ΑΡΙΣΤΟΤΕΛΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΟΝΙΚΗΣ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΝΕΥΡΩΝΙΚΑ ΔΙΚΤΥΑ ΚΑΙ ΒΑΘΙΑ ΜΑΘΗΣΗ
1^{Η} ΕΡΓΑΣΙΑ

Ιωάννης Δεϊρμεντζόγλου 10015

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΗΛΕΚΤΡΟΝΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

ΑΕΜ 10015 Email deirmentz@ece.auth.gr

Εισαγωγή

Στην παρούσα εργασία ζητείται η υλοποίηση ενός νευρωνικου πολυστρωματικού δικτύου **perceptron** , το οποίο θα είναι πλήρως συνδεδεμένο ή συνελικτικό ή και συνδυασμός αυτών με σκοπό την επίλυση προβλημάτων κατηγοριοποίησης . Σε αυτήν την εργασία χρησιμοποιήθηκε η βάση δεδομένων cifar 10 . Η CIFAR-10 περιλαμβάνει 60.000 εικόνες σε 10 διαφορετικές κατηγορίες. Κάθε εικόνα ανήκει σε μία από τις κατηγορίες: αεροπλάνα, αυτοκίνητα, πουλιά, γάτες, ελάφια, σκύλοι, βατράχια, ίπποι, πλοία και φορτηγά. Οι εικόνες έχουν διαστάσεις 32x32x3 (3 χρωματικά κανάλια –RGB εικόνες) .

Προεπεξεργασία των δεδομένων

Αρχικά , τα δεδομένα που φορτώνονται από την cifar 10 χωρίζονται σε train και test set με διαστάσεις 50.000x32x32x3 (50k εικόνες) και 10.000x32x32x3 αντίστοιχα . Οι εικόνες προκειμένου να κατηγοριοποιηθούν , κανονικοποιουνται μεταξύ 0 και 1 και μετατρέπονται σε 1d πίνακες. Επίσης , στα labels του train και test εφαρμόζεται one hot encoding , ώστε το μοντέλο να διαχωρίσει καλυτέρα τα labels μεταξύ τους και να έχει καλύτερη απόδοση .

Ανάλυση κυρίων συνιστωσών

Με ανάλυση κύριων συνιστωσών μειώνεται ο αριθμός των features τέτοιος ώστε να διατηρείται το 90% ή αντίστοιχα το 95 % της διακύμανσης των αρχικών δεδομένων . Γενικά , στόχος είναι να δημιουργήσουμε νέα features που προκύπτουν από το αρχικό σύνολο(3072 για την cifar 10) με σκοπό να διατηρηθεί όσο περισσότερη πληροφορία γίνεται σε πολύ μικρότερο αριθμό features (έχουμε 3072 features χωρίς PCA).

1. KNN και Nearest Centroid Κατηγοριοποιητές

Παρακάτω παρουσιάζονται τα αποτελέσματα εφαρμογής των κατηγοριοποιητών Nearest Centroid και k-Nearest Neighbor στη βάση δεδομένων **CIFAR-10.** Τα αποτελέσματα δεν ήταν καθόλου ικανοποιητικά. Κάποιοι από τους λογούς που συμβαίνει αυτό είναι ίσως είναι επειδή στην ουσία συγκρίνουν την γενικότερη χρωματική κατανομή των εικόνων ή το είδος του φόντου για να αποφασίσουν αν υπάρχει ομοιότητα και δεν έχουν την ικανότητα κατανόησης του σημασιολογικού περιεχομένου των εικόνων. Αντιμετωπίζουν κάθε feature εξίσου, που σημαίνει ότι ένα pixel που αντιπροσωπεύει το φόντο μπορεί να έχει το ίδιο βάρος με ένα pixel που αντιπροσωπεύει ένα κρίσιμο αντικείμενο ή χαρακτηριστικό μέσα σε μια εικόνα. Τα μοντέλα αυτών των κατηγοριοποιητών ενδέχεται να προσαρμόζονται κατά λάθος σε ορισμένα feature που δεν συμβάλλουν απαραίτητα στην ουσία του αντικειμένου που ταξινομείται.
Παρακάτω παρουσιάζονται τα αποτελέσματα των κατηγοριοποιητών :

- **KNN-- Nearest Neighbor with k =1 Neighbor**

```
Training time: 0.05073404312133789 seconds

Testing time: 10.155903577804565 seconds

For the 1-NN classifier accuracy is: 0.3858
For the 1-NN classifier, F1 score is: 0.3851
```

- **KNN-- Nearest Neighbor with k=3 Neighbors**

```
Training time: 0.046855926513671875 seconds

Testing time: 11.258288383483887 seconds

For the 3-NN classifier accuracy is: 0.3066
For the 3-NN classifier, F1 score is: 0.3569
```

- **Nearest Centroid with Euclidean Distance metric**

```
Training time: 0.03393840789794922 seconds

Testing time: 0.03734421730041504 seconds

For the Nearest Centroid classifier accuracy is: 0.3858
For the Nearest Centroid classifier, F1 score is: 0.2532
```

- **Nearest Centroid with Manhattan Distance metric**

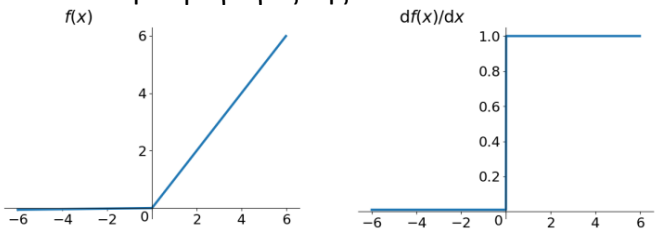
```
Training time: 0.14599990844726562 seconds

Testing time: 0.027576208114624023 seconds

For the Nearest Centroid classifier accuracy is: 0.3858
For the Nearest Centroid classifier, F1 score is: 0.2734
```

Παρατηρειται ότι τα αποτελέσματα δεν είναι ικανοποιητικά και οι παραπάνω κατηγοριοποιητες δεν διαχωρίζουν σωστά τις εικόνες του cifar 10 .

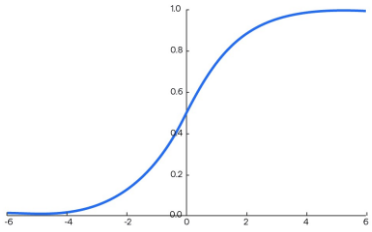
2. Classification με Multi-Layer Perceptron (Fully Densed)

Στόχος του classification της cifar 10 με MultiLayer Perceptron η επίτευξη μεγαλύτερου accuracy στο test set, σε σύγκριση με τις k Nearest Neighbors και Nearest Centroid μεθόδους. Σημαντικό κομμάτι της υλοποίησης είναι οι παράμετροι που θα επιλεγούν και θα καθορίσουν την απόδοση του νευρωνικου δικτύου. Κάποιες από αυτές τις παραμέτρους είναι το πλήθος των στρωμάτων, το πλήθος των νευρώνων κάθε στρώματος, το learning rate κλπ.
Για κάποιες παραμέτρους επιλέχθηκαν συγκεκριμένες τιμές , οι οποίες δεν αλλάζουν κατά την διάρκεια της εργασίας .

- Ως συνάρτηση ενεργοποίησης των κρυφών στρωμάτων (hidden layers) επιλέγεται η **ReLU** (Rectified Linear Unit), η οποία δίνει στην έξοδο 0 αν η είσοδος είναι αρνητική, αλλιώς δίνει την τιμή της εισόδου όταν αυτή είναι θετική. Η επιλογή αυτή είναι η πλέον διαδεδομένη για δομές MLP και CNN. Παρακατω , απεικονίζεται η ReLU και η παράγωγος της.



- Ως συνάρτηση ενεργοποίησης του στρώματος εξόδου επιλέγεται η **softmax**, η οποία μετατρέπει ένα διάνυσμα K πραγματικών αριθμών σε κατανομή πιθανότητας K πιθανών αποτελεσμάτων και δίνει στην έξοδο ένα διάνυσμα στο οποίο το άθροισμα των τιμών του ισούται με 1, επομένως μπορεί να θεωρηθεί ως πιθανότητες με τις οποίες ανήκει η είσοδος σε κάθε κλάση. Για το CIFAR-10 (10 κλάσεις), το διάνυσμα αυτό θα έχει μέγεθος 1x10 .



- Ως συνάρτηση κόστους (loss function) επιλέγεται η **categorical cross-entropy**, ειδικά για classification tasks που περιλαμβάνουν πολλαπλές κλάσεις. Εφαρμόζει λογαριθμικές συναρτήσεις στις προβλεπόμενες πιθανότητες, τιμωρώντας πιο σημαντικά σφάλματα, γεγονός που βοηθά στη σύγκλιση κατά τη διάρκεια του training. Στην συγκεκριμένη περιπτωση , η συναρτηση θα είναι :

$$\boldsymbol{CE\ Loss} = -\sum_{\boldsymbol{i=1}}^{\boldsymbol{number\ of\ Classes=10}} \boldsymbol{y_{test}} \times \boldsymbol{log}\ (\boldsymbol{y_{prediction}})$$

Όπου το y_{test} είναι η πραγματική κατανομή πιθανότητας της κάθε κλάσης για κάθε εικόνα μετα το one hot encoding (το label) πχ. [1 0 0 0 0 0 0 0 0 0] και το $y_{prediction}$ είναι η κατανομή πιθανότητας που προκύπτει από το μοντέλο. Η categorical cross-entropy loss function χρησιμοποιείται κατά την προσαρμογή των βαρών του μοντέλου κατά τη διάρκεια του training. Ο στόχος είναι να ελαχιστοποιηθεί το loss, δηλαδή όσο μικρότερη είναι η απώλεια τόσο καλύτερο είναι το μοντέλο. Θεωρητικά , ένα τέλειο μοντέλο θα είχε loss = 0 .

- Για την αρχικοποίηση των βαρών χρησιμοποιήθηκε η μέθοδος **HeNormal()** , η οποία προτείνεται στις περιπτώσεις , οπού ως συνάρτηση ενεργοποίησης χρησιμοποιείται η ReLU οποία παίρνει δείγματα από μια περικομμένη κανονική κατανομή με κέντρο το 0 και με τυπική απόκλιση ίση με **sqrt(2 / fan_in)** όπου fan_in είναι αριθμός εισόδων στον weight tensor (για πλήρως συνδεδεμένο δίκτυο ο αριθμός των νευρώνων του προηγουμένου layer.

- Ως μετρικές αξιολόγησης επιλέγονται στις περισσότερες περιπτώσεις το accuracy και το f1 score. Το accuracy λειτουργεί θεωρώντας σωστή την κατηγοριοποίηση αν η μεγαλύτερη πιθανότητα από την softmax βρίσκεται στο index της επιθυμητής κλάσης. Το **f1 score** είναι ο αρμονικός Μ.Ο. των μετρικών ***precision*** (ποσοστό positive προβλέψεων που ήταν όντως positive) και ***recall*** (ποσοστό όλων των positive που προβλέφθηκαν σωστά) και έχει μέγιστη τιμή 1. Δηλαδή το υπολογίζεται ως εξής : $\boldsymbol{f1\ score} = \mathbf{2} \times \frac{\boldsymbol{recall} \times \boldsymbol{precision}}{\boldsymbol{recall} + \boldsymbol{precision}}$

- Το **batch_size** τέθηκε ίσο με 128, καθώς επέφερε ικανοποιητικά αποτελέσματα. Δοκιμάστηκαν και άλλα batch sizes χωρίς μεγάλες διαφοροποιήσεις (32 , 128 , 256 , 512 , 1000 ,1200)

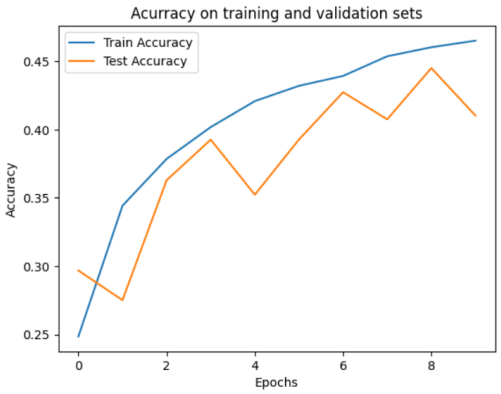
- Ως βελτιστοποιητής (optimizer) επιλέγεται ο **Adam**. Ο Adam είναι από τους πιο διαδεδομένους optimizer για classification . Επιτρέπει στον ρυθμό εκμάθησης(learning rate) να προσαρμόζεται κατά τη διάρκεια του training, γεγονός που μπορεί να οδηγήσει σε ταχύτερη σύγκλιση και καλύτερη απόδοση. Στην ουσία κάνει μεγάλα βήματα όταν τα gradients δεν διαφέρουν πολύ και πολύ μικρά βήματα όταν έχουμε ραγδαία αλλαγή. Έτσι οδηγούμαστε σε σχετικά καλό τοπικό ελάχιστο της συνάρτησης κόστους.

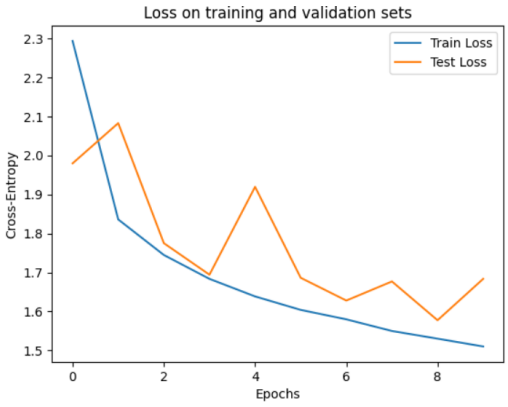
Δημιουργία βασικού μοντέλου

Για το αρχικό νευρωνικό δίκτυο επιλέχθηκε 1 δίκτυο με την απλή δομή των 2 layers. Για το ενδιάμεσο δίκτυο επιλέχθηκαν αυθαίρετα 512 νευρώνες μετά από κάποιες δοκιμές και προφανώς για το layer εξόδου 10 , όσες δηλαδή και οι κλάσεις .
Παράμετροι : epochs = 10 , validation_split = 0.2 , learning_rate = 0.001

Αποτελέσματα

```
Training time: 118.568 seconds
Test accuracy: 0.4368
Test F1 Score: 0.3480
```






```
Printing examples of correct classification:
```

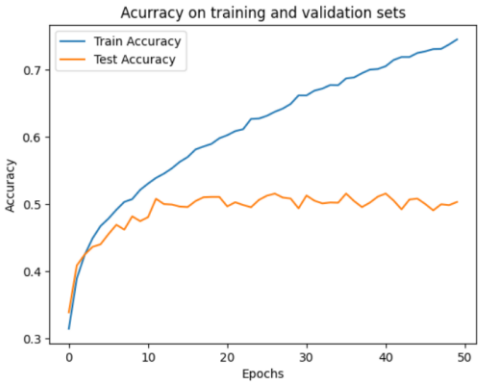

```
Printing examples of false classification:
```

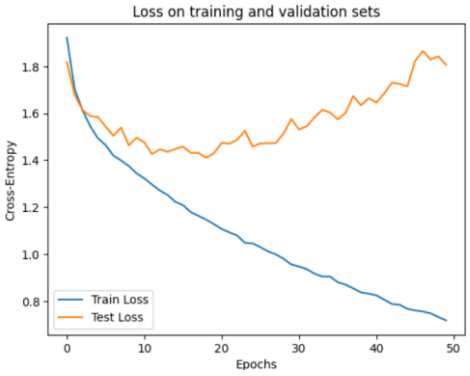
Μοντέλο 1

Για το 1ο μοντέλο ουσιαστικά προστέθηκε ένα 2ο ενδιάμεσο στρώμα νευρώνων στο οποίο χρησιμοποιήθηκαν 512 νευρώνες και αλλάξαν ως εξής οι παράμετροι: epochs = 50 ενώ για το validation testing χρησιμοποιήθηκε το 20% του train set . Το batch size παρέμεινε σταθερό 128 .

Αποτελέσματα

```
Training time: 804.081 seconds
Test accuracy: 0.5001
Test F1 Score: 0.4066
```





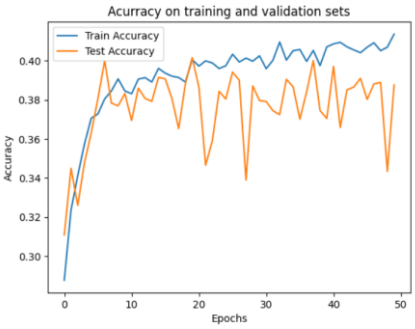
Printing examples of correct classification

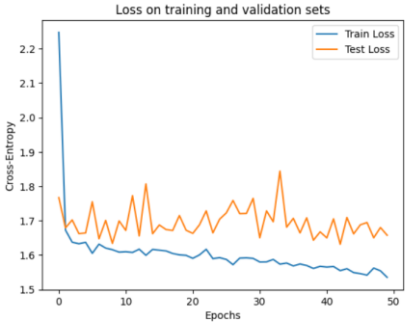
Printing examples of false classification:

Different Learning Rate

Σε περιπτωση αλλαγής του learning rate από 0.001 σε 0.01 παίρνω τα εξής αποτελέσματα :

```
Training time: 743.675 seconds
Test accuracy: 0.4021
Test F1 Score: 0.3523
```



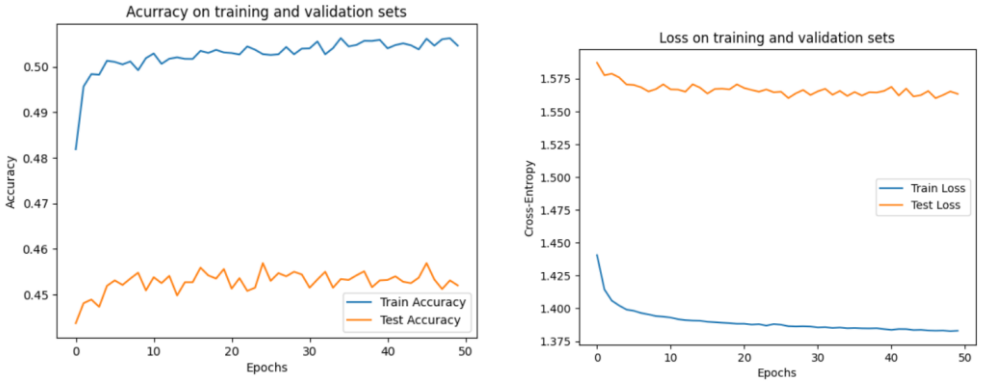


Παρατηρείται ότι η επιλογή του learning rate είναι ιδιαίτερα σημαντική και διαφοροποιεί τις επιδόσεις του μοντέλου κάτι που φαίνεται από την διαφοροποίηση των plots και την μείωση του accuracy. Δεκαπλασιάστηκε και για το ίδιο δίκτυο παρουσιάστηκαν σαφώς μεγαλύτερες αστάθειες. Θα είναι μια από τις παραμέτρους και στο Hyper-Parameter Tuning που θα παρουσιαστεί παρακάτω.

Different Optimizer: Stochastic Gradient Descent (SGD)

Σε περιπτωση που χρησιμοποιηθεί διαφορετικός optimizer έχω τα εξής αποτελέσματα :

```
Training time: 623.515 seconds
Test accuracy: 0.4474
```

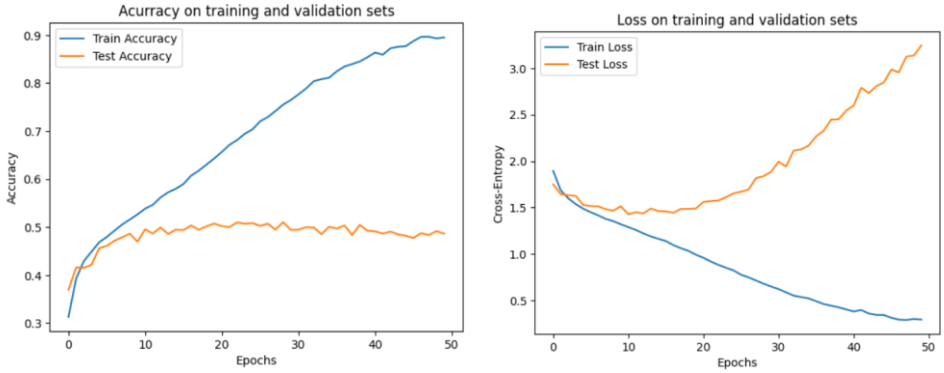


Παρατηρείται ότι υπαρχει μια μικρη μειωση στο accuracy . Στο υπολοιπο κομματι της εργασιας θα χρησιμοποιηθεί ο Adam .

Add one more hidden layer

Προστειθεται ένα ακομα ενδιαμεσο στρωμα με 256 νευρωνες και προκύπτουν τα εξής αποτελέσματα :

```
Training time: 764.378 seconds
Test accuracy: 0.4225
Test F1 Score: 0.3551
```



Παρατηρειται ότι παρα την προσθηκη ενός ακομα ενδιαμεσου layer δεν υπαρχει βελτιωση της αποδοσης του δικτυου και επομενως θα διατηρηθει η δομη του Model 1 με τα 2 που αναλυθηκε πριν με τα 2 layers . Επισης , από τα

διαγραμματα παρατηρειται ότι υπαρχει overfitting. Παρακατω , θα παρουσιαστουν μεθοδοι κανονικοποιησης για την απουφη του .

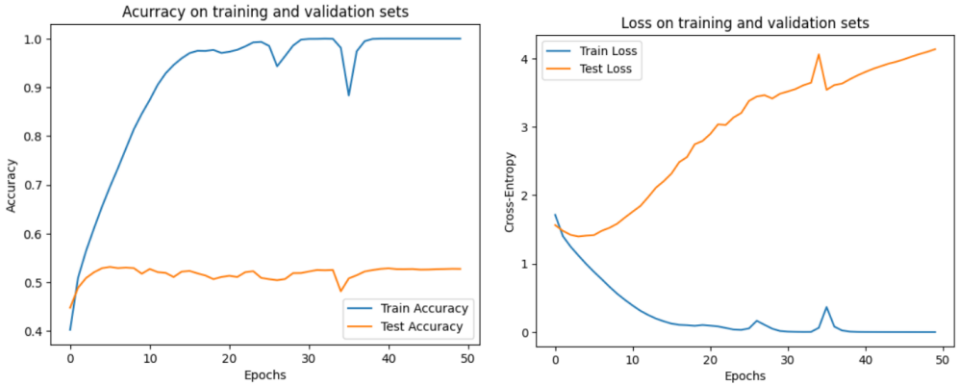
Στο υπόλοιπο κομμάτι της εργασίας χρησιμοποιείται το XTrain και το XTest που προκύπτουν μετα την μείωση της διάστασης όπου το 90% της διακύμανσης διατηρείται σε μόλις 99 features από 3072 που ήταν πριν , καθώς εκτελώντας τις προηγούμενες περιπτώσεις μετα την μείωση διάστασης με PCA → XTrainPCA και XTestPCA μειωθηκε αρκετά ο χρόνος εκπαίδευσης , χωρίς να επηρεαστεί σχεδόν καθόλου η απόδοση των παραπάνω δικτύων.

Weight initialization

Είναι μια σημαντική παράμετρος στο σχεδιασμό ενός μοντέλου νευρωνικών δικτύων και είναι απαραίτητη για τη σύγκλιση του δικτύου κατά τη φάση της εκπαίδευσης. Για την αρχικοποίηση των βαρών χρησιμοποιήθηκε η μέθοδος HeNormal() , η οποία προτείνεται στις περιπτώσεις , όπου ως συνάρτηση ενεργοποίησης χρησιμοποιείται η ReLU.

Αποτελέσματα :

```
Training time: 202.814 seconds
Test accuracy: 0.5330

f1Score: 0.4577
```



Παρατηρείται βελτίωση τόσο στο accuracy και όσο στο f1_measure . Παρόλα αυτά από τα διαγράμματα , φαίνεται ότι υπάρχει overfitting παρακάτω εξετάζονται κάποιες μέθοδοι για την αποφυγή του .

Regularization to avoid overfitting

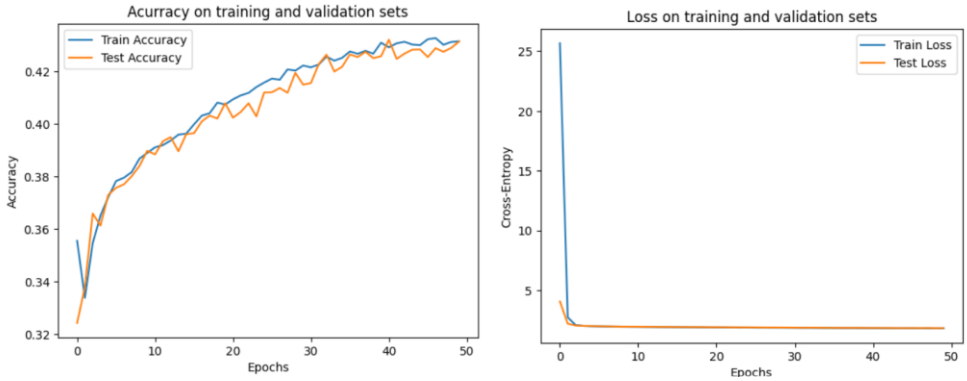
Η κανονικοποίηση είναι μια τεχνική που κάνει μικρές τροποποιήσεις στον αλγόριθμο μάθησης έτσι ώστε το μοντέλο να γενικεύεται καλύτερα. Αυτό με τη σειρά του βελτιώνει την απόδοση του μοντέλου.

L1 Regularization of Weights

Η Lasso Regularization χρησιμοποιείται για τον έλεγχο της πολυπλοκότητας του μοντέλου και την αποφυγή του overfitting. Η ιδέα πίσω από την τακτοποίηση L1 είναι να τροποποιηθεί η συνάρτηση κόστους έτσι ώστε τα μεγαλύτερα βάρη να προκαλούν τη συνάρτηση κόστους να αποκτά μεγαλύτερες τιμές. Τελικά, στην τακτοποίηση L1 ορισμένα βάρη θα λάβουν 0 τιμές που σημαίνει ότι η σύνδεση αυτού του νευρώνα δεν είναι ενεργή. Η συνάρτηση κόστους αυξάνεται μόνο κατά ένα μικρό ποσοστό της νόρμας L1 των βαρών, εδώ χρησιμοποιούμε συντελεστή 0,01.

Αποτελέσματα :

```
Training time: 213.350 seconds
Test accuracy: 0.4355
Test F1 Score: 0.4303
```



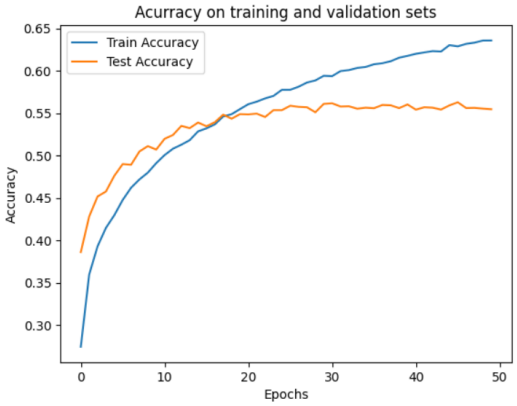
Παρατηρείται μείωση στο accuracy όμως από τα διαγράμματα διαπιστώνεται ότι δεν έχουμε overfitting πλέον .

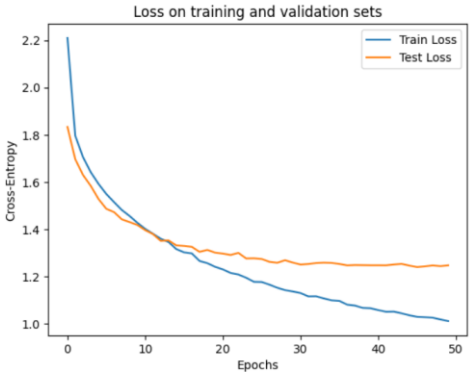
Dropout

Εφαρμόζεται σε κάποιο επίπεδο του δικτύου, ορίζοντας μία τιμή 0 < p < 1 η οποία καθορίζει το ποσοστό των νευρώνων του επιπέδου οι οποίοι θα απενεργοποιηθούν με τυχαία επιλογή για κάθε επανάληψη . Η τεχνική του dropout περιορίζει περιπτώσεις οπού υπάρχουν κόμβοι που διορθώνουν λάθη προηγουμένων, οδηγώντας σε όχι καλά αποτελέσματα και οδηγούν σε αδυναμία γενίκευσης σε διαφορετικά δεδομένα. Μετα από κάποιες δοκιμές για το συγκεκριμένο μοντέλο επιλέγεται η τιμή 0.4 για το p .

Αποτελέσματα :

```
Training time: 262.922 seconds
Test accuracy: 0.5627
```





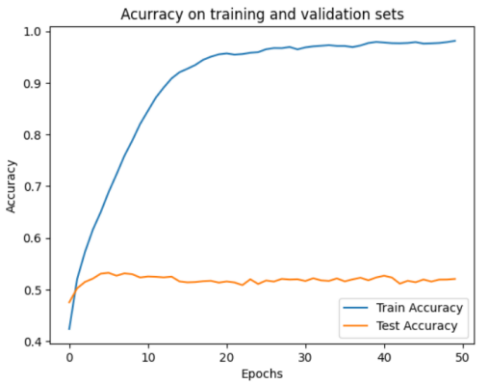
Παρατηρείται βελτίωση στο accuracy και στο f1 measure , και από τα διαγράμματα φαίνεται ότι στο train set το accuracy αλλά και η τιμή της loss function έφτασαν σε ικανοποιητικές τιμές αλλά όχι και στο test (validation set) και έχουμε σαφώς βελτίωση στο κομμάτι του overfitting.

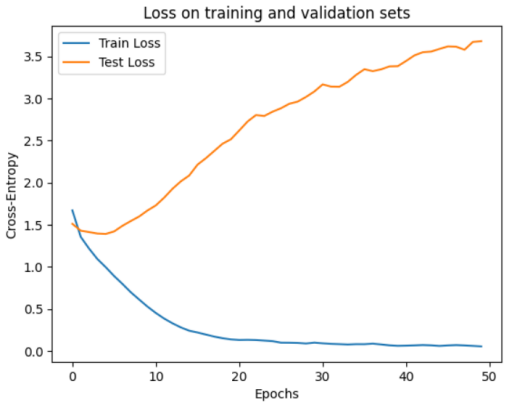
Batch Normalization

Με αυτήν την τεχνική γίνεται υπολογισμός της μέσης τιμής και διακύμανσης για κάθε διάσταση του διανύσματος εισόδου, κατά μήκος του batch, εκτελείται κανονικοποίηση (αφαίρεση μέσης τιμής και διαίρεση με τη διακύμανση) πριν την εφαρμογή της συνάρτησης ενεργοποίησης. Βοηθάει ιδιαίτερα στην διαδικασία εκπαίδευσης του μοντέλου. Δίνεται η δυνατότητα να χρησιμοποιηθούν υψηλότεροι ρυθμοί εκμάθησης, επιτρέποντας την ταχύτερη σύγκλιση.

Αποτελέσματα :

```
Training time: 263.405 seconds
Test accuracy: 0.5060
Test F1 Score: 0.3205
```





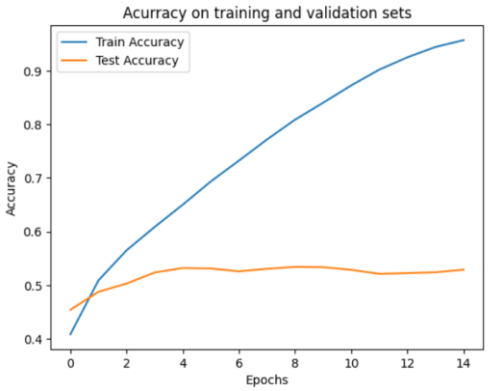
Παρατηρείται βελτίωση (όχι όσο στην περιπτωση του batch normalization) στο accuracy και στο f1 measure , και από τα διαγράμματα φαίνεται ότι στο train set το accuracy αλλά και η τιμή της loss function (αντίθετα αυξήθηκε) έφτασαν σε ικανοποιητικές τιμές αλλά όχι και στο test (validation set) .

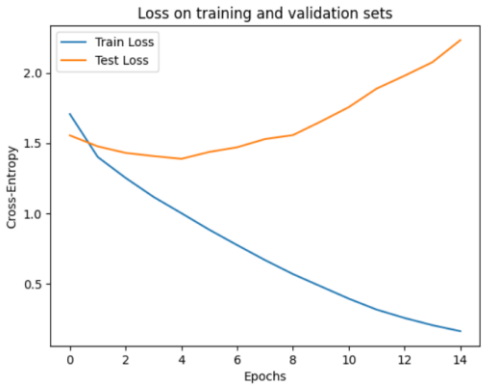
Early Stopping

Με βάση την επίδοση του μοντέλου πάνω στο validation set μπορεί να ληφθεί απόφαση για πρόωρο τερματισμό της εκπαίδευσης ώστε να αποφευχθεί η περίπτωση του overfitting. Η απόφαση λαμβάνεται όταν η συνάρτηση κόστους δεν έχει βελτιωθεί (δηλαδή βρεθεί σε νέο ελάχιστο) στη διάρκεια ενός συγκεκριμένου αριθμού εποχών εκπαίδευσης. Η παράμετρος "patience" καθορίζει τον αριθμό των εποχών ή των επαναλήψεων που περιμένει ο αλγόριθμος πριν σταματήσει την εκπαίδευση και επιλέγεται ίση με 10.

Αποτελέσματα :

```
Epoch 15: early stopping
Training time: 99.476 seconds
Test accuracy: 0.5215
Test F1 Score: 0.4196
```





Παρατήρηση : ένα από τα θέματα που παρατηρήθηκε είναι ο κατάλληλος αριθμός των εποχών για τις οποίες εκπαιδεύεται το δίκτυο.

Hyper-Parameter Tuning

Σε αυτό το στάδιο της εργασίας και μετά από τις δοκιμές για κάποιες υπερπαραμετρους θα γίνει προσπάθεια εύρεσης των καταλληλότερων για το επιλεγμένο dataset. Αυτοί θα είναι το πλήθος των στρωμάτων, το πλήθος των νευρώνων, και το learning rate και το p του dropout . Για την επιλογή των υπερπαραμετρων χρησιμοποιείται ο **RandomSearch** tuner από τη βιβλιοθήκη **KerasTuner.** Αντί να πραγματοποιηθεί ένα grid search στο οποίο θα ελεγχθούν όλοι οι πιθανοί συνδυασμοί που θα ορίσουμε, και επομένως το πλήθος των συνδυασμών θα αυξάνονταν εκθετικά για κάθε υπερπαράμετρο που θα προσθέταμε, ο RandomSearch tuner ελέγχει επιλεκτικά κάποιους πιθανούς συνδυασμούς από όλο το grid για να τους δοκιμάσει. Για κάθε συνδυασμό που επιλέγει έχει οριστεί να γίνεται training 2 φορές (executions_per_trial) με 50 epochs για να διασφαλιστεί ότι το τελικό αποτέλεσμα ώστε να μην υπάρχει μεγάλος

βαθμός τυχαιότητας στο τελικό μοντέλο . Ο RandomSearch tuner ενδέχεται όμως να μην προσφέρει βέλτιστες λύσεις.

- Learning Rate : [0.1 0.01 0.001]
- Neurons 1st Layer : [2048 1024 512]
- Neurons 2nd Layer : [1024 512 256]
- P of dropout : [0.1 0.2 0.3 0.4]

Αποτελέσματα

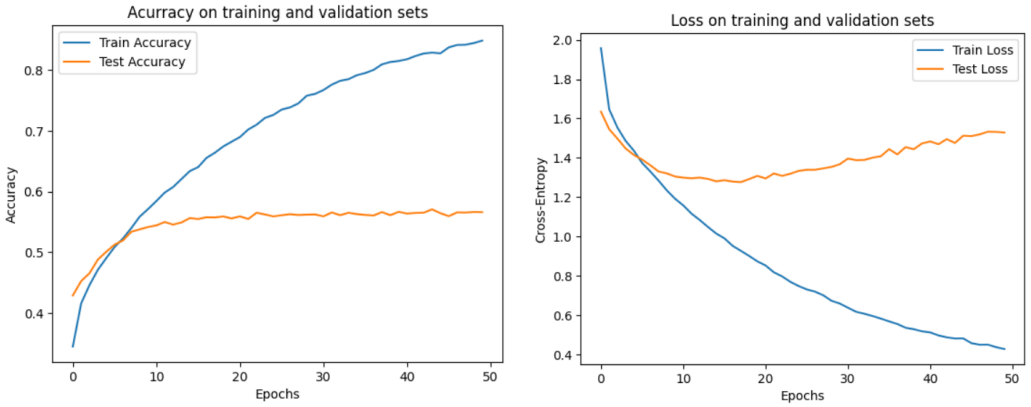
```
Trial 40 Complete [00h 23m 54s]
val_accuracy: 0.5676999986171722

Best val_accuracy So Far: 0.5676999986171722
Total elapsed time: 07h 12m 22s
Total time: 25941.732 seconds

Optimal number of neurons in 1st layer: 2048
Optimal number of neurons in 2nd layer: 1024
Optimal value of p for dropout: 0.4
Optimal value of learning rate: 0.001
```

Ως στόχος τέθηκε η μεγιστοποίηση του validation accuracy. Τα αποτελέσματα δείχνουν ότι το βέλτιστο μοντέλο είναι με **2048** νευρώνες στο 1ο layer, **1048** για το δεύτερο και ρυθμό εκμάθησης **0.001** και η πιθανότητα του dropout στο 0.4 **.**

Optimal Model

Για την δομή του MLP που προέκυψε μετα το hyparameter tuning παίρνουμε τα παρακάτω αποτελέσματα αν εκπαιδευτεί για 50 epochs :

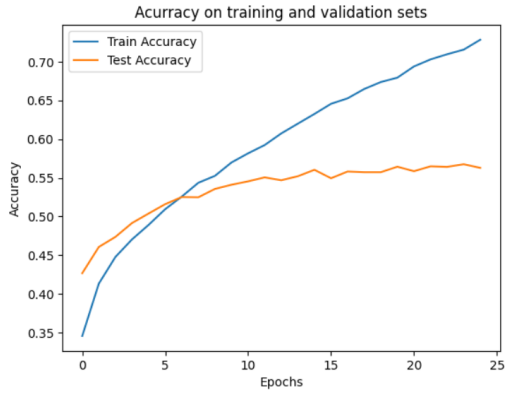
```
Training time: 1223.616 seconds
Test accuracy: 0.5690
f1Score: 0.4519
```

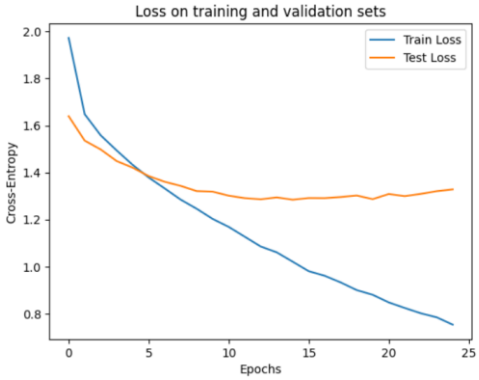


Παρατηρειται πως υπαρχει overfitting και επομενως θα δοκιμαστει να εκπαιδευτει το optimalModel για 25 epochs.

Τα αποτελέσματα που προκύπτουν είναι τα εξής :

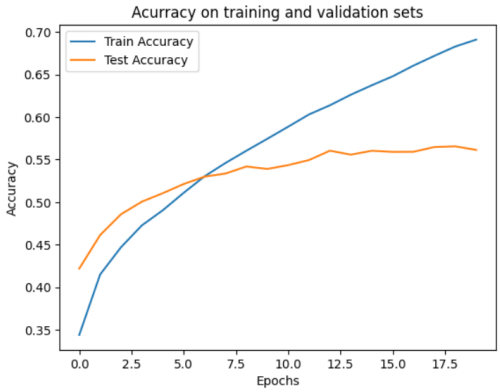
```
Training time: 624.244 seconds
Test accuracy: 0.5623
f1Score: 0.4268
```

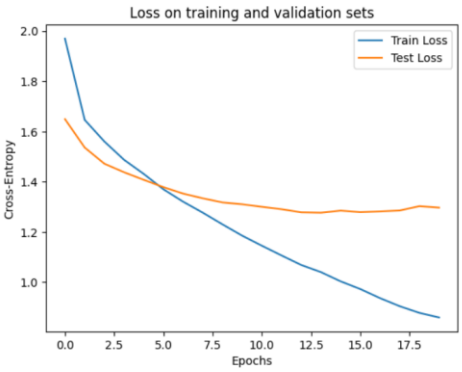




Και για 20 epochs :

```
Training time: 468.519 seconds
Test accuracy: 0.5611
f1Score: 0.4117
```





Printing examples of correct classification: Printing examples of false classification:

3. Classification με Convolutional Neural Network

Σε αυτό το στάδιο της εργασίας θα χρησιμοποιηθεί ένα συνελικτικό νευρωνικό δίκτυο στο CIFAR-10 dataset, με στόχο την περαιτέρω βελτίωση της ακρίβειας του μοντέλου.

Ένα σημαντικό χαρακτηριστικό των νευρωνικών δικτύων είναι ότι κάθε κόμβος ενός layer είναι συνδεδεμένος με μία υποπεριοχή του προηγούμενου επιπέδου. Αυτό προσφέρει μικρότερη πολυπλοκότητα, καθώς οι λιγότερες συνδέσεις οδηγούν και σε λιγότερες παραμέτρους προς εκμάθηση.

Η δομή ενός συνελικτικου νευρωνικου δικτύου αποτελείται από μία σειρά συνελικτικά επίπεδα, πιθανώς ακολουθούμενα και από άλλα επίπεδα το καθένα (π.χ. Batch Normalization, Dropout), και με σκοπό την εξαγωγή χαρακτηριστικών από τις εικόνες με διαδοχικές πράξεις συνέλιξης και μείωσης διαστάσεων. Στη συνέχεια υπάρχει ένα επίπεδο Flatten που χρησιμοποιείται για τη μετατροπή πολυδιάστατων δεδομένων σε μονοδιάστατο πίνακα , πριν από την τροφοδοσία τους σε πλήρως συνδεδεμένα (dense) επίπεδα, τα οποία αναμένουν εισόδους σε μονοδιάστατη μορφή.
Οι παράμετροι που επιλέγουμε σε κάθε convolutional layer :

- Αριθμός φίλτρων (**filter number**), ο αριθμός των διαφορετικών τοπικών χαρακτηριστικών.
- Μέγεθος παραθύρου συνέλιξης (**kernel size**), το μέγεθος του παραθύρου το οποίο θα βγάζει τα τοπικά χαρακτηριστικά.
- Το **zero-padding** για το οποίο επιλέγεται το ‘same’ ώστε να διατηρείται ίδιο το μέγεθος της εικόνας ανά συνελικτικό στρώμα.
- Η συναρτηση ενεργοποίησης όπως και στα ενδιάμεσα layers του MLP επιλέγεται να είναι η ReLU.

Μεταξύ των διαδοχικών layers του CNN συχνά εισάγονται επίπεδα υποδειγματοληψιας με σκοπό την μείωση των διαστάσεων των δεδομένων (άρα και του χρόνου εκπαίδευσης) και την εξαγωγή μόνο των πιο σημαντικών χαρακτηριστικών αν η υποδειγματοληψια εφαρμοστεί στα καταλληλά σημεία. Εδώ , επιλέγεται το **MaxPooling** στρώμα σε 2x2 διαστάσεις όπου τα δεδομένα χωρίζονται σε μικρές ομάδες ανάλογα με το βήμα της υποδειγματοληψίας και λαμβάνεται το μέγιστο από αυτές.

Παρατήρηση : Τα πλήρως συνδεδεμένα στρώματα στην έξοδο επιλέγονται ώστε να οδηγήσουν σε έξοδο 1x1x10 και να γίνει η σύγκριση των label.

Επίσης , για την αποφυγή του overfitting χρησιμοποιείται η τεχνική του dropout όπως και στο MLP .

1ο Μοντέλο

Για τα συνελικτικά μοντέλα που θα παρουσιαστούν σε αυτό το κομμάτι της εργασίας μετα από κάποιες δοκιμές στο 1ο CNN model η μέθοδος που προτιμήθηκε για την αποφυγή του overfitting είναι αυτή του dropout και η τιμή p τέθηκε στο 0.2

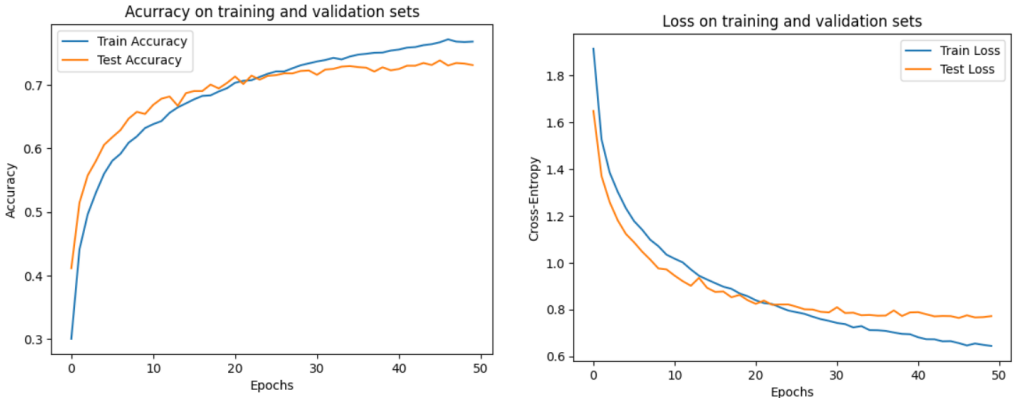
Για το 1ο μοντέλο του CNN χρησιμοποιείται η ακόλουθη δομή :

- Convolutional Layer (filter = 32x32 , kernel size = 4x4)
- MaxPooling Layer 4x4
- Convolutional Layer (filter = 64x64 , kernel size = 8x8)
- MaxPooling Layer 4x4
- Convolutional Layer (filter = 128x128 , kernel size = 8x8)
- MaxPooling Layer 4x4
- Στρώμα Flatten
- Dense Layer με 512 νευρώνες
- Dense Layer εξόδου με 10 νευρώνες και συναρτηση ενεργοποίησης την softmax

Σημείωση: οι εποχές για το 1ο CNN είναι 50 ενώ η παράμετρος validation split 0.2 Και το learning rate = 0.001 όπως προέκυψε από hyperparameter tuning για το optimal MLP NN .

Αποτελέσματα :

```
Training time: 2364.445 seconds
Test accuracy: 0.7324
Test F1 Score: 0.7247
```



Παρατηρούμε λοιπόν ότι η ακρίβεια αυξήθηκε σημαντικά σε σχέση με το fully densed MLP .Λόγω των φίλτρων των συνελικτικων επιπέδων τα CNN μπορούν να αναγνωρίσουν μοτίβα ανεξάρτητα από τη θέση τους μέσα σε μια εικόνα και το κάθε φίλτρο του CNN προσαρμόζεται και μαθαίνει συγκεκριμένο χαρακτηριστικό , βελτιώνοντας έτσι το accuracy και το f1 score σε σχέση με το MLP. Είναι εμφανής επίσης η εξάλειψη του overfitting από τα διαγράμματα .

Παραδείγματα σωστής και λάθους κατηγοριοποίησης

Printing examples of correct classification:

Printing examples of false classification:

2ο Μοντέλο

Για το 2ο μοντέλο του CNN χρησιμοποιείται η ακόλουθη δομή :

- Convolutional Layer (filter = 16x16 , kernel size = 4x4)
- Convolutional Layer (filter = 32x32 , kernel size = 4x4)
- MaxPooling Layer 2x2
- Convolutional Layer (filter = 64x64 , kernel size = 4x4)
- MaxPooling Layer 2x2
- Convolutional Layer (filter = 128x128 , kernel size = 4x4)
- MaxPooling Layer 2x2
- Στρώμα Flatten
- Dense Layer με 512 νευρώνες
- Dense Layer εξόδου με 10 νευρώνες και συναρτηση ενεργοποίησης την softmax

Ουσιαστικά προστίθενται 2 ακόμα επίπεδα με φίλτρα ένα 16x16 και ένα 128X128 . Επίσης , ελαττώνεται και το παράθυρο υποδειγματοληψίας σε 2x2 .

Τα αποτελέσματα για λογούς συνοπτικότητας δεν παρατίθενται και περνάμε κατευθείαν στο 3ο μοντέλο

Η διαφοροποίηση στο 3ο μοντέλο η προσθήκη διπλού συνελικτικου επιπέδου στη θέση του καθενός . Επίσης το Max Pooling είναι 2x2 όπως και το kernel size γίνεται 2x2 με σκοπό την περαιτέρω βελτίωση του μοντέλου .
https://d2l.ai/chapter_convolutional-modern/vgg.html

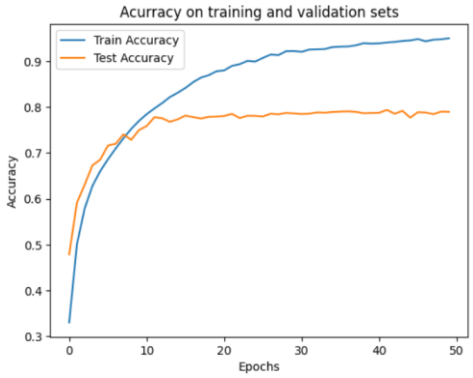
3ο Μοντέλο

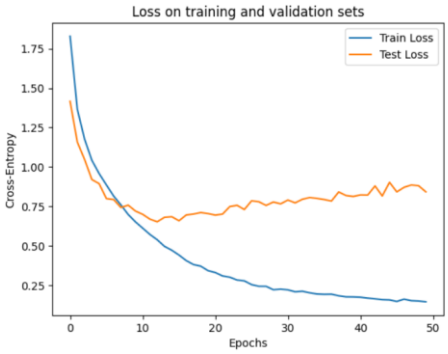
Για το 3ο μοντέλο του CNN χρησιμοποιείται η ακόλουθη δομή :

- Convolutional Layer (filter = 32x32 , kernel size = 2x2)
- Convolutional Layer (filter = 32x32 , kernel size = 2x2)
- MaxPooling Layer 2x2
- Convolutional Layer (filter = 64x64 , kernel size = 2x2)
- Convolutional Layer (filter = 64x64 , kernel size = 2x2)
- MaxPooling Layer 2x2
- Convolutional Layer (filter = 128x128 , kernel size = 2x2)
- Convolutional Layer (filter = 128x128 , kernel size = 2x2)
- MaxPooling Layer 2x2
- Στρώμα Flatten
- Dense Layer με 512 νευρώνες
- Dense Layer εξόδου με 10 νευρώνες και συναρτηση ενεργοποίησης την softmax

Τα αποτελέσματα που προέκυψαν παρατίθενται παρακάτω :

```
Training time: 7225.323 seconds
Test accuracy: 0.7825
Test F1 Score: 0.7845
```

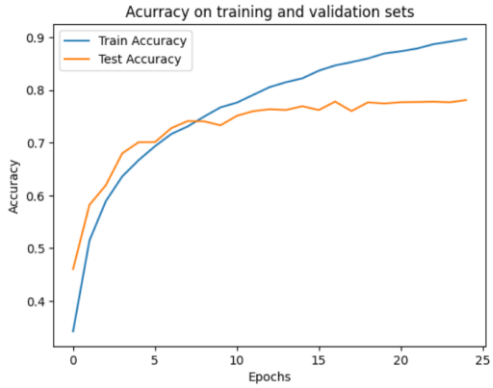


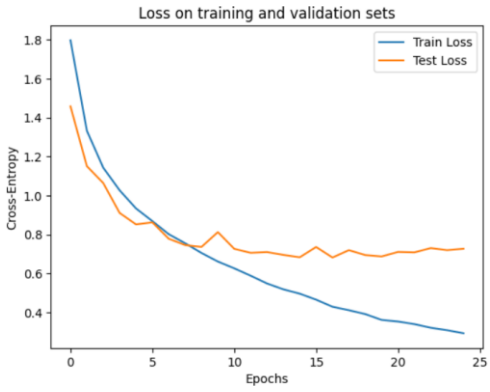


Παρατηρείται ότι το διπλό συνελικτικό layer όντως βοήθησε και πλέον η κατηγοριοποίηση φαίνεται ότι γίνεται σε ένα αρκετά ικανοποιητικό επίπεδο. Επίσης , φαίνεται ότι υπάρχει overfitting για τον συγκεκριμένο αριθμό epochs . Επομένως , μειώνεται στις 25 .

Αποτελέσματα για epochs = 25

```
Training time: 3680.151 seconds
Test accuracy: 0.7752
Test F1 Score: 0.7775
```

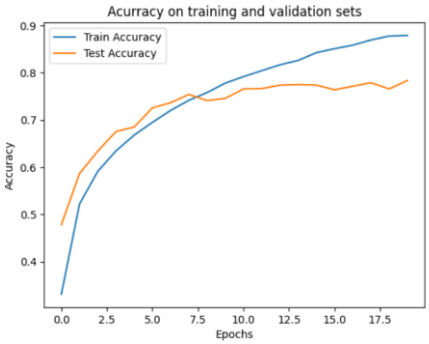


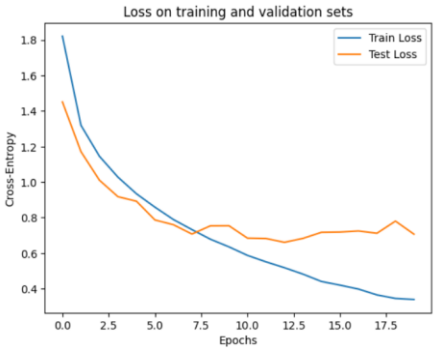


Παρακατω παρατιθενται και τα παραδειγματα σωστης και λανθασμενης κατηγοριοποιησης για το CNN .

Αν μειώσουμε ακόμα περισσότερο τις εποχές (epochs=20) τότε παίρνουμε τα παρακάτω αποτελέσματα :

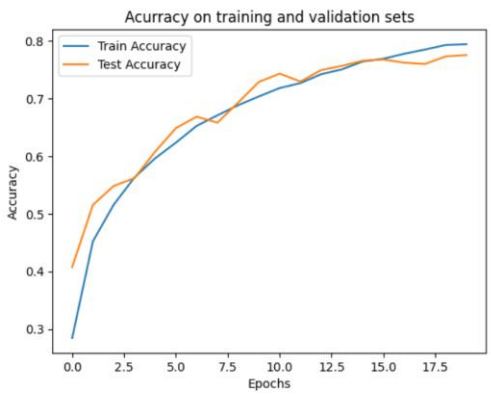
```
Training time: 2939.572 seconds
Test accuracy: 0.7644
Test F1 Score: 0.7676
```

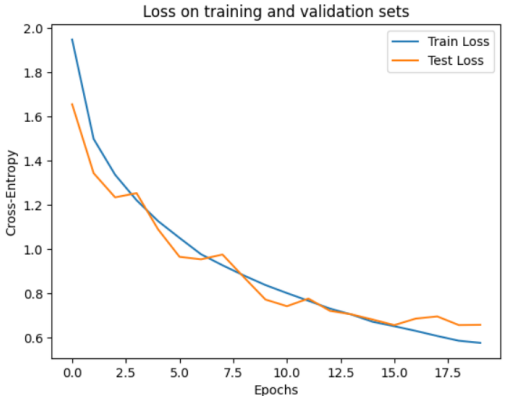




Παρατηρείται ότι δεν έχει αποφευχθεί πλήρως το overfitting. Επομένως , σε αυτή την περιπτωση προκειμένου να βελτιωθεί περαιτέρω το CNN θα αλλάξει η τιμή του p του dropout σε 0.3 (έγινε και με 0.4--δεν παρουσιάζεται διότι ικανοποιητικότερο αποτέλεσμα είχε το p=0.3) . Έτσι πάλι για 20 epochs παίρνουμε τα παρακάτω αποτελέσματα:

```
Training time: 3265.458 seconds
Test accuracy: 0.7666
Test F1 Score: 0.7657
```





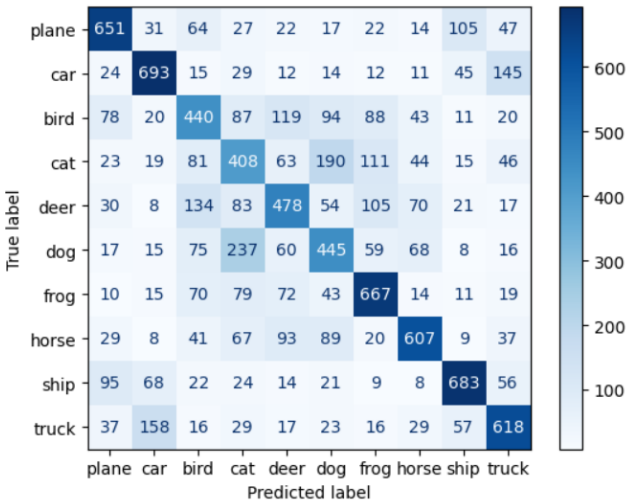
Είναι εμφανές πλέον ότι το overfitting εξαλείφθηκε και η απόδοση του CNN δεν επηρεάστηκε ιδιαίτερα .

Σύνοψη Αποτελεσμάτων

Στο τελικό κομμάτι της εργασίας παρουσιάζονται οι confusion matrices για κάθε μοντέλο.

MultiLayer Perceptron NN

	precision	recall	f1-score	support
0	0.65	0.65	0.65	1000
1	0.67	0.69	0.68	1000
2	0.46	0.44	0.45	1000
3	0.38	0.41	0.39	1000
4	0.50	0.48	0.49	1000
5	0.45	0.45	0.45	1000
6	0.60	0.67	0.63	1000
7	0.67	0.61	0.64	1000
8	0.71	0.68	0.70	1000
9	0.61	0.62	0.61	1000

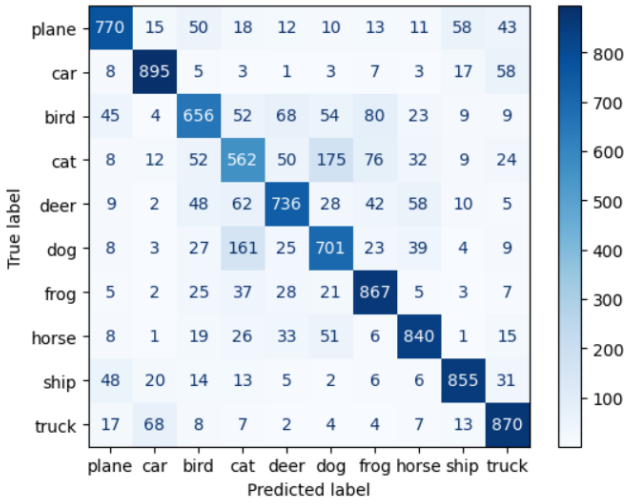


Από τον πίνακα σύγχυσης φαίνεται ότι την χειρότερη επίδοση την έχουν η γάτα , η οποία αναγνωρίζεται συχνά ως σκύλος αλλά και ως βάτραχος. Επίσης , το πουλί

αναγνωρίζεται συχνά και ως ελάφι αλλά και ως σκύλος . Το ελάφι αναγνωρίζεται συχνά ως πουλί και ως βάτραχος και ο σκύλος με την σειρά του πολύ συχνά ως γάτα. Τέλος , αν και έχει έχει σχετικά καλύτερη κατηγοριοποίηση από το φορτηγό αναγνωρίζεται ως αυτοκίνητο

Convolutional NN (3rd Model)

	precision	recall	f1-score	support
0	0.83	0.77	0.80	1000
1	0.88	0.90	0.89	1000
2	0.73	0.66	0.69	1000
3	0.60	0.56	0.58	1000
4	0.77	0.74	0.75	1000
5	0.67	0.70	0.68	1000
6	0.77	0.87	0.82	1000
7	0.82	0.84	0.83	1000
8	0.87	0.85	0.86	1000
9	0.81	0.87	0.84	1000
accuracy			0.78	10000
macro avg	0.77	0.78	0.77	10000
weighted avg	0.77	0.78	0.77	10000



Από τον πίνακα σύγχυσης παρατηρείται ότι το μοντέλο έχει αρκετά καλή επίδοση. Παρατηρείται ότι η γάτα έχει την χειρότερη κατηγοριοποίηση καθώς πολύ συχνά αναγνωρίζεται ως σκύλος (λιγότερο συχνά ως βάτραχος) όπως και ο σκύλος πολλές φορές αναγνωρίζεται ως γάτα . Το αυτοκίνητο , ο βάτραχος , το άλογο , το καράβι και το φορτηγό φαίνεται ότι αναγνωρίζονται σε ικανοποιητικό βαθμό .

Είναι εμφανές ότι τα CNN είναι πιο αποτελεσματικά σε datasets όπως της cifar 10. Τηη καλύτερη επίδοση είχε το CNN με ακρίβεια κοντά στα 78-80%, το οποίο είναι λογικό όταν έχουμε να αντιμετωπίσουμε classification tasks φωτογραφιών RGB (3 χρωματικά κανάλια) . Το fully-densed MLP δεν μπόρεσε να ξεπεράσει το 58% της ακρίβειας για το συγκεκριμένο σετ δεδομένων, το οποίο δείχνει και τα όριά του σε σετ με πολλές κλάσεις. Φυσικά, οι τεχνικές kNN και Nearest Centroid είναι ακατάλληλες για το συγκεκριμένο πρόβλημα καθώς αυτό που ουσιαστικά κάνουν είναι να ομαδοποιούν τις φωτογραφίες χρωματικά, το οποίο δεν οδηγεί

σε καλές επιδόσεις, αφού το background ή το χρώμα του αντικειμένου μπορούν να επηρεάσουν το label το οποίο τελικά θα ανήκει μια φωτογραφία. Στα μοντέλα MLP και CNN, καθοριστικό ρόλο στην αύξηση της επίδοσης και την μείωση του overfitting είχε η χρήση της τεχνικής Dropout.