Έκθεση Αποτελεσμάτων

Κατηγοριοποίηση Εικόνων CIFAR10 με συνδυασμό CNN και πλήρως συνδεδεμένου MLP

Ονοματεπώνυμο: Γεώργιος Πατιώς
ΑΕΜ: 4186

Περιεχόμενα

1 Εισαγωγή

- Στόχος της παρούσας εργασίας είναι η ανάπτυξη ενός πλήρως συνδεδεμένου νευρωνικού δικτύου για την κατηγοριοποίηση των εικόνων του CIFAR-10 . Θα γίνει σύγκριση των ποσοστών επιτυχίας του νευρωνικού με διαφορετικές τιμές παραμέτρων, αλλά και με άλλες μεθόδους κατηγοριοποίησης.

- Τα αποτελέσματα των προβλημάτων υπολογίστηκαν προγραμματιστικά στη γλώσσα προγραμματισμού Python και μπορούν να παραχθούν απο τα επισυναπτόμενα αρχεία πηγαίου κώδικα.

- Το CIFAR10 περιέχει 60.000 έγχρωμες εικόνες 32x32 pixels σε 10 κατηγορίες (50.000 στο training set και 10.000 στο test set). Κάθε κατηγορία περιέχει 6,000 εικόνες με τις κατηγορίες να περιλαμβάνουν αντικείμενα όπως αεροπλάνα, αυτοκίνητα, πουλιά, γάτες, ελάφια, σκύλους, βατράχους, άλογα, πλοία και φορτηγά.

- Το νευρωνικό δίκτυο χρησιμοποιεί πολλαπλά επίπεδα νευρώνων για την εκμάθηση χαρακτηριστικών από τα δεδομένα. Πιο συγκεκριμένα, η αρχιτεκτονική που χρησιμοποιήθηκε είναι συνδυασμός συνελικτικού και πλήρως συνδεδεμένου MLP δικτύου. Ακόμη, χρησιμοποιήθηκαν τεχνικές όπως dropout και batch normalization. Για σύγκριση χρησιμοποιήθηκαν οι μέθοδοι Nearest Neighbor και Nearest Class Centroid. Η μέθοδος Nearest Neighbor βασίζεται στην εύρεση της πιο κοντινής εικόνας στο σύνολο εκπαίδευσης με τον έλεγχο να γίνεται για 1 και 3 γείτονες. Επίσης, η μέθοδος Nearest Class Centroid χρησιμοποιεί τον μέσο όρο των χαρακτηριστικών κάθε κλάσης για την κατηγοριοποίηση.

2 Περιγραφή αλγορίθμου

2.1 Αρχιτεκτονική του Νευρωνικού Δικτύου

Περιγραφή της αρχιτεκτονικής:

- Το τελικό δίκτυο ειναι συνδυασμός συνελικτικού και πλήρως συνδεδεμένου δικτύου με δύο κρυφά επίπεδα CNN και MLP αντίστοιχα. Πιο συγκρικριμένα, δέχεται εικόνες 32x32 pixels με 3 κανάλια χρώματος/εισόδου. Αποτελείται από δύο CNN επίπεδα με 32 και 16 φίλτρα αντίστοιχα, με μέγεθος φίλτρου 3x3 , stride 1 και padding 1 . Ακολουθεί max pooling με μέγεθος παραθύρου 2x2 έπειτα από κάθε κρυφό συνελικτικό επίπεδο. Έπειτα, τα δεδομένα περνούν από δύο πλήρως συνδεδεμένα επίπεδα MLP με 256 και 128 νευρώνες αντίστοιχα. Στο τέλος, υπάρχουν 10 νευρώνες εξόδου για τις 10 κατηγορίες. Ο νευρώνας με τη μέγιση τιμή αντιστοιχεί στην πρόβλεψη του δικτύου.

- Μετά από κάθε κρυφό επίπεδο χρησιμοποιήθηκε η συνάρτηση ενεργοποίησης ReLU .

- Έγινε χρήση της τεχνικής dropout στα 3 από τα 4 κρυφά επίπεδα για την αποφυγή υπερεκπαίδευσης.

- Χρησιμοποιήθηκε η τεχνική batch normalization μετά από κάθε κρυφό επίπεδο.

2.2 Παράμετροι Εκπαίδευσης

Δοκιμάστηκαν διάφορες τιμές για τις παραμέτρους εκπαίδευσης του μοντέλου. Η σύγκριση των αποτελεσμάτων έγινε με βάση το ποσοστό επιτυχίας στο validation set σε συνδιασμό με τον χρόνο εκπαίδευσης και θα αναλυθεί παρακάτω. Στο τελικό μοντέλο χρησιμοποιήθηκαν οι παρακάτω τιμές:

- Learning rate = 0.001
- Batch size = 32

Ακόμη, χρησιμοποιήθηκε ο optimizer Adam και ως loss function το cross entropy . Σημειώνεται, επίσης, ότι χρησιμοποιήθηκε scheduler , που ενεργοποιείται μειώνοντας το learning rate στο μισό κάθε φορά που η ακρίβεια στο validation set δεν εμφανίζει βελτίωση για πολλαπλές εποχές στη σειρά.

2.3 Διαδικασία Εκπαίδευσης

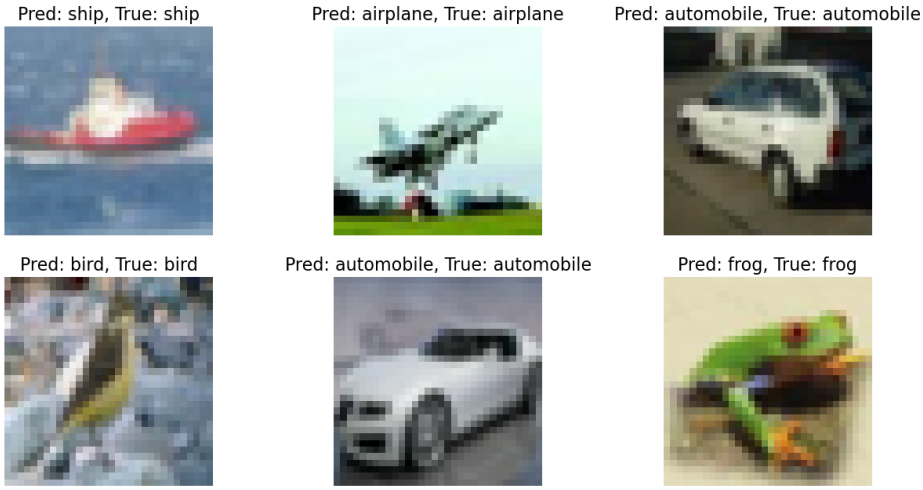
Αρχικά, τα δεδομένα κανονικοποιούνται όταν ακόμα είναι στη μορφή flat διανύσματος με 3072 διαστάσεις. Στη συνέχεια μετασχηματίζονται σε μορφή εικόνας 3x32x32 και χωρίζεται το training set σε training και validation set πριν γίνει εκπαίδευση του μοντέλου (80% training, 20% validation). Το βασικό training loop έπειτα αποτελείται από τα εξής βήματα:

1. Για κάθε εποχή εκπαίδευσης, τα δεδομένα του training set διατίθενται στο μοντέλο σε mini-batches μεγέθους 32.
2. Κάθε mini-batch αξιολογείται από το μοντέλο και υπολογίζονται οι προβλέψεις και οι αντίστοιχες απώλειες.
3. Υπολογίζεται η παράγωγος της συνάρτησης απώλειας ως προς τις παραμέτρους του μοντέλου.
4. Οι παράμετροι του μοντέλου ενημερώνονται με τον optimizer με βάση την παράγωγο.
5. Σε κάθε εποχή, τα δεδομένα του validation set χρησιμοποιούνται για να αξιολογηθεί η απόδοση του μοντέλου.

3 Παραδείγματα Κατηγοριοποίησης

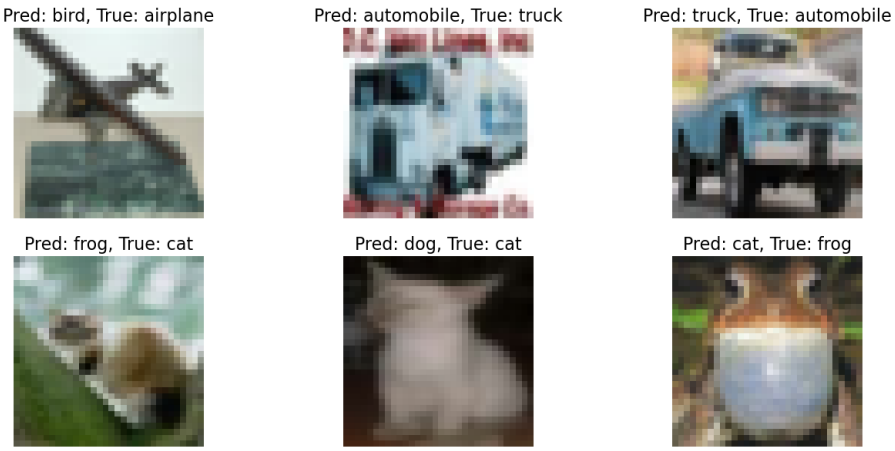
Παρακάτω απεικοντίζονται μερικά παραδείγματα εικόνων με σωστή και λανθασμένη πρόβλεψη. Οι εικόνες επιλέγονται τυχαία στο αρχείο **find_classification_examples.py** από τις σωστές και λάθος προβλέψεις του μοντέλου που εκπαιδεύτηκε και αποθηκεύτηκε στη μνήμη (αρχείο **cifar10_main_model.pth**) . Η τυχαία επιλογή ωστόσο επιστρέφει σταθερά τα ίδια αποτελέσματα, καθώς χρησιμοποιείται το ίδιο random seed . Ακόμη, η κάθε εικόνα έχει μια ετικέτα στο πάνω μέρος με τη μορφή predicted_label : [class] true_label : [class].

3.1 Ορθή Κατηγοριοποίηση



Σχήμα 1: Παραδείγματα σωστής κατηγοριοποίησης.

3.2 Εσφαλμένη Κατηγοριοποίηση



Σχήμα 2: Παραδείγματα εσφαλμένης κατηγοριοποίησης.

Αξίζει να σημειωθεί ότι οι λάθος προβλέψεις που απεικοντίζονται παρόλο που δεν έχουν τη σωστή ετικέτα, είναι αρκετά κοντά στην πραγματική κατηγορία της εικόνας. Αυτό παρατηρήθηκε και σε διαφορετικές τυχαίες επιλογές με διαφορετικό random seed . Δηλαδή, το αεροπλάνο που κατηγοριοποιήθηκε ως πουλί, τα φορτηγά που κατηγοριοποιήθηκαν ως

αυτοκίνητα και τα ζώα που κατηγοριοποιήθηκαν ως άλλα ζώα και όχι ως κάτι εκτός ζωικού βασιλείου είναι κάτι ενθαρρυντικό για τη σωστή λειτουργία του μοντέλου παρά την εσφαλμένη κατηγοριοποίηση.

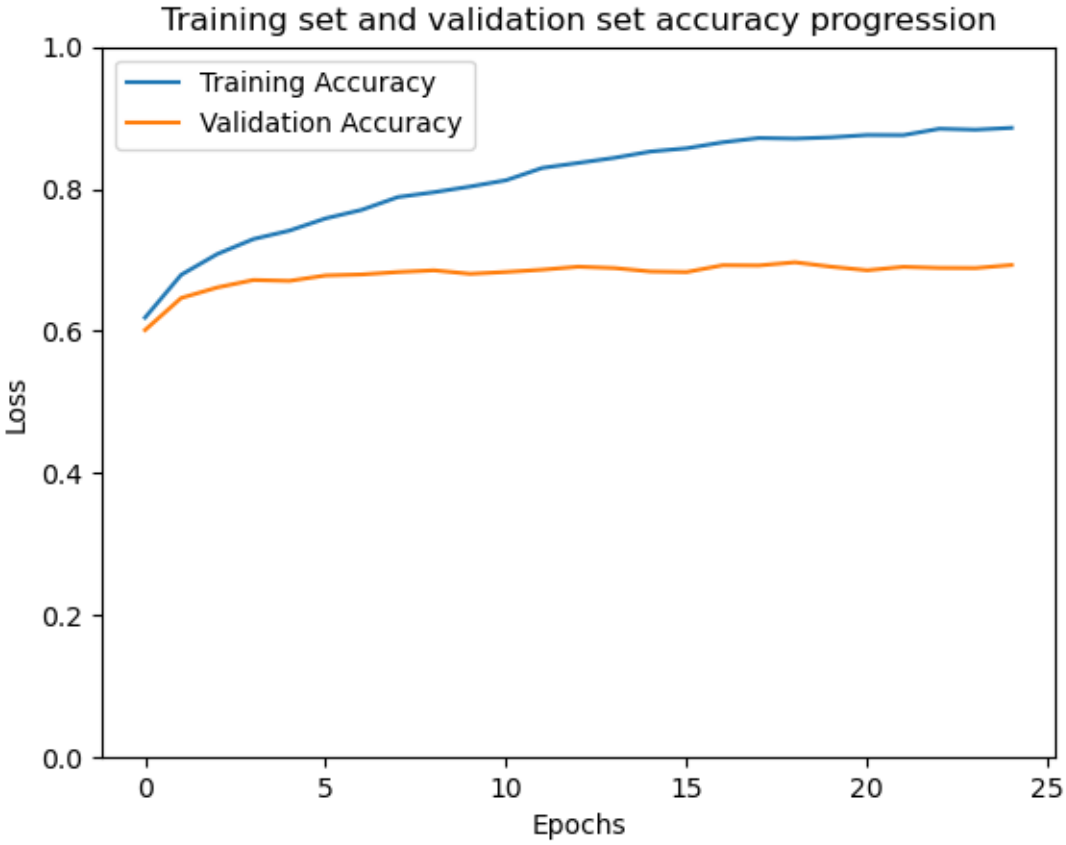
4 Αποτελέσματα και Συγκρίσεις

4.1 Επίδοση Νευρωνικού Δικτύου

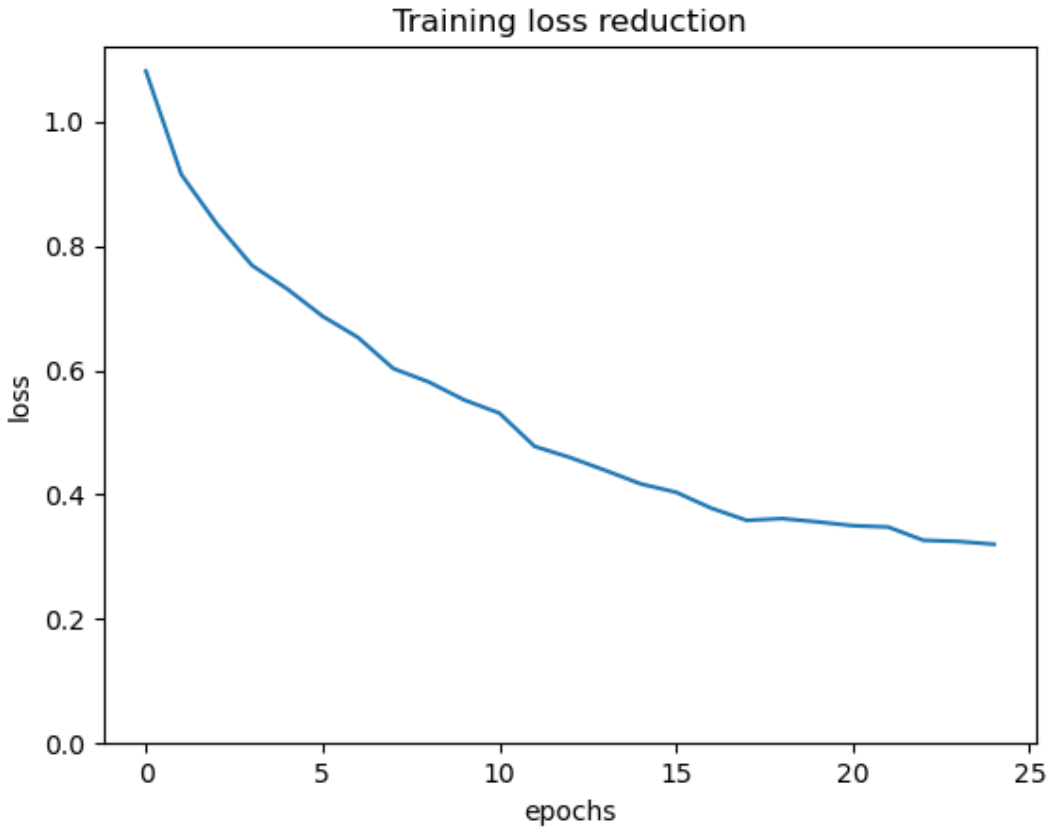
Το βασικό μοντέλο που περιγράφηκε παραπάνω εκπαιδεύτηκε για 25 εποχές και αποθηκεύτηκε στο αρχείο **cifar10_main_model.pth** για μελλοντική χρήση. Παρακάτω φαίνονται η πορεία της ακρίβειας στο training set και στο validation set (Σχήμα 3), η μείωση του loss (Σχήμα 4) και του learning rate (Σχήμα 5) κατά τη διάρκεια της εκπαίδευσης του νευρωνικού δικτύου στα δεδομένα του CIFAR10 .

Αξίζουν να σημειωθούν τα εξής:

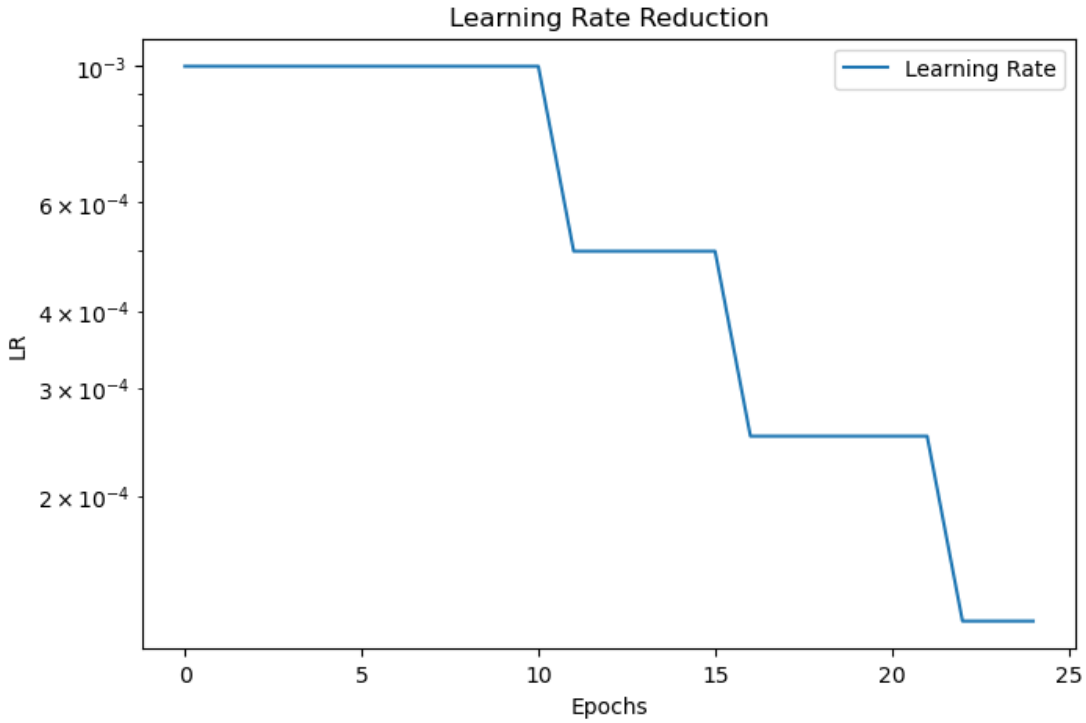
- Η ακρίβεια στο training set πλησιάζει το 90% και στο validation set το 70%.
- Η καμπύλη της ακρίβειας στο validation set δεν εμφανίζει σημαντική μείωση γεγονός που θα υποδήλωνε ότι το μοντέλο δεν πάσχει απο υπερεκπαίδευση. Αυτό μπορεί να αποδοθεί στη χρήση της τεχνικής dropout στα 3 απο τα 4 κρυφά επίπεδα.
- Το training loss μειώνεται όλο και πιο αργά, καθώς γίνεται η σύγκλιση του μοντέλου.
- Το learning rate μειώνεται κατά τη διάρκεια της εκπαίδευσης, αφού χρησιμοποιήθηκε scheduler , για να μπορέσει να συγκλίνει με μεγαλύτερη ακρίβεια στον τοπικό βελτιστοποιητή που θα βρει όταν η ακρίβεια φαίνεται να σταθεροποιείται.



Σχήμα 3: Πορεία της ακρίβειας στο training και validation set.



Σχήμα 4: Μείωση του loss στο training set κατά τη διάρκεια της εκπαίδευσης.



Σχήμα 5: Μείωση του learning rate κατά τη διάρκεια της εκπαίδευσης.

Ακόμη στον παρακάτω πίνακα φαίνεται αναλυτικά η επίδοση του μοντέλου στην πρόβλεψη κάθε κλάσης στο test set :

Class	Precision	Recall	F1-Score	Support
airplane	0.71	0.70	0.70	1000
automobile	0.79	0.81	0.80	1000
bird	0.61	0.56	0.58	1000
cat	0.49	0.49	0.49	1000
deer	0.62	0.65	0.63	1000
dog	0.56	0.60	0.58	1000
frog	0.75	0.77	0.76	1000
horse	0.75	0.73	0.74	1000
ship	0.79	0.79	0.79	1000
truck	0.79	0.75	0.77	1000
Accuracy			**0.69**	10000
Macro avg	0.69	0.69	**0.69**	10000
Weighted avg	0.69	0.69	0.69	10000

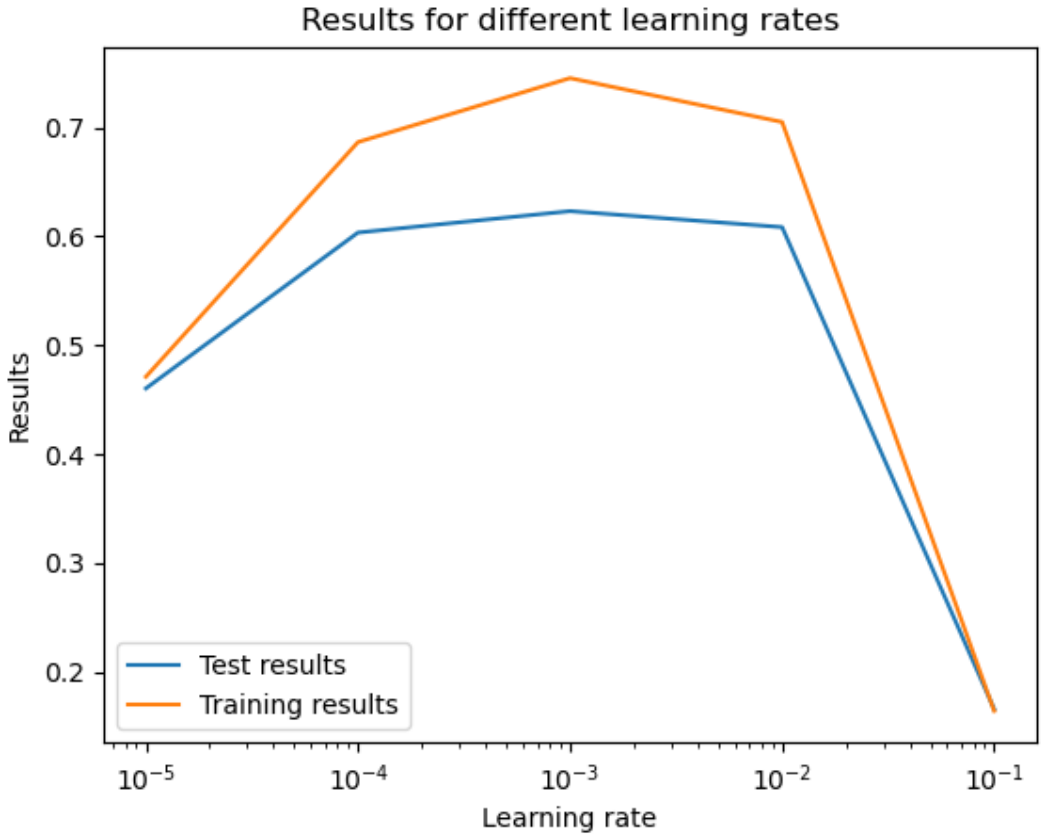
Πίνακας 1: Αναλυτική επίδοση του μοντέλου στην πρόβλεψη κάθε κλάσης στο test set .

4.2 Επίδοση Με Αλλαγή Παραμέτρων

Στην βελτίωση του μοντέλου δοκιμάστηκαν οι παρακάτω αλλαγές. Σημειώνεται ότι οι συγκρίσεις έγιναν για τον ίδιο αριθμό εποχών και στα γραφήματα που στον y άξονα υπάρχει ο

χρόνος εκπαίδευσης, η τιμή του αντιστοιχεί σε δευτερόλεπτα.

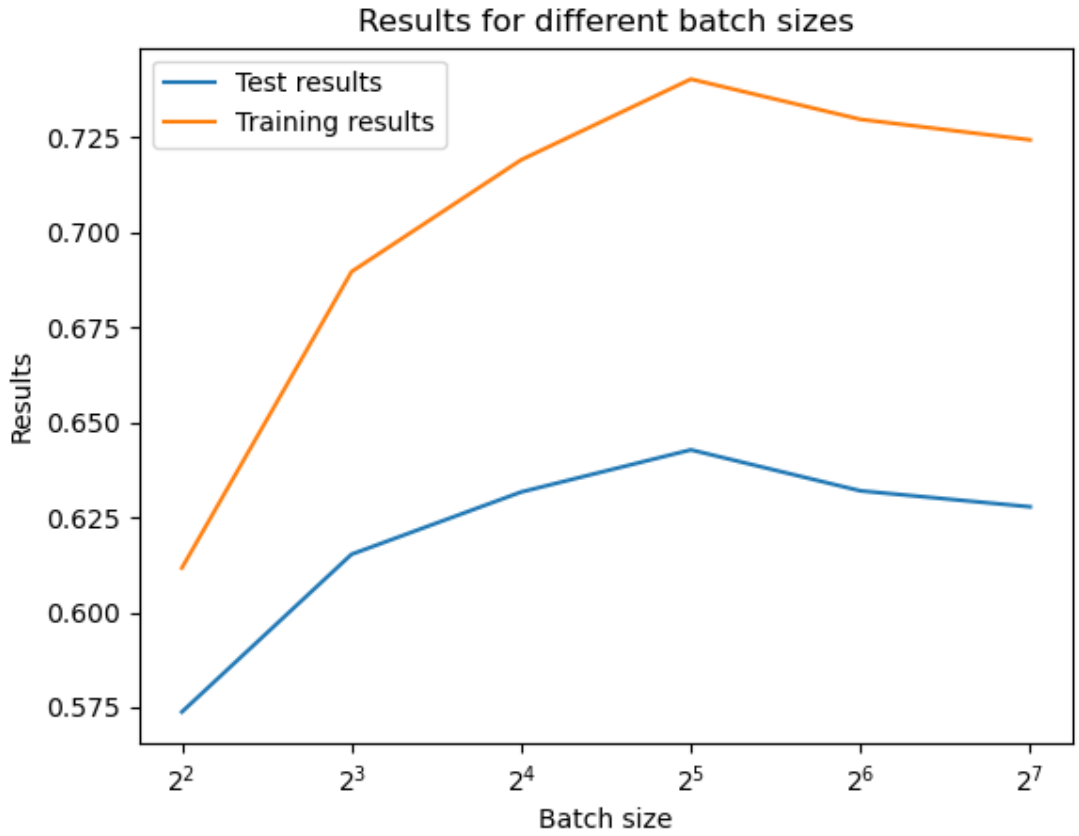
- Αυξομείωση του learning rate .



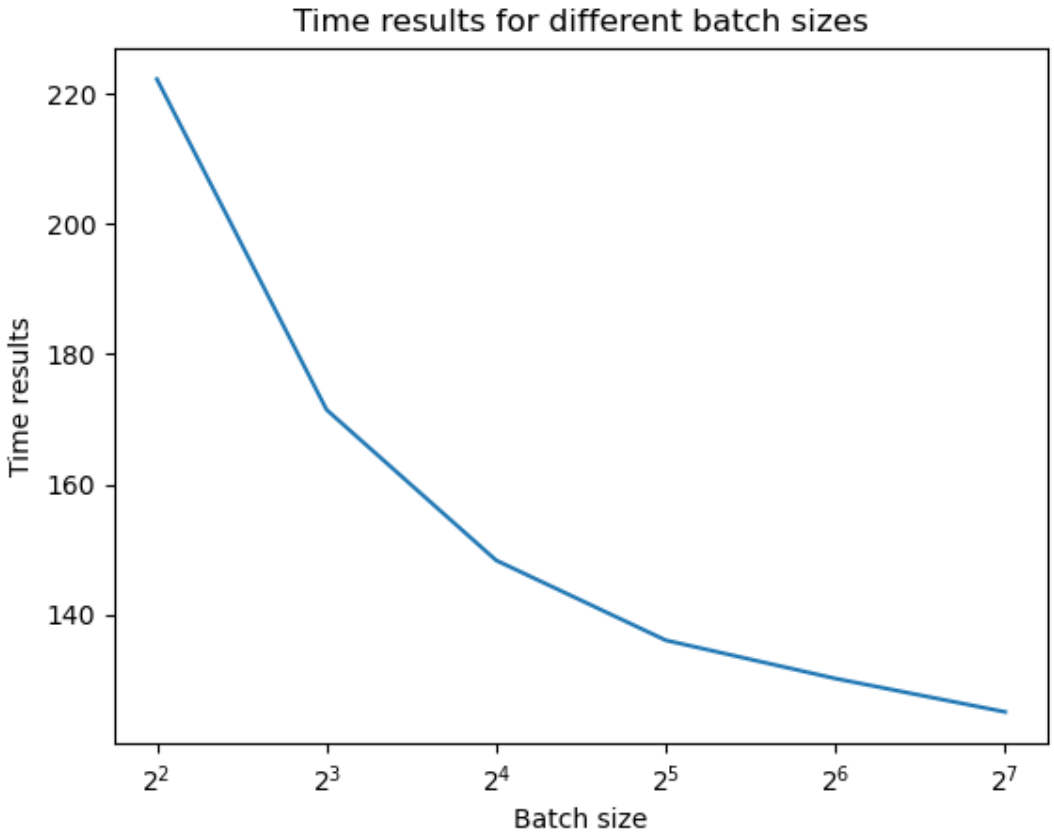
Σχήμα 6: Ακρίβεια στο training και στο validation set για διάφορες τιμές του learning rate.

Παρατηρήθηκε καλύτερη και γρηγορότερη σύγκλιση για το learning rate $= 0.001$. Για αυτό αρχικοποιήθηκε στο τελικό μοντέλο ο optimizer με αυτή την τιμή και ύστερα έγινε χρήση scheduler .

- Αυξομείωση του batch size .



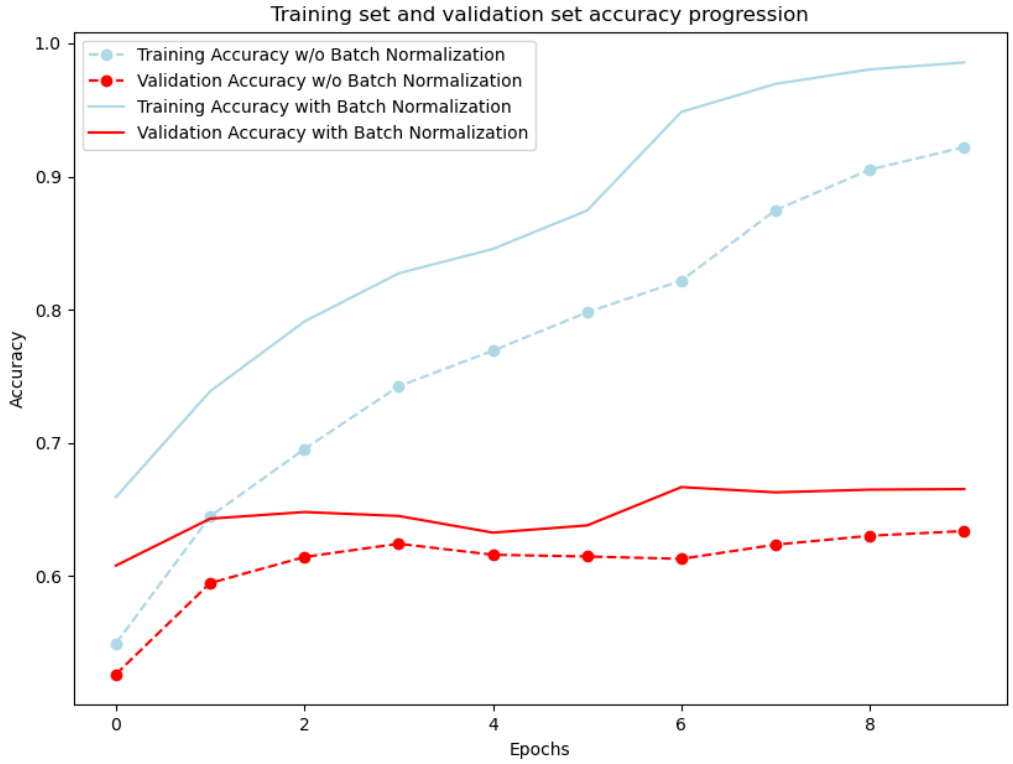
Σχήμα 7: Ακρίβεια στο training και στο validation set για διάφορα batch sizes.



Σχήμα 8: Χρόνος εκπαίδευσης μοντέλου για διάφορα batch sizes.

Η καλύτερη σύγκλιση για σταθερό αριθμό εποχών έγινε για 32 δείγματα ανά batch (Σχήμα 7). Ακόμη είναι προφανές οτι η αύξηση του batch size μειώνει τον χρόνο εκπαίδευσης του μοντέλου (Σχήμα 8).

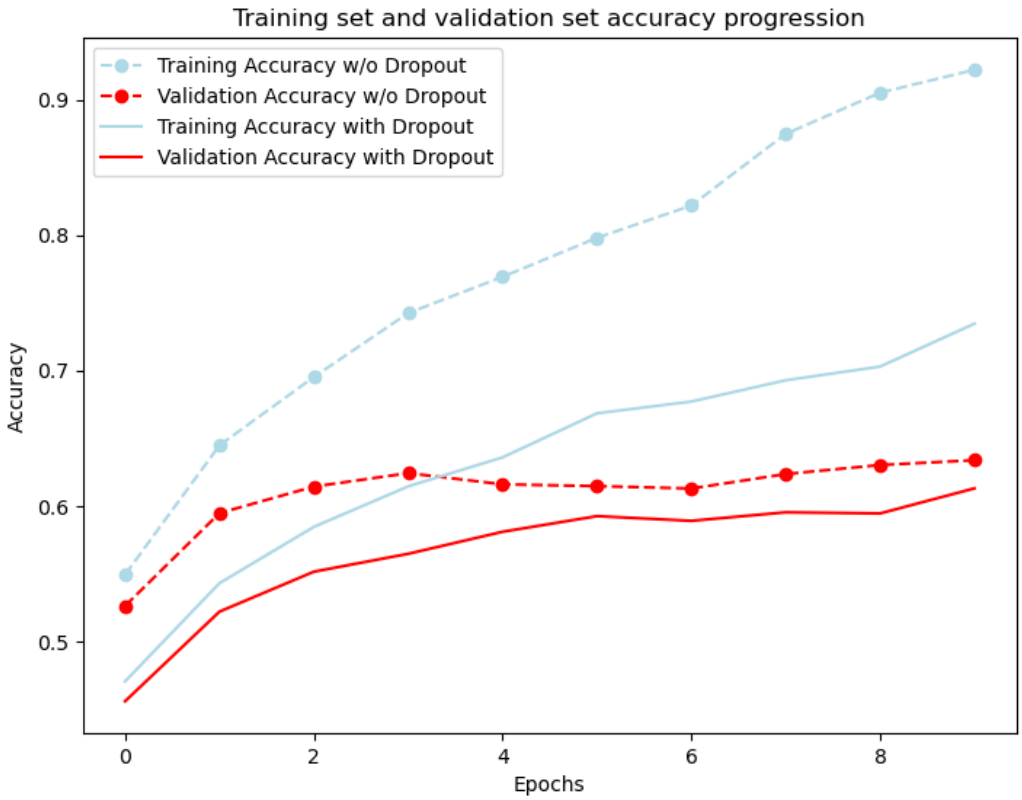
- Χρήση της τεχνικής batch normalization .



Σχήμα 9: Ακρίβεια στο training και στο validation set με και χωρίς batch normalization.

Η χρήση της τεχνικής batch normalization είχε σημαντική και εμφανή επίδραση στην αύξηση της απόδοσης του μοντέλου σε όλη τη διάρκεια της εκπαίδευσης (Σχήμα 9).

- Χρήση της τεχνικής dropout .

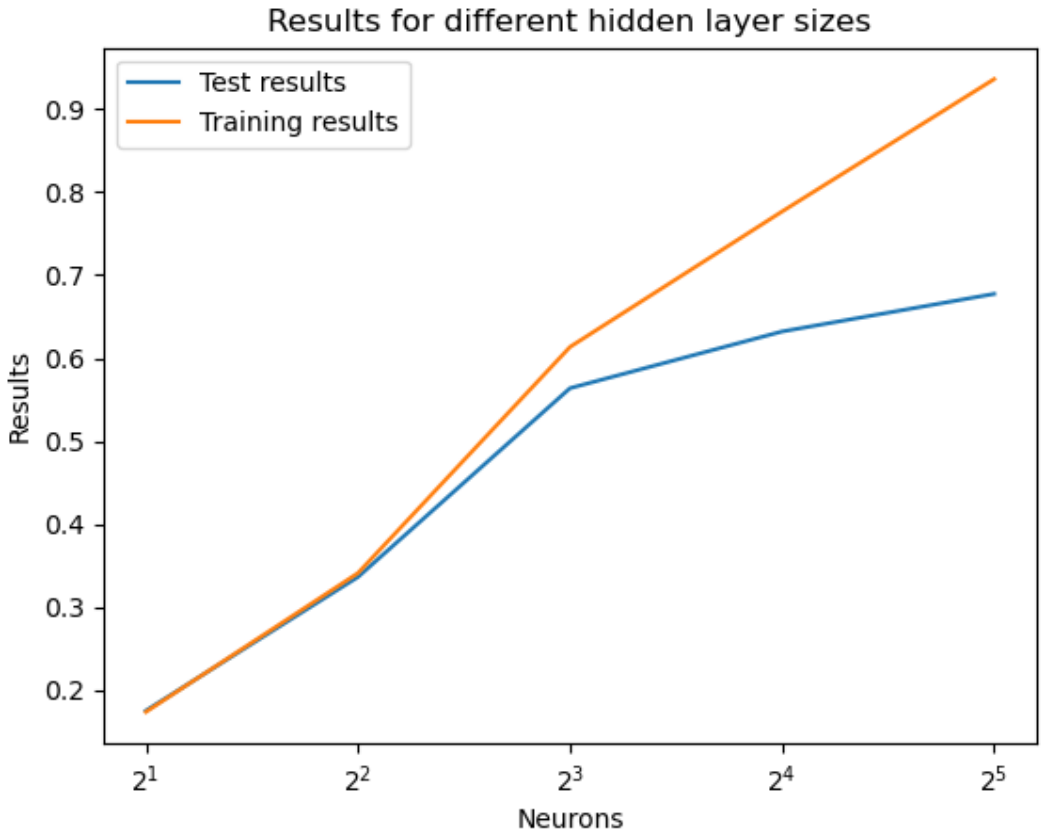


Σχήμα 10: Ακρίβεια στο training και στο validation set με και χωρίς dropout.

Η χρήση της τεχνικής dropout παρουσίασε ενδιαφέρον, διότι παρόλο που η ακρίβεια στο validation set συνέκλινε στα ίδια επίπεδα με την περίπτωση χωρίς dropout , η ακρίβεια στο training set ήταν σημαντικά χαμηλότερη. Αυτό υποδηλώνει ότι το μοντέλο αποφεύγει την υπερεκπαίδευση και είναι πιο γενικευμένο.

- Αυξομείωση πλήθους νευρώνων στα κρυφά επίπεδα. Για τη συστηματική δοκιμή διαφορετικού πλήθους νευρώνων στα κρυφά επίπεδα ορίστηκε n ίσο με το πλήθος των φίλτρων στο δεύτερο κρυφό επίπεδο (CNN layer). Τα υπόλοιπα κρυφά επίπεδα είχαν:
  - Πρώτο κρυφό επίπεδο (CNN): $2n$ φίλτρα
  - Τρίτο κρυφό επίπεδο (MLP): n^2 νευρώνες
  - Τέταρτο κρυφό επίπεδο (MLP): $n^2/2$ νευρώνες

  Αυτό το n είναι ο άξονας x των γραφημάτων που ακολουθούν.



Σχήμα 11: Ακρίβεια στο training και στο validation set για διάφορο πλήθος νευρώνων στα κρυφά επίπεδα.

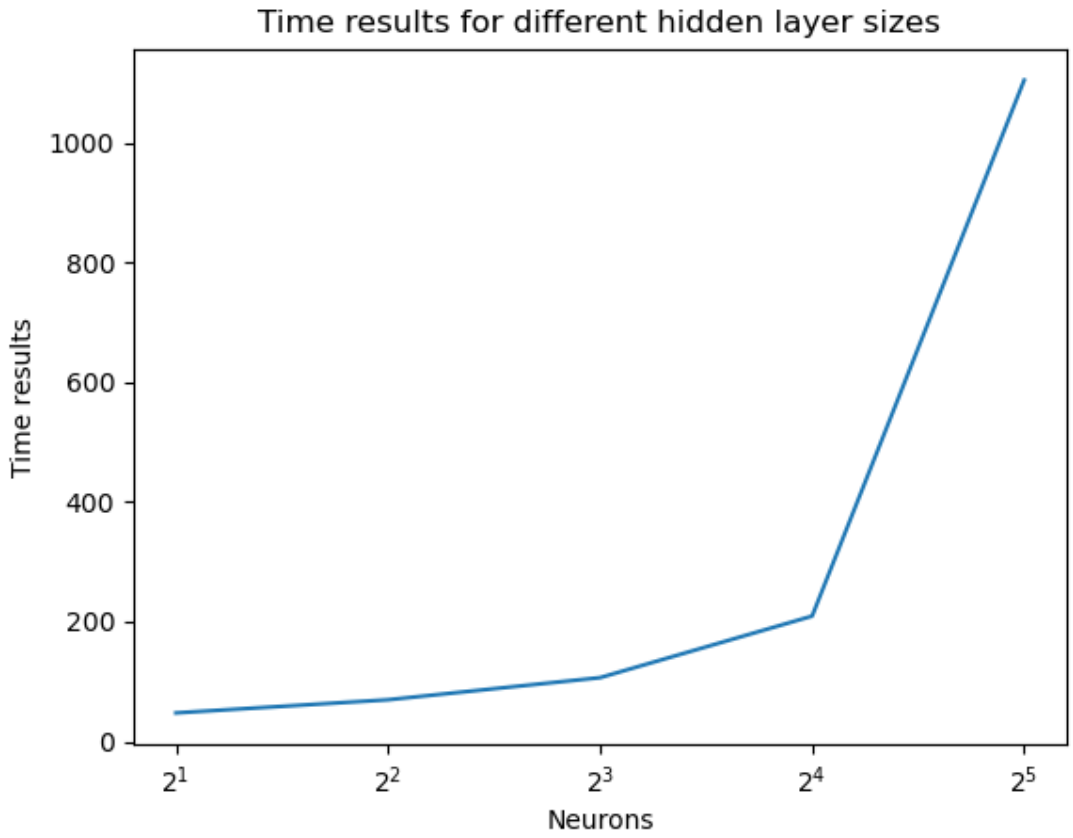
Αξίζει να σημειωθεί ότι η αύξηση του πλήθους των νευρώνων στα κρυφά επίπεδα οδηγεί σε αύξηση της ακρίβειας στο validation set , αλλά και σε σημανιτκή αύξηση του χρόνου εκπαίδευσης του μοντέλου. Μάλιστα, ο ρυθμός αύξησης της ακρίβειας δεν φαίνεται να μειώνεται, καθώς αυξάνεται το n που σημαίνει ότι τα πραγματικά όρια της συγκεκριμένης αρχιτεκτονικής είναι αρκετά πιο υψηλά. Ωστόσο, στο πλαίσιο της εργασίας επιλέχθηκε το $n = 16$ για ευχρηστία και αποδοτικότητα.

4.3 Σύγκριση Αποτελεσμάτων με Άλλες Μεθόδους

Μέθοδος	F1-score (test set)	Χρόνος Εκτέλεσης
Νευρωνικό Δίκτυο	0.69	447 sec
Nearest Neighbor (n=1)	0.35	21.9 sec
Nearest Neighbor (n=3)	0.32	22.1 sec
Nearest Class Centroid	0.25	0.7 sec

Πίνακας 2: Σύγκριση αποτελεσμάτων μεταξύ μεθόδων.

Από τα παραπάνω αποτελέσματα παρατηρούμε ότι οι επιπρόσθετοι αλγόριθμοι που χρησιμοποιήθηκαν έχουν μέσο F1-score που κυμαίνεται από 0.25 έως 0.35. Συγκεκριμένα, παρατηρείται ότι το F1-score του αλγορίθμου KNN με 1 γείτονα είναι υψηλότερο από ό,τι με



Σχήμα 12: Χρόνος εκπαίδευσης μοντέλου για διάφορο πλήθος νευρώνων στα κρυφά επίπεδα.

3 γείτονες γεγονός που καταδεικνύει τη σχετική τυχαιότητα των αποτελεσμάτων του αλγορίθμου, καθώς θεωρητικά οι 3 πλησιέστεροι γείτονες θα έπρεπε να δίνουν περισσότερη πληροφορία σχετικά με την κλάση της εκάστοτε εικόνας. Συμπερασματικά, η χρήση ενός νευρωνικού δικτύου (F1-score $\bar{0}.69$) καταφέρνει να ξεχωρίσει τις εικόνες μεταξύ τους αξιοποιώντας καλύτερα την πληροφορία που περιέχεται στις εικόνες του CIFAR10 , καθώς στο πλαίσιο αυτής της εργασίας δεν έγινε χρήση feature engineering . Ωστόσο, η εκπαίδευση ακόμα και σχετικά μικρού μεγέθους νευρωνικού δικτύου για 10 εποχές απαιτεί πολύ περισσότερο χρόνο και υπολογιστική ισχύ σε σχέση με τους επιπρόσθετους αλγορίθμους που χρησιμοποιήθηκαν.

5 Κώδικας

Για την εκτέλεση του κώδικα που υλοποιήθηκε στην παρούσα εργασία απαιτείται η χρήση των παρακάτω βιβλιοθηκών που δεν συμπεριλαμβάνονται στην εγκατάσταση της Python από προεπιλογή:

- **numpy** για αριθμητικές πράξεις – https://www.sympy.org
- **scikit-learn** για τη χρήση έτοιμων υλοποιημένων κατηγοριοποιητών – https://scikit-learn.org/stable/
- **pytorch** για την υλοποίηση του νευρωνικού δικτύου – https://pytorch.org

Για την παραγωγή των αποτελεσμάτων που σχολιάστηκαν στην εργασία αυτή, απαιτείται η εκτέλεση των αρχείων κώδικα που παρατίθονται παρακάτω:

- **knn.py** : Υλοποίηση του αλγορίθμου Nearest Neighbor με 1 και 3 γείτονες και εμφάνιση αποτελεσμάτων.
- **nearest_centroid.py** : Υλοποίηση του αλγορίθμου Nearest Class Centroid και εμφάνιση αποτελεσμάτων.
- **main_model.py** : Υλοποίηση του νευρωνικού δικτύου και εκπαίδευση στα δεδομένα του CIFAR10 , εμφάνιση αποτελεσμάτων σε γραφήματα και πίνακες και αποθήκευση του μοντέλου στον δίσκο.
- **read_data.py** : Φόρτωση των δεδομένων του CIFAR10 από τον δίσκο. Σημειώνεται ότι τα δεδομένα απο την ιστοσελίδα του CIFAR-10 αποθηκεύτηκαν στον φάκελο **cifar-10-batches-py** και ύστερα τοποθετήθηκαν σε φάκελο με όνομα **data** . Διαφορετική ιεραρχία φακέλων απαιτεί την αλλαγή των αντίστοιχων μεταβλητών στον κώδικα του αρχείου read_data.py .
- **preprocess_data.py** : Προεπεξεργασία των δεδομένων του CIFAR10 . Γίνονται μετασχηματισμοί, κανονικοποίηση και διαχωρισμός του training dataset σε training, validation set.
- **normalize_data.py** : Κανονικοποίηση των δεδομένων του CIFAR10 με χρήση του scaler της scikit-learn.
- **plot_different_params.py** : Εκτέλεση του νευρωνικού δικτύου με διαφορετικές παραμέτρους και εμφάνιση των αποτελεσμάτων σε γραφήματα.

- **find_classification_examples.py** : Εύρεση και εμφάνιση παραδειγμάτων εικόνων ταξινόμησης που έχουν ταξινομηθεί σωστά ή λανθασμένα από το μοντέλο που έχει αποθηκευτεί στον δίσκο.

- **accuracy_metrics.py** : Μέθοδοι για τον υπολογισμό των μετρικών ακρίβειας, ανάκλησης και F1-score .

Παρακάτω παρατίθεται το τελικό μοντέλο που χρησιμοποιήθηκε για την ταξινόμηση των εικόνων του CIFAR10 :

```
nn.Sequential(
    nn.Conv2d(input_channels, 32, kernel_size=3, padding=1),
    nn.BatchNorm2d(32),
    nn.ReLU(),
    nn.MaxPool2d(kernel_size=2, stride=2),
    nn.Dropout(0.15),

    nn.Conv2d(32, 16, kernel_size=3, padding=1),
    nn.BatchNorm2d(16),
    nn.ReLU(),
    nn.MaxPool2d(kernel_size=2, stride=2),  # 16 x 8 x 8
    nn.Flatten(),

    nn.Linear(16 * 8 * 8, 256),
    nn.BatchNorm1d(256),
    nn.ReLU(),
    nn.Dropout(0.3),

    nn.Linear(256, 128),
    nn.BatchNorm1d(128),
    nn.ReLU(),
    nn.Dropout(0.3),

    nn.Linear(128, output_size),
)
```