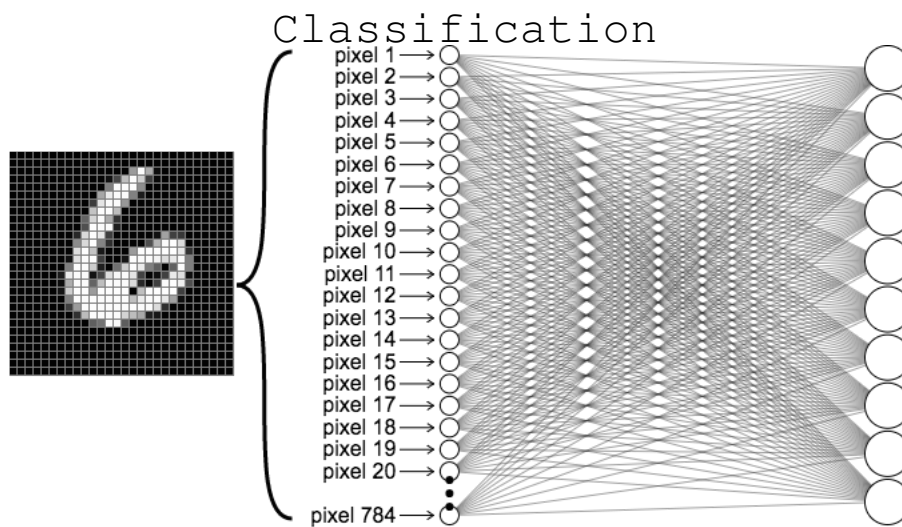


Επίλυση προβλήματος Ταξινόμησης MNIST με  
χρήση MLP νευρωνικού δικτύου  
Multi-Layer Perceptron



Χάρης Φίλης AEM: 9449  
Github Repository Link : [here](#)

October,2022

# Contents

0.1	Εισαγωγή - Abstract . . . . .	3
0.2	Διερεύνηση απόδοσης μοντέλου με διαφοροποίηση στον σχεδιασμό και την διαδικασία εκπαίδευσης . . . . .	3
0.2.1	Αλλαγή της μεθοδολογίας εκπαίδευσης του <b>dataset</b> . . . . .	4
0.2.2	Χρήση Διαφορετικών αλγορίθμων Βελτιστοποίησης και <b>Layer</b> . . . . .	6
0.2.3	Αρχικοποίηση συναπτικών βαρών με βάση κανονική κατανομή . . . . .	8
0.2.4	Διερεύνηση με προσθήκη Κανονικοποίησης συναπτικών βαρών καθώς και <b>Layer</b> που αφαιρούν κόμβους ( <b>Dropout</b> ) . . . . .	9
0.3	Fine tuning Υπερπαραμέτρων δικτύου . . . . .	11
0.3.1	Παραμετρικό Δίκτυο <b>Model_hp</b> . . . . .	11
0.3.2	Απόδοση μοντέλου . . . . .	13

# List of Figures

1	Καμπύλες Accuracy και Loss για το μοντέλο 1 . . . . .	4
2	Καμπύλες Accuracy και Loss για το μοντέλο 2 . . . . .	5
3	Καμπύλες Accuracy και Loss για το μοντέλο 3 . . . . .	6
4	Καμπύλες Accuracy και Loss για το μοντέλο 4 . . . . .	7
5	Καμπύλες Accuracy και Loss για το μοντέλο 5 . . . . .	7
6	Καμπύλες Accuracy και Loss για το μοντέλο 6 . . . . .	8
7	L2 Ridge Regression Regularizer $\lambda = a$ . . . . .	9
8	Καμπύλες Accuracy και Loss για το μοντέλο 7 . . . . .	9
9	Καμπύλες Accuracy και Loss για το μοντέλο 8 . . . . .	10
10	Καμπύλες Accuracy και Loss για το μοντέλο 4 . . . . .	10
11	Καμπύλες Accuracy και Loss για το μοντέλο 10 . . . . .	11
12	Keras Tuner Results . . . . .	12
13	Καμπύλη μάθησης του βέλτιστου μοντέλου . . . . .	13
14	Πίνακας σύγκρισης για πρόβλεψη στο test set και άλλες μετρικές μοντέλου Ταξινόμησης . . . . .	14
15	F1-SCORE . . . . .	14

## 0.1 Εισαγωγή - Abstract

Το αντικείμενο την συγκεκριμένης εργασίας είναι ή μελέτη της λειτουργίας των πυκνών μη συνελικτικών MLP νευρωνικών δικτύων στο κομμάτι της ταξινόμησης ενός dataset χειρόγραφων αριθμών 0-9 (input layer) στις αντίστοιχες κλάσεις τους (output layer). Πιο συγκεκριμένα γίνεται διερεύνηση μεταξύ διαφορετικών αρχιτεκτονικών δικτύου, διαφορετικών μορφών εκπαίδευσης με χρήση του αλγορίθμου back-propagation πάντα αλλά και διαφορετικού τρόπου βελτιστοποίησης βαρών του δικτύου με σκοπό την εύρεση του βέλτιστου δικτύου ως προς την μετρική της ακρίβειας εκτίμησης της κλάσης ενός αριθμού, αλλά και άλλων μετρικών μετέπειτα με χρήση της αντικειμενικής συνάρτησης κόστους categorical cross-entropy της οποίας ο μαθηματικός τύπος φαίνεται παρακάτω.

$$J(i) = - \sum_{e=1}^0 y_{i,e} \log_{\epsilon} y_{i,e}$$

Στην συνέχεια εφαρμόζεται ένας αποδοτικός τρόπος αυτόματης βελτιστοποίησης των υπερπαραμέτρων του νευρωνικού δικτύου.

## 0.2 Διερεύνηση απόδοσης μοντέλου με διαφοροποίηση στον σχεδιασμό και την διαδικασία εκπαίδευσης

Τα μοντέλα που παρουσιάζονται στην συνέχεια έχουν δύο κρυφά layers νευρώνων. Το πρώτο αποτελείται από 128 νευρώνες ενώ το δεύτερο αποτελείται από 256 νευρώνες. Η επιλογή της μη γραμμικότητας των νευρώνων δηλαδή η συνάρτηση ενεργοποίησης είναι η ReLU και αυτό ορίζεται εξ αρχής από την εργασία πιθανόν διότι ο αλγόριθμος back-propagation χρειάζεται τα gradients των βαρών να μην είναι πολύ μικρά και να υπάρχει θέμα λόγω αυτού στην εκπαίδευση, ένα πρόβλημα που εισάγεται από άλλες συναρτήσεις ενεργοποίησης. Τέλος εφόσον μιλάμε για πρόβλημα ταξινόμησης και συγκεκριμένα πολλών κλάσεων η συνάρτηση ενεργοποίησης του output layer είναι ή softmax που εμφανίζει ουσιαστικά πιθανότητες στην έξοδο του δικτύου και μάλιστα στους 10 νευρώνες που έχει το output layer. Τα εσωτερικά layer είναι πλήρως συν-δεμένα δηλαδή πρόκειται για Dense δίκτυο και όχι συνελικτικό. Η μετρική βελτιστοποίησης είναι η ακρίβεια ή accuracy και ή αντικειμενική συνάρτηση προς βελτιστοποίηση είναι η categorical cross-entropy όπως αναφέρθηκε παραπάνω. Η εκπαίδευση διαρκεί 100 εποχές και το validation set είναι το 20% του συνολικού dataset. Τα μοντέλα πέρα αυτών των βασικών χαρακτηριστικών διαφέρουν μεταξύ τους.

### 0.2.1 Αλλαγή της μεθοδολογίας εκπαίδευσης του **dataset**

Σε αυτό το πρώτο κομμάτι της διερεύνησης αυτό που ουσιαστικά αλλάζει είναι ο τρόπος με τον οποίο εκπαιδεύεται το δίκτυο. Συγκεκριμένα αλλάζει το μέγεθος του **batch size**, μιας υπερπαραμέτρου του **gradient descent** η οποία ελέγχει τον αριθμό των δειγμάτων εκπαίδευσης που δουλεύει ο αλγόριθμος βελτιστοποίησης πριν το μοντέλο αναβαθμίσει τις εσωτερικές του παραμέτρους και ουσιαστικά δουλέψει ο αλγόριθμος του **back-propagation**. Να σημειωθεί ότι γίνεται χρήση **output layer** με **softmax activation** συνάρτηση επομένως κατά τον υπολογισμό του **cross-entropy loss** δεν χρειάζεται περαιτέρω κανονικοποίηση των εξόδων (**logits=false**) και επίσης δεν χρησιμοποιείται η **sparse cross-entropy** που διδάχθηκε στην ενισχυτική διδασκαλία για τον ίδιο λόγο. Επίσης τα δεδομένα κανονικοποιούνται στην κλίμακα 0-1 αντί για 0-256 και τέλος τα δεδομένα **test** υπόκεινται σε **one-hot encoding** ώστε να εξοικονομηθεί υπολογιστικό χρόνος μιας και πρόκειται για κατηγορικά δεδομένα ουσιαστικά. Έχω έτσι **tensors** με τιμή 1 στην κλάση που μας ενδιαφέρει.

#### Model 1

Το πρώτο μοντέλο είναι το βασικό μοντέλο που αναλύθηκε παραπάνω με χρήση όμως το **RMSProp optimizer** με **learning rate** = 0.001 και  $\rho=0.9$ . Επίσης το **training** που γίνεται είναι **online** που σημαίνει ότι ο αλγόριθμος βελτιστοποίησης δουλεύει όλα τα δείγματα και υπολογίζει τα **gradients** και τρέχει τον αλγόριθμο βελτιστοποίησης για να αναβαθμιστεί το δίκτυο. Αυτό σημαίνει η διάρκεια εκτέλεσης επιβαρύνθηκε και συγκεκριμένα έφτασε τις 3 ώρες. Στα επόμενα σχήματα παρουσιάζονται οι καμπύλες ακρίβειας και κόστους για **training** και **validation set** αυτού του μοντέλου.

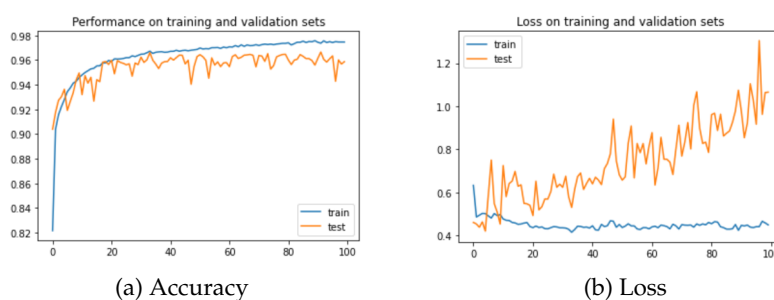


Figure 1: Καμπύλες Accuracy και Loss για το μοντέλο 1

Αυτό που παρατηρώ είναι ότι το συγκεκριμένο μοντέλο πετυχαίνει μεγάλη ακρίβεια για το σύνολο εκπαίδευσης πολύ γρήγορα δηλαδή σε λίγες εποχές ενώ στο **validation set** η ακρίβεια δεν σταθεροποιείται και μάλιστα παρατηρούμε και φαινόμενο χειροτέρευσης και επαναφοράς τύπου ταλάντωσης της ακρίβειας του μοντέλου στο πέρας των εποχών χωρίς όμως αυτό να σημαίνει ότι το μοντέλο δεν

είναι αποδοτικό και στο **validation set** με ακρίβεια πάνω του 0.9. Από την άλλη πλευρά στο κομμάτι του κόστους φαίνεται πως και εκεί το κόστος ελαχιστοποιείται άμεσα αλλά στο **validation set** έχουμε αύξηση του κόστους και ουσιαστικά έχουμε άμεση ένδειξη υπερμοντελοποίησης ή **overfitting** κάτι που σημαίνει ότι το μοντέλο αυτό αποστηθίζει τα δεδομένα εκπαίδευσης αντί ουσιαστικά να μαθαίνει να επιλύει το πρόβλημα της ταξινόμησης. Αυτό το πρόβλημα αντιμετωπίζεται με σχετικά απλό τρόπο είτε χρησιμοποιώντας **dropout layers** που προσδίδουν την έννοια του θορύβου στο νευρωνικό δίκτυο ώστε να μπορούμε να έχουμε ένα πιο **robust training** είτε κάνοντας καλύτερη κανονικοποίηση και ανακάτεμα των δεδομένων εισόδου ακόμα και **cross validation data split** το οποίο ίσως είναι μια καλύτερη λύση. Ο συνδυασμός των δύο μεθόδων αποτρέπει ακόμα περισσότερο την υπερμοντελοποίηση του νευρωνικού δικτύου.

## Model 2 Minibatch training

Το μοντέλο αυτό είναι το **default** μοντέλο όπως ορίστηκε παραπάνω αλλά χρησιμοποιούμε **batch** εκμάθησης μεγέθους **batch\_size = 256**. Αυτό έχει ήδη αναλυθεί πως προσφέρει σε ένα δίκτυο τόσο στο χρόνο εκτέλεσης αλλά θα δούμε και πως βοηθά το πρόβλημα του **overfitting** που εμφανίστηκε παραπάνω.

Στα επόμενα σχήματα παρουσιάζονται οι καμπύλες ακρίβειας και κόστους για **training** και **validation set** αυτού του μοντέλου.

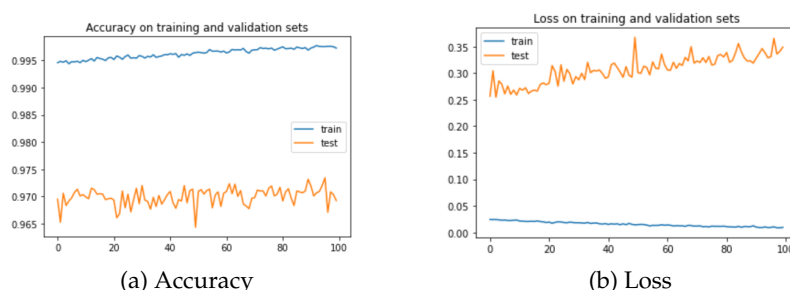


Figure 2: Καμπύλες Accuracy και Loss για το μοντέλο 2

Παρατηρείται πως η απόδοση του δικτύου είναι αρκετά καλή σε επίπεδα 0.995 μάλιστα τόσο στο σύνολο εκπαίδευσης όσο και στο σύνολο επικύρωσης. Επιπλέον οι τιμές της συνάρτησης κόστους για το **validation** ή **test set** όπως φαίνεται στα υπομνήματα σταθεροποιείται (αυτό θα ήταν πιο εμφανές αν είχαμε περισσότερες εποχές). Δηλαδή δεν έχουμε διαρκή αύξηση του σφάλματος στο **test set** και αυτό δείχνει σημάδια ότι το μοντέλο μαθαίνει κανονικά μια και το **train loss** είναι κοντά στο **test loss**. Επομένως το μοντέλο όπως λέγεται "γενικεύει καλά". Φυσικά ο χρόνος εκπαίδευσης είναι αρκετά μικρότερος εξαιτίας ακριβώς του **batch learning**. Προσθέτουμε στην αντιμετώπιση του **overfitting** και την μέθοδο του **minibatch learning**.

### Model 3 Batch=Ntrain

Εδώ το μοντέλο είναι ακριβώς το ίδιο με τα παραπάνω απλά το μέγεθος του batch είναι ίσο με το μέγεθος των δειγμάτων του συνόλου εκπαίδευσης. Στα επόμενα σχήματα παρουσιάζονται οι καμπύλες ακρίβειας και κόστους για **training** και **validation set** αυτού του μοντέλου. Από τα διαγράμματα παρατηρείται

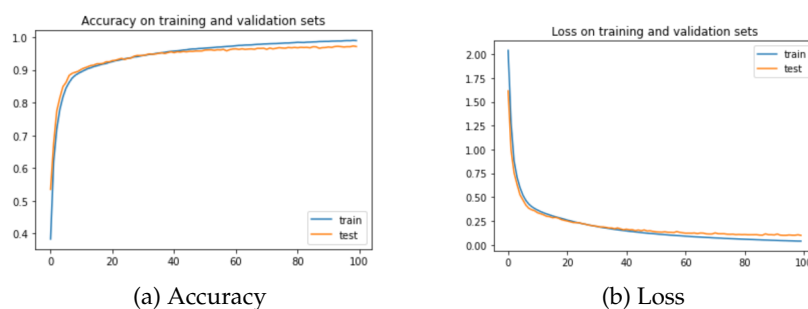


Figure 3: Καμπύλες Accuracy και Loss για το μοντέλο 3

ότι μετά τις πρώτες 10 εποχές οι τιμές της ακρίβειας και του κόστους φτάνουν σταθεροποιούνται σε καλή τιμή για όλες τις επόμενες εποχές και μάλιστα οι καμπύλες κινούνται σχεδόν η μία πάνω στην άλλη. Όσον αφορά το **training set** φαίνεται ότι αυτό φτάνει και την απόλυτη ακρίβεια ενώ το **loss** οδηγείται στο μηδέν. Θα μπορούσε να διατυπωθεί ότι σε αυτό το σημείο έχουμε το τέλειο μοντέλο μιας και βλέπουμε καλή πορεία των καμπυλών. Παρόλα αυτά το γεγονός ότι μετά από τις πρώτες εποχές οι καμπύλες έχουν πολύ μικρή κλίση δείχνει ότι η εκπαίδευση δεν είναι απαραίτητη για 100 εποχές. Τέλος χρονικά το συγκεκριμένο μοντέλο ολοκληρώνει την εκπαίδευση σε 40 δευτερόλεπτα κάτι που είναι μακράν λιγότερο χρόνος εκτέλεσης εκπαίδευσης σε σχέση με τα δύο προηγούμενα μοντέλα.

### 0.2.2 Χρήση Διαφορετικών αλγορίθμων Βελτιστοποίησης και Layer

Μιας και το τελευταίο μοντέλο δεν φαίνεται να εκπαιδεύεται για περισσότερες από 10 εποχές εξακολουθεί η χρήση του μοντέλου με το **minibatch training** μιας και φάνηκε να είναι το μοντέλο που γενικεύει καλύτερα και είναι και εκπαιδευσιμο.

### Model 4

Για αυτό το μοντέλο χρησιμοποιήθηκε ο **RMSProp** με **learning rate**  $lr = 0.001$  και  $\rho$  ή **rho**  $= 0.01$ . Στα επόμενα σχήματα παρουσιάζονται οι καμπύλες ακρίβειας και κόστους για **training** και **validation set** αυτού του μοντέλου.

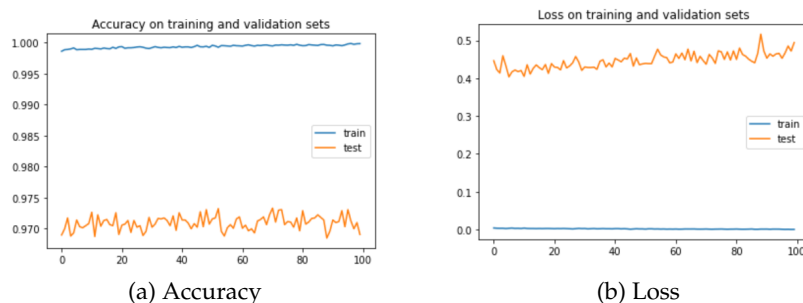


Figure 4: Καμπύλες Accuracy και Loss για το μοντέλο 4

Παρατηρούμε ότι το accuracy (0.99) του training set καθώς και το loss(0.01) παρουσιάζει πολύ καλές τιμές έως άριστες ενώ το accuracy και του loss του validation set είναι και αυτά σε καλές τιμές άρα γενικεύει καλά το μοντέλο. Είναι γνωστό από την θεωρία, ότι στον RMSProp optimizer η μικρή τιμή της υπερπαραμέτρου  $\rho$ , η οποία ελέγχει την κλίμακα του κινούμενου μέσου όρου που εφαρμόζεται στα βάρη, ευνοεί ουσιαστικά την κίνηση του αλγορίθμου προς παλιότερες διευθύνσεις του αθροιστικού τετραγωνικού gradient με αποτέλεσμα να μην υπάρχει βελτίωση αν έχει επιτευχθεί η καλύτερη μέχρι στιγμής τιμή(τωρινή εποχή) του accuracy. Για αυτό τον λόγο δεν μπορεί να κριθεί αυτή η περίπτωση ως περίπτωση overfitting ωστόσο το μικρό learning rate είναι αυτό που εγκλωβίζει σε αυτή την κατάσταση τον αλγόριθμο και κάνει την εκπαίδευση στάσιμη.

### Model 5

Σε αυτό το μοντέλο ανεβαίνει το learning rate και ταυτόχρονα τοποθετούμε την παράμετρο  $\rho$  στην τιμή 0.99 που είναι πολύ κοντά στο όριο της. Στα επόμενα σχήματα παρουσιάζονται οι καμπύλες ακρίβειας και κόστους για training και validation set αυτού του μοντέλου.

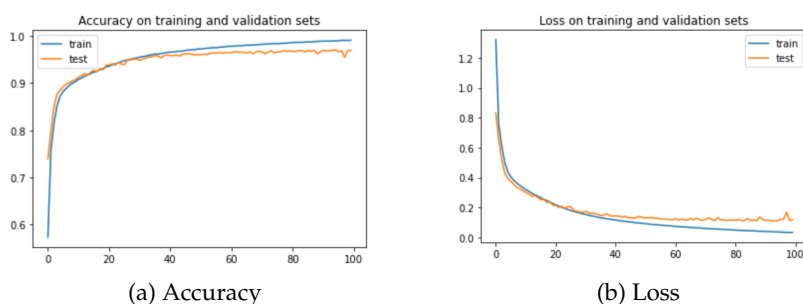


Figure 5: Καμπύλες Accuracy και Loss για το μοντέλο 5



Αυτό που παρατηρώ στα δυο γραφήματα είναι ότι παρόλο που η παράμετρος  $\rho$  είναι πολύ μεγάλη που σημαίνει ότι το μοντέλο οδηγείται προς την παρούσα διεύθυνση των ανθροιστικών τετραγωνικών **gradient**, αυτή η κατεύθυνση λόγω αύξησης του **learning rate** τελικά είναι σωστή και το μοντέλο αποφεύγει την προηγούμενη σκόπελο. Επομένως φαίνονται πολύ καλά **learning curves** αλλά και στο κομμάτι του **accuracy** και του **loss** παρόλο που στο **loss** ξεκινάμε από αρκετά μεγάλη τιμή.

### 0.2.3 Αρχικοποίηση συναπτικών βαρών με βάση κανονική κατανομή

#### Model 6

Το συγκεκριμένο μοντέλο πέραν του ότι χρησιμοποιείται άλλος αλγόριθμος βελτιστοποίηση και συγκεκριμένα ο **Stochastic Gradient Descent** με **learning rate** = 0.01 και **default momentum** = 0, εφαρμόζεται σε αυτό και αρχικοποίηση συναπτικών βαρών με χρήση **kernel** κανονικής κατανομής με μέση τιμή το 10. Στα παρακάτω σχήματα φαίνονται οι καμπύλες ακρίβειας και κόστους για τα **training** και **validation sets**. Αυτό που παρατηρείται στο κομμάτι της ακρίβειας (**accuracy**),

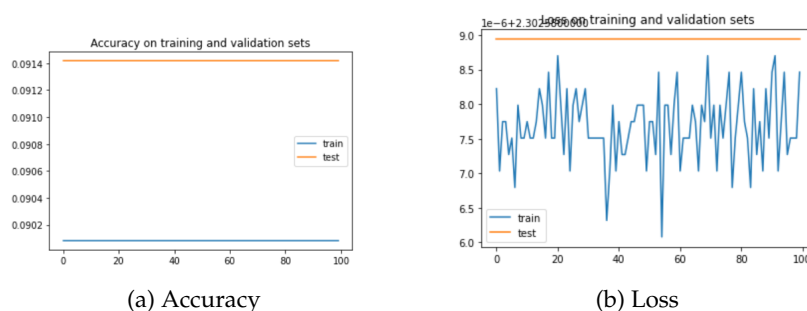


Figure 6: Καμπύλες Accuracy και Loss για το μοντέλο 6

είναι πως το μοντέλο δεν προσαρμόζεται καθόλου στα δεδομένα εκπαίδευσης και φαίνεται από το γεγονός ότι βλέπουμε μεγαλύτερη ακρίβεια στα **test data** παρά στα **training**. Έτσι στην συγκεκριμένη περίπτωση μιλάμε για **underfitting**. Ενδιαφέρον περίπτωση είναι αυτή του **loss** στο **training set**. Φαίνεται μια ανεξέλεγκτη πορεία και αυτό κατά πάσα πιθανότητα οφείλεται στην αρχικοποίηση των βαρών με μια κατανομή με μεγάλη μέση τιμή σε σχέση με τις τιμές των δεδομένων. Ουσιαστικά σε αυτή την περίπτωση οι αρχικές αυτές τιμές των **gradients** είναι τόσο μεγάλες σε σχέση με τα δεδομένα που οδηγεί σε έντονες ταλαντώσεις και αδυναμία σύγκλισης (**exploding gradients**). Επομένως η εισαγωγή θορύβου κατά μία έννοια στα βάρη κρίνεται μη αποτελεσματική μιας και δεν μπορούν να φτάσουν στην βέλτιστη τιμή τους μετέπειτα και το νευρωνικό δίκτυο δεν εκτελεί το **task** σωστά.

### 0.2.4 Διερεύνηση με προσθήκη Κανονικοποίησης συναπτικών Βαρών καθώς και **Layer** που αφαιρούν κόμβους (**Dropout**)

#### Model 7

Για το μοντέλο 7 χρησιμοποιήθηκαν οι ίδιες υπέρ-παράμετροι με το μοντέλο 6 ωστόσο εδώ εφαρμόζεται μια μορφής κανονικοποίηση στα συναπτικά βάρη με την μορφή **penalty** με την χρήση της L2-νόρμας δηλαδή της απόστασης ο οποίος εφαρμόζεται απευθείας στα στρώματα του δικτύου που ουσιαστικά στο keras περιβάλλον μιλάμε για τα βάρη και έχει  $a = 0.1$ . Λειτουργεί μέσω αλγορίθμου **Ridge Regression** και αυτή η κανονικοποίηση προσθέτει την ποινή καθώς η πολυπλοκότητα του μοντέλου αυξάνεται. Η παράμετρος  $a$  μπαίνει σαν ποινή σε όλες τις παραμέτρους εκτός αν αποτρέπει την σύγκλιση του μοντέλου και τον κάνει να μην γενικεύει καλά ή να υπερμοντελοποιείται. Ο Ridge regression αλγόριθμος με τον οποίο λειτουργεί προσθέτει το τετράγωνο του  $a$  σαν ποινή στην συνάρτηση κόστους.

$$J(\theta) = \frac{1}{2m} \left[ \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2 + \lambda \sum_{j=1}^n \theta_j^2 \right]$$

$$\min_{\theta} J(\theta)$$

Figure 7: L2 Ridge Regression Regularizer  $\lambda = a$

Διαγράμματα Όπως φαίνεται από τα παραπάνω διαγράμματα, το μοντέλο αυτό εί-

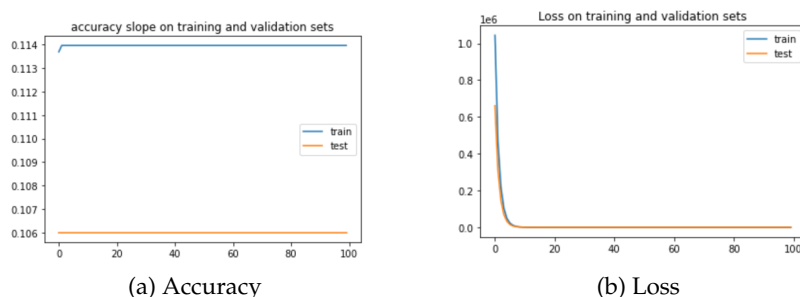


Figure 8: Καμπύλες Accuracy και Loss για το μοντέλο 7

ναι αναποτελεσματικό στο να κάνει την ταξινόμηση των δεδομένων. Δεν έχουμε καμία βελτίωση καθόλη την διάρκεια της εκπαίδευσης και ειδικότερα στην συνάρτηση κόστους η αρχική τιμή είναι εξαιρετικά μεγάλη (λογικά λόγω του μεγάλου **penalize** μια και η αρχικοποίηση των βαρών γίνεται ουσιαστικά με θόρυβο). Παρ όλη την διόρθωσή αυτού σχετικά σε μικρό αριθμό εποχών δεν βλέπουμε το μοντέλο να βελτιώνεται παραπέρα. Επομένως καταλήγουμε σε ίδια συμπεράσματα με εξαίρεση ότι πλέον έχουμε μια σχετική βελτίωση του εξαιρετικά κακού **loss**.

### Model 8

Σε αυτό το μοντέλο η παράμετρος  $\alpha$  μειώνεται σε 0.01 για να περιοριστεί αυτή η ποινή. Σε αυτά τα διαγράμματα φαίνεται ότι το κομμάτι του loss πλέον φτάνει

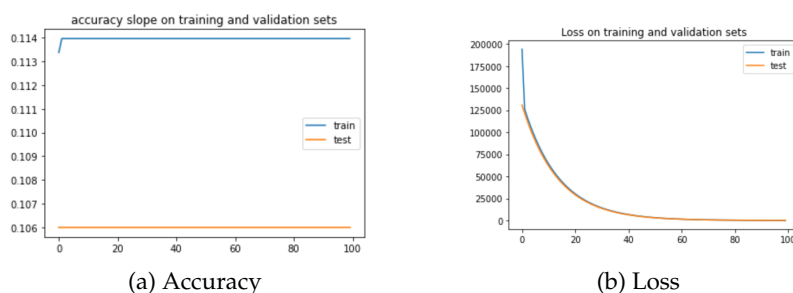


Figure 9: Καμπύλες Accuracy και Loss για το μοντέλο 8

σε χαμηλότερη κλίμακα και μάλιστα συνεχίζει και αρχίζει να συγκλίνει στο 0 ακόμα από την 60η εποχή κάτι που ισχύει και για το **validation** και για το **test set**. Παρόλα αυτά, το μοντέλο συνεχίζει να μην είναι αποδοτικό καθώς η ακρίβεια εκτίμησης του παραμένει στο 0.1 χωρίς όμως να έχουμε φαινόμενα υπερμοντελοποίησης. Επομένως δεν πρόκειται για ένα καλό μοντέλο.

### Model 9

Μειώνεται περαιτέρω η παράμετρος της L2 νόρμας στο  $\alpha = 0.001$ . Στα παρακάτω διαγράμματα φαίνονται οι καμπύλες ακρίβειας και κόστους για **training set** και **validation set**. Δεν βλέπουμε κάποια βελτίωση σε κανένα από τα δυο διαγράμ-

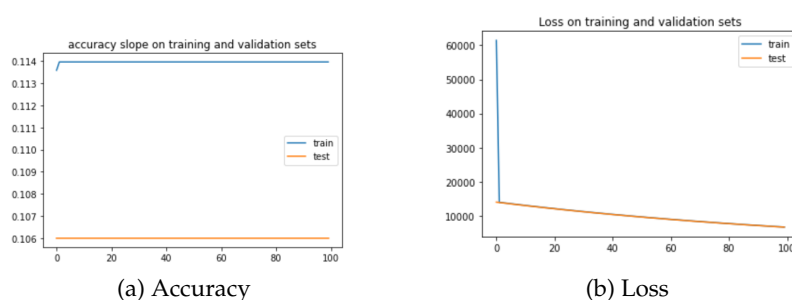


Figure 10: Καμπύλες Accuracy και Loss για το μοντέλο 4

ματα και επομένως μιλάμε για **underfitting** μιας και εμφανίζονται όλα τα φαινόμενα των τεράστιων **gradients**, των στάσιμων καμπυλών καθώς και των κακών τιμών ακρίβειας και κόστους. Σαν προσωπική σημείωση θα σημειώνα ότι η χρήση της L2 νόρμας σε συνδυασμό με τον θόρυβο στα βάρη είναι άστοχη στην συγκεκριμένη περίπτωση καθώς το δεύτερο οδηγεί σε **exploding gradients** και το

δεύτερο λόγω μεγάλων **gradients** οδηγεί σε συνεχόμενες ποινές που σε αυτή την περίπτωση λόγω του μικρού συντελεστή  $\alpha$  δεν έχουν και κάποια ουσιαστική επίδραση.

### Model 10

Στο συγκεκριμένο μοντέλο γίνεται στροφή στον τρόπο κανονικοποίησης μέσω της L1 νόρμα και του αντίστοιχου **regularizer** αλγορίθμου που εφαρμόζεται ( $\alpha=0.001$ ). Επιπλέον ενσωματώνονται στην αρχιτεκτονική του δικτύου **dropout layers** τα οποία αφαιρούν νευρώνες πρακτικά με πιθανότητα 0.3. Ως μέθοδος βελτιστοποίησης χρησιμοποιήθηκε **RMSprop** με  $\rho=0.9$  και  $\text{lr}=0.01$ . Στα επόμενα σχήματα παρουσιάζονται οι καμπύλες ακρίβειας και κόστους για **training** και **validation set** α-τού του μοντέλου. Παρατηρείται και εδώ αδυναμία βελτίωσης του μοντέλου

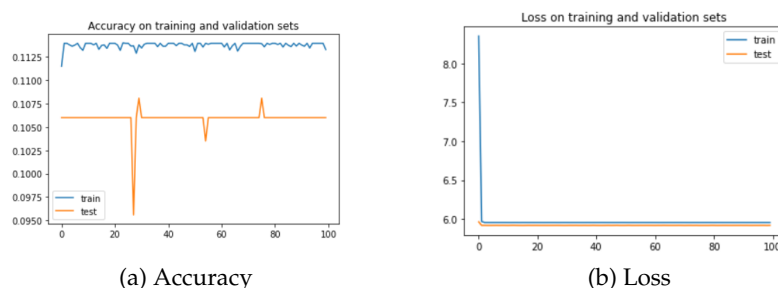


Figure 11: Καμπύλες Accuracy και Loss για το μοντέλο 10

παρόλο που έχουμε βελτίωση σε σχέση με τα προηγούμενα μοντέλα ωστόσο το **loss** παραμένει αρκετά μεγάλο και στάσιμο και το **accuracy** παλινδρομεί σε πολύ χαμηλές τιμές.

## 0.3 Fine tuning Υπερπαραμέτρων δικτύου

Στο προηγούμενο κομμάτι έγινε μια διερεύνηση μοντέλων για την επίλυση του προβλήματος ταξινόμησης του **MNIST dataset** με χρήση **MLP** δικτύου. Κάποια αποδείχθηκαν αρκετά αποδοτικά κάποια οδηγούνταν σε υπερμοντελοποίηση αποσπώντας το **dataset** και κάποια σε υπόμοντελοποίηση με επιδόσεις κακές και με φαινόμενα μη εφαρμογής στα δεδομένα. Είναι σαφές λοιπόν η ανάγκη μιας πιο συστηματικής μεθόδου ρύθμισης των παραμέτρων που επηρεάζουν αυτά τα μοντέλα.

### 0.3.1 Παραμετρικό Δίκτυο **Model\_hp**

Στο συγκεκριμένο κομμάτι θα λάβουμε ένα δίκτυο που βελτιστοποιείται με **RMSPProp** αλγόριθμο, χρησιμοποιεί αλγόριθμο κανονικοποίησης **L2**, αρχικοποιεί τα βάρη με **Henormalization** και έχει 2 **hidden layers**.

Βέβαια στο κομμάτι των υπόλοιπων υπερπαραμέτρων θα γίνει αυτόματο **tuning**. Αυτό θα μπορούσε να γίνει μεσω αλγορίθμου **Grid Search** και με **cross validation 5-fold** ώστε να βρεθούν οι πιο αποδοτικές παράμετροι. Βέβαια το πακέτο **keras** παρέχει έναν πολύ πιο γρήγορο και αποδοτικό **tuner** που είναι εύκολος και στην χρήση. Επομένως γίνεται μια χρήση του **randomSearch tuner** απλά για πειραματισμό και στην συνέχεια γίνεται χρήση του **hyperband tuner**.

Οι υπερπαραμέτροι που πρόκειται να βελτιστοποιηθούν είναι οι εξής:

- νευρώνων 1-hidden-layer - {64,128}
- νευρώνων 2-hidden-layer - {256,512}
- παράμετρος ποινής του **reguralizer a** - {0.1,0.01,0.001,1e-06}
- **Learning rate** - {0.1,0.01,0.001}

Τέλος η μετρική που χρησιμοποιείται στο συγκεκριμένο πείραμα είναι η **F-score** καθώς και δημιουργείται σήμα - **callback** για **Early Stopping** του **tuner** μιας και ή λειτουργία του είναι επαναληπτική και δεν θέλουμε να συνεχίζει την αναβάθμιση της επιλογής βέλτιστου μοντέλου αν δεν μεγαλώνει το **F1\_score**.

Ο **Tuner** έτρεξε για πάνω από 30 **trials** και έβγαλε τα παρακάτω αποτελέσματα με **early stopping**. Άρα οι βέλτιστες τιμές των υπερπαραμέτρων που επιλέχθηκαν

```
INFO:tensorflow:Oracle triggered exit
Optimal number of neurons for the 1st hidden layer is: 128neurons
Optimal number of neurons for the 2nd hidden layer is: 256neurons
Optimal learning rate for the optimizer RMSProp is : 0.01
Optimal a parameter for weight reguralization using L2 norm(distance) is 1e-06

Trial 31 Complete [00h 00m 22s] val_f1_score: 0.019168173894286156

Best val_f1_score So Far: 0.9251760244369507 Total elapsed time: 00h 08m 21s Optimal number of neurons for the 1st hidden layer is: 128neurons Optimal number of
neurons for the 2nd hidden layer is: 512neurons Optimal learning rate for the optimizer RMSProp is : 0.01 Optimal a parameter for weight reguralization using L2
norm(distance) is 1e-06
```

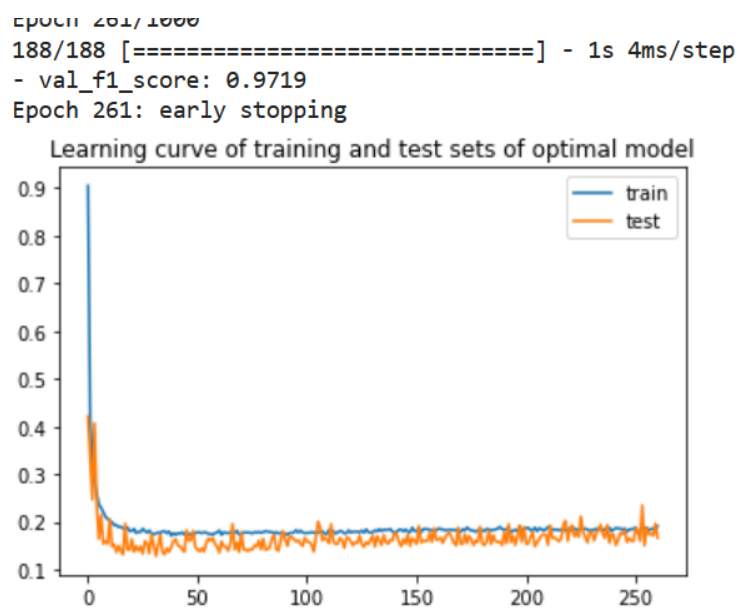
Figure 12: Keras Tuner Results

είναι:

- νευρώνων 1-hidden-layer = 128
- νευρώνων 2-hidden-layer = 256
- παράμετρος ποινής του **reguralizer a** = 1e-06
- **Learning rate** = 0.01

### 0.3.2 Απόδοση μοντέλου

Στα επόμενα σχήματα παρουσιάζονται οι καμπύλες ακρίβειας και κόστους για **training** και **validation set** αυτού του μοντέλου.



(a) Learning Curve

Figure 13: Καμπύλη μάθησης του βέλτιστου μοντέλου

Όπως μπορούμε να δούμε στο Σχ.13, οι τιμές της συνάρτησης κόστους τόσο για **test** όσο και για το **train set** είναι αρκετά μικρές και μάλιστα η καμπύλη και για τα δύο **set** έχει απότομη καθοδική πορεία ειδικά για το **training set**. Στο **testing set** είναι πιο ομαλή και τραχιά αυτή η μετάβαση. Φαίνεται ότι το μοντέλο φτάνει την βέλτιστη τιμή σχετικά νωρίς μόλις από τις πρώτες 20 εποχές και εδώ γίνεται αν και καθυστερημένα χρήση του σήματος **early stopping** για αυτό και η εκπαίδευση του δικτύου σταματά στις 250 εποχές το **F1\_score** είναι αρκετά καλό τόσο για το **validation (0.9719) set** όσο και για το **training set**. Πιθανότατα θα έπρεπε να μειωθεί η τιμή της μεταβλητής **patience** ώστε να σταματήσει πιο γρήγορα το αχρείαστο **training** του δικτύου μιας όπως είπαμε το μοντέλο συγκλίνει και γενικεύει καλά σε λιγότερο από 50 εποχές.

### Confusion Matrix

Μια άλλη μετρική που αξιολογείται στο κομμάτι αυτό είναι ο πίνακας σύγχυσης (**confusion matrix**)

	precision	recall	f1-score	support
0	0.97	0.99	0.98	980
1	0.98	0.99	0.98	1135
2	0.96	0.98	0.97	1032
3	0.97	0.98	0.97	1010
4	0.99	0.95	0.97	982
5	0.98	0.96	0.97	892
6	0.97	0.98	0.98	958
7	0.97	0.97	0.97	1028
8	0.97	0.96	0.96	974
9	0.97	0.96	0.96	1009
accuracy			0.97	10000
macro avg	0.97	0.97	0.97	10000
weighted avg	0.97	0.97	0.97	10000

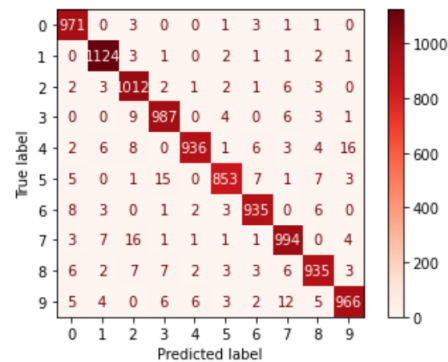


Figure 14: Πίνακας σύγχυσης για πρόβλεψη στο **test set** και άλλες μετρικές μοντέλου Ταξινόμησης

Στο κομμάτι του πίνακα σύγχυσης τα στοιχεία συγκεντρώνονται στην κύρια διαγώνιο κατά κύρια πλειοψηφία με ορισμένες περιπτώσεις όπως το 4 που ταξινομείται για 9 16 φορές κάτι που είναι λογικό δεδομένου ότι τα δύο νούμερα έχουν παρόμοια δομή και λογικά καταλαμβάνουν παραπλήσια **pixel** στο **cmap**. Επίσης το ίδιο συμβαίνει με το 5 και το 3 αλλά και το 7 και το 2. Ωστόσο κατά την κύρια πλειοψηφία το μοντέλο είναι άριστο.

Όσον αφορά τις κλασσικές μετρικές που παρέχει η **sklearn** για την ταξινόμηση, **accuracy**, **precision**, **recall**, **F1-score** όπου: μπορούμε να δούμε ότι

$$F_1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

Figure 15: F1-SCORE

φτάνει το 0.97-0.98 που είναι αρκετά καλή τιμή για την ταξινόμηση του **MNIST**.

Τέλος γίνεται εκτίμηση 9 αριθμών στο **notebook** για πειραματισμό με το δίκτυο. Αλγόριθμοι **RandomSearch** τοποθετήθηκαν στο τέλος επίσης υπάρχουν πειραματικά και κελιά στα οποία πειραματίστηκα με το χτίσιμο μοντέλων χωρίς το **keras** με βάση τις σημειώσεις τα οποία δεν χρησιμοποιήθηκαν λόγω του ότι το **keras** περιέχει πολύ πιο βελτιστοποιημένους αλγόριθμους. Τέλος έγιναν και άλλες εκτελέσεις ορισμένων **cell** και πιθανόν ορισμένα διαγράμματα να είναι διαφορετικά με ορισμένες αλλαγές που έγιναν. Ωστόσο τα συγκεκριμένα διαγράμματα υπάρχουν στο **notebook** σε προηγούμενα **commit** του **github** αλλά και στον φάκελο **img**. Αυτές οι αλλαγές οι εκτελέσεις έγιναν γιατί έκανα ορισμένες μικρο διορθώσεις.