Νευρωνικά Δίκτυα - Βαθιά Μάθηση

Τμήμα Πληροφορικής ΑΠΘ – 7ο Εξάμηνο

Φθινόπωρο 2021

Υλοποίηση νευρωνικού δικτύου πολυστρωματικού perceptron (MLP) για επίλυση προβλημάτων κατηγοριοποίησης

Σπυράκης Άγγελος (9352)

aspyrakis@ece.auth.gr

Τμήμα Ηλεκτρολόγων Μηχανικών & Μηχανικών Η/Υ

Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης

1 Εισαγωγή

Στην πρώτη ενδιάμεση εργασία του μαθήματος παρουσιάστηκαν τα αποτελέσματα εφαρμογής των κατηγοριοποιητών Nearest Centroid και k-Nearest Neighbor στη βάση δεδομένων **CIFAR-10**. Τα αποτελέσματα δεν ήταν καθόλου ικανοποιητικά και τα ποσοστά ακρίβειας ήταν σημαντικά χαμηλά. Συνεπώς, θα εξετάσουμε σε αυτή την εργασία διαφορετικές πρακτικές οι οποίες ενδέχεται να επιφέρουν καλύτερα αποτελέσματα. Οι δομές που θα δοκιμαστούν αφορούν πολυστρωματικά perceptron (MLP) και ενδεχομένως συνελικτικά νευρωνικά δίκτυα.

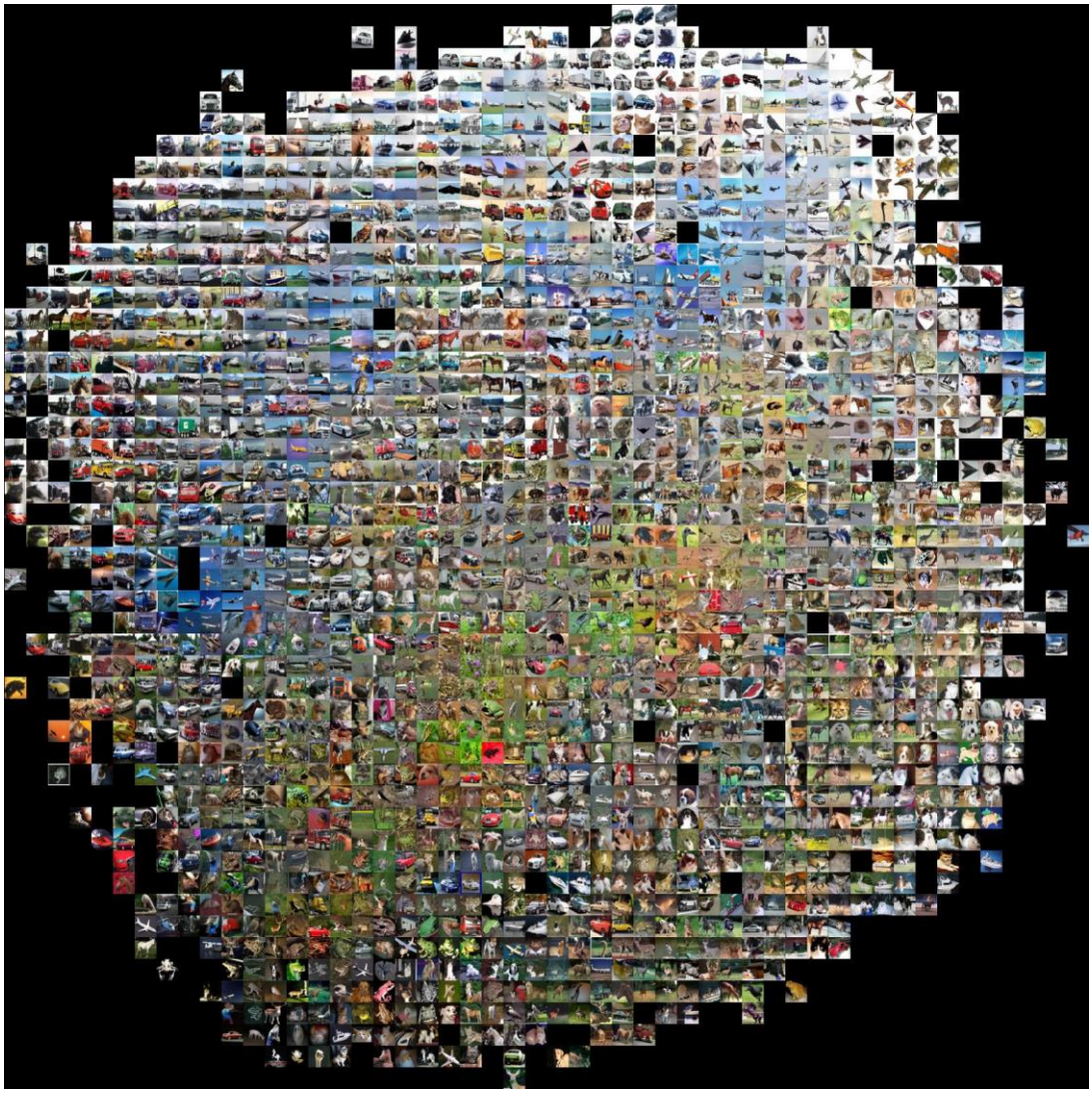
Τα αποτελέσματα των προηγούμενων δοκιμών παρουσιάζονται ξανά στον επόμενο πίνακα ώστε να χρησιμοποιηθούν σαν βάση για τα επόμενα πειράματα.

Πίνακας 1 - Αποτελέσματα k-NN και NC.

Προηγούμενα αποτελέσματα		
Method	**Accuracy**	**Time elapsed**
Nearest Neighbor (k=3)	**35.69%**	50.71 sec
Nearest Centroid	27.74%	0.52 sec

Ο λόγος για τον οποίο οι μέθοδοι αυτοί δεν λειτουργούν αποτελεσματικά είναι επειδή στην ουσία συγκρίνουν την γενικότερη χρωματική κατανομή των εικόνων ή το είδος του φόντου για να αποφασίσουν αν υπάρχει ομοιότητα. Δηλαδή, θεωρητικά μια εικόνα με ένα σκύλο και μια εικόνα με μια γάτα στις οποίες το φόντο είναι άσπρο δύναται να θεωρηθούν ότι ανήκουν στην ίδια κλάση. Χρησιμοποιώντας την τεχνική t-SNE[1], μπορούμε να οπτικοποιήσουμε την ομοιότητα των εικόνων βάσει απόστασης. Στην παρακάτω εικόνα, φωτογραφίες που είναι κοντά στον 2D χώρο έχουν και κοντινή ευκλείδεια απόσταση.

[1] https://lvdmaaten.github.io/tsne/



Εικόνα 1 - Οπτικοποίηση των όμοιων βάσει απόστασης εικόνων (t-SNE).

Παρατηρούμε ότι σχεδόν όλες οι εικόνες ανήκουν σε ένα ενιαίο cluster, στο οποίο γειτονικές εικόνες έχουν το ίδιο χρωματικό pattern (π.χ. γαλάζιο ουρανού και θάλασσας, πράσινο βλάστησης κ.ο.κ.). Μεγεθύνοντας στην γαλάζια περιοχή, θα δούμε ότι πλοίο, φορτηγό, αεροπλάνο και πουλί θα κατηγοριοποιηθούν στην ίδια κλάση, λόγω του background τους (Εικόνα 2).

Εικόνα 2 - Μεγέθυνση της t-SNE εικόνας στην γαλάζια περιοχή.

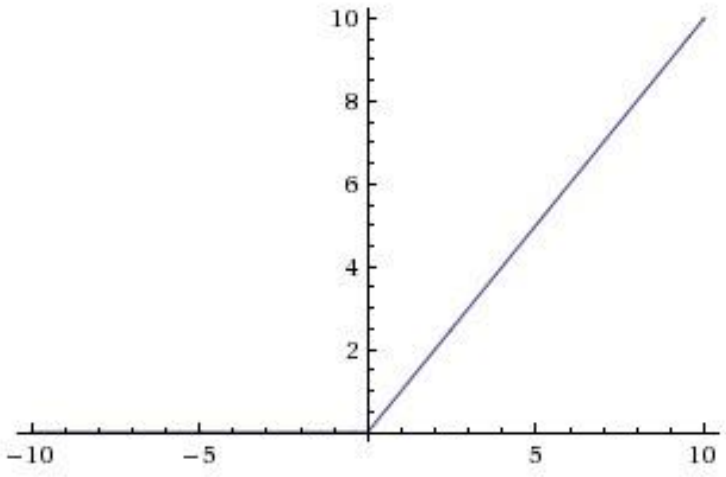
2 Κατηγοριοποίηση με Multi-Layer Perceptron

Στόχος της χρήσης της MLP δομής δικτύου θα είναι η επίτευξη μεγαλύτερου accuracy στο test set, σε σύγκριση με τις NN και NC μεθόδους. Η διαδικασία όμως αυτή δεν είναι απλή, καθώς οι παράμετροι που θα επιλεγούν θα καθορίσουν σε μεγάλο βαθμό την επιτυχία του νέου δικτύου. Οι παράμετροι αυτοί ενδέχεται να είναι το πλήθος των στρωμάτων, το πλήθος των νευρώνων κάθε στρώματος, ο ρυθμός μάθησης, παράγοντες κανονικοποίησης, και πολλά άλλα.

Λόγω του ότι επιδιώκουμε να λύσουμε πρόβλημα κατηγοριοποίησης, και επειδή το dataset δεν είναι τόσο περίπλοκο και μεγάλο, γίνονται οι εξής a priori επιλογές για την δομή του MLP, οι οποίες δεν θα αλλάζουν κατά τη διάρκεια του fine tuning του δικτύου:

- Ως βελτιστοποιητής (optimizer) επιλέγεται ο **Adam**[2]. Ο Adam είναι από τους πιο διαδεδομένους για προβλήματα κατηγοριοποίησης, καθώς συνδυάζει τα πλεονεκτήματα των επεκτάσεων του Stochastic Gradient Descent (SGD), την AdaGrad και την RMSProp. Αντί να προσαρμόζει το learning rate βάσει των πρώτων ροπών (δηλαδή της μέσης τιμής) όπως η RMSProp, ο Adam χρησιμοποιεί και τον Μ.Ο. των δεύτερων ροπών των gradient (δηλαδή την “uncentered” διακύμανση). Στην ουσία κάνει μεγάλα βήματα όταν οι gradients δεν διαφέρουν πολύ και πολύ μικρά βήματα όταν έχουμε ραγδαία αλλαγή. Έτσι οδηγούμαστε σε σχετικά καλό τοπικό ελάχιστο της συνάρτησης κόστους και αποφεύγουμε τις ταλαντώσεις.
- Ως αρχικοποιητής των βαρών επιλέγεται η κλάση **HeNormal**[3], η οποία παίρνει δείγματα από μια περικομμένη κανονική κατανομή με κέντρο το 0 και με τυπική απόκλιση ίση με $\sqrt{2/input\_units\_of\_tensor}$. Γενικότερα είναι προτιμότερο τα βάρη να έχουν τιμές κοντά στο μηδέν ώστε να επιτυγχάνεται πιο εύκολα η γενίκευση στο dataset μας.
- Ως συνάρτηση ενεργοποίησης των κρυφών στρωμάτων επιλέγεται η **ReLU**, η οποία δίνει στην έξοδο 0 αν η είσοδος είναι αρνητική, αλλιώς δίνει την τιμή της εισόδου όταν αυτή είναι θετική (Εικόνα 3). Η επιλογή αυτή είναι η πλέον διαδεδομένη για δομές MLP και CNN.
- Ως συνάρτηση ενεργοποίησης του στρώματος εξόδου επιλέγεται η **softmax**, η οποία 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, το διάνυσμα αυτό θα έχει μέγεθος 1x10.

[2] https://arxiv.org/abs/1412.6980
[3] https://www.cv-foundation.org/openaccess/content_iccv_2015/html/He_Delving_Deep_into_ICCV_2015_paper.html



Εικόνα 3 - Συνάρτηση ενεργοποίησης ReLU.

- Ως συνάρτηση κόστους (loss function) επιλέγεται η **categorical cross-entropy**, η οποία αφορά προβλήματα κατηγοριοποίησης που κάθε εικόνα μπορεί να ανήκει μόνο σε μία κλάση. Γι' αυτό επιλέχθηκε και παραπάνω η softmax για το στρώμα εξόδου. Στην περίπτωσή μας ορίζεται ως:

$$loss = -\sum_{i=1}^{10} y_i * \log \hat{y}_i$$

  όπου y_i η επιθυμητή έξοδος και $\hat{y}_i$ η πραγματική. Φυσικά, στην προεπεξεργασία του dataset τα labels υπέστησαν **one-hot encoding** για να είναι και αυτά σε μορφή διανύσματος και για να μην υπάρχει λογική αριθμητική συσχέτιση μεταξύ των labels στην έξοδο.
- Ως μετρικές αξιολόγησης επιλέγονται οι **categorical accuracy** και **f1_measure**. Η categorical accuracy θεωρεί σωστή την κατηγοριοποίηση αν η μεγαλύτερη πιθανότητα από την softmax βρίσκεται στο index της επιθυμητής κλάσης και η f1_measure 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.
- Τέλος, το **batch_size** τέθηκε ίσο με 1000, καθώς επέφερε ικανοποιητικά αποτελέσματα με ισοσταθμισμένο χρόνο εκπαίδευσης.

Τώρα που το backbone του δικτύου έχει οριστεί, πρέπει να επιλεχθούν οι σωστοί **υπερπαράμετροι** για το επιλεγμένο dataset. Όπως αναφέρθηκε παραπάνω, αυτοί θα είναι το πλήθος των στρωμάτων, το πλήθος των νευρώνων, και το learning rate.

Όσον αφορά το πλήθος των στρωμάτων, θεωρητικά 1 layer μπορεί να μάθει τέλεια το dataset, αλλά αυτό χωρίς να σημαίνει ότι έχει την ικανότητα της γενίκευσης. Επίσης για μεγάλο πλήθος νευρώνων σε ένα στρώμα, δεν έχουμε αποδοτικούς αλγόριθμους που να μπορούν να εκπαιδεύσουν το δίκτυο γρήγορα. Γι' αυτό, και επειδή το dataset δεν είναι ιδιαίτερα μεγάλο και με πολλές κλάσεις, τοποθετούνται **2 κρυφά στρώματα** στο δίκτυο.

Για την επιλογή των καταλληλότερων νευρώνων ανά στρώμα και του learning rate χρησιμοποιείται ο **RandomSearch** tuner από τη βιβλιοθήκη **KerasTuner**[4]. 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 60 epochs για να διασφαλιστεί ότι το τελικό αποτέλεσμα δεν βασίζεται σε τυχαιότητα της στιγμής. Ο RandomSearch tuner ενδέχεται να μην προσφέρει βέλτιστες λύσεις, αλλά υποβέλτιστες.

Η διαδικασία του fine-tuning έφερε τα εξής αποτελέσματα:

```
Best val_f1 metric So Far: 0.5190

Total elapsed time: 01h 42m 30s

The hyperparameter search is complete. The optimal number of
units in the first densely-connected layer is 1024, in the second
densely-connected layer is 512, and the optimal learning rate
for the optimizer is 0.001.
```

Ως στόχος τέθηκε η μεγιστοποίηση του validation f1_measure. Βλέπουμε ότι οι επιλογές που έκανε ήταν **1024** νευρώνες για το πρώτο στρώμα, **512** για το δεύτερο και ρυθμό εκμάθησης **0.001**. Η όλη διαδικασία διήρκησε 1 ώρα και 42 λεπτά. Οι συνδυασμοί από τους οποίους είχε να διαλέξει ήταν οι εξής:

Units_first → [2048, 1024, 512, 256]

Unit_second → [2048, 1024, 512, 256]

Learning_rate → [0.1, 0.01, 0.001]

48 πιθανοί συνδυασμοί

Ενδεικτικά παρουσιάζονται τα αποτελέσματα της εκπαίδευσης για διάφορους συνδυασμούς πάνω στο σετ ελέγχου.

[4] https://keras.io/keras_tuner/

Πίνακας 2 - Αποτελέσματα MLP με διάφορους συνδυασμούς neurons για σταθερό lr.

Neurons #1	Neurons #2	Learning Rate	Training Duration	Loss	Accuracy	F1
2048	1024	0.001	57.13sec	1.4970	52.22%	0.5145
1024	512	0.001	51.26sec	1.4711	**52.57%**	**0.5192**
1024	1024	0.001	53.82sec	1.4777	51.59%	0.4986
1024	256	0.001	46.77sec	1.4123	52.11%	0.5064
512	512	0.001	42.84sec	1.4285	52.22%	0.5019
512	256	0.001	41.79sec	**1.4226**	51.57%	0.4860

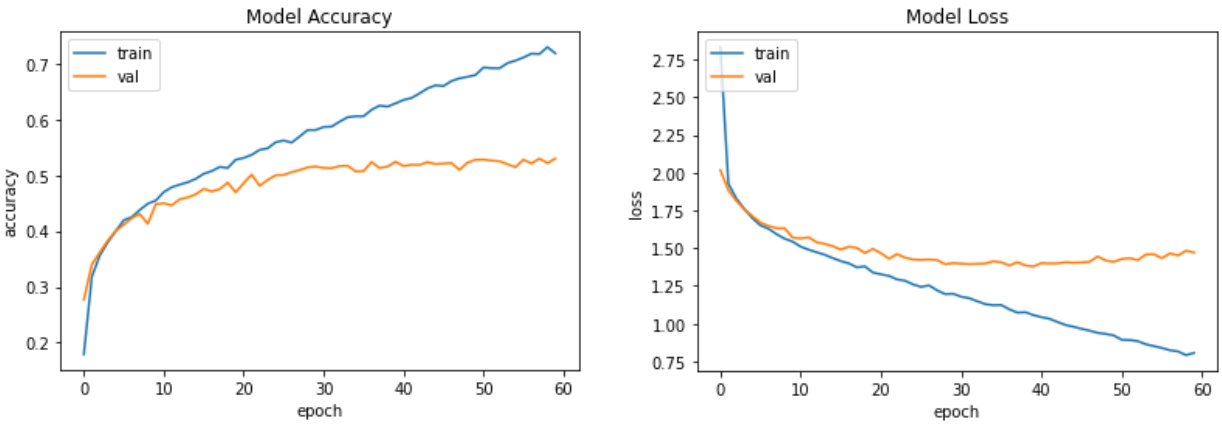
Παρατηρούμε λοιπόν ότι την μεγαλύτερη ακρίβεια την είχαμε όντως για 1024 και 512 νευρώνες. Όσο μειώνεται το συνολικό πλήθος των νευρώνων, τόσο μειώνεται και ο χρόνος εκπαίδευσης, αφού ο back propagation αλγόριθμος έχει λιγότερα βάρη να ανανεώσει λόγω των λιγότερων συνδέσεων. Η μετρική f1 είναι η καλύτερη μέθοδος αξιολόγησης του μοντέλου, καθώς εξετάζει συνολικά την ακρίβεια της κατηγοριοποίησης. Π.χ. για τον συνδυασμό [512, 256] το loss ήταν χαμηλότερο απ' ότι στο [1024, 512] (χαμηλότερο = καλύτερο), αλλά η ακρίβεια και η f1 ήταν χαμηλότερη (χαμηλότερη = χειρότερο).

Πίνακας 3 - Αποτελέσματα MLP με σταθερό πλήθος neurons και διαφορετικό lr.

Neurons #1	Neurons #2	Learning Rate	Training Duration	Loss	Accuracy	F1
1024	512	0.001	51.26sec	**1.4711**	**52.57%**	**0.5192**
1024	512	0.01	52.77sec	2.3025	10.01%	0.0
1024	512	0.1	47.07sec	2.3050	10.00%	0.0

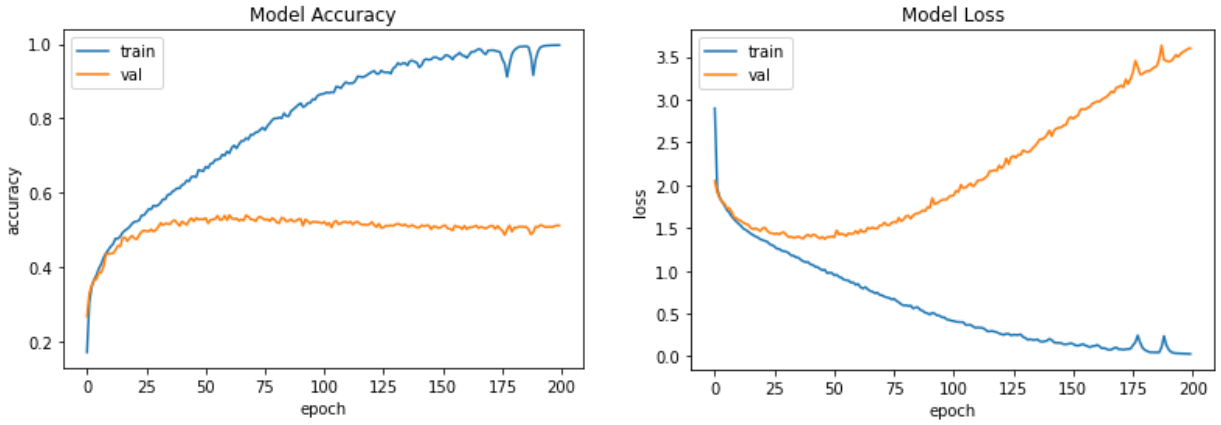
Στον πίνακα 3 φαίνεται ξεκάθαρα η σημαντικότητα του learning rate στην ακρίβεια του μοντέλου. Βλέπουμε ότι για μεγαλύτερο learning rate από 0.001, άρα και μεγαλύτερη μεταπήδηση στη καμπύλη, το μοντέλο έχει αχρηστευτεί. Το 10% στην στήλη του accuracy δηλώνει ότι το μοντέλο κάνει **τυχαία επιλογή** σε κάθε περίπτωση, και επειδή το πρόβλημα αποτελείται από 10 κλάσεις η ακρίβεια προκύπτει κατά Μ.Ο. 100/10 = 0.1. Στην ουσία έχουμε ταλάντωση χωρίς να συγκλίνει ποτέ.

Ας δούμε τώρα τις γραφικές που προκύπτουν για το **βέλτιστο μοντέλο (1024, 512, 0.001)**:



Σχήμα 1 - Accuracy και loss για τον συνδυασμό (n1, n2, lr) = (1025, 512, 0.001).

Παρατηρούμε λοιπόν ότι ενώ η καμπύλη του validation στο accuracy και στο loss έχει σχετικά σταθεροποιηθεί, η καμπύλη του train συνεχίζει να αυξάνεται και να μειώνεται αντίστοιχα, το οποίο σημαίνει ότι το μοντέλο συνεχίζει να υπερεκπαιδεύεται. Επειδή όμως οπτικά βλέπουμε μια πολύ μικρή αύξηση στο val_accuracy, θα δοκιμάσουμε μήπως η υπερεκπαίδευση μας οδηγήσει και σε καλύτερα validation αποτελέσματα. Ενδεικτικά, για τον ίδιο συνδυασμό και για 200 epochs προκύπτουν τα εξής διαγράμματα:



Σχήμα 2 - Accuracy και loss για τον ίδιο συνδυασμό με epochs=200.

Είναι εμφανές πλέον ότι αν συνεχίσουμε να εκπαιδεύουμε του μοντέλο τότε αυτό δεν γενικεύει σωστά στο validation set με αποτέλεσμα να αυξάνεται σημαντικά το loss και να μειώνεται το accuracy. Σχηματικά, η καλύτερη περιοχή για την εκπαίδευση ήταν όντως μεταξύ 50 και 60 epochs.

Σαν επόμενο βήμα, για να δοκιμάσουμε αν μπορεί να γίνει περαιτέρω **γενίκευση** στο μοντέλο και να επιτευχθεί μεγαλύτερη ακρίβεια στο test set, δοκιμάζεται η προσθήκη **Dropout**. Το Dropout είναι μια τεχνική κατά την οποία μερικοί έξοδοι ενός στρώματος αγνοούνται με τυχαίο τρόπο. Αυτό κάνει το συγκεκριμένο στρώμα να φαίνεται ότι έχει λιγότερους κόμβους σε σχέση με πριν, με αποτέλεσμα η διαδικασία της εκπαίδευσης να γίνεται πιο “noisy”, αναγκάζοντας

κάποιους κόμβους να αναλαμβάνουν μεγαλύτερη «ευθύνη» διαχείρισης των εισόδων. Αυτό μας συμφέρει διότι οι κόμβοι μπορεί να αλλάξουν με τρόπο που να διορθώνουν τα λάθη των άλλων κόμβων, λάθη που μπορούν να οδηγήσουν σε πολύπλοκα **co-adaptations** (συν-προσαρμογές). Αυτό με τη σειρά του οδηγεί σε overfitting επειδή τα co-adaptations **δεν γενικεύονται** σε άγνωστα δεδομένα.

Για να δούμε αρχικά σε ποιο layer πρέπει να μπει το Dropout, θα το εφαρμόσουμε στο προηγούμενο μοντέλο με πιθανότητα 0.3 ώστε να γίνει η επιλογή της θέσης. Για κάθε περίπτωση θα ληφθεί ο Μ.Ο. από 5 εκπαιδεύσεις. Τα αποτελέσματα παρουσιάζονται στον παρακάτω πίνακα.

Πίνακας 3 - Δοκιμή Dropout στο μοντέλο (n1, n2, lr) = (1024, 512, 0.001).

Dropout (0.3)	Loss	Accuracy	F1
not used	1.4709	52.62%	0.5187
after first layer	1.3305	52.46%	0.4486
after second layer	1.3654	**53.55%**	0.5029
after both layers	1.3649	51.31%	0.3973

Από τις δοκιμές προκύπτει ότι η περίπτωση στην οποία εισήχθη το Dropout **μετά το δεύτερο στρώμα** έφερε μεγαλύτερη ακρίβεια στο test set και ελάχιστα μικρότερο f1 score απ' ότι στην εκδοχή χωρίς Dropout. Οι περιπτώσεις των Dropout μετά το πρώτο στρώμα και μετά από κάθε στρώμα είχαν λίγο χαμηλότερη ακρίβεια και σημαντικά μικρότερο f1 score, οπότε αποκλείονται.

Αξίζει επομένως να γίνει ένα **νέο tuning**, στο οποίο θα δοκιμαστούν οι ίδιοι συνδυασμοί πλήθους νευρώνων, με σταθερό learning rate ίσο με 0.001, με executions_per_trial = 4 και με **διαφορετικές πιθανότητες** dropout μετά το δεύτερο στρώμα. Συγκεκριμένα θα δοκιμαστούν οι πιθανότητες [0.25, 0.3, 0.4]. Χαμηλότερες τιμές δεν δοκιμάζονται γιατί στην ουσία η επίδραση του Dropout θα χαθεί. Οι υψηλότερες τιμές δεν δοκιμάζονται επίσης επειδή έτσι θα χάνονταν πολλές από τις συνδέσεις του δικτύου και θα μειωνόταν σημαντικά η επίδοση.

Το αποτέλεσμα του tuning με επιπλέον χρήση Dropout είναι το εξής, και προέκυψε μετά από 34 trials (αντί για σύνολο 48):

```
Best val_f1_metric So Far: 0.5364420413970947
Total elapsed time: 01h 35m 32s
Tuning time = 5732.1351146698 second(s)

    The hyperparameter search is complete. The optimal number of units in the first densely-connected
    layer is 2048, in the second densely-connected layer is 1024 and the optimal dropout probability
    is 0.25.
```

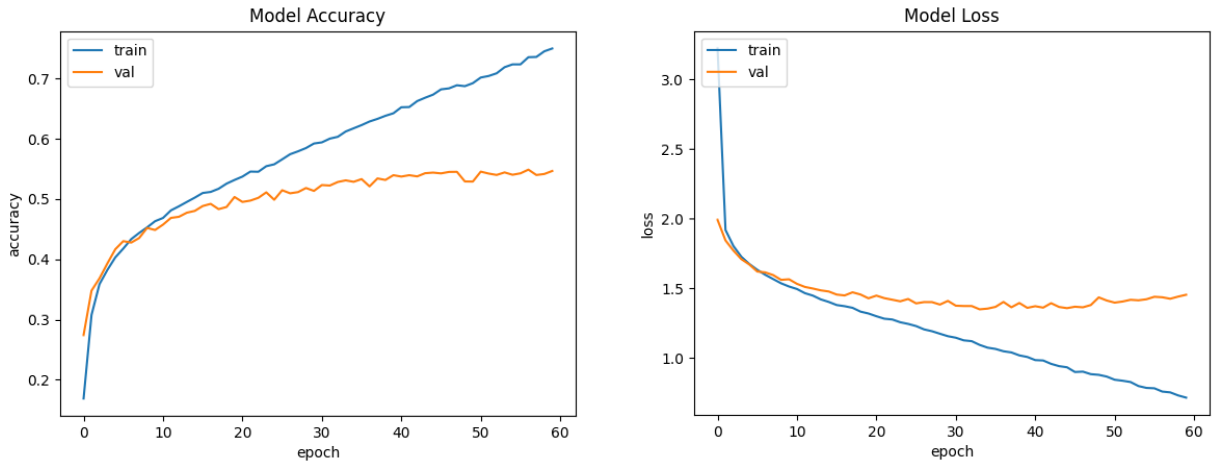
Άρα συνοψίζοντας, τα δύο μοντέλα που έχουν προκύψει με RandomSearch tuning είναι:

Πίνακας 4 - Τα μοντέλα που προέκυψαν μέσω fine-tuning.

Neurons #1	Neurons #2	Learning Rate	Dropout	Val_f1
1024	512	0.001	No used	0.5190
2048	**1024**	0.001	0.25	**0.5364**

Αξίζει να σημειωθεί ότι για το νέο μοντέλο που προέκυψε, η εκπαίδευσή του χωρίς Dropout είχε επιφέρει f1 = 0.5145, επομένως έχουμε βελτίωση. Επίσης οι δύο αυτοί συνδυασμοί νευρώνων (1024, 512) και (2048, 1024) είχαν σχεδόν ίδια επίδοση στο πρώτο tuning, οπότε ο νέος συνδυασμός που προέκυψε είναι λογικός.

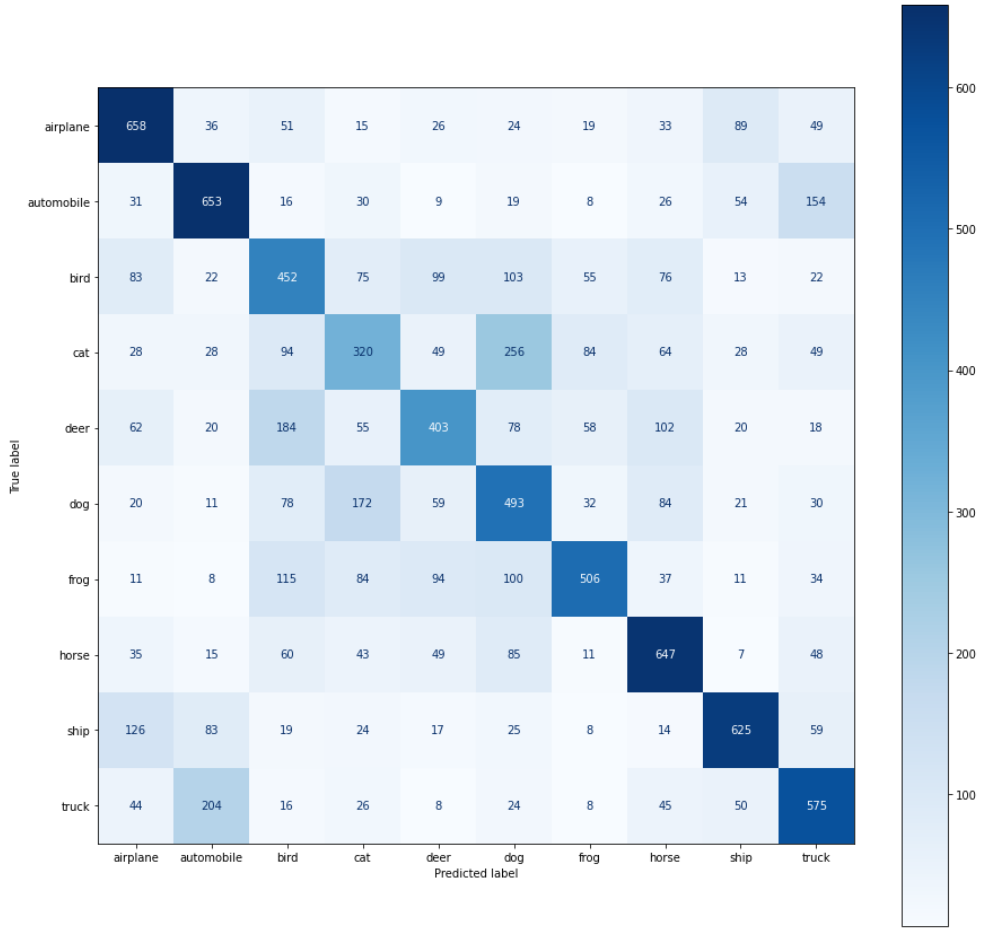
Οι **καμπύλες μάθησης** που προκύπτουν από αυτό το μοντέλο καθώς και η επίδοσή του παρουσιάζονται παρακάτω:



Σχήμα 3 – Learning curves για το τελικό MLP μοντέλο.

Πίνακας 5 - Επίδοση τελικού μοντέλου στο σετ ελέγχου.

Final Model's Performance on Test Set			
Training Duration	**Loss**	**Accuracy**	**F1**
59.20sec	1.4350	53.84%	0.5301



Σχήμα 4 - Confusion matrix τελικού μοντέλου.

Παρατηρούμε λοιπόν από τον πίνακα σύγχυσης ότι το μοντέλο έχει σχετικά καλή επίδοση. Σαν κλάση, την χειρότερη επίδοση είχε η «γάτα», η οποία συχνά χαρακτηρίζονταν ως σκύλος ή πουλί. Αντίστοιχα, στην περίπτωση του αυτοκινήτου βλέπουμε ότι πολλές φορές αντιστοιχίζεται στην κλάση «φορτηγό», το οποίο είναι λογικό, και το αντίστροφο. Μερικά παραδείγματα εσφαλμένης και σωστής κατηγοριοποίησης όπως προκύπτουν από το μοντέλο παρουσιάζονται παρακάτω:

Εικόνα 4 - Παραδείγματα ορθής κατηγοριοποίησης.

Εικόνα 5 - Παραδείγματα λανθασμένης κατηγοριοποίησης.

Συνοψίζοντας, οι βέλτιστες μέχρι τώρα επιδόσεις με κάθε τεχνική που χρησιμοποιήθηκε παρουσιάζονται ξανά στον παρακάτω πίνακα.

Πίνακας 6 - Σύνοψη αποτελεσμάτων KNN, NC, dense MLP.

Σύνοψη αποτελεσμάτων		
Method	**Accuracy**	**Time elapsed**
Nearest Neighbor (k=3)	35.69%	50.71 sec
Nearest Centroid	27.74%	0.52 sec
Densely Connected MLP	**53.84%**	59.2 sec Train 2.12 sec Test

3 Κατηγοριοποίηση με Convolutional Neural Network

Σε αυτό το στάδιο της εργασίας θα δούμε την επίδοση ενός συνελικτικού νευρωνικού δικτύου στο CIFAR-10 dataset, με στόχο την περαιτέρω βελτίωση της ακρίβειας του μοντέλου.

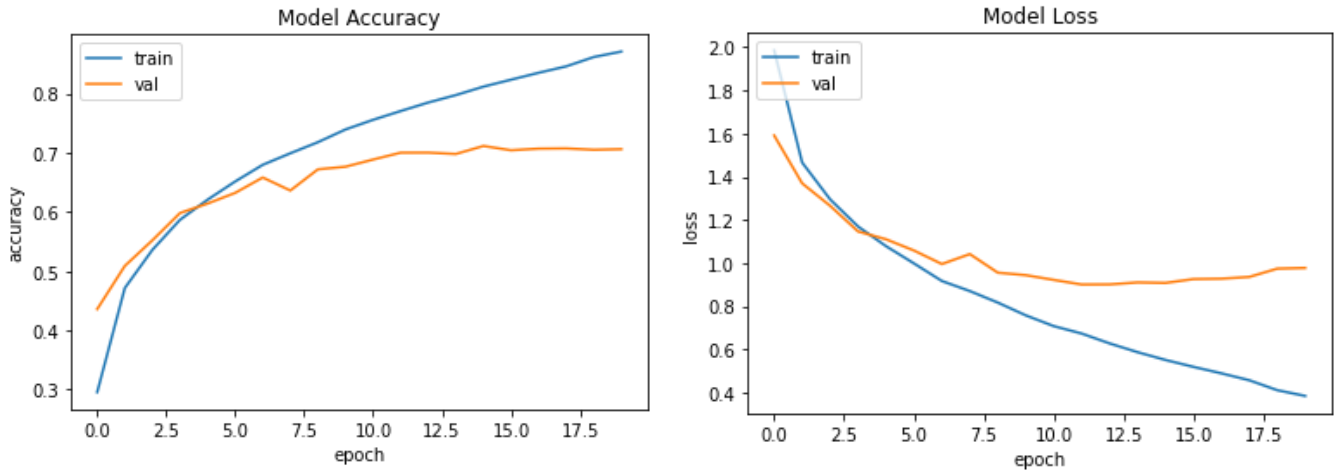
Αρχικά θα δοκιμαστεί μια σχετικά απλή δομή με 2 **VGG**[5] μπλοκ, όπου το κάθε μπλοκ αποτελείται από 2 συνελικτικά στρώματα και 1 MaxPooling, και ακολουθούν 1 fully-connected στρώμα με ReLU και 1 επίσης fully-connected στρώμα στην έξοδο

[5] *Very Deep Convolutional Networks for Large-Scale Image Recognition:* https://arxiv.org/abs/1409.1556

με συνάρτηση ενεργοποίησης softmax. Οι επιλογές αυτές γίνονται για τους εξής λόγους:

- Το **στοίβαγμα** πολλών μικρών CONV στρωμάτων έχουν καλύτερη επίδοση από CONV στρώματα με μεγαλύτερο μέγεθος φίλτρου. Αν για παράδειγμα επιλεγούν δύο στρώματα με μέγεθος φίλτρου (receptive field) **3x3** έναντι ενός με 7x7, οι μη γραμμικότητες που περιέχουν τα στοιβαγμένα θα αποδώσουν καλύτερα σε σχέση με την γραμμικότητα του ενός, διότι τα χαρακτηριστικά τους θα είναι πιο «εκφραστικά».
- Ως **padding** επιλέγεται το “same” ώστε να διατηρείται ίδιο το μέγεθος της εικόνας ανά συνελικτικό στρώμα και την δουλεία της μείωσης να την αναλαμβάνουν μόνο τα MaxPooling στρώματα.
- Είναι σύνηθες να εισάγετε περιοδικά ένα επίπεδο **MaxPooling** μεταξύ διαδοχικών επιπέδων CONV σε μια CNN αρχιτεκτονική. Η λειτουργία του είναι να μειώνει προοδευτικά το χωρικό μέγεθος της αναπαράστασης για να μειώσει το πλήθος των παραμέτρων και των υπολογισμών στο δίκτυο, και ως εκ τούτου να ελέγξει επίσης το overfitting. Το MaxPooling επίπεδο λειτουργεί ανεξάρτητα σε κάθε depth slice[6] της εισόδου και αλλάζει το μέγεθός του χωρικά, χρησιμοποιώντας τη λειτουργία MAX.
- Τα πλήρως συνδεδεμένα στρώματα στην έξοδο επιλέγονται ώστε να οδηγήσουν σε vector [1, 1, 10] για να γίνει η σύγκριση των label.

Τα πρώτα αποτελέσματα που προέκυψαν παρουσιάζονται παρακάτω:



Σχήμα 5 - Accuracy και loss για τον πρώτο συνδυασμό CNN.

Οι εποχές τέθηκαν ίσες με **20 επειδή σε προηγούμενη δοκιμή παρατηρήθηκε overfitting για μεγαλύτερο πλήθος.*

[6] Για είσοδο 32x32x3 και για συνελικτικό στρώμα με π.χ. 20 φίλτρα, η έξοδος θα προκύψει 32x32x20, όπου 20 το depth.

Πρώτος συνδυασμός CNN δικτύου			
Στρώμα Conv2D			
filters	**kernel_size**	**padding**	**activation**
32	3x3	same	ReLU
Στρώμα Conv2D			
filters	**kernel_size**	**padding**	**activation**
32	3x3	same	ReLU
Στρώμα MaxPooling2D (2, 2)			
Στρώμα Conv2D			
filters	**kernel_size**	**padding**	**activation**
64	3x3	same	ReLU
Στρώμα Conv2D			
filters	**kernel_size**	**padding**	**activation**
64	3x3	same	ReLU
Στρώμα MaxPooling2D (2, 2)			
Στρώμα Flatten			
Στρώμα Dense			
neurons		**activation**	
128		ReLU	
Στρώμα Dense			
neurons		**activation**	
10		softmax	
Αποτελέσματα			
Loss		0.9967	
Accuracy		69.58%	
F1		0.7184	
Training time		115.12 sec	

Πίνακας 7 - Πρώτη μορφή CNN.

Παρατηρούμε λοιπόν ότι η ακρίβεια αυξήθηκε σημαντικά σε σχέση με το dense MLP της προηγούμενης παραγράφου (**αύξηση 16%**). Αυτό συμβαίνει επειδή κάθε φίλτρο του CNN προσαρμόζεται και μαθαίνει συγκεκριμένο χαρακτηριστικό, το οποίο βοηθάει σημαντικά στην σωστή κατηγοριοποίηση.

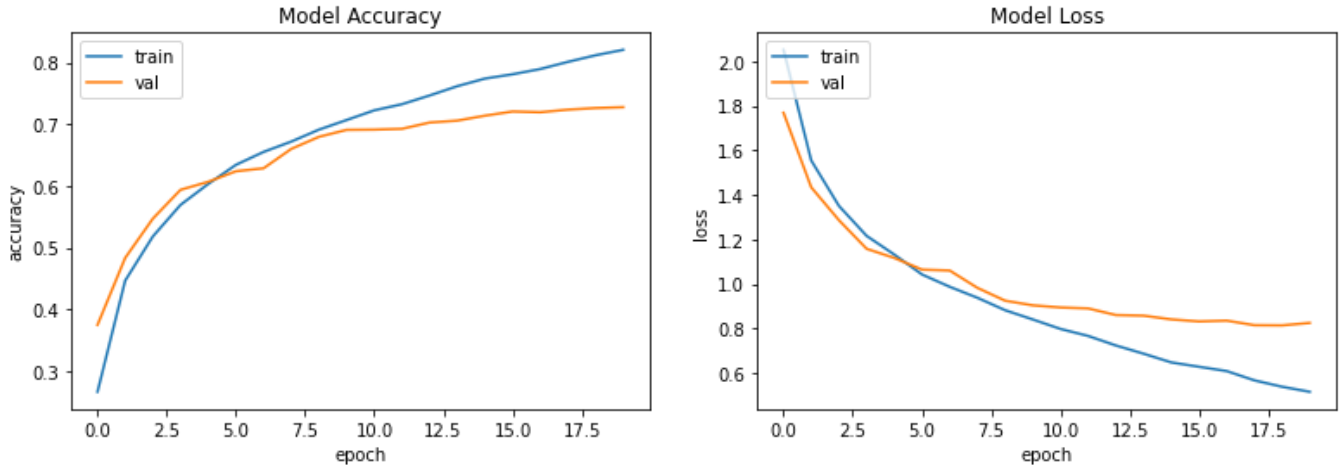
Αν τώρα προσθέσουμε και **τρίτο VGG μπλοκ** μετά από τα δύο ήδη υπάρχοντα, με στόχο την μάθηση περισσότερων features, το αποτέλεσμα είναι το εξής:

Αποτελέσματα	
Loss	1.1332
Accuracy	71.74%
F1	0.7184
Training time	155.59 sec

Πίνακας 8 - Αποτελέσματα για προσθήκη επιπλέον VGG μπλοκ.

Η ακρίβεια φαίνεται να αυξήθηκε ελάχιστα με την νέα προσθήκη, ενώ η μετρική f1 παρέμεινε ακριβώς ίδια, επομένως ως προς το πλήθος των φίλτρων φαίνεται να ήταν επαρκή τα προηγούμενα.

Όπως έγινε και στην περίπτωση του dense MLP, θα δοκιμάσουμε την προσθήκη **Dropout** στρώματος στην αρχική αρχιτεκτονική, αμέσως μετά τα 2 VGG μπλοκ και πριν το στρώμα Flatten, με πιθανότητα 0.3. Τα αποτελέσματα που προκύπτουν είναι αρκετά ικανοποιητικά:



Σχήμα 6 - Learning curves για 2 VGG blocks και χρήση Dropout(0.3).

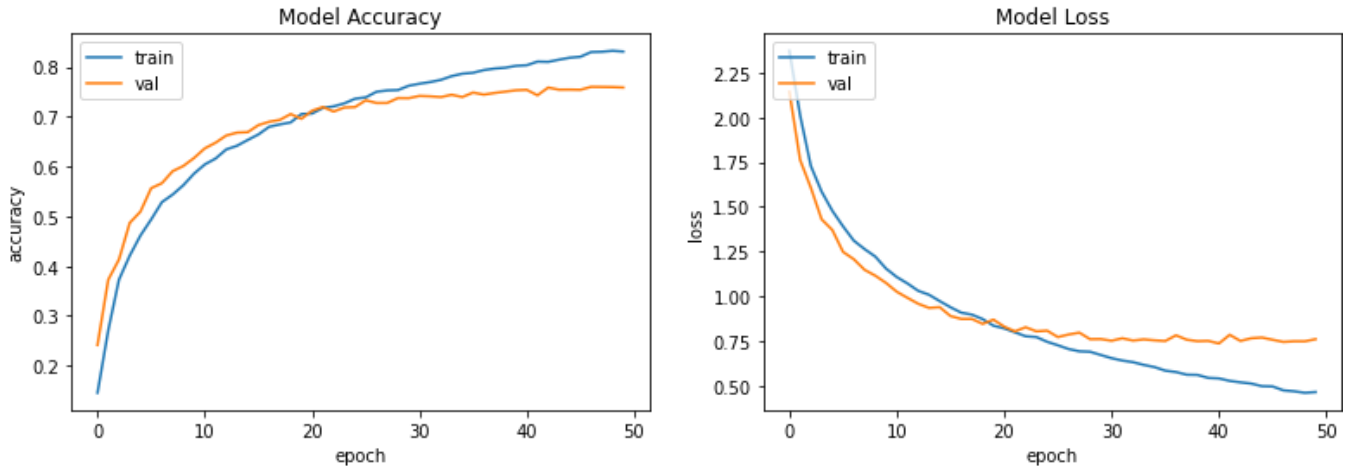
Αποτελέσματα	
Loss	0.8410
Accuracy	72.29%
F1	0.7207
Training time	143.24 sec

Πίνακας 9 - Αποτελέσματα για 2 VGG μπλοκ και Dropout(0.3)

Παρατηρείται, λοιπόν, αύξηση και στην ακρίβεια και στην μετρική F1. Παραμένει όμως ένα overfitting στα διαγράμματα, το οποίο καλό θα ήταν να εξαλειφθεί. Θα δοκιμάσουμε επομένως χρήση **Dropout μεταξύ κάθε VGG μπλοκ και πριν την έξοδο** (το training θα γίνει σε **50 epochs**).

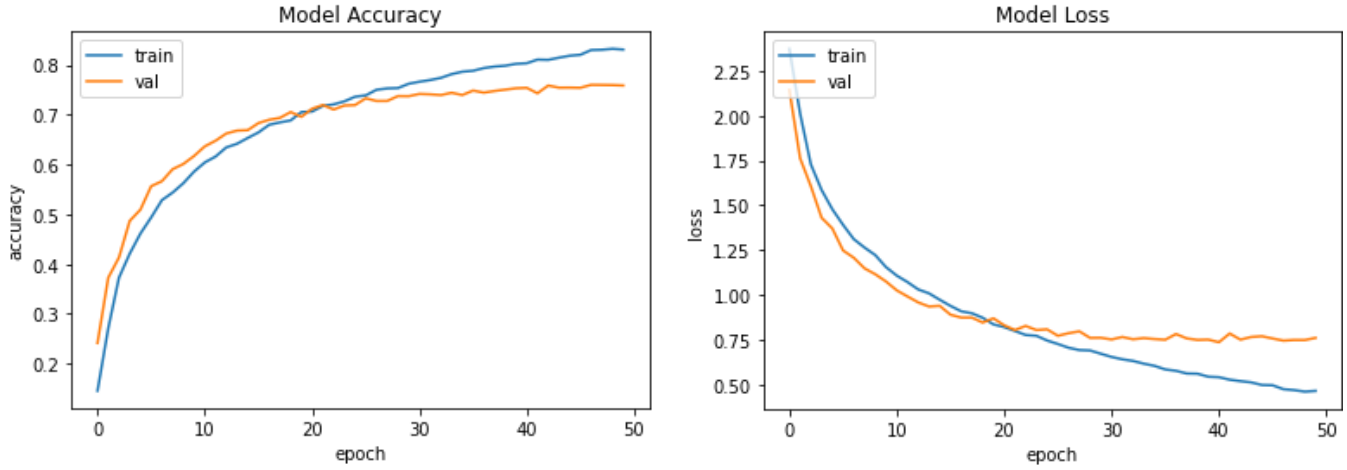
Αποτελέσματα	
Loss	0.7905
Accuracy	75.51%
F1	0.7566
Training time	323.49 sec

Πίνακας 10 - Αποτελέσματα με χρήση Dropout(0.2) μετά από κάθε VGG και πριν το στρώμα εξόδου.



Σχήμα 7 - Learning curves για Dropout(0.2) μετά από κάθε VGG μπλοκ και πριν στο στρώμα εξόδου.

Παρατηρούμε ότι η χρήση επιπλέον Dropout επιπέδων είχε ακόμα καλύτερη επίδοση, με f1 = 0.7566, και οι καμπύλες άρχισαν να συγκλίνουν περισσότερο. Θα δοκιμάσουμε τώρα ξανά **τρία μπλοκ VGG**, με Dropout ενδιάμεσα, για να δούμε αν το δίκτυο θα επωφεληθεί με τα περισσότερα φίλτρα και από την χρήση Dropout για εξάλειψη του overfitting.



Σχήμα 8 - Learning curves για 3 VGG μπλοκ με Dropout(0.2) μεταξύ κάθε μπλοκ και πριν την έξοδο.

Αποτελέσματα	
Loss	0.6449
Accuracy	79.64%
F1	0.7959
Training time	323.45 sec

Πίνακας 11 - Αποτελέσματα δομής 3 VGG με Dropout(0.2).

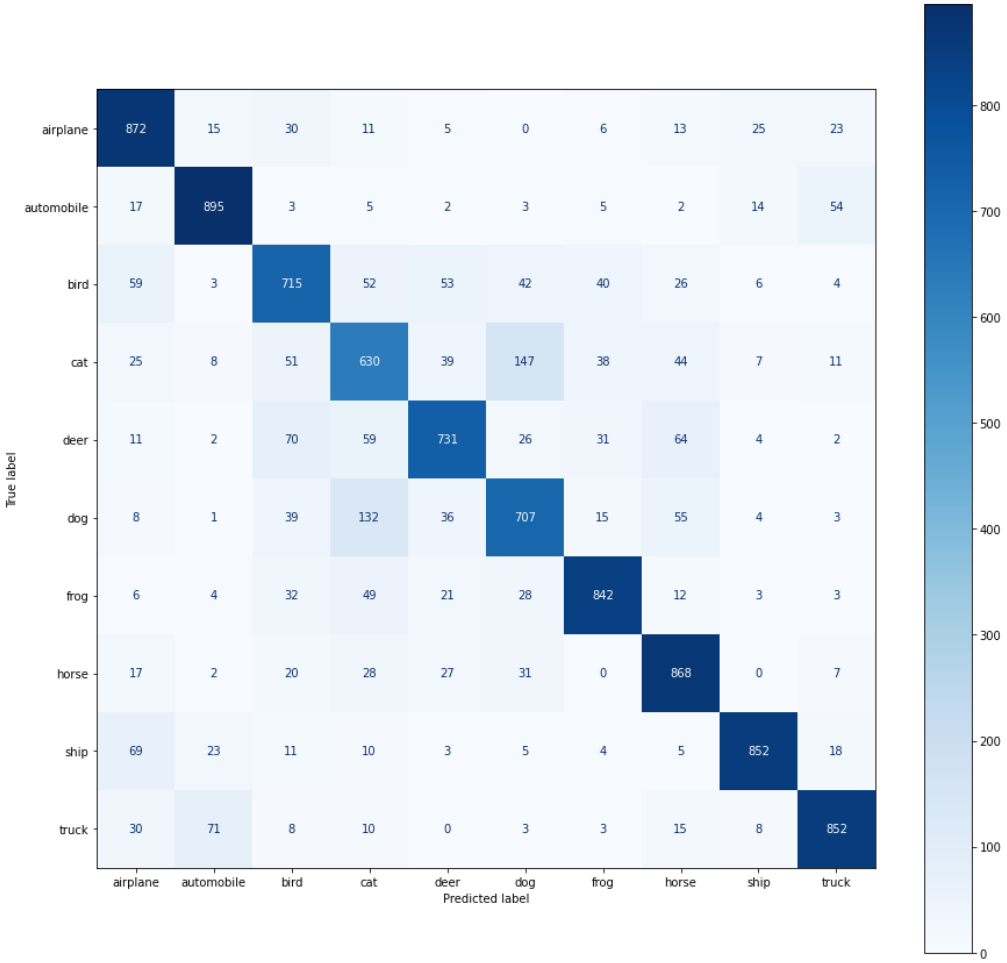
Όπως γίνεται αντιληπτό, η νέα δομή επέφερε πολύ καλά αποτελέσματα, με την ακρίβεια να αγγίζει το 80% και την μετρική f1 το 0.8.

Θα μπορούσαν να δοκιμαστούν μεγαλύτερες αρχιτεκτονικές δικτύων ή να γίνει περαιτέρω fine-tuning των μοντέλων, αλλά σαν διαδικασία θα ήταν πολύ χρονοβόρα, ειδικά κατά την διάρκεια της εκπαίδευσης, οπότε δεν θα συμπεριληφθούν. Επομένως, η **τελική μορφή του CNN μοντέλου** είναι:

Τελικός συνδυασμός CNN δικτύου			
Στρώμα Conv2D			
filters	**kernel_size**	**padding**	**activation**
32	3x3	same	ReLU
Στρώμα Conv2D			
filters	**kernel_size**	**padding**	**activation**
32	3x3	same	ReLU
Στρώμα MaxPooling2D (2, 2)			
Dropout(0.2)			
Στρώμα Conv2D			
filters	**kernel_size**	**padding**	**activation**
64	3x3	same	ReLU
Στρώμα Conv2D			
filters	**kernel_size**	**padding**	**activation**
64	3x3	same	ReLU
Στρώμα MaxPooling2D (2, 2)			
Dropout(0.2)			
Στρώμα Conv2D			
filters	**kernel_size**	**padding**	**activation**
128	3x3	same	ReLU
Στρώμα Conv2D			
filters	**kernel_size**	**padding**	**activation**
128	3x3	same	ReLU
Στρώμα MaxPooling2D (2, 2)			
Dropout(0.2)			
Στρώμα Flatten			
Στρώμα Dense			
neurons		**activation**	
128		ReLU	
Dropout(0.2)			
Στρώμα Dense			
neurons		**activation**	
10		softmax	

Πίνακας 12 - Τελική αρχιτεκτονική CNN δικτύου.

Τέλος, παρουσιάζονται ο πίνακας σύγχυσης και παραδείγματα εσφαλμένης/σωστής κατηγοριοποίησης.



Σχήμα 9 – Confusion matrix τελικού CNN μοντέλου.

Εικόνα 6 - Παραδείγματα σωστής κατηγοριοποίησης με CNN.

Εικόνα 7 - Παραδείγματα εσφαλμένης κατηγοριοποίησης με CNN.

4 Σύνοψη

Στην παρούσα εργασία έγινε προσπάθεια επίλυσης του προβλήματος κατηγοριοποίησης για την βάση δεδομένων CIFAR-10, χρησιμοποιώντας τις τεχνικές K-Nearest Neighbor, Nearest Centroid, Densely-Connected MLP και Convolutional Neural Network. Ο πίνακας 13 περιέχει συγκεντρωτικά όλα τα αποτελέσματα που προέκυψαν από την συγκεκριμένη μελέτη.

Σύνοψη αποτελεσμάτων		
Method	**Accuracy**	**Time elapsed**
Nearest Neighbor (k=3)	35.69%	50.71 sec
Nearest Centroid	27.74%	0.52 sec
Densely Connected MLP	53.84%	59.2 sec Train 2.12 sec Test
CNN	**79.64%**	323.45 sec Train 5.51 sec Test

Πίνακας 13 - Σύνοψη αποτελεσμάτων των τεχνικών που υλοποιήθηκαν.

Είναι προφανές ότι τη καλύτερη επίδοση είχε το συνελικτικό νευρωνικό δίκτυο με ακρίβεια ~80%, το οποίο ήταν αναμενόμενο για κατηγοριοποίηση φωτογραφιών πολλών κλάσεων. Το dense MLP όσο fine-tuning και να γινόταν δεν μπορούσε να ξεπεράσει το φράγμα της περίπου 55% ακρίβειας για το συγκεκριμένο σετ δεδομένων, το οποίο δείχνει και τα όριά του σε σετ με πολλές κλάσεις. Φυσικά, οι τεχνικές kNN και NC είναι ακατάλληλες για το συγκεκριμένο πρόβλημα καθώς το καλύτερο που μπορούσαν να 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 την απόφαση.

Στα μοντέλα dense MLP και CNN, καθοριστικό ρόλο στην αύξηση της επίδοσης και την μείωση του overfitting είχε η χρήση της τεχνικής Dropout, τεχνική δηλαδή που οδηγεί το δίκτυο σε ασθενέστερες ενώσεις (εξάρτηση) και συνεπώς σε μεγαλύτερη γενίκευση. Καθοριστική σημασία είχαν επίσης η κανονικοποίηση των δεδομένων και η χρήση decaying learning rate (μέσω του βελτιστοποιητή Adam).

Όλος ο κώδικας της εργασίας έτρεξε στο περιβάλλον του **Google Colab** και βρίσκεται σε μορφή python notebook. Το τελευταίο cell του notebook αφορά CNN υλοποίηση σε **Grayscale** εικόνες της CIFAR-10, το οποίο είχε επίδοση 75% και για λόγους συνοπτικότητας δεν εμφανίζεται στην παρούσα εργασία. Γενικότερα, θεωρείται κλασική τεχνική οι εικόνες να γίνονται πρώτα grayscale και μετά να επεξεργάζονται από συνελικτικά δίκτυα, ώστε να μειώνεται το μέγεθος των δεδομένων και η εκπαίδευση να γίνεται ταχύτερη. Εξάλλου, τα συνελικτικά δίκτυα ανιχνεύουν ακμές και σχήματα, οπότε ο χρωματισμός δεν έχει τόση σημασία.