Εργασία 3, Επίλυση προβλήματος ταξινόμησης με χρήση Multi-layer Pereceptron δικτύου

Νίκος Λαδιάς

Αύγουστος 2021

Η εργασία είναι υλοποιημένη σε ένα αρχείο .ipynb με την βοήθεια του Jupyter Notebook , όπου μας δίνεται η δυνατότητα να εκτελούμε κελιά κώδικα κάθε φορά και να βλέπουμε τα αντίστοιχα μηνύματα της κονσόλας ακριβώς κάτω απο το κάθε κελί. Αυτό μας προσφέρει καλύτερη οπτικοποίηση των αποτελεσμάτων και οργάνωση του κώδικα. Σε κάθε κελί υπάρχουν και τα αντίστοιχα σχόλια (στα Αγγλικά) επεξήγησης του κώδικα.

Περιεχόμενα

1 Διερεύνηση απόδοσης μοντέλου για διαφορετικές τεχνικές σχεδίασης και εκπαίδευσης

Αφού αρχικά εγκαταστάθηκαν οι κατάλληλες βιβλιοθήκες tensorflow, keras και οποιαδήποτε άλλη χρειάστηκε και ενσωματώθηκαν στον κώδικα για την χρήση τους αργότερα, ενσωματώθηκε και η βάση δεδομένων MNIST. Η οποία είναι η βάση πάνω στην οποία η εργασία βασίζεται. Όπως ορίζει η εκφώνηση, δηλώνονται όλες οι απαραίτητες παράμετροι ενώ σε οποιεσδήποτε δεν αναφέρονται ρητά η τιμή τους, θεωρείται οτι λαμβάνει by default μια τιμή. Τα νευρωνικά δίκτυα που θα σχεδιαστούν και θα εκπαιδευτούν με minibatch μέθοδο με batch size= 256 και για 100 εποχές. Έπειτα γίνεται η κατάλληλη προεπεξεργασία των δεδομένων καθώς και η κανονικοποίηση των τιμών των εικόνων στο [0 , 1] . Σύμφωνα με την εκφώνηση ένα 20% του συνόλου των δεδομένων εκπαίδευσης παρακρατείται για

χρήση ως σύνολο επικύρωσης, αυτό ακριβώς συμβαίνει στο επόμενο κελί. Η minibatch λογική θα χρησιμοποιηθεί για την εκπαίδευση όλων των δικτύων ως συνδυασμός της μεθόδου με τις υπόλοιπες , καθώς παρατηρήθηκες πως **η εκπαίδευση των δικτύων με χρήση minibatch λογικής επιταχύνει πολύ την διαδικασία εκπαίδευσης**. Για κάθε συνδυασμό μεθόδων τυπώνονται τα αντίστοιχα ζητούμενα γραφήματα (καμπύλες ακρίβειας και κόστους για training και validation sets) στο αμέσως επόμενο κελί κώδικα. Τα μοντέλα των νευρωνικών δικτύων, οι ζητούμενες ενεργοποιήσεις των στρωμάτων, καθώς και η εκπαίδευση τους με γίνεται με την βοήθεια του keras. Κατα την διαδικασία compile η συνάρτηση κόστους είναι η CategoricalCrossentropy όπως ζητείται .

Έτσι πρώτα ορίζονται δύο μοντέλα νευρωνικών δικτύων : Ένα με RMSprop optimizer και ένα με SGD opimizer. Το μοντέλο που χρησιμοποιεί για optimize τον SGD optimizer χρησιμοποιεί επίσης και έναν initializer για την αρχικοποίηση των συναπτικών βαρών κάθε στρώματος (hidden layers) με βάση μια κανονική κατανομή με mean=10, όπως άλλωστε αναφέρεται και στην εκφώνηση. Για τον RMSprop optimizer δημιουργούνται δύο optimizers με βάση την παραμέτρο ρ (rho), χρησιμοποιούνται και οι δύο εναλλάξ στα μοντέλα που φτιάχνουμε για εκπαίδευση. Σε κάθε παραγόμεο γράφημα απεικονίζονται τα δεδομένα εκπαίδευσης (train) με μπλέ και τα δεδομένα επικύρωσης (test) με πορτοκαλί.

1.1 RMSprop optimizer και SGD optimizer

Οι επιπτώσεις της αρχικοποίησης των βαρών στα νευρωνικά δίκτυα

Η απόδοση ενός νευρωνικού δικτύου εξαρτάται σε μεγάλο βαθμό από τον τρόπο με τον οποίο αρχικοποιούνται οι παράμετροί του όταν ξεκινάει η εκπαίδευσή του. Επιπλέον, αν το αρχικοποιήσουμε τυχαία για κάθε εκτέλεση, είναι βέβαιο ότι θα είναι μη επαναλήψιμο (σχεδόν) και ακόμη και όχι τόσο αποδοτικό επίσης. Από την άλλη πλευρά, αν το αρχικοποιήσουμε με συνεχείς τιμές, μπορεί να του πάρει πάρα πολύ χρόνο για να συγκλίνει. Με αυτό, εξαλείφουμε επίσης την ομορφιά της τυχαιότητας, η οποία με τη σειρά της δίνει σε ένα νευρωνικό δίκτυο τη δύναμη να φτάσει σε σύγκλιση πιο γρήγορα χρησιμοποιώντας μάθηση βασισμένη στην κλίση.

Μελέτες έχουν δείξει ότι η αρχικοποίηση των βαρών με τιμές δειγματοληψίας από μια τυχαία κατανομή αντί για σταθερές τιμές, όπως μηδενικά και μονάδες, βοηθάει στην πραγματικότητα ένα νευρωνικό δίκτυο να εκπαιδευτεί καλύτερα και ταχύτερα. Η επιβαλλόμενη τυχαιότητα δεν είναι μόνο πολύ κατάλληλη για τεχνικές βελτιστοποίησης που βασίζονται σε κλίση, αλλά βοηθά επίσης ένα δίκτυο να καθοδηγεί καλύτερα ποια βάρη θα ενημερώσει. Διαισθητικά, με μια σταθερή αρχικοποίηση βαρών, όλες οι έξοδοι των επιπέδων κατά το αρχικό πέρασμα προς τα εμπρός ενός δικτύου είναι ουσιαστικά οι ίδιες και αυτό καθιστά πολύ δύσκολο για το δίκτυο να καταλάβει ποια βάρη πρέπει να ενημερωθούν.

Ιδανικά θα θέλαμε οι τιμές του διανύσματος βαρών να είναι τέτοιες ώστε να μην καταλήγουν να προκαλούν απώλεια δεδομένων στο διάνυσμα εισόδου. Τελικά πολλαπλασιάζουμε το διάνυσμα βαρών με το διάνυσμα εισόδου, οπότε πρέπει να είμαστε πολύ προσεκτικοί. Έτσι, είναι συχνά μια καλή πρακτική να κρατάμε τις τιμές του διανύσματος βαρών να είναι όσο το δυνατόν μικρότερες, αλλά όχι πολύ μικρές ώστε να καταλήγουν να προκαλούν αριθμητική αστάθεια. Αρχικοποιώντας τα βάρη με βάση μια κανονική κατανομή , οι περισσότερες τιμές θα συγκεντρώνονται στην περιοχή της μέσης τιμής. Έτσι προκειμένου να κρατήσουμε αυτές τις αρχικές τιμές βαρών κοντά στο 0, θα χρησιμοποιούσαμε μια κανονική κατανομή με μέση τιμή το 0. Απο την εκφώνηση όμως μας δίνεται μια μέση τιμή 10. Τι συμβαίνει τότε;

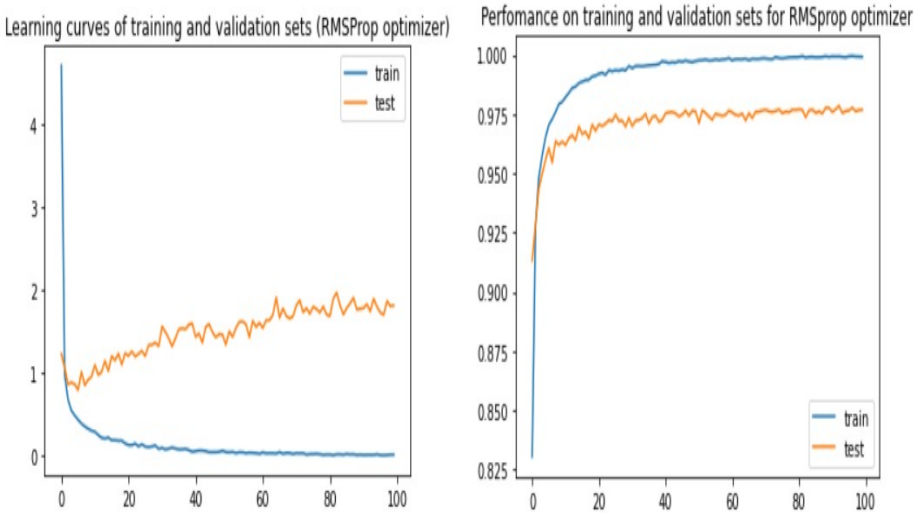
Όπως βλέπουμε καθαρά στην εικόνα 1, η απώλεια επικύρωσης και η απώλεια εκπαίδευσης αποκλίνουν σε μεγάλο βαθμό η μία από την άλλη και η ακρίβεια επικύρωσης παραμένει σταθερή σε όλες τις εποχές. Αυτό δείχνει ότι το μοντέλο μας πραγματικά δυσκολεύεται να εκπαιδευτεί. Αυτό συμβαίνει κυρίως επειδή αρχικά τα βάρη του μοντέλου μας ξεκινούν με μεγάλες τιμές . Ως εκ τούτου, οι ενημερώσεις των βαρών που συμβαίνουν λόγω της οπισθοδιάδοσης δεν αποδεικνύονται αρκετά αποτελεσματικές για να συγκλίνει το μοντέλο.

(a) Καμπύλη κόστους για τα training και validation sets με SGD optimizer

(b) Καμπύλη ακρίβειας για τα training και validation sets με SGD optimizer

Εικόνα 1: Καμπύλες εκπαίδευσης για το πρώτο μοντέλο με χρήση SGD optimizer και αρχικοποίηση των βαρών

Όσο αφορά το μοντέλο του RMSprop optimizer, όπως βλέπουμε στην εικόνα 2, το σφάλμα στα δεδομένα εκπαίδευσης μειώνεται συνεχώς και γρήγορα πιάνει το μηδέν. Αντιθέτως όμως στα δεδομένα επικύρωσης, το σφάλμα μειώνεται μέχρι



(a) Καμπύλη κόστους για τα training και validation sets με RMSprop optimizer

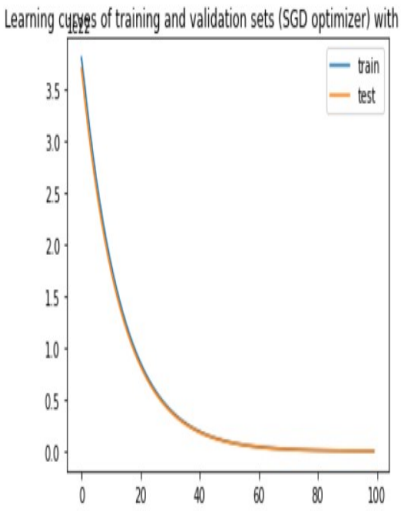
(b) Καμπύλη ακρίβειας για τα training και validation sets με RMSprop optimizer

Εικόνα 2: Καμπύλες εκπαίδευσης για το πρώτο μοντέλο με χρήση RMSprop optimizer

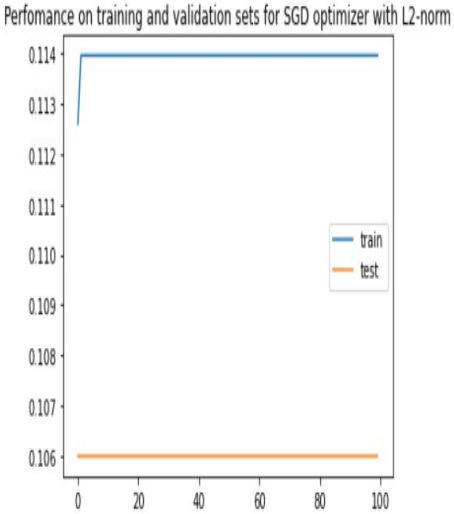
ένα σημείο απο το οποίο και μετά αυξάνεται συνεχώς. Απο εκείνο το σημείο και μετά το μοντέλο έχει περάσει στο overfitting. Αυτό επιβεβαιώνεται και απο την καμπύλη ακρίβειας του μοντέλου όπου για τα δεδομένα εκπαίδευσης φτάνει το 1 γρήγορα ,ενώ για τα δεδομένα επικύρωσης απο το αντίστοιχο πάλι σημείο και μετά παύει να αυξάνεται.

1.2 Κανονικοποίηση με L_2 νόρμα

Αξιοποιώντας τους διαθέσιμους regularizers που προσφέρει το API του keras, ξαναχτίζονται δύο πανομοιότυπα μοντέλα με πρίν με μόνη διαφορά την προσθήκη του regularizer L_2 με παράμετρο κανονικοποίησης α=0.01. Τα αποτελέσματα για το μοντέλο που χρησιμοποιεί τον SGD optimizer (ο οποίος με βάση την εκφώνηση έρχεται πάντα πακέτο με την αρχικοποίηση των βαρών με τον τρόπο που αναφέρθηκε) είναι ελαφρώς καλύτερα, καθώς το κόστος στα δεδομένα επικύρωσης φαίνεται να ακολουθάει το κόστος των δεδομένων εκπαίδευσης. Παρόλα αυτά η ακρίβεια εδώ παραμένει πολύ χαμηλή και παραμένει σταθερή και για τα δύο datasets, αφού στη προσπάθεια να αποφύγουμε το overfitting με κάποιον regularizer , το πρόβλημα της κακής εκπαίδευσης του μοντέλου λόγω κακής αρχικοποίησης των βαρών παραμένει.



(a) Καμπύλη κόστους για τα training και validation sets



(b) Καμπύλη ακρίβειας για τα training και validation sets

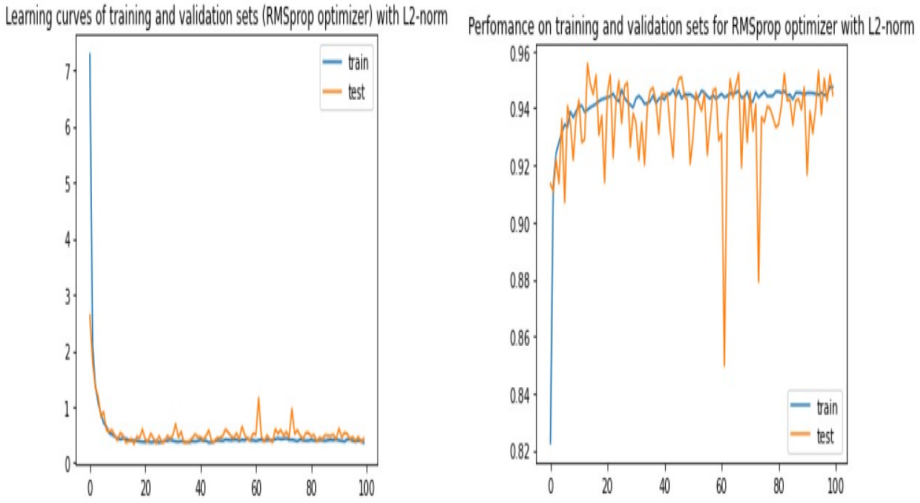
Για το μοντέλο με τον RMSprop optimizer, παρατηρείται πως η λειτουργία της κανονικοποίησης κατάφερε να εξαλείψει το φαινόμενο του overfitting. Η καμπύλη κόστους για τα δεδομένα τώρα ακολουθεί αυτήν των δεδομένων εκπαίδευσης, με κάποιες αποκλίσεις αλλά όχι και πολύ ανησυχητικές. Αυτό επιβεβαιώνεται και απο τις καμπύλες ακρίβειας, όπου η ακρίβεια επικύρωσης φαίνεται να έχει μεγάλες αποκλίσεις, με βάση την κλίμακα όμως οι απόκλισεις είναι μικρές.

1.3 Κανονικοποίηση με L_1-νόρμα και χρήση dropout layers

Το Dropout είναι μια μέθοδος κανονικοποίησης που προσεγγίζει την παράλληλη εκπαίδευση ενός μεγάλου αριθμού νευρωνικών δικτύων με διαφορετικές αρχιτεκτονικές.

Κατά τη διάρκεια της εκπαίδευσης, κάποιος αριθμός εξόδων στρώματος αγνοείται τυχαία ή "εγκαταλείπεται". Αυτό έχει ως αποτέλεσμα το στρώμα να μοιάζει και να αντιμετωπίζεται όπως ένα στρώμα με διαφορετικό αριθμό κόμβων και συνδεσιμότητα σε σχέση με το προηγούμενο στρώμα. Το Dropout μπορεί να εφαρμοστεί σε οποιοδήποτε ή όλα τα κρυφά στρώματα του δικτύου, καθώς και στο ορατό στρώμα ή στο στρώμα εισόδου. Δεν χρησιμοποιείται στο στρώμα εξόδου.

Εισάγεται μια νέα υπερπαράμετρος που καθορίζει την πιθανότητα με την οποία οι έξοδοι του στρώματος εγκαταλείπονται, ή αντίστροφα, την πιθανότητα με την



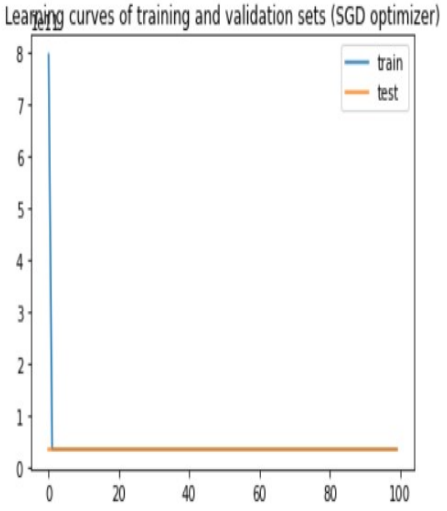
(a) Καμπύλη κόστους για τα training και validation sets

(b) Καμπύλη ακρίβειας για τα training και validation sets

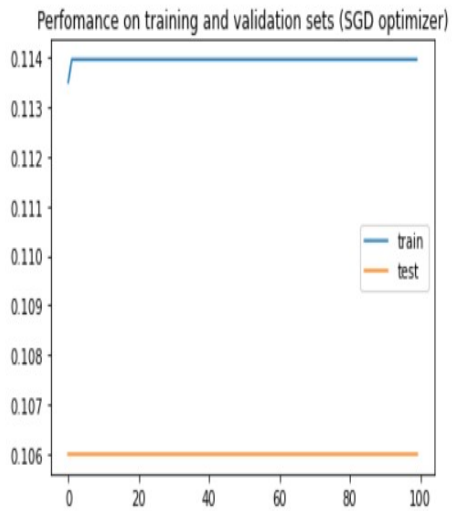
οποία οι έξοδοι του στρώματος διατηρούνται. Μια κοινή τιμή είναι μια πιθανότητα 0.5 για τη διατήρηση της εξόδου κάθε κόμβου σε ένα κρυφό στρώμα και μια τιμή κοντά στο 1.0, όπως 0.8, για τη διατήρηση των εισόδων από το ορατό στρώμα. Η δική μας πιθανότητα είναι 0.3. Στα δικά μας καινούργια μοντέλα Dropout στρώματα τοποθετούνται στα δύο κρυφά στρώματα του δικτύου. Οι υπόλοιπες ιδιότητες του δικτύου παραμένουν ως έχει εκτός απο προφανώς και την προσθήκη της κανονικοποίησης με L_1-νόρμα.

Για το μοντέλο με SGD optimizer, με την χρήση των dropout layers , η απόδοση του μοντέλου εκπαίδευσης φαίνεται να μην έχει επηρεαστεί ως προς την ακρίβεια του μοντέλου. Ως προς την ταχύτητα εκπαίδευσης φαίνεται το σφάλμα να πέφτει σχεδόν κατακόρυφα απο μια αρκετά μεγάλη τιμή, άρα παρουσιάζεται μεγαλύτερη ταχύτητα σύγκλισης. Γενικά όπως προαναφέρθηκε η αρχικοποίηση των βαρών σε αυτό το μοντέλο δεν αφήνει πολλά περιθώρια βελτίωσης με οποιδήποτε τρόπο regularization.

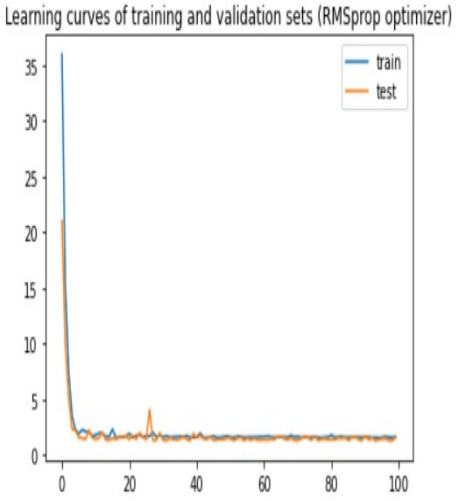
Στο μοντέλο του RMSprop optimizer, το κόστος φαίνεται να παρακολουθεί ικανοποιητικά το κόστος στα δεδομένα εκπαίδευσης ,ενώ η ταχύτητα σύγκλισης του κόστους στο μηδέν φαίνεται επίσης να είναι πολύ μεγαλύτερη. Παρόλα αυτά η ακρίβεια ειδικά στο validation set , επηρεάστηκε αρκετά, με την καμπύλη της να μην είναι πλέον τόσο ομαλή, αλλά να παρουσιάζει "spikes"(=καρφιά). Αυτό συμβαίνει λόγω της τυχαιότητας με την οποία νευρώνες αποκόπτονται απο το δίκτυο απο επανάληψη σε επανάληψη λόγω του dropout.



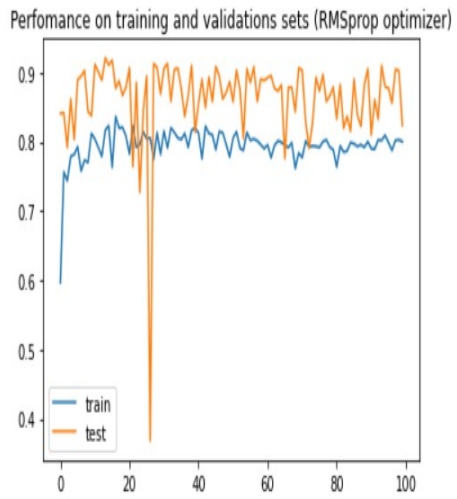
(a) Καμπύλη κόστους για τα training και validation sets για το μοντέλο με SGD optimizer και αρχικοποίηση βαρών



(b) Καμπύλη ακρίβειας για τα training και validation sets για το μοντέλο με SGD optimizer και χρήση dropout



(a) Καμπύλη κόστους για τα training και validation sets για το μοντέλο με RMSprop optimizer και χρήση dropout



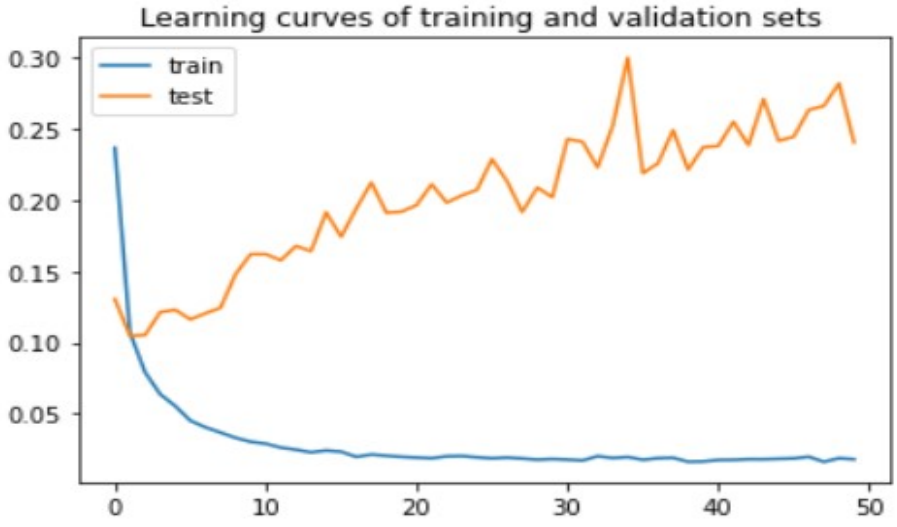
(b) Καμπύλη ακρίβειας για τα training και validation sets για το μοντέλο με RMSprop optimizer

2 Fine tuning δικτύου

Σε αυτό το κομμάτι της εργασίας σκοπός είναι εύρεση των βέλτιστων τιμών για μερικές υπερπαραμέτρους του δικτύου και η τελική εκπαίδευση και αξιολόγηση

ενός μοντέλου με βάση τις επιλεγμένες τιμές. Το δίκτυο θα χρησιμοποιεί για optimizer τον RMSprop, θα εφαρμοστεί σε κάθε στρώμα του κανονικοποίηση με L_2-νόρμα και η αρχικοποίηση των συναπτικών βαρών κάθε γίνεται με βάση το He-normalization. Αφού ενσωματωθούν στον κώδικα οι αντίστοιχες απαραίτητες πάλι βιβλιοθήκες, ορίζονται οι συναρτήσεις recall, precision , f1_score, καθώς μας βοηθάνε να υπολογίσουμε τις ζητούμενες μετρικές. Έτσι , χτίζοντας ένα μοντέλο με την βοήθεια μιας συνάρτησης όπου ενσωματώνουμε τις διάφορες τιμές για την κάθε υπερπαράμετρο στο όρισμα της συνάρτησης, κάνουμε ως γνωστόν compile το μοντέλο αφού το χτίσουμε με βάση τις ιδιότητες που αναφέρθηκαν (He-normalization για αρχικοποίηση , L_2-νόρμα, RMSprop optimizer κλπ).

Ορίζεται έπειτα με βάση το keras tuner , ένας tuner με βάση την κλάση Hyperband, καθώς και μια μέθοδος Early stopping με patience=200 όπως ζητείται . Η διαδικασία tuning ακολουθεί (διαρκεί περίπου 10 λεπτά, 1000 εποχές για κάθε δοκιμή) και οι καλύτερες υπερπαράμετροι χρησιμοποιούνται για να χτίσουμε το μοντέλο που θα εκπαιδευτεί με τις βέλτιστες αυτές τιμές. Το μοντέλο εκπαιδεύετε για 50 εποχές, καθώς η εκπαίδευση δεν γίνεται με minibatch μέθοδο και διαρκεί αρκετά , ενώ επίσης στις 50 εποχές είναι αρκετά φανερά τα χαρακτηριστικά της εκπαίδευσης ήδη. Αυτό σημαίνει πως η καμπύλη εκμάθησης όπως φαίνεται παρακάτω υποδηλώνει ξεκάθαρα overfitting στα δεδομένα testing (το σφάλμα στο testing set ανεβαίνει σταδιακά). Όμως αυτό που είναι πολύ ενδιαφέρον είναι το πολύ μικρό σφάλμα εκμάθησης απο το οποίο αρχίζουν και οι δύο καμπύλες άρα ίσως αυτό σημαίνει ότι το μοντέλο φαίνεται τουλάχιστον να είναι "καλά στημένο" ως προς τις παραμέτρους του.

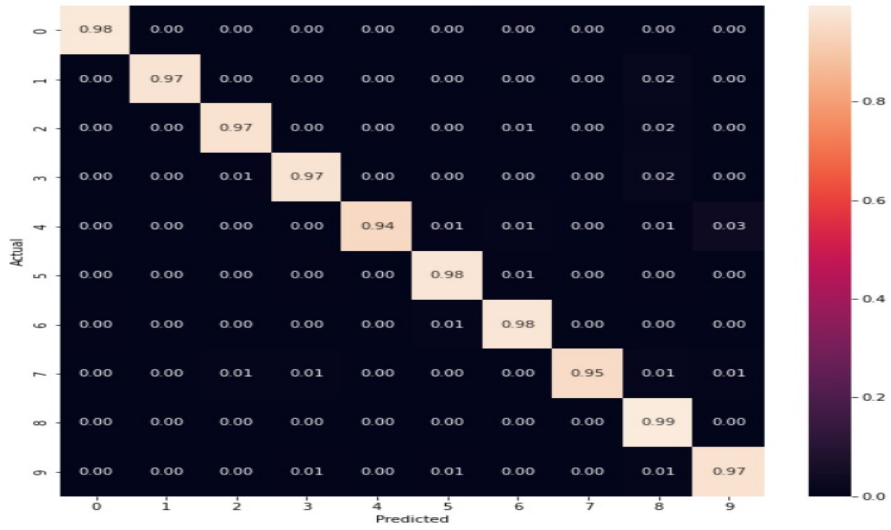


Εικόνα 7: Καμπύλη εκμάθησης για training και validation (testing) δεδομένα

Επίσης οι μετρικές f1_score, accuracy , recall ,precision τυπώνονται με την

βοήθεια τις evaluate μεθόδου του υπερμοντέλου που χτίζεται με την hypermodel.build μέθοδο του tuner, με βάση τις καλύτερες υπερπαραμέτρους προφανώς όπως αναφέρθηκε. Για να μπορέσουν να υπολογιστούν όμως αυτές οι μετρικές με αυτόν τον τρόπο έπρεπε να εισαχθούν **μαζί με την μετρική accuracy, στις μετρικές αξιολόγησης του μοντέλου** .

Ο πίνακας σύγχυσης υπολογίζεται με την βοήθεια των metrics της scikit-learn βιβλιοθήκης . Αφού κανονικοποιηθούν οι τιμές του , με την βοήθεια της seaborn βιβλιοθήκης (ειδική στην παραγωγή στατιστικών γράφων και γραφημάτων) , οπτικοποιούμε τον πίνακα . Ο πίνακας χρωματίζεται με μια κλίμακα χρωμάτων που εκφράζει την αλλαγή της τιμής που έχει αποδώσει ο πίνακας η οποία φαίνεται δίπλα απο τον γράφο.



Εικόνα 8: Γράφημα πίνακα σύγχυσης

Παρατηρούμε έτσι αμέσως πως ο πίνακας έχει αποδώσει τιμές πολύ κοντά στην μονάδα στην διαγώνιο των ψηφίων , εκεί όπου το προβλεπόμενο απο το μοντέλο δηλαδή ψηφίο είναι ίδιο με το πραγματικό. Αυτό σημαίνει πως το μοντέλο καταφέρνει αρκετά ικανοποιητικά να προβλέψει τα ψηφία που του δίνονται , τουλάχιστον στο σύνολο test δεδομένων του MNIST dataset όπως φαίνεται .