθέσιμο στο αποθετήριο GitHub του συγγραφέα της διπλωματικής [14]

Function	Type	Category	Description
cpu_{freq}	Float Tuple	CPU	Creates a tuple of the current, minimum and maximum frequency of the CPU.
$cpu_{percent}$	Float Array	CPU	Creates a list of float pointer numbers representing the current system-wide CPU utilization as a percentage for each CPU core at regular intervals.
$disk_{usage}$	Mix Tuple Array	HDD	Creates a tuple including total, used and free space expressed in bytes, plus the percentage usage of each HDD not excluded.
$netI/O_{count}$	JSON Array	Network	Creates a JSON Array containing the following metrics regarding network activity for each not excluded network adapter: • Number of bytes sent • Number of bytes received • Number of packets sent • Number of packets received
$getGPUs$	JSON Array	GPU	Creates a JSON Array containing the following metrics regarding the GPU activity for each not excluded GPU adapter: • Device name • Current load percentage • Current memory utilization • Current memory used in bytes • Current free memory in bytes
$time$	Long Integer	Time	Returns the epoch timestamp for logging.

Σχήμα 7.3 Χαρακτηριστικά καταγραφής πόρων

7.4. Υλοποίηση μοντέλου και σύγκριση με άλλα μοντέλα

Το μοντέλο HBES και το μοντέλο παλινδρόμησης πολλαπλών εξόδων GRU (HBES-GRU) υλοποιήθηκαν σε Python 3 χρησιμοποιώντας τις βιβλιοθήκες NumPy, pandas, statsmodels, Scikit-Learn, SciPy, Scikit-Optimize, TensorFlow 2 και το API Keras υψηλότερου επιπέδου. Το περιβάλλον που χρησιμοποιήσαμε είναι το Google Colaboratory. Ο πηγαίος κώδικας του πειράματος είναι διαθέσιμος για κάθε είδους αναπαραγωγή και επανεξέταση στο αποθετήριο GitHub [25].

Συγκρίναμε το HBES-GRU, με μια προσέγγιση βάσης χρονοσειρών, την πρόβλεψη χρήσης πόρων (AUCROP) που βασίζεται σε μετα-μοντέλο ML, το Auto-sklearn, το XGBoost, το προηγούμενο LSTM μοντέλο με τον γενετικό αλγόριθμο (GA-LSTM) και το Keras-Tuner. Όλα αυτά τα μοντέλα έχουν περιγραφεί σε προηγούμενες ενότητες ή περιγράφονται στην συνέχεια.

Το Autoregressive Integrated Moving Average (ARIMA) είναι ένα αρκετά δημοφιλές στατιστικό μοντέλο για προβλέψεις χρονοσειρών που βασίζεται στο autoregression και σταθμίζει μια σειρά από καθυστερημένες παρατηρήσεις με βάση το πόσο πρόσφατες είναι. Το ARIMA έχει χρησιμοποιηθεί για να αποφευχθεί η υπολειτουργία των πόρων ή η υπερβολική παροχή σε κέντρα δεδομένων. Το μοντέλο ARIMA διατίθεται από τα python module statsmodels τα οποία χρησιμοποιήσαμε για την πειραματική μας σύγκριση.

Οι προσεγγίσεις μηχανικής μαθησης δημιουργούν μοντέλα βασισμένα σε ιστορικά δεδομένα για να κάνουν προβλέψεις σε νέες περιπτώσεις. Η μηχανική μάθηση χρησιμοποιείται ευρέως σε υπολογιστές νέφους για τρεις σκοπούς: (α) τον χαρακτηρισμό και πρόβλεψη του φόρτου εργασίας, (β) την τοποθέτηση εικονικών μηχανών και την ενοποίηση συστήματος και (γ) την ελαστικότητα και αποκατάσταση εφαρμογών. Μια πρόσφατη μέθοδος πρόβλεψης φόρτου εργασίας ομαδοποίησης της CPU και RAM έχει προταθεί στο και επισημαίνει τις βελτιώσεις στη διαχείριση πόρων. Ενώ υπάρχουν πολλά διαθέσιμα κλασικά μοντέλα μηχανικής μάθησης, επιλέξαμε για τα πειράματά μας να χρησιμοποιήσουμε το XGBoost [26] επειδή είναι δημοφιλές για τις επιτυχίες του στους Kaggle και άλλους διακεκριμένους διαγωνισμούς μηχανικής μαθησης. Το XGBoost χρησιμοποιεί ως επί το πλείστον δέντρα λήψης αποφάσεων και είναι διαθέσιμο ως βιβλιοθήκη λογισμικού ανοιχτού κώδικα.

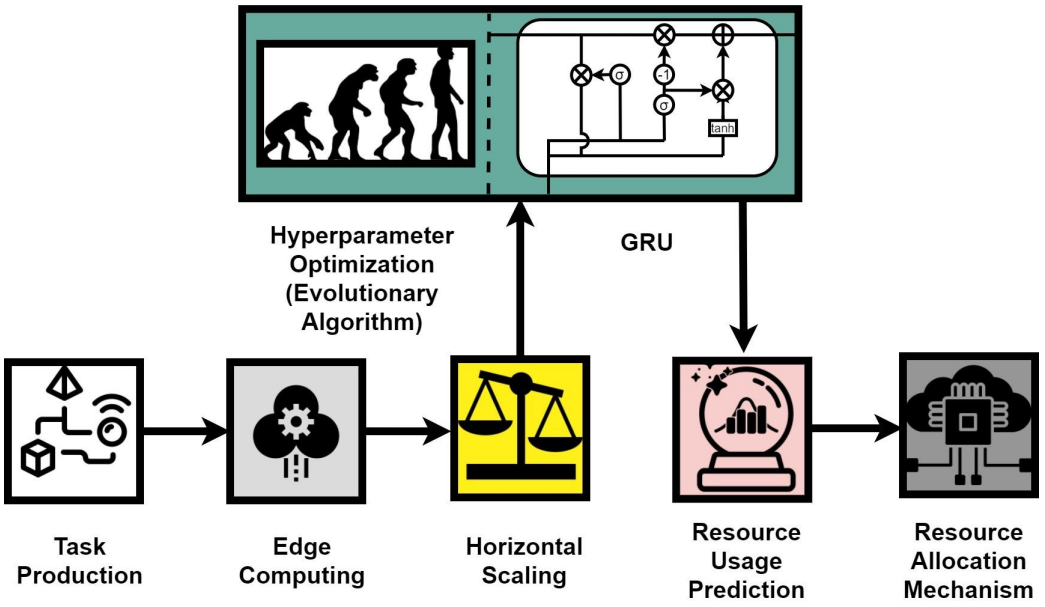
Ο περιορισμός των προσεγγίσεων μηχανικής μαθησης είναι ότι κάθε φορά που η υποδομή cloud-edge, η συμπεριφορά του χρήστη ή η εφαρμογή αλλάζουν, νέα μοντέλα πρέπει να εκπαιδεύονται από το μηδέν με ανθρώπινη βοήθεια. Η αυτοματοποιημένη μηχανική μαθηση επιτυγχάνει με έναν αυτόματο τρόπο καθοδήγηση της μαθησιακής διαδικασίας των μοντέλων, μεγιστοποιώντας την απόδοση, ελαχιστοποιώντας τον υπολογιστικό προϋπολογισμό και χωρίς ανθρώπινη συμμετοχή. Στον τομέα του cloud computing, το Application and User Context Resource Predictor (AUCROP) [27] έχει προταθεί για την αυτοματοποιημένη χρήση κλασικών αλγορίθμων μηχανικής μάθησης. Επιπλέον, ένα γενικό αυτοματοποιημένο μετα-μοντέλο μηχανικής μάθησης για προεπεξεργασία δεδομένων, παλινδρόμηση και επιλογή υπερπαραμέτρων μέσω της βελτιστοποίησης Bayesian είναι το Auto-sklearn [28].

Το Keras-Tuner [29] είναι η προσέγγιση του Keras για την αυτοματοποίηση της επιλογής των υπερπαραμέτρων. Το Keras είναι μία από τις πιο δημοφιλείς βιβλιοθήκες στην κοινότητα του DL. Το Keras-Tuner έχει το πλεονέκτημα ότι τα υπερμοντέλα, το εύρος των υπερπαραμέτρων και η διαδικασία συντονισμού ενσωματώνονται ομαλά στο Keras αλλά υποστηρίζει μόνο ως βελτιστοποιητές, την τυχαία αναζήτηση, το Hyperband, που είναι μια τυχαία αναζήτηση με πρόωρη διακοπή και τη Μπεϋζιανή βελτιστοποίηση. Στην πειραματική μας αξιολόγηση χρησιμοποιήσαμε Keras-Tuner, AUCROP και Auto-sklearn.

Το RNN είναι μια κατηγορία DL μοντέλων ικανών για προγνώσεις χρονοσειρών. Το RNN και συγκεκριμένα το LSTM έχουν χρησιμοποιηθεί επιτυχώς στο παρελθόν για πρόβλεψη χρήσης πόρων που ξεπερνά το μοντέλο ARIMA σε προβλέψεις χρονοσειρών για τη χρήση της CPU. Επίσης έχουμε χρησιμοποιήσει το προηγούμενο μοντέλο πρόβλεψης για την χρήση πόρων GA-LSTM [30] που χρησιμοποιεί γενετικούς αλγορίθμους σε συνδυασμό με LSTM νευρωνικά δίκτυα. Οι γενετικοί αλγόριθμοι είναι μια κατηγορία εξελικτικών αλγορίθμων που μπορούν να χρησιμοποιηθούν για την εξέλιξη υποψήφιων μοντέλων DL μέσω επαναλήψεων crossover, μετάλλαξης και επιλογής.

Τα GRU και LSTM έχουν πολλά κοινά, καθώς και τα δύο βασίζονται σε μοντέλα RNN με το GRU, ωστόσο, να έχει απλούστερη δομή και καλύτερη ευελιξία. Μια ανάλυση των GRU και μια πειραματική σύγκριση μεταξύ αυτών των δύο παραλλαγών του RNN είχε ζητηθεί στο συνέδριο που

παρουσιάσαμε την δημοσίευση [19]. Θέλαμε να επεκτείνουμε την έρευνα των εξελικτικών αλγορίθμων σχετικά με το hypertuning διατηρώντας τα πλεονεκτήματα της Μπευζιανής βελτιστοποίησης. Έτσι προτείνουμε σε αυτό το άρθρο την καινοτόμο HBES ως εξέχουσα αυτοματοποιημένη DL λύση και τη μεθοδολογία GRU-HBES για ακριβή πρόβλεψη χρήσης πόρων.
Το GRU-RNN με το μοντέλο HBES προτείνεται ως ένα ακριβές και αποτελεσματικό μοντέλο πρόβλεψης που ικανοποιεί τις ιδιαιτερότητες των μετρήσεων χρήσης των πόρων στην υπολογιστική των άκρων. Επειδή οι μετρήσεις πόρων όπως η CPU, η μνήμη RAM, ο δίσκος και το εύρος ζώνης εξαρτώνται από την χρονική ακολουθία, το RNN είναι μια κατάλληλη προσέγγιση. Το RNN συνδυάζει τα πλεονεκτήματα του DL με τα χαρακτηριστικά της πρόβλεψης χρονοσειρών. Υπάρχουν διαφορετικοί τύποι αρχιτεκτονικών RNN. Στο παρόν, προτείνουμε την GRU επειδή μπορούν να μάθουν μακροχρόνιες χρονικές εξαρτήσεις, είναι υπολογιστικά αποδοτικές και έχουν καλή απόδοση σε μικρότερα και λιγότερο πυκνά σύνολα δεδομένων.
Η κατάρτιση του RNN είναι μια χρονοβόρα και υπολογιστικά έντονη διαδικασία που εξαρτάται από αρχιτεκτονικές αποφάσεις όπως ο αριθμός των επαναλαμβανόμενων στρωμάτων, τα στρώματα τροφοδοσίας προς τα εμπρός, οι νευρώνες, ο τύπος των συναρτήσεων ενεργοποίησης και οι βελτιστοποιητές. Η μαθησιακή διαδικασία εκτιμά τα βάρη μεταξύ των νευρώνων αλλά, η απόφαση μιας βέλτιστης αρχιτεκτονικής RNN είναι ένα πρόβλημα βελτιστοποίησης που δεν μπορεί να λυθεί με μια αναλυτική προσέγγιση. Αυτό μας ωθεί να προτείνουμε τη μέθοδο HBES που κάνει μια έξυπνη αναζήτηση στο χώρο των αρχιτεκτονικών RNN και ANN. Η μέθοδος HBES συνδυάζει το BO προκειμένου να αναζητήσει τις ονομαστικές αρχιτεκτονικές αποφάσεις όπως τον τύπο των λειτουργιών ενεργοποίησης και το ES που εκτιμά τις αριθμητικές υπερπαραμέτρους όπως τον αριθμό των νευρώνων.



Σχήμα 7.4 Βελτιστοποίηση του GRU-RNN για έξυπνη ενορχήστρωση υπολογιστικών άκρων

Η ροή εργασίας της πρόβλεψης χρήσης πόρων σε ένα περιβάλλον υπολογιστικών άκρων απεικονίζεται στο Σχήμα 7.4. Στην αρχή, οι συσκευές άκρων δημιουργούν εργασίες που εκφορτώθηκαν εν μέρει ή πλήρως στην υποδομή. Η ενορχήστρωση υπολογιστικών άκρων περιλαμβάνει έναν έξυπνο μηχανισμό οριζόντιας κλιμάκωσης που εκτιμά ότι μία κοντά στην βέλτιστη αρχιτεκτονική GRU-RNN χρησιμοποιώντας τον αλγόριθμο HBES. Το GRU-RNN παρέχει την πρόβλεψη χρήσης πόρων που μπορεί να χρησιμοποιηθεί για μια προσαρμοστική κατανομή πόρων,

εκφόρτωση εργασιών ή προνοητική ανοχή σε σφαλμάτα. Στις επόμενες ενότητες, θα περιγράψουμε τα πειραματικά αποτελέσματα των GRU-RNN και του HBES για την πρόβλεψη χρήσης πόρων.

7.5. Αποτελέσματα και συζήτηση αξιολόγησης

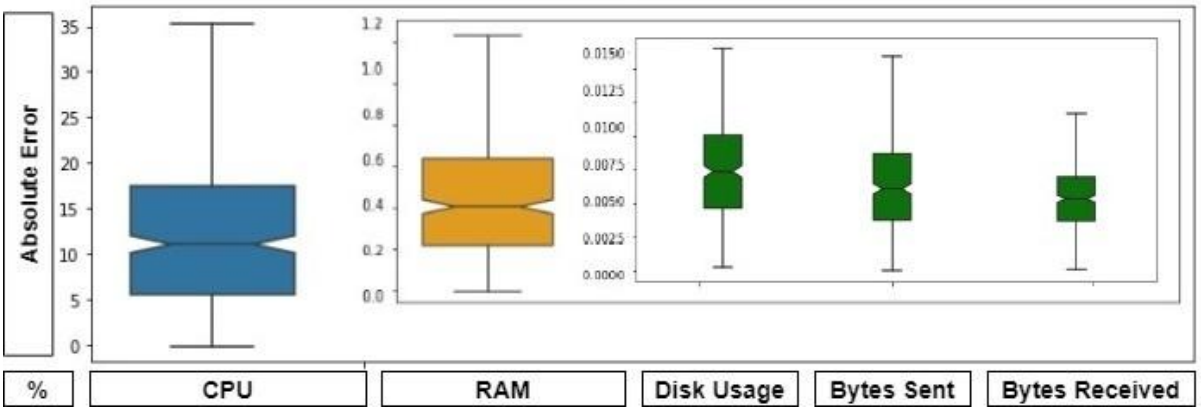
Στην πειραματική αξιολόγηση, ξεκινήσαμε με την ανάλυση χρονοσειρών και βρήκαμε θετικούς συσχετισμούς για καθυστερήσεις σε εύρος τιμών από 1 έως 22 δευτερόλεπτα. Αυτό επιβεβαιώνει την ισχυρή ιδιότητα της ομοιότητας που έχουν οι ακολουθίες των μετρήσεων χρήσης πόρων. Στη συνέχεια, χρησιμοποιώντας το μοντέλο πρόβλεψης ARIMA αξιολογήσαμε τις προβλέψεις των μετρήσεων πόρων. Για παράδειγμα, η CPU RMSE ήταν 18.474. Συγκρίνοντας τα αποτελέσματα των στατιστικών μοντέλων με τις προσεγγίσεις μηχανικής μάθησης και DL διαπιστώσαμε ότι οι DL προσεγγίσεις είχαν μια βελτίωση που ξεπερνά το 20% RMSE στις περισσότερες περιπτώσεις. Από αυτή την άποψη, αποφασίσαμε να επικεντρωθούμε στα μοντέλα μηχανικής μάθησης και DL.
Στον Πίνακα 7.1 συνοψίζουμε τα πειραματικά αποτελέσματα με τα μοντέλα και τις βιβλιοθήκες που περιγράψαμε προηγουμένως. Μπορούμε να δούμε ότι σε όλες τις μετρήσεις η HBES-GRU είχε την καλύτερη ή τη δεύτερη καλύτερη απόδοση. Οι δύο πρώτες στήλες με τίτλο RMSE και MAE παρέχουν για κάθε μοντέλο πρόβλεψης το συγκεντρωτικό RMSE και MAE αντίστοιχα. Αυτές περιλαμβάνουν όλες τις τιμές δοκιμής, τις συσκευές και τις μετρήσεις πόρων. Για το RMSE, το οποίο δίνει ένα επιπλέον βάρος σε προβλέψεις με σημαντικά σφάλματα, μπορούμε να δούμε ότι η HBES-GRU είχε την καλύτερη απόδοση. Στη στήλη με τίτλο MAE, βλέπουμε ότι τα δύο καλύτερα μοντέλα είναι το auto-sklearn και το HBES-GRU. Τα λάθη πρόβλεψής τους είναι πολύ κοντά το ένα με το άλλο και έχουν σημαντικά καλύτερη απόδοση σε σύγκριση με άλλα μοντέλα.

Μέθοδος	RMSE	MAE	CPU-1 (%)		RAM-1 (%)		Χρόνος πρόβλεψης (s)	
			RMSE	MAE	RMSE	MAE	μεμονωμένη	ομαδική
HBES-GRU	0.0641	0.0276	15.918	12.815	1.694	0.580	0.033	0.038
GA-LSTM	0.0674	0.0338	16.099	12.838	1.746	0.917	0.020	0.024
Keras-Tuner	0.0785	0.0377	16.291	13.290	2.631	0.818	0.042	0.042
AUCROP	0.0814	0.0414	17.235	14.009	2.480	1.482	0.004	0.011
XGBoost	0.1139	0.0599	16.457	13.569	1.515	0.472	0.060	0.010
AutoSKLearn	0.1055	0.0243	52.659	17.856	1.546	0.526	0.263	0.572

Πίνακας 7.1 Αξιολόγηση μεθόδων πρόβλεψης.

Οι στήλες CPU-1 και RAM-1 αντιπροσωπεύουν το RMSE και MAE για τον κόμβο επεξεργασίας που είχε τις περισσότερες προβλέψεις στην υποδομή. Επιπλέον, το Σχήμα 7.5 απεικονίζει το 25ο και 75ο εκατοστημόριο, τη διάμεση τιμή, το ελάχιστο και το μέγιστο των μετρήσεων της τιμής σφάλματος. Αυτές οι μετρήσεις περιλαμβάνουν CPU, RAM, χρήση δίσκου και το εύρος ζώνης όσον αφορά τα byte που αποστέλλονται και λαμβάνονται. Από ότι βλέπουμε στη χρήση του δίσκου και το εύρος ζώνης, τα σφάλματα πρόβλεψης ήταν ασήμαντα. Αυτό δεν οφείλεται μόνο στην ικανότητα του HBES-GRU να παρέχει ακριβείς προβλέψεις, αλλά και τη μικρή διακύμανση σε αυτές τις δύο

μετρήσεις πόρων. Η διακύμανση της CPU είναι πολύ ισχυρότερη από τη RAM και το HBES-GRU καταγράφει καλύτερα τις αλλαγές σε σύγκριση με τα άλλα μοντέλα. Το XGBoost έχει καλύτερη απόδοση από το HBES-GRU στη μνήμη RAM. Αυτό μπορεί να δικαιολογηθεί από τη δομή του συνόλου που έχει το XGBoost. Το XGBoost μπορεί να δημιουργήσει συγκεκριμένα δέντρα αποφάσεων για τα error residuals της μνήμης RAM και να αναγνωρισει μια συμπεριφορά αργής αλλαγής.



Σχήμα 7.5. GRU-RNN με σφάλματα πρόβλεψης HBES των μετρήσεων χρήσης πόρων

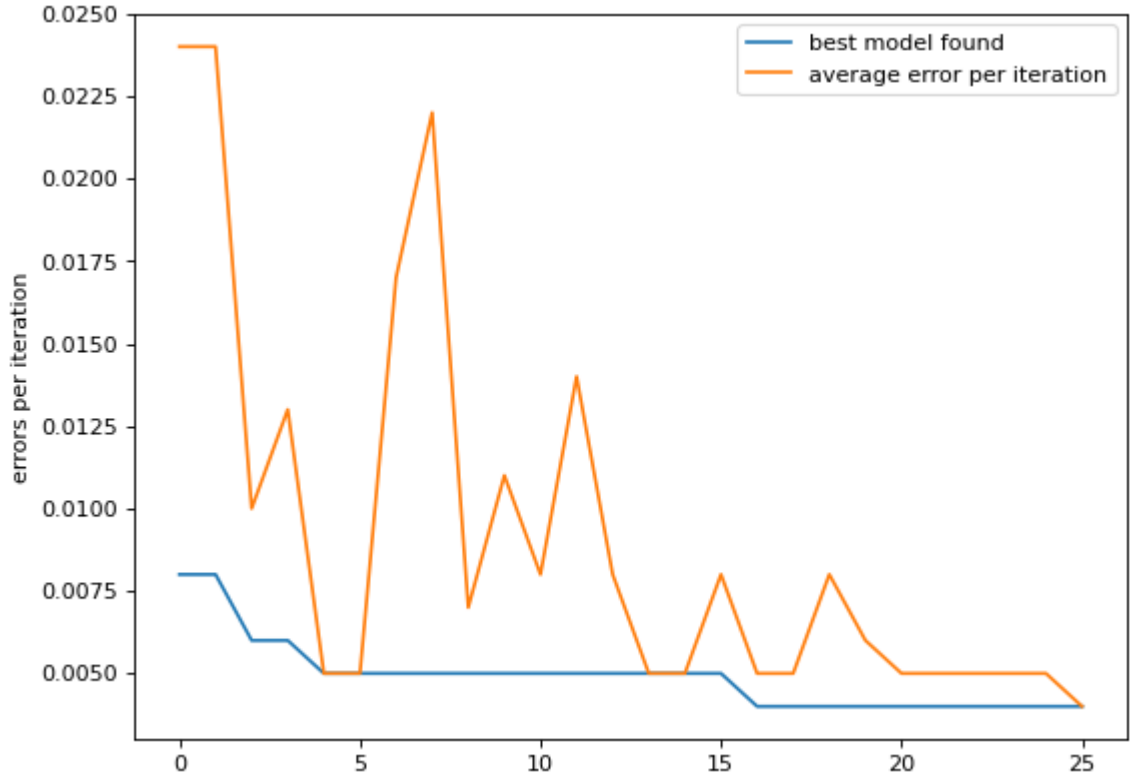
Τέλος, βλέπουμε τους χρόνους για να εξαχθούν τα συμπεράσματα των μοντέλων προκειμένου να παρέχουμε μία μόνο πρόβλεψη ή μια δέσμη με εκατό προβλέψεις. Όλες οι μετρήσεις χρόνου είναι σε δευτερόλεπτα. Όλοι οι χρόνοι συμπερασμάτων, εκτός από αυτές του Auto-sklearn, κυμαίνονται από 4 έως 60 msec. Αυτοί οι χρόνοι συμπερασμάτων καθιστούν την πρόβλεψη χρήσης πόρων μια έγκαιρη διαδικασία και κατάλληλη για εφαρμογές ευαίσθητες στο χρόνο (time sensitive). Σε αυτήν την έρευνα δεν συγκρίναμε τους χρόνους εκπαίδευσης επειδή θέλαμε να κάνουμε μια εξαντλητικά έξυπνη αναζήτηση στο χώρο υπόθεσης και να δούμε τα όρια ακρίβειας που επιτυγχάνουν τα μοντέλα. Έχουμε κάνει πειράματα με πολλαπλά χρονικά διαστήματα. Οι μετρήσεις στον Πίνακα 7.1 είναι με χρόνο 8 λεπτών. Επιλέξαμε να απεικονίσουμε αυτό το χρονικό πλαίσιο επειδή πλησιάζει στον χρόνο ανάπτυξης (deployment) ενός VM.

Βλέπουμε ότι ακόμη και αν οι GRU είναι απλούστερες στη δομή τους σε σύγκριση με το LSTM και δεν διαθέτουν την πύλη εξόδου, έχουν καλύτερη απόδοση. Αυτό το συμπέρασμα επιβεβαιώνεται από τη βιβλιογραφία σε περίπτωση μικρών και λιγότερο συχνών παρατηρήσεων δεδομένων που χαρακτηρίζουν την πρόβλεψη χρήσης πόρων [9].

Τρία σημαντικά συμπεράσματα που έχουμε είναι: Στις περισσότερες μετρήσεις τα μοντέλα DL (HBES-GRU, GA-LSTM, Keras-Tuner) έχουν καλύτερη απόδοση από τα μοντέλα μηχανικής μάθησης (AUCROP, XGBoost, Auto-sklearn). Στη συνέχεια, οι εξελικτικοί αλγόριθμοι (HBES-GRU και GA-LSTM) έχουν καλύτερη απόδοση σε σύγκριση με την απλή Μπεϋζιανή βελτιστοποίηση του Keras-Tuner. Τελευταίο αλλά όχι λιγότερο σημαντικό, βλέπουμε μια σημαντική βελτίωση χρησιμοποιώντας την υβριδική προσέγγιση HBES και τη στρατηγική εξέλιξης σε σύγκριση με τους απλούς γενετικούς αλγόριθμους.

7.6. Σύγκλιση της υβριδικής εξελικτικής στρατηγικής Bayesian

Η σύγκλιση και η θέση του ολικού βέλτιστου είναι δύο από τα πιο σημαντικά θέματα στον τομέα των εξελικτικών αλγορίθμων. Η σύγκλιση σημαίνει ότι καθώς ο πληθυσμός εξελίσσεται, τα άτομα πλησιάζουν τη βέλτιστη λύση, μειώνοντας το σφάλμα τους. Αλλά δεν μπορούμε να είμαστε σίγουροι αν τα σημεία σύγκλισης στο χώρο του γονότυπου είναι τα ολικά ή τα τοπικά ελάχιστα. Για το λόγο αυτό, ο αλγόριθμος HBES στην αρχή της διαδικασίας εξέλιξης εκφράζει μια ισχυρή διακύμανση στη μετάλλαξη που εξασθενεί με την πάροδο των επαναλήψεων. Ταυτόχρονα, διατηρούμε τον καλύτερο γονότυπο που βρέθηκε σε όλες τις επαναλήψεις.



Σχήμα 7.6. Η σύγκλιση του HBES για σχεδόν το βέλτιστο RNN.

Η σύγκλιση του HBES απεικονίζεται στο Σχήμα 7.6 Βλέπουμε ότι στην αρχή το μέσο σφάλμα του πληθυσμού ανά επανάληψη έχει μια έντονη διακύμανση. Σε ορισμένες επαναλήψεις εγκλωβίζεται στα τοπικά ελάχιστα όπως για παράδειγμα στις επαναλήψεις έξι έως έντεκα. Σε μερικές άλλες επαναλήψεις βρισκεται σε περιοχές οροπέδιων όπως βλέπουμε στις επαναλήψεις είκοσι έως είκοσι πέντε. Όμως, τελικά χρησιμοποιώντας τη μετάλλαξη τα άτομα ξεφεύγουν από τις περιοχές του οροπέδιου και τα τοπικά ελάχιστα και μετακινούνται προς τις βέλτιστες περιοχές.
Αυτές οι περιοχές που βρίσκονται κοντά στις βέλτιστες περιοχές στον χώρο του γονότυπου αποκωδικοποιούνται στην πλησιέστερη στις βέλτιστες αρχιτεκτονικές GRU-RNN στον χώρο των φαινοτύπων. Αυτές οι αρχιτεκτονικές GRU-RNN παρέχουν τις πιο ακριβείς προβλέψεις χρήσης πόρων για χρήση CPU, RAM, δίσκου και εύρους ζώνης σε υποδομή υπολογιστών άκρων.

8. Συμπεράσματα

Στην παρούσα διπλωματική εργασία παρουσιάσαμε το μοντέλο των νευρωνικών δικτύων, την διαδικασία εκπαίδευσης του και βελτιστοποίησης των υπερπαραμέτρων του. Έπειτα παρουσιάσαμε πώς μπορούμε να εφαρμόσουμε μια προσέγγιση βαθιάς μάθησης με νευρώνες LSTM προκειμένου να αντιμετωπίσουμε το πρόβλημα της πρόβλεψης της επόμενης θέσης ενός κινούμενου αντικειμένου. Είδαμε ότι το μοντέλο βαθιάς μάθησης μπορεί να είναι ένα ισχυρό εργαλείο με αποτελέσματα υψηλής ακρίβειας, εάν έχει μια καλά σχεδιασμένη αρχιτεκτονική. Χρησιμοποιήσαμε μια ευρετική προσέγγιση για να καταλήξουμε στην αρχιτεκτονική ANN χρησιμοποιώντας έναν γενετικό αλγόριθμο. Επιπλέον, είδαμε ότι όταν ένα ANN αρχικοποιείται στην αρχή της εκπαιδευτικής διαδικασίας με τις παραμέτρους ενός καλά εκπαιδευμένου μοντέλου σε ένα διαφορετικό σύνολο δεδομένων, τότε χρειάζεται πολύ λιγότερο χρόνο εκπαίδευσης από μια τυχαία προετοιμασία παραμέτρων.

Πιστεύουμε ότι βρισκόμαστε στην αρχή μιας μακροχρόνιας έρευνας πάνω στην χρήση των RNN για τις ανάγκες της ανάλυσης τροχιάς. Τα θέματα που δεν έχουμε αγγίξει καν σε αυτήν τη διπλωματική είναι πολλά. Πρώτον, πρέπει να διερευνηθεί περαιτέρω η μέθοδος ανάκτησης των τοπολογιών ANN στο πλαίσιο της μεταφοράς γνώσης. Ο λόγος που βρίσκουμε ομάδες κοινών μοτίβων στα ANN μοντέλα θα πρέπει να εξηγηθεί. Πρέπει να το θεωρήσουμε ως ένα τυχαίο αποτέλεσμα του γενετικού αλγορίθμου ή προκαλούνται πραγματικά από κοινές συμπεριφορές στην κίνηση; Θα πρέπει να εφαρμόσουμε διαφορετικές τεχνικές βελτιστοποίησης υπερπαραμέτρων και να συγκριθούν τα αποτελέσματά τους.

Μια ενδιαφέρον τεχνική στο DL είναι οι μηχανισμοί προσοχής (Attention) που επιτρέπουν στο ANN να επικεντρωθεί σε κάποια μέρη της ακολουθίας εισόδου περισσότερο από κάποια άλλα κατά τη διαδικασία πρόβλεψης. Είναι μια ανοιχτή ερώτηση εάν και πώς μπορεί να εφαρμοστεί ο μηχανισμός προσοχής προκειμένου να βελτιωθεί η ακρίβεια ενός μοντέλου πρόβλεψης επόμενης θέσης, και αν είναι πιθανό να υπάρξει κάποια σημαντική βελτίωση στα αποτελέσματα.

Το δεύτερο σενάριο αξιολόγησης προτείνει την εφαρμογή ANN για την πρόβλεψη χρήσης πόρων που έχει σκοπό την έξυπνη ενορχήστρωση μιας υποδομής υπολογιστικής των άκρων. Συγκεκριμένα, παρουσιάσαμε πώς η προσαρμοστική κατανομή πόρων, η εκφόρτωση εργασιών και η προληπτική ανοχή σφαλμάτων μπορούν να βελτιωθούν από την πρόβλεψη της χρήσης πόρων. Προτείναμε, αναλύσαμε και αξιολογήσαμε πειραματικά τη χρήση μιας προσέγγισης χρονοσειρών με GRU-RNN για τη μοντελοποίηση χρήσης πόρων.

Η δυναμικότητα και η ετερογένεια του φόρτου εργασίας και των κόμβων των άκρων μας ώθησαν να διεξάγουμε έρευνα για μια συστηματική προσέγγιση για την κατασκευή μιας βέλτιστης ANN αρχιτεκτονικής. Σχεδιάσαμε έναν αλγόριθμο hypertuning που συνδυάζει την εξελικτική στρατηγική με τη Μπεϋζιανή βελτιστοποίηση και είναι σημαντικό επίτευγμα ότι η μέθοδος αυτή ξεπερνά ευρέως χρησιμοποιούμενα μοντέλα όπως το Keras-Tuner.

Είμαστε πεπεισμένοι για τη μεθοδολογία μας και τη δυνατότητα εφαρμογής των RNN και συγκεκριμένα των GRU για την πρόβλεψη χρήσης πόρων. Ο κύριος στόχος μας ως επόμενα βήματα είναι να δημιουργήσουμε μηχανισμούς εκφόρτωσης εργασιών και ανοχής σφαλμάτων που αξιοποιούν την πρόβλεψη χρήσης πόρων με ANN σε ένα περιβάλλον προσομοιωσης όπως το CloudSim Plus.

Ευρετήριο όρων

DL	Deep Learning	Βαθιά Μάθηση
ANN	Artificial Neural Network	Τεχνητό Νευρωνικό Δίκτυο
SGD	Stochastic Gradient Descent	Στοχαστική Κατάβαση Κλίσης
RNN	Recurrent Neural Network	Αναδρομικά Νευρωνικά Δίκτυα
LSTM	Long Short-Term Memory	Μακράς-Βραχείας Μνήμης
MLP	Multi-Layer Perceptron	Πολυστρωματικό νευρωνικό δίκτυο
ML	Machine Learning	Μηχανική Μάθηση
MSE	Mean Squared Error	Μέσο Τετραγωνικό Σφάλμα
MAE	Mean Absolute Error	Μέσο Απόλυτο Σφάλμα
RMSE	Root Mean Squared Error	Ρίζα του Μέσου Τετραγωνικού Σφάλματος
IoT	Internet of Things	Διαδίκτυο των πραγμάτων
ReLU	Rectified Linear Unit	Διορθωμένης Γραμμικής Μονάδας
Tanh	Hyperbolic Tangent Function	Συνάρτηση Υπερβολικής Εφαπτομένης
SELU	Scaled Exponential Linear Unit	Κλιμακούμενη Εκθετική Γραμμική Μονάδα
MLF	Multi Layer Feedforward	Πολυστρωματικά Τροφοδοτούμενα προς τα Εμπρός
BRNN	Bidirectional Recurrent Neural Network	Αμφίδρομο Επαναλαμβανόμενο Νευρωνικό Δίκτυο
GRU	Gated Recurrent Unit	Ανατροφοδοτούμενη Μονάδας με Πύλη
CNN	Convolutional Neural Network	Συνελικτικά Νευρωνικά Δίκτυα

ES	Evolutionary Strategy	Εξελικτική Στρατηγική
BO	Bayesian Optimization	Μπεϋζιανή Βελτιστοποίηση
RMSProp	Root Mean Square Propagation	Ρίζα της Μέσης Τετραγωνικής Διάδοσης
Adagrad	Adaptive Gradient Algorithm	Προσαρμοστικός Αλγόριθμος Κλησης
Adam	Adaptive Moment Estimation	Προσαρμοστική Εκτίμηση Ροπής
HBES	Hybrid Bayesian Evolution Strategy	Υβριδική Μπεϋζιανή Εξελικτική Βελτιστοποίηση
SLA	Service Level Agreements	Συμφωνίες Επιπέδου Εξυπηρέτησης

Βιβλιογραφικές αναφορές

[1] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[2] "TensorFlow." https://www.tensorflow.org/?authuser=1 (accessed Mar. 28, 2021).

[3] J. Violos, K. Tserpes, A. Papaoikonomou, M. Kardara, and T. Varvarigou, "Clustering Documents using the 3-Gram Graph Representation Model," 2014, pp. 1–5, doi: 10.1145/2645791.2645812.

[4] P. Ramachandran, B. Zoph, and Q. V. Le, "Searching for Activation Functions," *ArXiv171005941 Cs*, Oct. 2017, Accessed: Mar. 28, 2021. [Online]. Available: http://arxiv.org/abs/1710.05941.

[5] D. Svozil, V. Kvasnicka, and J. Pospichal, "Introduction to multi-layer feed-forward neural networks," *Chemom. Intell. Lab. Syst.*, vol. 39, no. 1, pp. 43–62, Nov. 1997, doi: 10.1016/S0169-7439(97)00061-0.

[6] M. Hüsken and P. Stagge, "Recurrent neural networks for time series classification," *Neurocomputing*, vol. 50, pp. 223–235, Jan. 2003, doi: 10.1016/S0925-2312(01)00706-8.

[7] M. Schuster and K. Paliwal, "Bidirectional recurrent neural networks," *Signal Process. IEEE Trans. On*, vol. 45, pp. 2673–2681, Dec. 1997, doi: 10.1109/78.650093.

[8] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997, doi: 10.1162/neco.1997.9.8.1735.

[9] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling," *ArXiv14123555 Cs*, Dec. 2014, Accessed: Mar. 28, 2021. [Online]. Available: http://arxiv.org/abs/1412.3555.

[10] S. Albawi, T. A. Mohammed, and S. Al-Zawi, "Understanding of a convolutional neural network," in *2017 International Conference on Engineering and Technology (ICET)*, Aug. 2017, pp. 1–6, doi: 10.1109/ICEngTechnol.2017.8308186.

[11] "How to Use Convolutional Neural Networks for Time Series Classification | by Rehan Ahmed Sayyad | Medium." https://medium.com/@Rehan_Sayyad/how-to-use-convolutional-neural-networks-for-time-series-classification-80575131a474 (accessed Mar. 28, 2021).

[12] D. Choi, C. J. Shallue, Z. Nado, J. Lee, C. J. Maddison, and G. E. Dahl, "On Empirical Comparisons of Optimizers for Deep Learning," *ArXiv191005446 Cs Stat*, Jun. 2020, Accessed: Mar. 28, 2021. [Online]. Available: http://arxiv.org/abs/1910.05446.

[13] S. Zeng, J. Gou, and L. Deng, "An antinoise sparse representation method for robust face recognition via joint l1 and l2 regularization," *Expert Syst. Appl.*, vol. 82, pp. 1–9, Oct. 2017, doi: 10.1016/j.eswa.2017.04.001.

[14] A. Labach, H. Salehinejad, and S. Valaee, "Survey of Dropout Methods for Deep Neural Networks," *ArXiv190413310 Cs*, Oct. 2019, Accessed: Mar. 28, 2021. [Online]. Available: http://arxiv.org/abs/1904.13310.

[15] L. Prechelt, "Early Stopping - But When?," in *Neural Networks: Tricks of the Trade*, G. B. Orr and K.-R. Müller, Eds. Berlin, Heidelberg: Springer, 1998, pp. 55–69.

[16] D. van Kuppevelt, C. Meijer, F. Huber, A. van der Ploeg, S. Georgievska, and V. T. van Hees, "Mcfly: Automated deep learning on time series," *SoftwareX*, vol. 12, p. 100548, Jul. 2020, doi: 10.1016/j.softx.2020.100548.

[17] M. Srinivas and L. M. Patnaik, "Genetic algorithms: a survey," *Computer*, vol. 27, no. 6, pp. 17–26, Jun. 1994, doi: 10.1109/2.294849.

[18] I. Rechenberg, "Evolution Strategy: Nature's Way of Optimization," in *Optimization: Methods and Applications, Possibilities and Limitations*, Berlin, Heidelberg, 1989, pp. 106–126, doi: 10.1007/978-3-642-83814-9_6.

[19] P. I. Frazier, "A Tutorial on Bayesian Optimization," *ArXiv180702811 Cs Math Stat*, Jul. 2018, Accessed: Mar. 28, 2021. [Online]. Available: http://arxiv.org/abs/1807.02811.

[20] J. Violos, S. Tsanakas, M. Androutsopoulou, G. Palaiokrassas, and T. Varvarigou, "Next Position Prediction using LSTM Neural Networks," in *11th Hellenic Conference on Artificial Intelligence*,

New York, NY, USA, Sep. 2020, pp. 232–240, doi: 10.1145/3411408.3411426.
[21] A. Valsamis, K. Tserpes, D. Zissis, D. Anagnostopoulos, and T. Varvarigou, "Employing traditional machine learning algorithms for big data streams analysis: The case of object trajectory prediction," *J. Syst. Softw.*, vol. 127, pp. 249–257, May 2017, doi: 10.1016/j.jss.2016.06.016.
[22] M. Feurer and F. Hutter, "Hyperparameter Optimization," in *Automated Machine Learning: Methods, Systems, Challenges*, F. Hutter, L. Kotthoff, and J. Vanschoren, Eds. Cham: Springer International Publishing, 2019, pp. 3–33.
[23] J. H. Friedman, "Stochastic Gradient Boosting," *Comput Stat Data Anal*, vol. 38, no. 4, pp. 367–378, Feb. 2002, doi: 10.1016/S0167-9473(01)00065-2.
[24] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, and C. Liu, "A Survey on Deep Transfer Learning," in *Artificial Neural Networks and Machine Learning – ICANN 2018*, Cham, 2018, pp. 270–279, doi: 10.1007/978-3-030-01424-7_27.
[25] "STsanakas - Overview," *GitHub*. https://github.com/STsanakas (accessed Mar. 28, 2021).
[26] T. Chen and C. Guestrin, "XGBoost: A Scalable Tree Boosting System," *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discov. Data Min.*, pp. 785–794, Aug. 2016, doi: 10.1145/2939672.2939785.
[27] J. Violos, E. Psomakelis, K. Tserpes, F. Aisopos, and T. Varvarigou, "Leveraging User Mobility and Mobile App Services Behavior for Optimal Edge Resource Utilization," in *Proceedings of the International Conference on Omni-Layer Intelligent Systems*, Crete, Greece, May 2019, pp. 7–12, doi: 10.1145/3312614.3312620.
[28] M. Feurer, A. Klein, K. Eggensperger, J. T. Springenberg, M. Blum, and F. Hutter, "Auto-sklearn: Efficient and Robust Automated Machine Learning," in *Automated Machine Learning: Methods, Systems, Challenges*, F. Hutter, L. Kotthoff, and J. Vanschoren, Eds. Cham: Springer International Publishing, 2019, pp. 113–134.
[29] "Keras Tuner." https://keras-team.github.io/keras-tuner/ (accessed Feb. 20, 2021).
[30] J. Violos, E. Psomakelis, D. Danopoulos, S. Tsanakas, and T. Varvarigou, "Using LSTM Neural Networks as Resource Utilization Predictors: The Case of Training Deep Learning Models on the Edge," in *Economics of Grids, Clouds, Systems, and Services*, Cham, 2020, pp. 67–74, doi: 10.1007/978-3-030-63058-4_6.

Παράρτημα

Κώδικας Πειραματική αξιολόγηση με πρόβλεψη τροχιάς σε κινούμενα πλοία

```
mypath=''
dataset = mypath+'Datasets/'+'Demo.csv'
from os import path
import os
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2'
import numpy as np
import pandas as pd
import time
import random
import statistics
import pandas
import math
import csv
import random
import logging
from functools import reduce
from operator import add
from tqdm import tqdm
import geopy.distance
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error
from tensorflow import keras
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import LSTM
from tensorflow.keras.layers import Activation
from tensorflow.keras.callbacks import EarlyStopping
from Conv2Ang import transform_dataset

training_percentage=0.67
myverbose=0
if (mypath==''):
        print('Project folder not set')
        print('Please set your folder path in TrajectoryPredictionLSTM.py in line 1 and run the code again')
        exit()
EarlyStopper = EarlyStopping(patience=3, monitor='loss', mode='min')
best_score=10000000.0


def preprocessing(dataset):
  dataset_X = transform_dataset(dataset)
  dataset_Y = np.delete(dataset_X, 0, 0)
  dataset_X = np.delete(dataset_X, -1, 0)
  Coords = pd.read_csv(dataset, engine='python').values.astype('float32')
  Coords=np.roll(Coords, -1, axis=0)
  scaler_X = MinMaxScaler(feature_range=(0, 1))
  scaler_X.fit(dataset_X)
  dataset_X = scaler_X.transform(dataset_X)
  scaler_Y = MinMaxScaler(feature_range=(0, 1))
```

```
  scaler_Y.fit(dataset_Y)
  dataset_Y = scaler_Y.transform(dataset_Y)
  train_size = int(len(dataset_X) * training_percentage)
  trainX, testX = dataset_X[0:train_size,:], dataset_X[train_size:len(dataset_X)+1,:]
  trainY, testY = dataset_Y[0:train_size,:], dataset_Y[train_size:len(dataset_X)+1,:]
  Coords = Coords[train_size:len(dataset_X)+1,:]
  return trainX, trainY, testX, testY, scaler_X, scaler_Y, Coords

def compile_model(network, trainX, trainY):
  nb_neurons=[]
  activation=[]
  # Get our network parameters.
  nb_layers = network['nb_layers']
  lstms=network['lstms']
  implementation1=network['implementation1']
  units1=network['units1']
  lstm_activation1=network['lstm_activation1']
  recurrent_activation1=network['recurrent_activation1']
  implementation2=network['implementation2']
  units2=network['units2']
  lstm_activation2=network['lstm_activation2']
  recurrent_activation2=network['recurrent_activation2']
  nb_neurons.append(network['nb_neurons1'])
  nb_neurons.append(network['nb_neurons2'])
  nb_neurons.append(network['nb_neurons3'])
  nb_neurons.append(network['nb_neurons4'])
  nb_neurons.append(network['nb_neurons5'])
  activation.append(network['activation1'])
  activation.append(network['activation2'])
  activation.append(network['activation3'])
  activation.append(network['activation4'])
  activation.append(network['activation5'])
  optimizer = network['optimizer']
  model = Sequential()

  if(lstms==1):
                        model.add(LSTM(units1, input_shape=(1, 2), activation=lstm_activation1,
recurrent_activation=recurrent_activation1, implementation=implementation1))
  elif (lstms==2):
                        model.add(LSTM(units1, input_shape=(1, 2), activation=lstm_activation1,
recurrent_activation=recurrent_activation1, implementation=implementation1, return_sequences=True))
                model.add(LSTM(units2, activation=lstm_activation2, recurrent_activation=recurrent_activation2,
implementation=implementation2))
  for i in range(nb_layers):
            model.add(Dense(nb_neurons[i], activation=activation[i]))
  model.add(Dense(2))
  model.compile(loss='mean_squared_error', optimizer=optimizer)
  trainX=trainX.reshape(len(trainX),1,2)
  model.fit(trainX, trainY, epochs=100, batch_size=1, verbose=myverbose, callbacks=[EarlyStopper])
  return model, lstms

def evaluate(model, trainX, testX, trainY, testY, scaler_X, scaler_Y, Coords):
  global best_score
  if not isinstance(model.get_layer(index=0), keras.layers.Dense):
```

```
    testX=testX.reshape(len(testX),1,2)
    testPredict = scaler_Y.inverse_transform(model.predict(testX))
    testY = scaler_Y.inverse_transform(testY)
    testScore=0
    predictions=0
    for i in range(len(Coords)-1):
      R = 6378.1 #Radius of the Earth
      brng = testY[i,1]*(math.pi/180) #Bearing is 90 degrees converted to radians.
      d = testY[i,0] #Distance in km
      brng3 = testPredict[i,1]*(math.pi/180) #Bearing is 90 degrees converted to radians.
      d3 = testPredict[i,0] #Distance in km
      lat1 = math.radians(Coords[i,0]) #Current lat point converted to radians
      lon1 = math.radians(Coords[i,1]) #Current long point converted to radians
      lat3 = math.asin( math.sin(lat1)*math.cos(d3/R) + math.cos(lat1)*math.sin(d3/R)*math.cos(brng3))
      lon3 = lon1 + math.atan2(math.sin(brng3)*math.sin(d3/R)*math.cos(lat1), math.cos(d3/R)-math.sin(lat1)*math.sin(lat3))
      true=(Coords[i+1,0],Coords[i+1,1])
      mypred=(math.degrees(lat3),math.degrees(lon3))
      testScore += geopy.distance.geodesic(true, mypred).km
      predictions=predictions+1
    testScore=testScore/predictions
    if(best_score>testScore):
      best_score=testScore
      model.save(mypath+'Models/bestLSTMnetworkGenetic.h5')
    return testScore

  def train_and_score(network, dataset):
    trainX, trainY, testX, testY, scalerX, scalerY, Coords = preprocessing(dataset)
    model, lstms = compile_model(network, trainX, trainY)
    error = evaluate(model, trainX, testX, trainY, testY, scalerX, scalerY, Coords)
    return error


  class Network():

    def __init__(self, nn_param_choices=None):
                self.accuracy = 0.
                self.nn_param_choices = nn_param_choices
                self.network = {}  # (dic): represents MLP network parameters

    def create_random(self):
                for key in self.nn_param_choices:
                            self.network[key] = random.choice(self.nn_param_choices[key])

    def create_set(self, network):
                self.network = network

    def train(self, dataset):
                if self.accuracy == 0.:
                            self.accuracy = train_and_score(self.network, dataset)

    def print_network(self):
      print("Network error: %.2f m" % (1000*self.accuracy))
      print("%d LSTM layers" % self.network['lstms'])
      if (self.network['lstms']>0):
```

```
        print("first LSTM layer: ", self.network['units1'], self.network['lstm_activation1'], self.network['recurrent_activation1'], self.network['implementation1'])
        if (self.network['lstms']>1):
                print("second LSTM layer: ", self.network['units2'], self.network['lstm_activation2'], self.network['recurrent_activation2'], self.network['implementation2'])
    print("First Hidden Layer: %d neurons, with" % self.network['nb_neurons1'], self.network['activation1'], "activation")
    if (self.network['nb_layers']>1):
        print("Second Hidden Layer: %d neurons, with" % self.network['nb_neurons2'], self.network['activation2'], "activation")
    if (self.network['nb_layers']>2):
        print("Third Hidden Layer: %d neurons, with" % self.network['nb_neurons3'], self.network['activation3'], "activation")
    if (self.network['nb_layers']>3):
        print("Fourth Hidden Layer: %d neurons, with" % self.network['nb_neurons4'], self.network['activation4'], "activation")
    if (self.network['nb_layers']>4):
        print("Fifth Hidden Layer: %d neurons, with" % self.network['nb_neurons5'], self.network['activation5'], "activation")
    print('Optimizer: ', self.network['optimizer'])
    print('Model is saved in '+mypath+'Models/bestLSTMnetworkGenetic.h5')
    print('-'*80)



class Optimizer():

  def __init__(self, nn_param_choices, retain=0.4, random_select=0.1, mutate_chance=0.2):
        self.mutate_chance = mutate_chance
        self.random_select = random_select
        self.retain = retain
        self.nn_param_choices = nn_param_choices

  def create_population(self, count):
        pop = []
        for _ in range(0, count):
            # Create a random network.
            network = Network(self.nn_param_choices)
            network.create_random()
            # Add the network to our population.
            pop.append(network)
        return pop

  @staticmethod
  def fitness(network):
        return network.accuracy

  def grade(self, pop):
        summed = reduce(add, (self.fitness(network) for network in pop))
        return summed / float((len(pop)))

  def breed(self, mother, father):
        children = []
        for _ in range(2):
            child = {}
```

```
                # Loop through the parameters and pick params for the kid.
                for param in self.nn_param_choices:
                        child[param] = random.choice([mother.network[param], father.network[param]])
                # Now create a network object.
                network = Network(self.nn_param_choices)
                network.create_set(child)
                # Randomly mutate some of the children.
                if self.mutate_chance > random.random():
                        network = self.mutate(network)
                children.append(network)
        return children

def mutate(self, network):
        # Choose a random key.
        mutation = random.choice(list(self.nn_param_choices.keys()))
        # Mutate one of the params.
        network.network[mutation] = random.choice(self.nn_param_choices[mutation])
        return network

def evolve(self, pop):

        # Get scores for each network.
        graded = [(self.fitness(network), network) for network in pop]

        # Sort on the scores.
        graded = [x[1] for x in sorted(graded, key=lambda x: x[0], reverse=False)]

        # Get the number we want to keep for the next gen.
        retain_length = int(len(graded)*self.retain)

        # The parents are every network we want to keep.
        parents = graded[:retain_length]

        # For those we aren't keeping, randomly keep some anyway.
        for individual in graded[retain_length:]:
                if self.random_select > random.random():
                        parents.append(individual)

        # Now find out how many spots we have left to fill.
        parents_length = len(parents)
        desired_length = len(pop) - parents_length
        children = []

        # Add children, which are bred from two remaining networks.
        while len(children) < desired_length:

                # Get a random mom and dad.
                male = random.randint(0, parents_length-1)
                female = random.randint(0, parents_length-1)

                # Assuming they aren't the same network...
                if male != female:
                        male = parents[male]
                        female = parents[female]
```

```
                                # Breed them.
                                babies = self.breed(male, female)

                                # Add the children one at a time.
                                for baby in babies:
                                        # Don't grow larger than desired length.
                                        if len(children) < desired_length:
                                                children.append(baby)

                parents.extend(children)

                return parents


def train_networks(networks, dataset):

  pbar = tqdm(total=len(networks))
  for network in networks:
                network.train(dataset)
                pbar.update(1)
  pbar.close()


def generate(generations, population, nn_param_choices, dataset):
  optimizer = Optimizer(nn_param_choices)
  networks = optimizer.create_population(population)

  # Evolve the generation.
  for i in range(generations):
                print("***Doing generation %d of %d***" %(i + 1, generations))

                # Train and get accuracy for networks.
                train_networks(networks, dataset)
                networks = sorted(networks, key=lambda x: x.accuracy, reverse=False)


                print('Best score up until now:', int(1000*best_score))
                print('-'*80)

                # Evolve, except on the last iteration.
                if i != generations - 1:
                        # Do the evolution.
                        networks = optimizer.evolve(networks)

  # Sort our final population.
  networks = sorted(networks, key=lambda x: x.accuracy, reverse=False)
  networks[:1][0].print_network()


def main():
  generations = 10  # Number of times to evole the population.
  population = 20  # Number of networks in each generation.
```

```
  nn_param_choices = {
            'lstms':[1,2],
            'implementation1':[1,2],
            'units1':[2,8,16,32,64,128],
            'lstm_activation1':['tanh', 'tanh', 'sigmoid', 'relu', 'linear', 'hard_sigmoid'],
            'recurrent_activation1':['hard_sigmoid', 'tanh', 'sigmoid', 'relu', 'linear', 'hard_sigmoid'],
            'implementation2':[1,2],
            'units2':[2,8,16,32,64,128],
            'lstm_activation2':['tanh', 'tanh', 'sigmoid', 'relu', 'linear', 'hard_sigmoid'],
            'recurrent_activation2':['hard_sigmoid', 'tanh', 'sigmoid', 'relu', 'linear', 'hard_sigmoid'],
            'nb_layers': [1, 2, 3, 4, 5],
    'nb_neurons1': [2,8,16,32,64,128],
            'activation1': ['tanh', 'sigmoid', 'linear', 'relu'],
    'nb_neurons2': [2,8,16,32,64,128],
    'activation2': ['tanh', 'sigmoid', 'linear', 'relu'],
    'nb_neurons3': [2,8,16,32,64,128],
    'activation3': ['tanh', 'sigmoid', 'linear', 'relu'],
    'nb_neurons4': [2,8,16,32,64,128],
    'activation4': ['tanh', 'sigmoid', 'linear', 'relu'],
    'nb_neurons5': [2,8,16,32,64,128],
    'activation5': ['tanh', 'sigmoid', 'linear', 'relu'],
            'optimizer': ['rmsprop', 'adam', 'sgd', 'adagrad', 'adadelta', 'adamax', 'nadam'],
  }
  print("***Evolving %d generations with population %d***" %(generations, population))
  generate(generations, population, nn_param_choices, dataset)

main()
```

Κώδικας: πειραματική αξιολόγηση με πρόβλεψη χρήσης πόρων σε υποδομές των άκρων

```
!pip install scikit-optimize
platform='colab'
import pandas as pd
import numpy as np
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import Flatten
from tensorflow.keras.layers import Dropout
from tensorflow.keras.layers import Activation
from tensorflow import keras
from tensorflow.keras.layers import Conv1D, GRU
from tensorflow.keras.layers import LSTM
from tensorflow.keras.layers import Flatten
from sklearn.metrics import mean_absolute_error
import numpy.ma as ma
import tqdm
import math
import re
import os
import random
import time
from skopt.space import Categorical, Real
from skopt.utils import use_named_args
from skopt import gp_minimize
import tensorflow
from tensorflow.keras import optimizers
from tensorflow.keras.callbacks import EarlyStopping
EarlyStopper = EarlyStopping(patience=3, monitor='loss', mode='min')
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2'
best_error=1
np.set_printoptions(precision=3)
import warnings
mypath= createpath(platform)
warnings.filterwarnings('ignore', message='The objective has been evaluated at this point before.')

dimensions  = [Categorical(['tanh','sigmoid','linear','relu'], name='activationFunction'), #Search space, all parameters are nomminal
        Categorical(['RMSProp','Adam','SGD','Adagrad', 'Adadelta', 'Adamax', 'Nadam'], name='optimizer')]
defact = ['tanh','sigmoid','linear','relu']
defopt =  ['RMSProp','Adam','SGD','Adagrad', 'Adadelta', 'Adamax', 'Nadam']
default_parameters = [random.choice(defact), random.choice(defopt)]
@use_named_args(dimensions=dimensions) #Combine Objective function with its search space
def gp_minimize_opt_function(activationFunction, optimizer):
  global dataset, myparams, bayes_time, bt
  bayes_time=bayes_time+time.time()-bt
  loss = compile_model(activationFunction, optimizer)
  bt=time.time()
  return loss
```

```
def split_sequences(sequences, n_steps_in, n_steps_out):
	X, y = list(), list()
	for i in range(len(sequences)):
		end_ix = i + n_steps_in
		out_end_ix = end_ix + n_steps_out
		if out_end_ix > len(sequences):
			break
		seq_x, seq_y = sequences[i:end_ix, :], sequences[end_ix:out_end_ix, :]
		X.append(seq_x)
		y.append(seq_y)
	return np.array(X), np.array(y)

def clear(search, mypath):
  filenamestart='bestES'+search
  prefixed = [filename for filename in os.listdir(mypath+'Models/') if filename.startswith(filenamestart)]
  for i in range(len(prefixed)):
    os.remove(mypath+'Models/'+prefixed[i])
  return

def metrics(model, X_test, y_test, scaler):
	prediction=model.predict(X_test)
	mse = mean_squared_error(y_test, prediction)
	mae = mean_absolute_error(y_test, prediction)
	y_test=scaler.inverse_transform(y_test)
	prediction=scaler.inverse_transform(prediction)
	mse_ram = mean_squared_error(y_test[:,10], prediction[:,10])
	mae_ram = mean_absolute_error(y_test[:,10], prediction[:,10])
	mse_cpu = mean_squared_error(y_test[:,4], prediction[:,4])
	mae_cpu = mean_absolute_error(y_test[:,4], prediction[:,4])
	return mse, mae, mse_ram, mae_ram, mse_cpu, mae_cpu

def inference(model, X):
	i=random.randrange(len(X))
	startS=time.time()
	prediction=model.predict(X[[i]])
	timeS=time.time()-startS
	i=random.randrange(len(X)-100)
	startB=time.time()
	prediction=model.predict(X[i:i+101])
	timeB=time.time()-startB
	return timeS, timeB

def average_params(params, prev):
	params=np.array(params)
	myparams = ma.masked_array(params, mask=[0])
	for i in range(len(params)):
		layers=int(round(1 + (params[i][0]*4)))
		if (layers<2):
			myparams[i] = ma.masked_array(params[i], mask=[0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0,
0])
```

```
                elif (layers<3):
                        myparams[i] = ma.masked_array(params[i], mask=[0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0,
0])
                elif (layers<4):
                        myparams[i] = ma.masked_array(params[i], mask=[0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0,
0])
                elif (layers<5):
                        myparams[i] = ma.masked_array(params[i], mask=[0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0,
0])
                if (params[i][16]<0.5):
                        myparams[i] = ma.masked_array(myparams[i], mask=[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 1])
                        print(myparams[i])
        best=ma.array((myparams[0], myparams[1], myparams[2], myparams[3])).mean(axis=0)
        print(best)
        for i in range(len(best)):
                if (ma.is_masked(best[i])):
                        best[i]=prev[i]
        print(best)
        return best

def Results(mydatasetfile, model, method):
        dataset_name = mydatasetfile.split("/")[-1]
        trainX,trainY,testX,testY,scaler=data_preparation(mydatasetfile)
        model.fit(trainX, trainY, epochs = 100, validation_data = (testX, testY))
        mse, mae, mse_ram, mae_ram, mse_cpu, mae_cpu = functions.metrics(model, testX, testY, scaler)
        infS, infB = functions.inference(model, testX)

        print(method+' Algorithm results using', dataset_name, 'as dataset')
        print('Best mse: %.6f  mae: %.6f  rmse: %.6f  training time: %.0f s' % (mse, mae,
math.sqrt(mse), mytime))
        print('Single inference time: %.3f s  batch inference time: %.3f s ' % (infS, infB))
        print('RAM mse: %.6f  RAM mae: %.6f  RAM rmse: %.6f' % (mse_ram, mae_ram, math.sqrt(mse_ram)))
        print('CPU mse: %.6f  CPU mae: %.6f  CPU rmse: %.6f' % (mse_cpu, mae_cpu, math.sqrt(mse_cpu)))
        print(method+',%.0f,%.5f,%.5f,%.0f,%.0f,%.3f,%.3f,%.3f,%.3f' % (mytime, math.sqrt(mse), mae, 1000*infS,
1000*infB, math.sqrt(mse_cpu), mae_cpu, math.sqrt(mse_ram), mae_ram),
file=open(mypath+'Results/ResourcePredictionResults.csv','a'))

def StartTimer():
        return time.time()
def StopTimer(start):
        return time.time()-start


def remove_dupes(i, o):
        for j in range(len(i)):
                for k in range(len(i)):
                        if (i[j]==i[k] and j!=k):
                                if (o[j]>o[k]):
                                        i.pop(j)
                                        o.pop(j)
                                        return remove_dupes(i, o)
                                else:
                                        i.pop(k)
```

```
                                        o.pop(k)
                                        return remove_dupes(i, o)
        return i, o

def createpath(platform):
        if (platform=='pc'):
                path = 'local path'
        elif (platform=='colab'):
                path= 'drive path'
        return path



def descale(scaled, limits):
  descaled = limits[0] + (scaled*(limits[1]-limits[0]))
  return descaled

def getOptimizer(opt,lr):
  if opt=='RMSProp':
    myopt = optimizers.RMSprop(learning_rate=descale(lr,learningrateL))
  elif opt=='Adam':
    myopt = optimizers.Adam(learning_rate=descale(lr,learningrateL))
  elif opt=='SGD':
    myopt = optimizers.SGD(learning_rate=descale(lr,learningrateL))
  elif opt=='Adagrad':
    myopt = optimizers.Adagrad(learning_rate=descale(lr,learningrateL))
  elif opt=='Adadelta':
    myopt = optimizers.Adadelta(learning_rate=descale(lr,learningrateL))
  elif opt=='Adamax':
    myopt = optimizers.Adamax(learning_rate=descale(lr,learningrateL))
  elif opt=='Nadam':
    myopt = optimizers.Nadam(learning_rate=descale(lr,learningrateL))
  return myopt


def data_preparation(datasetfile):
  dataframe = pd.read_csv(datasetfile, engine='python')
  dataset = dataframe.values
  dataset = dataset.astype('float32')
  scaler = MinMaxScaler(feature_range=(0, 1))
  scaler.fit(dataset)
  dataset = scaler.transform(dataset)
  return dataset, scaler

def compile_model(activation, optimizer):
        #params = [0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5]
        global dataset, myparams, best_error
        if (search=='dense'):
                lookback = 1
        else:
                lookback = int(round(descale(myparams[15],layersL)))
        graph = tensorflow.Graph()
        with tensorflow.compat.v1.Session(graph=graph):
                 opt=getOptimizer(optimizer, myparams[13])
```

```
		if (search=='dense'):
			lookback = 1
		else:
			lookback = int(round(descale(myparams[15],layersL)))
		model = Sequential()

		if (search=='lstm'):
			retseq=False
			if (myparams[16]>0.5):
				retseq=True
			  model.add(LSTM(int(round(descale(myparams[14],neuronsL))), input_shape=(lookback, dataset.shape[1]), activation=activation, recurrent_activation='sigmoid', return_sequences=retseq))
			if (myparams[16]>0.5):
				model.add(LSTM(int(round(descale(myparams[17],neuronsL))), input_shape=(lookback, dataset.shape[1]), activation=activation, recurrent_activation='sigmoid'))
		elif (search=='conv'):
			model.add(Conv1D(filters=int(round(descale(myparams[14],neuronsL))), kernel_size=lookback, activation=activation, input_shape=(lookback, dataset.shape[1])))
			if (myparams[16]>0.5):
				model.add(Conv1D(filters=int(round(descale(myparams[17],neuronsL))), kernel_size=1, activation=activation))
			model.add(Flatten())
		elif (search=='gru'):
			retseq=False
			if (myparams[16]>0.5):
				retseq=True
				model.add(GRU(units=int(round(descale(myparams[14],neuronsL))), return_sequences=retseq))
			if (myparams[16]>0.5):
				model.add(GRU(units=int(round(descale(myparams[17],neuronsL)))))

		for i in range(1,int(round(descale(myparams[0],layersL)+1))):
			model.add(Dense(round(descale(myparams[(2*i)-1],neuronsL)), activation=activation))
			model.add(Dropout(descale(myparams[2*i],dropoutL)))
		model.add(Dense(dataset.shape[1]))
		model.compile(loss='mse',
	optimizer=opt,
	metrics=['mae', 'mse'])
		datasetX,datasetY=split_sequences(dataset,lookback,1)
		if (search=='dense'):
			datasetX=datasetX.reshape(datasetX.shape[0],datasetX.shape[2])
		datasetY=datasetY.reshape(datasetY.shape[0],datasetY.shape[2])
		train_size = int(len(datasetX) * 0.67)
		trainX, testX = datasetX[0:train_size,:], datasetX[train_size:len(dataset),:]
		trainY, testY = datasetY[0:train_size,:], datasetY[train_size:len(dataset),:]
		  model.fit(trainX, trainY, epochs=int(round(descale(myparams[11],epochsL))), batch_size=int(round(descale(myparams[12],batchsizeL))), verbose=0, callbacks=[EarlyStopper])

		loss, _, _ = model.evaluate(testX, testY, verbose=0)
		if (loss<best_error):
			clear(search, mypath)
			best_error = loss
			model.save(mypath+'Models/bestES'+search+'_'+str(lookback)+'.h5')
		del model, datasetX, datasetY, trainX, testX, trainY, testY
```

```
        return loss

def eval_model(params):
        global best_error, dataset, myparams, bn, bayes_inputs, bayes_results, bayes_time, bt
        calls=3
        myparams=params
        bt=time.time()
        if not bayes_inputs:
                bayes = gp_minimize(func=gp_minimize_opt_function, dimensions=dimensions, acq_func='EI', n_calls=calls, n_random_starts=1, x0=default_parameters, model_queue_size=1)
        else:
                bayes = gp_minimize(func=gp_minimize_opt_function, dimensions=dimensions, acq_func='EI', n_calls=calls, n_random_starts=1, x0=bayes_inputs, y0=bayes_results, model_queue_size=1)
        bayes_time=bayes_time+time.time()-bt
        bn=bn+1
        bayes_inputs.extend(bayes.x_iters)
        bayes_results.extend(bayes.func_vals)
        keep=calls*bn
        bayes_inputs=bayes_inputs[-keep:]
        bayes_results=bayes_results[-keep:]
        bayes_inputs, bayes_results=remove_dupes(bayes_inputs, bayes_results)
        error=min(bayes_results[-calls:])
        return error

mydatasetfile=mypath+'Datasets/train_imdb.csv'
search='gru'
iterations=30
neuronsL=[2,200]
epochsL=[20,200]
learningrateL=[0.001,0.1]
batchsizeL=[1,128]
dropoutL=[0.0,0.5]
layersL=[1,5]
bn=0
bayes_inputs = []
bayes_results = []
time_total=[]
bayes_time_total=[]
bt=0
bayes_time=0
def optimize(params, top_n = 5, n_pop = 20, n_iter = 10, sigma_error = 0.15, error_weight = 1, decay_rate = 0.95, min_error_weight = 0.01 ):
 global dataset
 # Model weights have been randomly initialized at first
 best_params = params



 for i in range(n_iter):
          # Generating the population of parameters
           pop_params = [best_params + error_weight*sigma_error*np.random.randn(*np.shape(params)) for i in range(n_pop)]
 pop_params = [x.clip(0, 1)  for x in pop_params]
```

```
 # Evaluating the population of parameters
 evaluation_values=[eval_model(parameters) for parameters in pop_params]
 #print(evaluation_values)
 average=np.average(evaluation_values)
 # Sorting based on evaluation score
 param_eval_list = zip(evaluation_values, pop_params)

 param_eval_list = sorted(param_eval_list, key = lambda x: x[0], reverse = False)

 evaluation_values, pop_params = zip(*param_eval_list)

 # Taking the mean of the elite parameters
 best_params = average_params(pop_params[:top_n], best_params)
 #Decaying the weight
 error_weight = max(error_weight*decay_rate, min_error_weight)

 params = best_params
 return best_params, param_eval_list[0][0], average # Instantiating our model object
# Running it for 200 steps
best_params = [0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5]
dataset, myscaler=data_preparation(mydatasetfile)
best_periteration=np.ones(iterations)
average_periteration=np.ones(iterations)
start=time.time()
for i in tqdm.tqdm(range(iterations)):
        best_params, best_periteration[i], average_periteration[i]= optimize(best_params, top_n = 4, n_pop = 10, n_iter =
3)
        print(best_periteration)
        print(average_periteration)
        bayes_time_total.append(bayes_time)
        time_total.append(time.time()-start)

mytime=time.time()-start

print(best_periteration)
print(average_periteration)
filenamestart='bestES'+search
prefixed = [filename for filename in os.listdir(mypath+'Models/') if filename.startswith(filenamestart)]
modelname=prefixed[0]
lookback = int(re.findall("(\d+)", modelname)[0])

mymodel=keras.models.load_model(mypath+'Models/'+modelname)
mydatasetfile=mypath+'Datasets/train_imdb.csv'
dataset_name = mydatasetfile.split("/")[-1]
dataset, scaler = data_preparation(mydatasetfile)
dataset,datasetY=split_sequences(dataset,lookback,1)
if (search=='dense'):
  dataset=dataset.reshape(dataset.shape[0],dataset.shape[2])
datasetY=datasetY.reshape(datasetY.shape[0],datasetY.shape[2])
train_size = int(len(dataset) * 0.67)
test_size = len(dataset) - train_size
trainX, testX = dataset[0:train_size,:], dataset[train_size:len(dataset),:]
trainY, testY = datasetY[0:train_size,:], datasetY[train_size:len(dataset),:]
mse, mae, mse_ram, mae_ram, mse_cpu, mae_cpu = metrics(mymodel, testX, testY, scaler)
```

```
infS, infB = inference(mymodel, testX)


print(search+' Evolution Strategy Algorithm results using', dataset_name, 'as dataset')
print('Best mse: %.6f mae: %.6f rmse: %.6f training time: %.0f s' % (mse, mae, math.sqrt(mse), mytime))
print('Single inference time: %.3f s batch inference time: %.3f s ' % (infS, infB))
print('RAM mse: %.6f RAM mae: %.6f RAM rmse: %.6f' % (mse_ram, mae_ram, math.sqrt(mse_ram)))
print('CPU mse: %.6f CPU mae: %.6f CPU rmse: %.6f' % (mse_cpu, mae_cpu, math.sqrt(mse_cpu)))
print('Evolution'+search+',%.0f,%.5f,%.5f,%.0f,%.0f,%.3f,%.3f,%.3f,%.3f' % (mytime, math.sqrt(mse), mae, 1000*infS, 1000*infB, math.sqrt(mse_cpu), mae_cpu, math.sqrt(mse_ram), mae_ram), file=open(mypath+'Results/ResourcePredictionResults.csv','a'))


import matplotlib.pyplot as plt
plt.plot(best_periteration)
plt.plot(average_periteration)
plt.ylabel('errors per iteration')
plt.savefig(mypath+'Results/'+search+'ESconvergence.png', bbox_inches='tight')
```