

# 1 Η ΥΠΟΧΡΕΩΤΙΚΗ ΕΡΓΑΣΙΑ ΣΤΟ ΜΑΘΗΜΑ «ΝΕΥΡΩΝΙΚΑ ΔΙΚΤΥΑ – ΒΑΘΙΑ ΜΑΘΗΣΗ»

Η παρούσα εργασία ασχολείται με την υλοποίηση ενός Συνελικτικού Νευρωνικού Δικτύου (Convolutional Neural Network - CNN), με στόχο την ταξινόμηση εικόνων του dataset [CIFAR-10](#). Το CIFAR-10 είναι ένα σύνολο δεδομένων που περιλαμβάνει 60.000 RGB εικόνες μεγέθους 32x32 pixel, κατανεμημένες σε 10 κατηγορίες.

Η παρούσα μελέτη αναπτύσσει ένα CNN εξ ολοκλήρου με ιδιόγραφη κωδικοποίηση. Θα παρουσιαστούν οι βοηθητικές συναρτήσεις και μέθοδοι που υλοποιήθηκαν για τη σωστή λειτουργία του δικτύου.

## Βοηθητικές Συναρτήσεις

### 1. `get_im2col_indices`

- Η συνάρτηση αυτή δημιουργεί τους δείκτες (indices) που απαιτούνται για την αναδιάταξη των στοιχείων ενός τεσσάρων διαστάσεων του tensor εισόδου (N, C, H, W) σε έναν tensor μορφής im2col.
- Στόχος είναι να διευκολυνθεί η πράξη της συνελικτικής μετατροπής, αναδιοργανώνοντας τα δεδομένα της εισόδου, έτσι ώστε η πράξη να μπορεί να εκτελεστεί σαν πολλαπλασιασμός matrix.
- Παράμετροι: Οι είσοδοι περιλαμβάνουν το σχήμα της εισόδου, το ύψος και το πλάτος του φίλτρου, την ποσότητα padding, και το βήμα (stride) της συνελικτικής πράξης.
- Αποτελέσματα: Επιστρέφει τους δείκτες k, i, j, που χρησιμοποιούνται για την ανακατασκευή της εισόδου σε μορφή im2col.

### 2. `im2col_indices`

- Αυτή η συνάρτηση χρησιμοποιεί τους δείκτες από τη `get_im2col_indices` για να εξαγάγει όλα τα patches από το array της εισόδου με padding και να τα διαμορφώσει σε μορφή im2col.
- Στόχος: Μετατρέπει το τεσσάρων διαστάσεων σχήμα της εισόδου σε μια πιο βολική διάταξη για αποδοτική συνελικτική πράξη μέσω πολλαπλασιασμού με CuPy.
- Αποτέλεσμα: Επιστρέφει ένα matrix im2col, η οποία περιέχει τα patches της εισόδου σε διατεταγμένη μορφή.

**3. col2im\_indices**

- a. col2im\_indices αναστρέφει τη διαδικασία της im2col, παίρνοντας το matrix cols και επανατοποθετώντας τα δεδομένα σε μια μορφή τεσσάρων διαστάσεων.
- b. Επανασυναρμολογεί την αρχική είσοδο από το im2col
- c. Επιστρέφει τον ανακατασκευασμένο matrix της εισόδου, αφαιρώντας το padding εάν αυτό εφαρμόστηκε.

**4. Softmax**

- a. Μετατρέπει τα στοιχεία ενός διανύσματος εξόδου σε πιθανότητες που αθροίζουν σε 1.
- b. Επιστρέφει τις πιθανότητες για κάθε κατηγορία, οι οποίες είναι κατάλληλες για την τελική ταξινόμηση.

**5. cross\_entropy\_loss**

- a. Η cross\_entropy\_loss είναι η συνάρτηση απώλειας, υπολογίζει τη διαφορά ανάμεσα στις προβλεπόμενες πιθανότητες και στις πραγματικές ετικέτες (labels).
- b. Υπολογίζει την αρνητική λογαριθμική πιθανότητα των προβλεπόμενων τιμών για τις σωστές κατηγορίες και εξάγει τη μέση τιμή για όλες τις παρατηρήσεις.
- c. Επιστρέφει μια μέση τιμή απώλειας, η οποία δείχνει πόσο κοντά βρίσκεται το μοντέλο στις πραγματικές τιμές. Αυτή η απώλεια καθοδηγεί την εκπαίδευση του μοντέλου, προσαρμόζοντας τα βάρη ώστε να ελαχιστοποιηθεί η απόκλιση από τις πραγματικές ετικέτες.

**6. softmax\_backward**

- a. Η softmax\_backward υπολογίζει την παράγωγο της απώλειας ως προς τις εισόδους της συνάρτησης softmax. Αυτό το βήμα είναι απαραίτητο για backpropagation και την ενημέρωση των βαρών κατά την εκπαίδευση.
- b. Λειτουργία: Ξεκινάει αντιγράφοντας τις προβλεπόμενες πιθανότητες (y\_pred) και αφαιρεί 1 από την πρόβλεψη της σωστής κατηγορίας για κάθε δείγμα. Στη συνέχεια, διαιρεί με τον αριθμό των δειγμάτων για να λάβει τη μέση παράγωγο.
- c. Επιστρέφει το dx, δηλαδή την παράγωγο της απώλειας ως προς τις εισόδους της softmax, η οποία χρησιμοποιείται για την αναπροσαρμογή των βαρών στο δίκτυο.

**7. random\_horizontal\_flip**

- a. Η συνάρτηση random\_horizontal\_flip εκτελεί τυχαία οριζόντια αναστροφή στις εικόνες με δοσμένη πιθανότητα p.

# Κλάσεις

## 1. ConvLayer

Η κλάση ConvLayer υλοποιεί ένα συνελικτικό στρώμα για το (CNN) μας.

### a. Κατασκευαστής `__init__`

- i. Αρχικοποιεί τις παραμέτρους του συνελικτικού στρώματος.
- ii. `in_channels`: Ο αριθμός των καναλιών εισόδου
- iii. `out_channels`: Ο αριθμός των εξόδων που θα εφαρμοστούν.
- iv. `kernel_size`: Το μέγεθος του kernel
- v. `stride`: Το βήμα της συνελικτικής πράξης
- vi. `padding`: Το πλήθος των μηδενικών που προστίθενται γύρω από την εικόνα εισόδου.
- vii. Αρχικοποίηση βαρών (τεχνική He) και biases (0)

### b. Forward

- i. Αποθηκεύει το σχήμα της εισόδου και εφαρμόζει zero-padding.
- ii. Μετατρέπει την είσοδο σε μορφή `im2col` για αποδοτικό convolution.
- iii. Υπολογίζει την έξοδο εκτελώντας πολλαπλασιασμό matrix, προσθέτοντας τα biases.
- iv. Διαμορφώνει και επιστρέφει την έξοδο στην επιθυμητή μορφή (`N`, `out_channels`, `out_height`, `out_width`).

### c. Backward

- i. Η έξοδος του επόμενου στρώματος `dout` ανασχηματίζεται (`transpose` και `reshape`) για να ταιριάζει με τις διαστάσεις που χρειάζονται για τον υπολογισμό των βαθμών μεταβολής ως προς τα βάρη.
- ii. Υπολογίζονται οι βαθμοί μεταβολής για τα βάρη `W` μέσω του πολλαπλασιασμού matrix μεταξύ `dout_reshaped` και των στηλών της εισόδου (μετασχηματισμένες μέσω `im2col`).
- iii. Το αποτέλεσμα ανασχηματίζεται στη διάσταση των αρχικών βαρών.
- iv. Οι βαθμοί μεταβολής για τα biases υπολογίζονται με το άθροισμα των στοιχείων του `dout_reshaped` κατά μήκος της κατάλληλης διάστασης.
- v. Ανασχηματίζουμε τα βάρη `W` σε δισδιάστατη μορφή για `backpropagation` των βαθμών μεταβολής.
- vi. Πραγματοποιούμε πολλαπλασιασμό matrix μεταξύ των ανασχηματισμένων βαρών και του `dout_reshaped` για να λάβουμε το `dcols`.
- vii. Μετατρέπουμε το `dcols` πίσω στη μορφή της εισόδου χρησιμοποιώντας τη συνάρτηση `col2im_indices`, που επανατοποθετεί τα δεδομένα στην αρχική τους μορφή.
- viii. Εάν έχει εφαρμοστεί `padding`, αφαιρείται από τον βαθμό μεταβολής της εισόδου.
- ix. Ενημερώνονται τα βάρη και τα biases του στρώματος
- x. Η συνάρτηση επιστρέφει τον βαθμό μεταβολής `dx`

## 2. ReLU

### a. Forward

- i. Αποθηκεύει τα δεδομένα εισόδου  $x$  στη μεταβλητή `self.x` για χρήση στο `backpropagation`.
- ii. Επιστρέφει το μέγιστο μεταξύ του μηδενός και των τιμών του  $x$ . Όλες οι αρνητικές τιμές μετατρέπονται σε μηδέν, ενώ οι θετικές παραμένουν ίδιες.

### b. Backward

- i. Υπολογίζει τον βαθμό μεταβολής  $dx$  πολλαπλασιάζοντας το `dout` με μια μάσκα που οι τιμές της `self.x` είναι θετικές και 0 όπου είναι αρνητικές ή μηδενικές.
- ii. Αυτή η μάσκα διασφαλίζει ότι το `backpropagation` επιτρέπει τη διάδοση του σφάλματος μόνο για τις θετικές τιμές εισόδου.

## 3. MaxPool

Η κλάση `MaxPool` υλοποιεί τη διαδικασία της μέγιστης υποδειγματοληψίας (`Max Pooling`), πραγματοποιεί τη μείωση των διαστάσεων των δεδομένων εισόδου, διατηρώντας τα σημαντικά χαρακτηριστικά.

### a. `__init__`

- i. `size`: Το μέγεθος του παραθύρου υποδειγματοληψίας
- ii. `stride`: Το βήμα της υποδειγματοληψίας
- iii. Η μέγιστη υποδειγματοληψία εφαρμόζεται σε παραθύρια `size x size` με καθορισμένο βήμα.

### b. Forward

- i. Αποθηκεύει το σχήμα των δεδομένων εισόδου  $x$  για χρήση στην `backpropagation`.
- ii. Υπολογίζει το ύψος και το πλάτος της εξόδου, ανάλογα με τις διαστάσεις της εισόδου, το μέγεθος του παραθύρου, και το βήμα.
- iii. Αναδιατάσσει το `array` εισόδου για να εφαρμόσει τη μέγιστη υποδειγματοληψία στα κατάλληλα παραθυράκια.
- iv. Εφαρμόζει τη συνάρτηση `np.max` για να βρει τη μέγιστη τιμή σε κάθε παράθυρο και δημιουργεί μια μάσκα (`mask`) που δείχνει πού βρίσκονται αυτές οι μέγιστες τιμές.
- v. Επιστρέφει την έξοδο της μέγιστης υποδειγματοληψίας, μειώνοντας τις διαστάσεις των δεδομένων

### c. backward

- i. Χρησιμοποιεί τη μάσκα για να διατηρήσει μόνο τους βαθμούς μεταβολής που αντιστοιχούν στις θέσεις των μέγιστων τιμών στο παράθυρο υποδειγματοληψίας.
- ii. Ανασχηματίζει το `array` των βαθμών μεταβολής στην αρχική διάσταση της εισόδου.
- iii. επιστρέφει τον βαθμό μεταβολής  $dx$ .

**d. FCLayer**

Η κλάση FCLayer υλοποιεί το fully connected στρώμα.

**i. `__init__`**

1. `in_size`: Το μέγεθος της εισόδου (αριθμός χαρακτηριστικών).
2. `out_size`: Το μέγεθος της εξόδου (αριθμός νευρώνων).
3. Αρχικοποίηση βάρων (τεχνική He) και biases (0)

**ii. Forward**

1. Αποθηκεύει την είσοδο  $x$  για χρήση στην backpropagation
2. Υπολογίζει το γινόμενο του tensor εισόδου  $x$  με τα βάρη  $W$  και προσθέτει τα biases  $b$ .
3. Επιστρέφει την έξοδο  $out$ , η οποία είναι το αποτέλεσμα του πλήρως συνδεδεμένου στρώματος.

**iii. Backward**

1. Υπολογίζει τον βαθμό μεταβολής της εισόδου  $dx$  ως το γινόμενο του  $dout$  με τα transposed βάρη  $W$ .
2. Υπολογίζει τον βαθμό μεταβολής για τα βάρη  $dW$  ως το γινόμενο του transposed tensor εισόδου  $x$  με το  $dout$ .
3. Υπολογίζει τον βαθμό μεταβολής για τα biases  $db$  με το άθροισμα των στοιχείων του  $dout$  κατά μήκος του άξονα των δειγμάτων.
4. Ενημερώνει τα βάρη και τα biases χρησιμοποιώντας τον ρυθμό μάθησης (`learning_rate`).

**e. Dropout**

Η κλάση Dropout χρησιμοποιείται για τη μείωση της πιθανότητας υπερεκπαίδευσης (overfitting). Η ιδέα του Dropout είναι να «απενεργοποιεί» τυχαία νευρώνες κατά τη διάρκεια της εκπαίδευσης, επιτρέποντας στο δίκτυο να γενικεύσει καλύτερα.

**i. `__init__`**

1. Αρχικοποιεί το ποσοστό dropout και τη μάσκα.

**ii. Forward**

1. Εάν η εκτέλεση βρίσκεται στη φάση της εκπαίδευσης (`is_training=True`), δημιουργεί μια μάσκα τυχαίων τιμών με το ίδιο σχήμα όπως η είσοδος  $x$ . Οι τιμές της μάσκας είναι 1 όπου ο νευρώνας διατηρείται και 0 όπου απενεργοποιείται, ανάλογα με το ποσοστό dropout.
2. Πολλαπλασιάζει την είσοδο  $x$  με τη μάσκα, «απενεργοποιώντας» έτσι ορισμένους νευρώνες.

**iii. Backward**

1. Εάν έχει δημιουργηθεί η μάσκα κατά το forward pass, οι βαθμοί μεταβολής  $dout$  πολλαπλασιάζονται με τη μάσκα. Έτσι, μόνο οι ενεργοί νευρώνες κατά την εμπρόσθια διάδοση λαμβάνουν βαθμούς μεταβολής.
2. Επιστρέφει τους τροποποιημένους βαθμούς μεταβολής  $dout$

**iv. OneCycleLR**

Η κλάση OneCycleLR υλοποιεί τη στρατηγική One Cycle Learning Rate. Η ιδέα είναι να ξεκινάμε από μια χαμηλή τιμή ρυθμού μάθησης, να την αυξάνουμε γραμμικά προς τη μέγιστη τιμή κατά τη διάρκεια της φάσης warm-up, και στη συνέχεια να τη μειώνουμε ξανά στη φάση decay.

**1. \_\_init\_\_**

- a. max\_lr: Ο μέγιστος ρυθμός μάθησης που θα χρησιμοποιηθεί κατά τη διάρκεια της εκπαίδευσης.
- b. total\_steps: Ο συνολικός αριθμός βημάτων εκπαίδευσης.
- c. pct\_start: Το ποσοστό των συνολικών βημάτων κατά το οποίο αυξάνεται ο ρυθμός μάθησης.
- d. anneal\_strategy: Η στρατηγική απόσβεσης.
- e. div\_factor: Ο παράγοντας που καθορίζει τη σχέση μεταξύ του αρχικού και του μέγιστου ρυθμού μάθησης.
- f. Ο αρχικός ρυθμός μάθησης (initial\_lr) υπολογίζεται διαιρώντας τον max\_lr με τον div\_factor.
- g. Ο τελικός ρυθμός μάθησης (final\_lr) υπολογίζεται ως το initial\_lr διαιρούμενο ξανά με τον div\_factor.

**2. get\_lr**

- a. Εάν η εκπαίδευση βρίσκεται στη φάση προθέρμανσης, ο ρυθμός μάθησης αυξάνεται γραμμικά από το initial\_lr έως το max\_lr.
- b. Εάν η εκπαίδευση βρίσκεται στη φάση απόσβεσης, ο ρυθμός μάθησης μειώνεται από το max\_lr έως το final\_lr, με τη στρατηγική που επιλέχθηκε.
- c. Επιστρέφει τον υπολογισμένο ρυθμό μάθησης.

**3. Step**

- a. καλείται μετά από κάθε βήμα εκπαίδευσης για να ενημερώσει τη στρατηγική.

**4. Reset**

- a. Χρησιμοποιείται όταν θέλουμε να επανεκκινήσουμε τη στρατηγική εκπαίδευσης.

**v. BatchNorm**

Η κλάση BatchNorm υλοποιεί τη μέθοδο κανονικοποίησης batch, η οποία χρησιμοποιείται για τη σταθεροποίηση και την επιτάχυνση της εκπαίδευσης. επιτρέπει τη χρήση υψηλότερων ρυθμών μάθησης, βελτιώνοντας την απόδοση του μοντέλου.

**1. \_\_init\_\_**

- a. num\_features: Ο αριθμός των χαρακτηριστικών εισόδου
- b. momentum: Ο συντελεστής για τον υπολογισμό των κινούμενων μέσων όρων της μέσης τιμής και της διακύμανσης.
- c. epsilon: Μια μικρή σταθερά για την αποφυγή διαίρεσης με το μηδέν.
- d. gamma: Παράμετρος κλιμάκωσης.

- e. `beta`: Παράμετρος μετατόπισης.
- f. `running_mean`: Κινούμενος μέσος όρος της μέσης τιμής για χρήση στην πρόβλεψη.
- g. `running_var`: Κινούμενος μέσος όρος της διακύμανσης για χρήση στην πρόβλεψη.

## 2. Forward

- a. Κατά την εκπαίδευση, υπολογίζει τη μέση τιμή και τη διακύμανση για το τρέχον batch.
- b. Κανονικοποιεί τα δεδομένα αφαιρώντας τη μέση τιμή και διαιρώντας με την τυπική απόκλιση, στη συνέχεια εφαρμόζει την κλιμάκωση και τη μετατόπιση (`gamma` και `beta`).
- c. Ενημερώνει τους κινούμενους μέσους όρους της μέσης τιμής και της διακύμανσης.
- d. Κατά την πρόβλεψη, χρησιμοποιεί τους κινούμενους μέσους όρους για την κανονικοποίηση.
- e. Επιστρέφει το κανονικοποιημένο batch των δεδομένων εισόδου

## 3. Backward

- a. Υπολογίζει τους βαθμούς μεταβολής για τις παραμέτρους κλιμάκωσης `gamma` και μετατόπισης `beta`.
- b. Υπολογίζει τον βαθμό μεταβολής για την κανονικοποιημένη είσοδο.
- c. Εφαρμόζει chain rule για την backpropagation μέσα από τη διαδικασία κανονικοποίησης, υπολογίζοντας τα `dvar` και `dmean`.
- d. Ενημερώνει τις παραμέτρους κλιμάκωσης και μετατόπισης χρησιμοποιώντας τον ρυθμό μάθησης (`learning_rate`).
- e. Υπολογίζει τον βαθμό μεταβολής για την είσοδο `dx`.
- f. Ενημερώνει τις παραμέτρους `gamma` και `beta`.
- g. Επιστρέφει τον βαθμό μεταβολής της εισόδου `dx`

# Κατασκευή του Μοντέλου

Αφού περιγράψαμε τις βοηθητικές συναρτήσεις και τις βασικές κλάσεις που δημιουργήσαμε, προχωράμε στη διαδικασία κατασκευής του CNN για την ταξινόμηση του CIFAR-10. Το μοντέλο μας βασίζεται σε μια σειρά από συνελκτικά και πλήρως συνδεδεμένα στρώματα.

## Υπερπαράμετροι

- learning\_rate: 3 (επιτρέπεται λόγω του **OneCycleLR** και **BatchNorm**)
- num\_epochs: 70
- batch\_size: 64

## Αρχιτεκτονική του Μοντέλου

### 1. Πρώτο Συνελκτικό Μπλοκ

- a. ConvLayer με 3 κανάλια εισόδου και 32 εξόδους. Το μέγεθος του πυρήνα είναι 3x3, και χρησιμοποιούμε padding 1.
- b. BatchNorm: Κανονικοποίηση.
- c. ReLU: Συνάρτηση ενεργοποίησης, για τη μη γραμμικότητα.
- d. MaxPool: Στρώμα μέγιστης υποδειγματοληψίας με μέγεθος παραθύρου 2x2 και βήμα 2 για τη μείωση των διαστάσεων.

### 2. Δεύτερο Συνελκτικό Μπλοκ

- a. ConvLayer με 32 κανάλια εισόδου και 64 εξόδους, πυρήνας 3x3 και padding 1.
- b. BatchNorm, ReLU, MaxPool: Παρόμοιες λειτουργίες όπως στο πρώτο μπλοκ.

### 3. Τρίτο Συνελκτικό Μπλοκ

- a. ConvLayer: Συνελκτικό στρώμα με 64 κανάλια εισόδου και 128 εξόδους, πυρήνας 3x3 και padding 1.
- b. BatchNorm, ReLU, MaxPool: Παρόμοιες λειτουργίες όπως στα προηγούμενα μπλοκ.

### 4. Πλήρως Συνδεδεμένα Στρώματα

- a. Κάνουμε flatten την έξοδο του τρίτου ConvLayer
- b. FCLayer: Πλήρως συνδεδεμένο στρώμα με 256 εξόδους.
- c. ReLU: Συνάρτηση ενεργοποίησης.
- d. Dropout: Dropout στρώμα με ποσοστό 50% για την αποφυγή υπερεκπαίδευσης.
- e. FCLayer: Τελικό πλήρως συνδεδεμένο στρώμα με 10 εξόδους (μία για κάθε κλάση CIFAR-10).



# Training του Μοντέλου

Μετά την κατασκευή CNN, η διαδικασία εκπαίδευσης του μοντέλου πραγματοποιείται μέσω ενός βρόχου.

## 1. Παράμετροι

- `num_batches`: Ο αριθμός των batches που προκύπτουν όταν διαιρούμε τον αριθμό των δεδομένων εκπαίδευσης με το μέγεθος του batch (`batch_size`). Αυτός ο αριθμός καθορίζει πόσες φορές θα ενημερώσουμε τα βάρη του δικτύου ανά εποχή.
- `loss_history`: Μια λίστα που αποθηκεύει την απώλεια κάθε εποχής, για τη μελέτη της σύγκλισης του μοντέλου.
- `test_loss_history`: Μια λίστα που αποθηκεύει την απώλεια του συνόλου δοκιμών ανά εποχή.

## 2. Αρχικοποίηση του OneCycleLR

### Περιγραφή του Βρόχου Εκπαίδευσης

- Επανάληψη για κάθε εποχή
  - Ο βρόχος εκπαίδευσης εκτελείται για `num_epochs` εποχές. Σε κάθε εποχή, ανακατεύουμε (`shuffle`) τα δεδομένα εκπαίδευσης για να διασφαλίσουμε ότι το μοντέλο δεν προσαρμόζεται σε συγκεκριμένη σειρά των δειγμάτων.
  - Χρησιμοποιούμε τη βιβλιοθήκη `torch` για να εμφανίσουμε μια γραμμή προόδου που μας δίνει πληροφορίες για την πρόοδο της εκπαίδευσης.
- Περίοδος Batches**
  - Δημιουργία Batch: Παίρνουμε ένα υποσύνολο των δεδομένων (`X_batch`, `y_batch`). Εφαρμόζουμε τυχαία οριζόντια αναστροφή στα δεδομένα εικόνας.
  - Forward pass: Περνάμε το batch μέσω του δικτύου καλώντας τη συνάρτηση `forward_pass`. Στη συνέχεια, χρησιμοποιούμε τη συνάρτηση `softmax` για να μετατρέψουμε τα `scores` του μοντέλου σε πιθανότητες και υπολογίζουμε την απώλεια χρησιμοποιώντας τη `cross_entropy_loss`.
  - Backward pass: Υπολογίζουμε τους βαθμούς μεταβολής (`gradients`) μέσω της συνάρτησης `softmax_backward` και ενημερώνουμε τα βάρη του μοντέλου καλώντας τη `backward_pass`. Χρησιμοποιούμε τον τρέχοντα ρυθμό μάθησης από το One Cycle LR για τις ενημερώσεις.
  - Ενημέρωση Learning Rate: Ενημερώνουμε το One Cycle LR με τη συνάρτηση `scheduler.step()`.
- Υπολογισμός Μέσης Απώλειας Εποχής:**
  - Υπολογίζουμε τη μέση απώλεια της εποχής διαιρώντας τη συνολική απώλεια της εποχής με τον αριθμό των batches. Καταγράφουμε τη μέση απώλεια στο `loss_history`.

#### 4. Υπολογισμός Απώλειας Test

- a. Μετά από κάθε εποχή, αξιολογούμε το μοντέλο στο σύνολο δοκιμών χωρίς την εφαρμογή Dropout. Υπολογίζουμε την απώλεια του συνόλου δοκιμών και την καταγράφουμε στο `test_loss_history`.

## Evaluation του Μοντέλου

Αφού ολοκληρώσουμε τη διαδικασία εκπαίδευσης, πρέπει να αξιολογήσουμε την απόδοσή του σε μη γνωστά δεδομένα.

### 1. Αρχικοποίηση Μεταβλητών

- a. `num_samples`: Ο συνολικός αριθμός δειγμάτων στο σύνολο δεδομένων `X`.
- b. `num_batches`: Ο αριθμός των `batches` που μπορούν να δημιουργηθούν από τα δείγματα, με βάση το μέγεθος του `batch`.
- c. `correct`: Μετρητής για τον αριθμό των σωστών προβλέψεων.
- d. `total`: Μετρητής για τον συνολικό αριθμό δειγμάτων που έχουν αξιολογηθεί.

### 2. Βρόχος Αξιολόγησης

- a. Παίρνουμε ένα υποσύνολο των δεδομένων `X_batch` και τις αντίστοιχες ετικέτες `y_batch`.
- b. Forward pass: Εκτελούμε `forward_pass` για να πάρουμε τις πιθανότητες των κλάσεων (`y_pred`).
- c. Χρησιμοποιούμε τη `softmax` για να μετατρέψουμε τα `scores` σε πιθανότητες και επιλέγουμε την κλάση με τη μέγιστη πιθανότητα με τη συνάρτηση `np.argmax`.
- d. Ενημερώνουμε τον μετρητή `correct` προσθέτοντας τον αριθμό των σωστών προβλέψεων, συγκρίνοντας τις προβλέψεις με τις πραγματικές ετικέτες `y_batch`.
- e. Ενημερώνουμε τον μετρητή `total` με το μέγεθος του τρέχοντος `batch`.

### 3. Υπολογισμός Ακρίβειας

- a. Επιστρέφουμε την ακρίβεια ως τελική μέτρηση απόδοσης του μοντέλου.

# Αποτελέσματα και Ανάλυση

Μετά την ολοκλήρωση της εκπαίδευσης του (CNN) και την αξιολόγησή του στο σύνολο δοκιμών, καταγράψαμε τα ακόλουθα αποτελέσματα (LR = 3, epochs=70, batch\_size=64):

- Ακρίβεια Εκπαίδευσης (Training Accuracy): 92.16%
- Ακρίβεια Δοκιμών (Test Accuracy): 83.53%
- Runtime: ✓ 40m 13.8s

Η διαφορά μεταξύ της ακρίβειας εκπαίδευσης και της ακρίβειας δοκιμών (93.56% έναντι 83.22%) δείχνει ότι υπάρχει κάποια απόκλιση, η οποία μπορεί να οφείλεται σε overfitting.

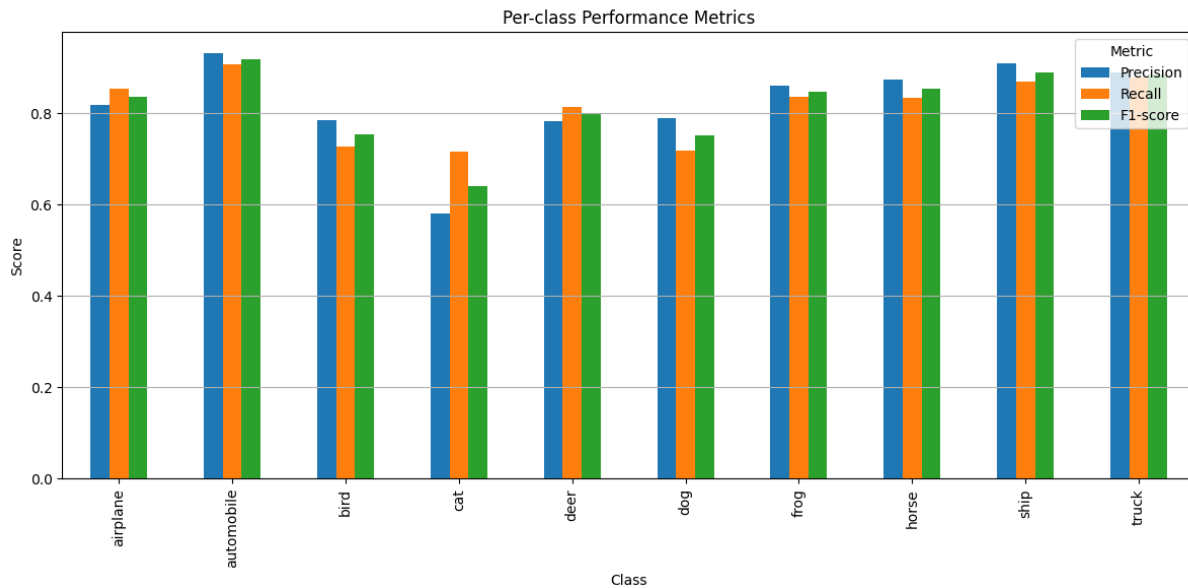


Το παραπάνω γράφημα παρουσιάζει την εξέλιξη της απώλειας εκπαίδευσης (Training Loss) και της απώλειας δοκιμών (Test Loss) ανά εποχή, παρέχοντας μια εικόνα της απόδοσης του μοντέλου κατά τη διαδικασία εκπαίδευσης. Παρατηρείται ότι, παρόλο που υπάρχει κάποια διακύμανση στην καμπύλη της απώλειας εκπαίδευσης, οι επιλεγμένες υπερπαραμέτροι και οι τεχνικές εκπαίδευσης, ήταν οι βέλτιστες, με βάση τις δοκιμές μου (ψηλό Test Accuracy και μικρή διαφορά με το Training Accuracy).



Ο πίνακας σύγχυσης παρέχει μια εικόνα της απόδοσης του μοντέλου μας στην ταξινόμηση του συνόλου CIFAR-10, αποτυπώνοντας τον αριθμό των σωστών και εσφαλμένων προβλέψεων για κάθε κατηγορία. Κάθε γραμμή του πίνακα αντιπροσωπεύει τις πραγματικές ετικέτες (True Label), ενώ κάθε στήλη αντιπροσωπεύει τις προβλεπόμενες ετικέτες (Predicted Label).

Plotting per-class metrics...



Το γράφημα απεικονίζει την ακρίβεια (Precision), την ανάκληση (Recall), και το F1-score για κάθε μία από τις 10 κατηγορίες του συνόλου δεδομένων CIFAR-10

- Οι κατηγορίες *airplane*, *automobile*, *frog*, *ship*, και *truck* παρουσιάζουν υψηλές τιμές για όλες τις μετρικές (precision, recall, και F1-score), που βρίσκονται κοντά στο 0.8 ή υψηλότερα. Αυτό υποδηλώνει ότι το μοντέλο ταξινομεί αυτές τις κατηγορίες με μεγάλη ακρίβεια και ανακτά αποτελεσματικά τα δείγματα αυτών των κατηγοριών. Άρα το μοντέλο έχει καλή ισορροπία μεταξύ της ικανότητάς του να προβλέπει σωστά (precision) και της ικανότητάς του να μην παραλείπει δείγματα (recall) σε αυτές τις κατηγορίες
- Οι κατηγορίες *bird*, *cat* και *dog* παρουσιάζουν χαμηλότερα scores, γεγονός που δηλώνει ότι το μοντέλο έχει δυσκολία στη σωστή διάκριση αυτών των κατηγοριών. Το μοντέλο τείνει να προβλέπει λανθασμένα τα δείγματα επειδή τα χαρακτηριστικά αυτών των κατηγοριών είναι λιγότερο διακριτά (πχ *cat-dog*). Αυτό φαινόταν από πριν στον πίνακα σύγχυσης
- Επίσης σε κάποιες κατηγορίες εμφανίζεται ανισορροπία μεταξύ Precision και Recall. Για παράδειγμα, η κατηγορία *dog* έχει υψηλότερη ακρίβεια σε σύγκριση με την ανάκληση, υποδεικνύοντας ότι το μοντέλο είναι επιλεκτικό στη θετική πρόβλεψη αυτής της κατηγορίας, παραλείποντας ορισμένα σωστά δείγματα. Αντίθετα, στην κατηγορία *cat*, η ανάκληση είναι υψηλότερη από την ακρίβεια, κάτι που σημαίνει ότι το μοντέλο ανακτά περισσότερα δείγματα αυτής της κατηγορίας, αλλά με μεγαλύτερο ποσοστό εσφαλμένων προβλέψεων.


Classification Report:				
	precision	recall	f1-score	support
airplane	0.82	0.85	0.84	998
automobile	0.93	0.91	0.92	999
bird	0.79	0.73	0.76	999
cat	0.58	0.72	0.64	997
deer	0.78	0.81	0.80	1000
dog	0.79	0.72	0.75	997
frog	0.86	0.84	0.85	1000
horse	0.87	0.83	0.85	997
ship	0.91	0.87	0.89	997
truck	0.89	0.88	0.88	1000
accuracy	0.82	0.82	0.82	9984

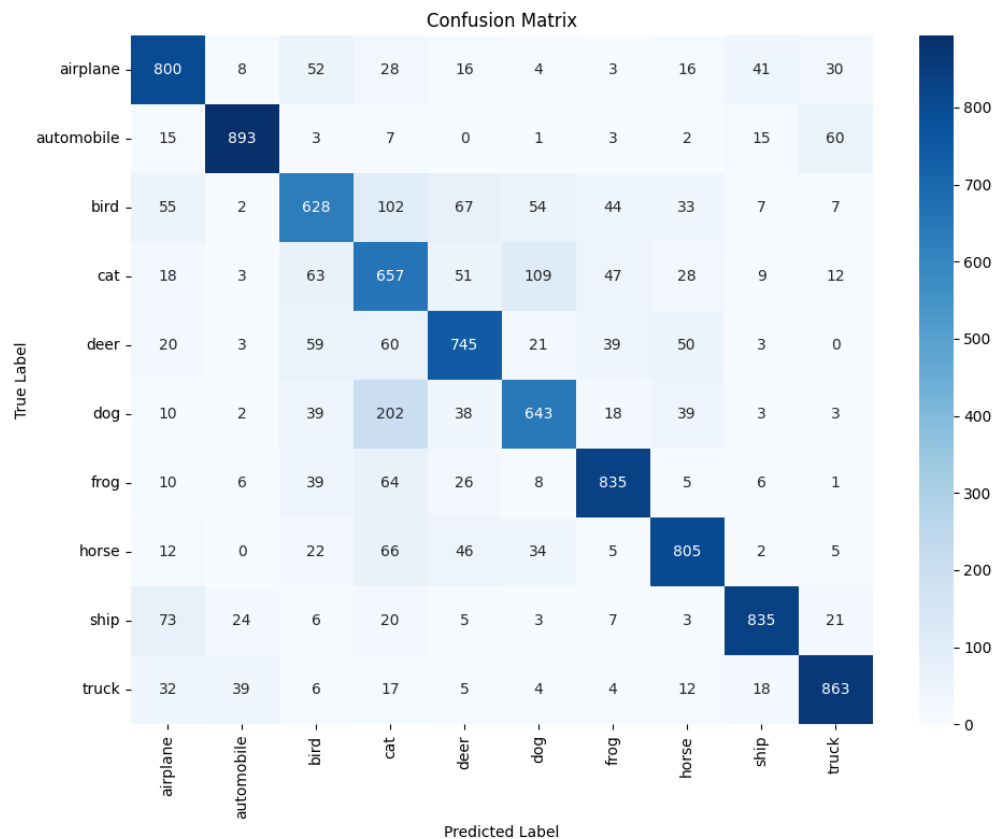
# Αποτελέσματα με epoch=30

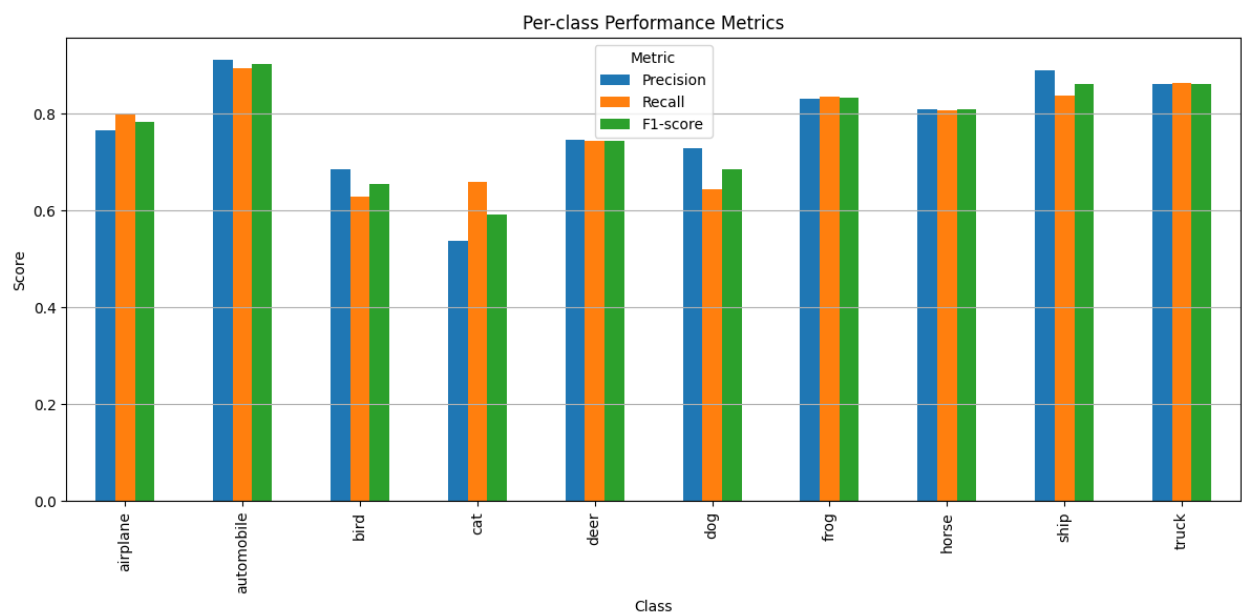
## Υπερπαράμετροι

- learning\_rate: 3
- num\_epochs: 30
- batch\_size: 64

## Αποτελέσματα

- Ακρίβεια Εκπαίδευσης (Training Accuracy): 84.17%
- Ακρίβεια Δοκιμών (Test Accuracy): 80.03%
- Runtime:  16m 52.2s





Classification Report:				
	precision	recall	f1-score	support
airplane	0.77	0.80	0.78	998
automobile	0.91	0.89	0.90	999
bird	0.68	0.63	0.66	999
cat	0.54	0.66	0.59	997
deer	0.75	0.74	0.75	1000
dog	0.73	0.64	0.68	997
frog	0.83	0.83	0.83	1000
horse	0.81	0.81	0.81	997
ship	0.89	0.84	0.86	997
truck	0.86	0.86	0.86	1000
accuracy			0.77	9984
macro avg	0.78	0.77	0.77	9984
weighted avg	0.78	0.77	0.77	9984

# Αποτελέσματα με LR=0.01

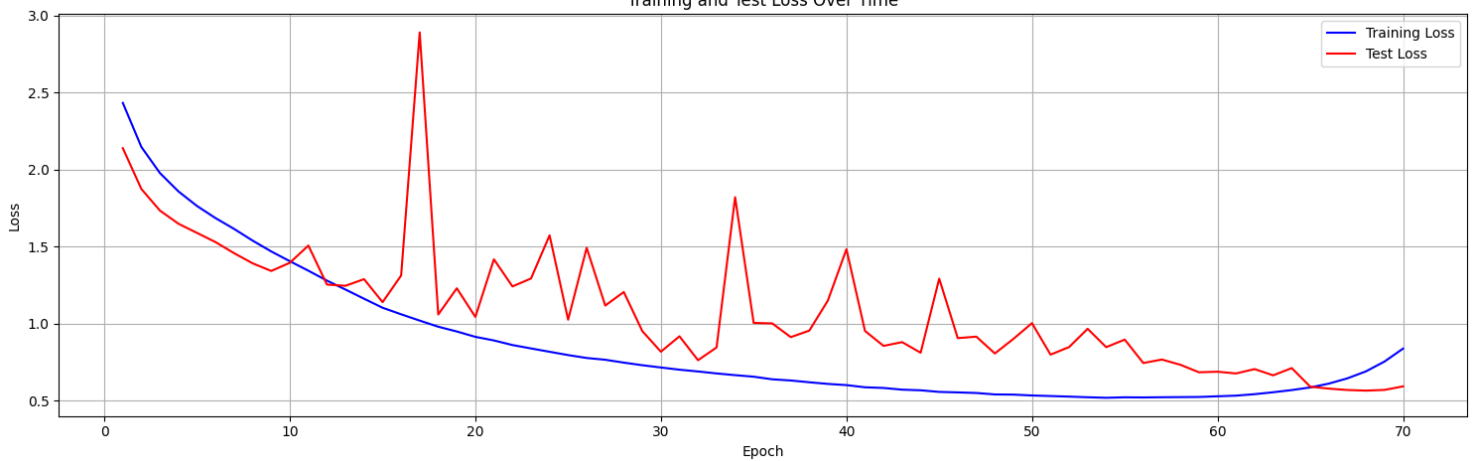
## Υπερπαράμετροι

- learning\_rate: 0.01
- num\_epochs: 70
- batch\_size: 64

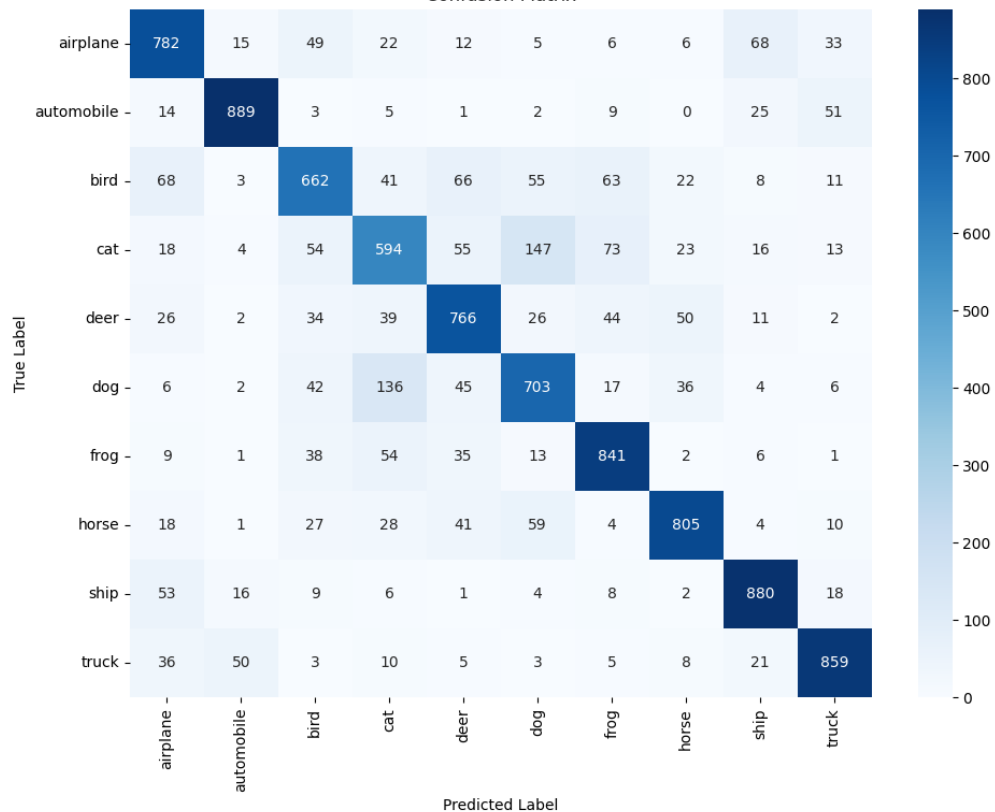
## Αποτελέσματα

- Ακρίβεια Εκπαίδευσης (Training Accuracy): 84.49%
- Ακρίβεια Δοκιμών (Test Accuracy): 80.75%
- Runtime: ✓ 48m 3.2s

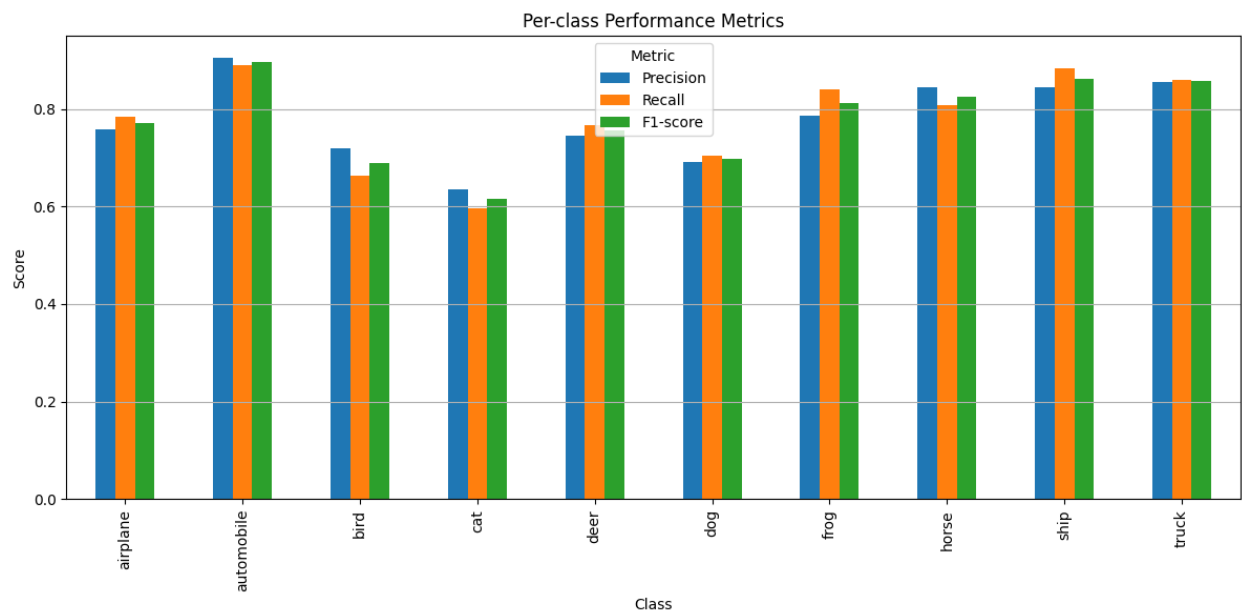
Training and Test Loss Over Time



Confusion Matrix







Classification Report:				
	precision	recall	f1-score	support
airplane	0.76	0.78	0.77	998
automobile	0.90	0.89	0.90	999
bird	0.72	0.66	0.69	999
cat	0.64	0.60	0.61	997
deer	0.75	0.77	0.76	1000
dog	0.69	0.71	0.70	997
frog	0.79	0.84	0.81	1000
horse	0.84	0.81	0.83	997
ship	0.84	0.88	0.86	997
truck	0.86	0.86	0.86	1000
accuracy			0.78	9984
macro avg	0.78	0.78	0.78	9984
weighted avg	0.78	0.78	0.78	9984

# ΔΟΚΙΜΕΣ ΜΕ ΑΛΛΕΣ ΑΡΧΙΤΕΚΤΟΝΙΚΕΣ

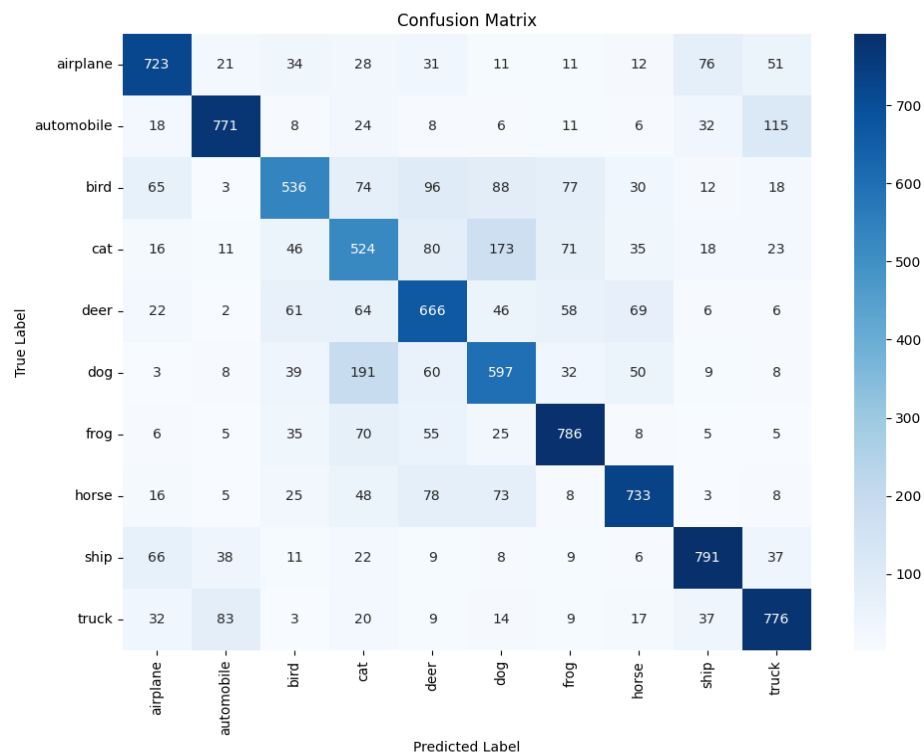
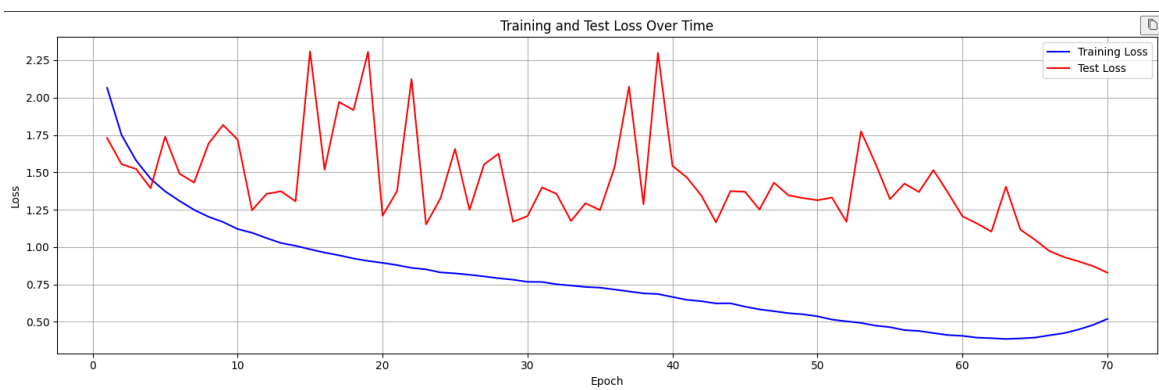
## Αποτελέσματα με μονό 32-ConvLayer

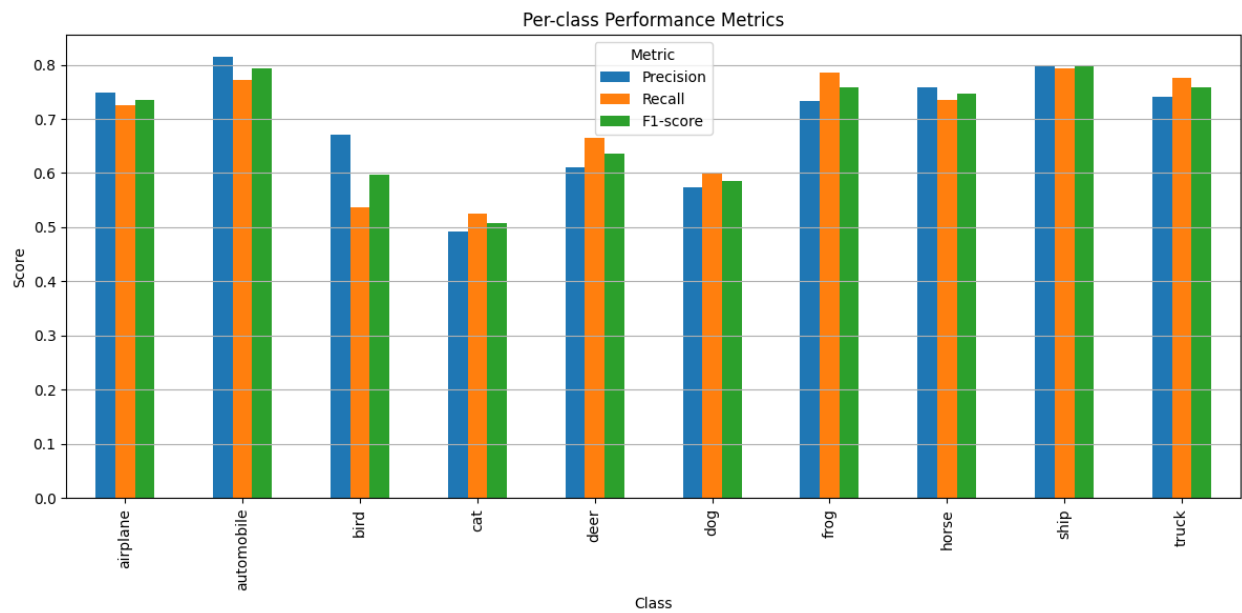
### Υπερπαράμετροι

- learning\_rate: 0.1
- num\_epochs: 70
- batch\_size: 64

### Αποτελέσματα

- Ακρίβεια Εκπαίδευσης (Training Accuracy): 92.91%
- Ακρίβεια Δοκιμών (Test Accuracy): 73.72%
- Runtime: ✓ 18m 12.8s





Classification Report:				
	precision	recall	f1-score	support
airplane	0.75	0.72	0.74	998
automobile	0.81	0.77	0.79	999
bird	0.67	0.54	0.60	999
cat	0.49	0.53	0.51	997
deer	0.61	0.67	0.64	1000
dog	0.57	0.60	0.59	997
frog	0.73	0.79	0.76	1000
horse	0.76	0.74	0.75	997
ship	0.80	0.79	0.80	997
truck	0.74	0.78	0.76	1000
accuracy			0.69	9984
macro avg	0.69	0.69	0.69	9984
weighted avg	0.69	0.69	0.69	9984

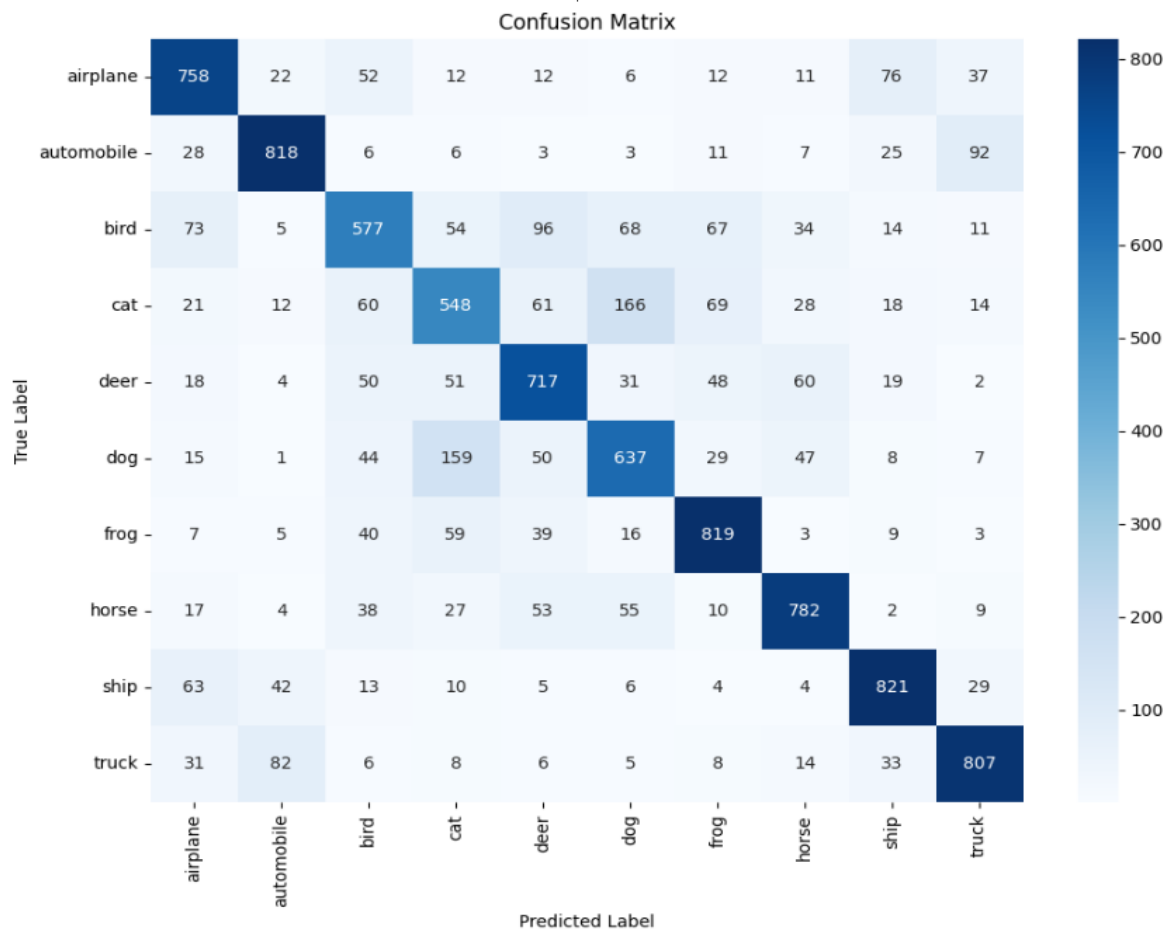
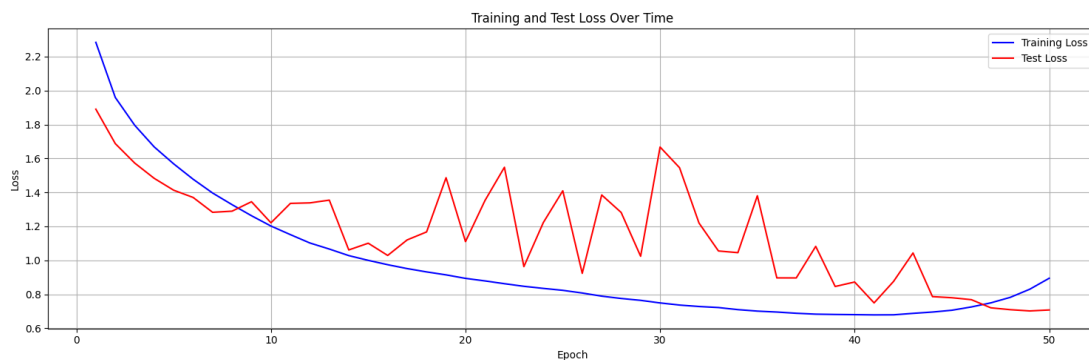
# Αποτελέσματα με 64-128 ConvLayers

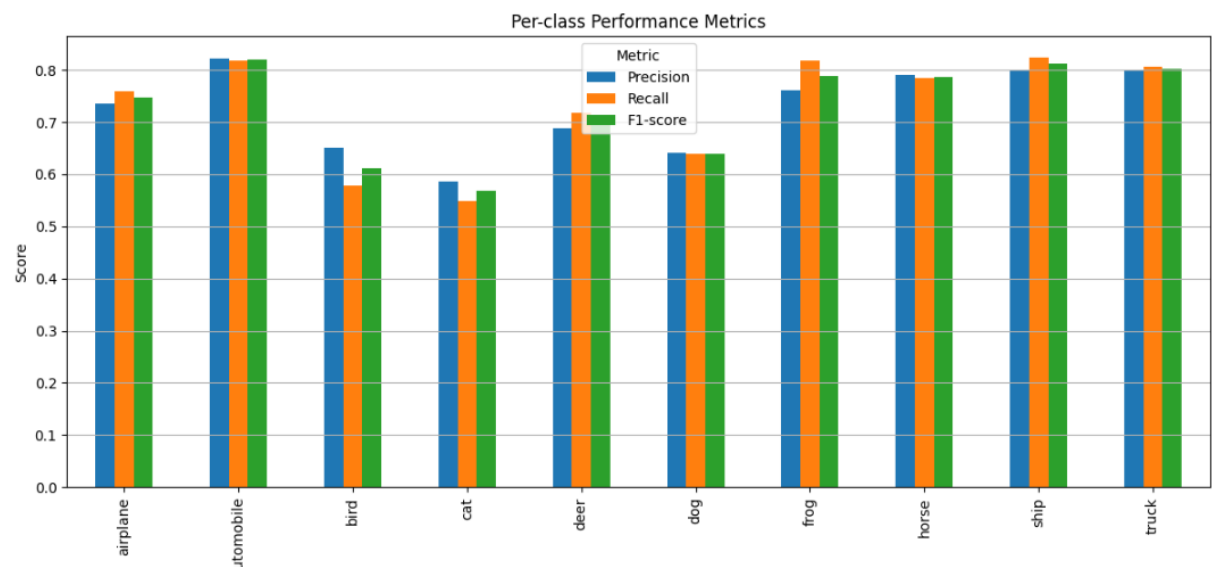
## Υπερπαράμετροι

- learning\_rate: 0.01
- num\_epochs: 50
- batch\_size: 64

## Αποτελέσματα

- Ακρίβεια Εκπαίδευσης (Training Accuracy): 78.75%
- Ακρίβεια Δοκιμών (Test Accuracy): 75.68%
- Runtime: ✓ 48m 53.3s





Classification Report:

	precision	recall	f1-score	support
airplane	0.74	0.76	0.75	998
automobile	0.82	0.82	0.82	999
bird	0.65	0.58	0.61	999
cat	0.59	0.55	0.57	997
deer	0.69	0.72	0.70	1000
dog	0.64	0.64	0.64	997
frog	0.76	0.82	0.79	1000
horse	0.79	0.78	0.79	997
ship	0.80	0.82	0.81	997
truck	0.80	0.81	0.80	1000
accuracy			0.73	9984
macro avg	0.73	0.73	0.73	9984
weighted avg	0.73	0.73	0.73	9984

# Μερικές αλλαγές στο δίκτυο

Έγινε υλοποίηση και αλλαγή σε κάποιες συναρτήσεις

## Cutout

εφαρμόζει data augmentation, Αυτή η τεχνική εισάγει τυχαία ορθογώνια κενά στις εικόνες κατά τη διάρκεια της εκπαίδευσης με πιθανότητα  $p$ .

## L2 με Weight Decay

υλοποίηση Weight Decay στις κατάλληλες συναρτήσεις

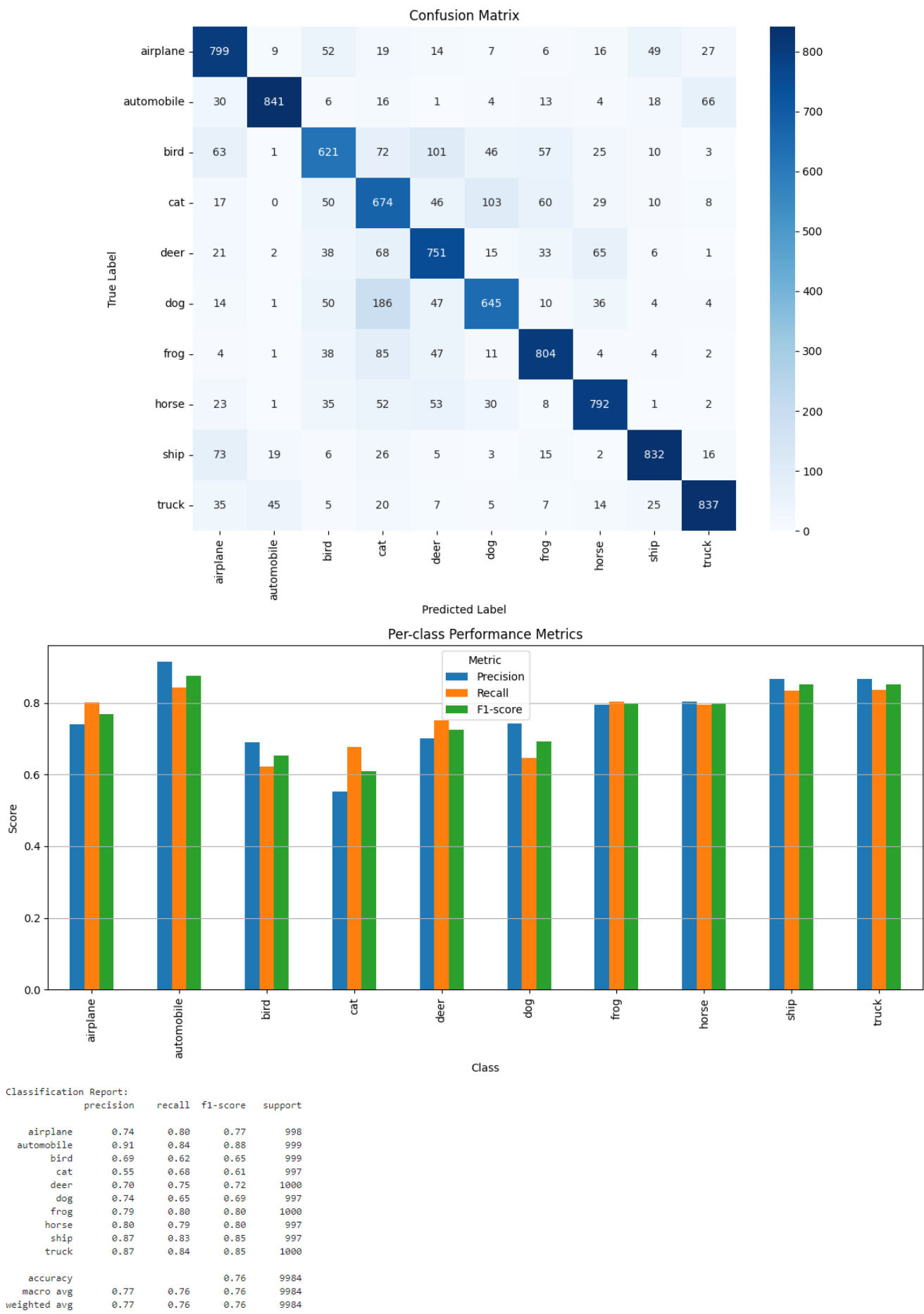
## Υπερπαράμετροι

- learning\_rate: 1
- num\_epochs: 30
- batch\_size: 64

## Αποτελέσματα

- Ακρίβεια Εκπαίδευσης (Training Accuracy): 81.56%
- Ακρίβεια Δοκιμών (Test Accuracy): 79.55%
- Runtime: 33 min



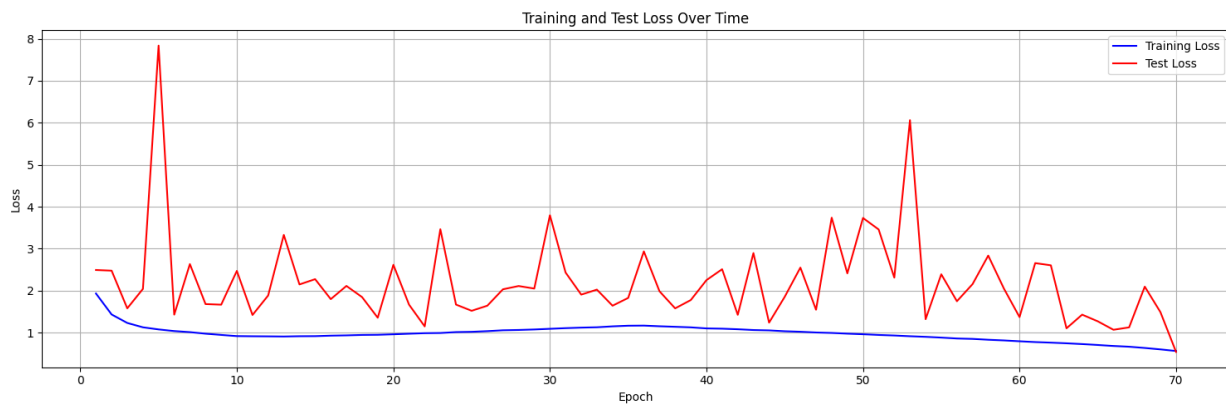


## Υπερπαράμετροι

- learning\_rate: 0.8
- num\_epochs: 70
- batch\_size: 64

## Αποτελέσματα

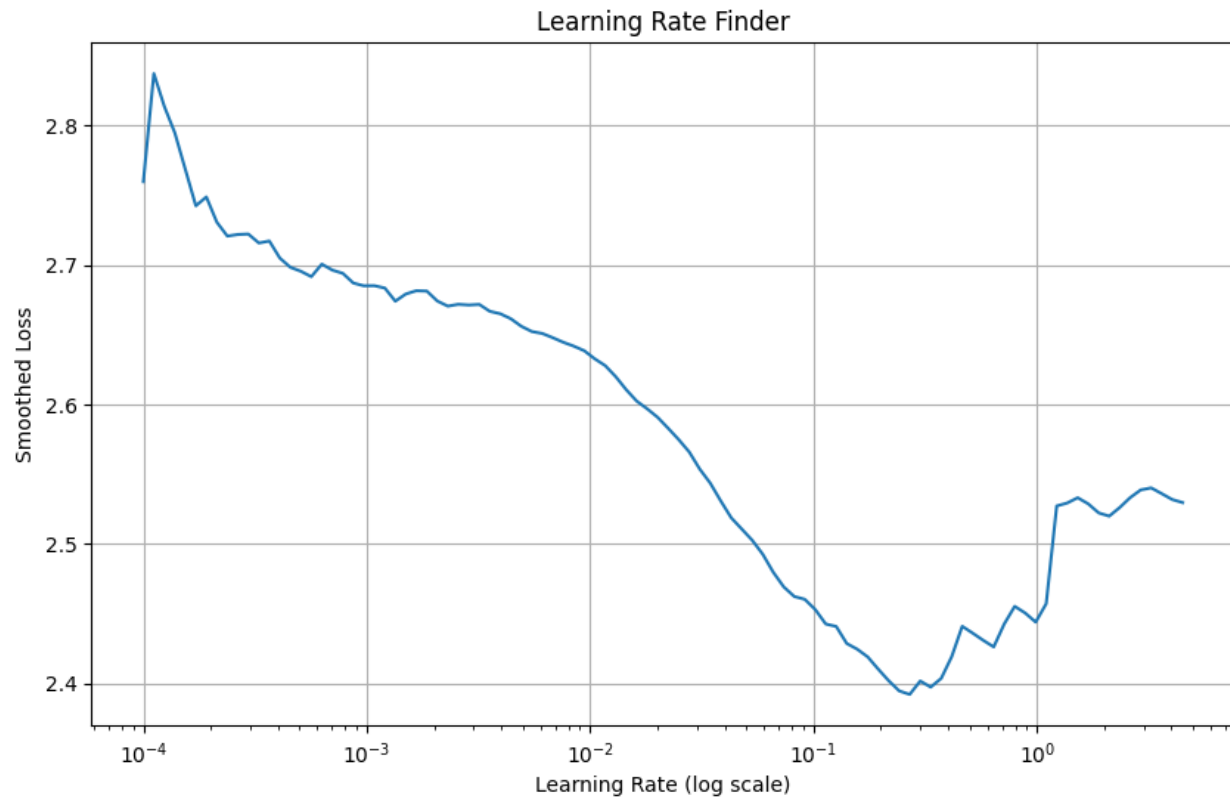
- Ακρίβεια Εκπαίδευσης (Training Accuracy): 86.62%
- Ακρίβεια Δοκιμών (Test Accuracy): 82.44%
- Runtime: 81 min



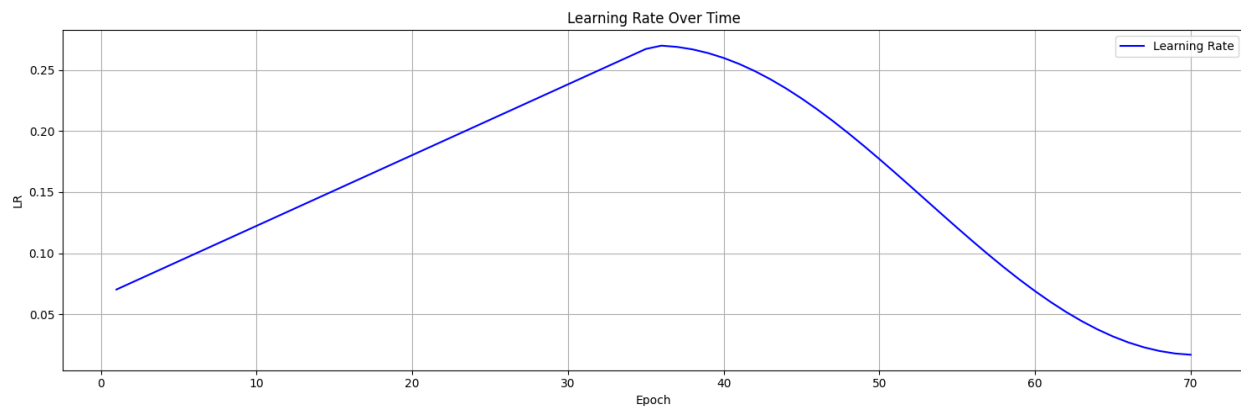
## Έυρεση βέλτιστου Learning Rate

Δημιούργησα ένα test, για να βρω τον βέλτιστο ρυθμό μάθησης για το μοντέλο μας (τρέχω σε υποσύνολο του dataset ). Αυξάνω εκθετικά τον ρυθμό μάθησης από μια χαμηλή τιμή σε κάθε επανάληψη. Έπειτα, δημιουργώ το διάγραμμα της απώλειας σε σχέση με τον ρυθμό μάθησης.





Έχω minimum loss στο LR = 2.69e-01. Σύμφωνα με το paper ([arXiv:1506.01186](https://arxiv.org/abs/1506.01186)). Τοποθετώ  $\text{max\_lr}=0.27$   
 $\text{min\_lr}=\frac{0.27}{4}$  στο OneCycleLR. Ορίζω **total\_steps**= num\_epochs\*iterations\_per\_epoch με αποτέλεσμα ο scheduler μου να μην κάνει reset.



Επίσης δημιούργησα κλάση **CNNModel**, στην οποία υλοποιούνται οι μέθοδοι **forward\_pass** **backward\_pass**, αλλά γίνεται και αρχικοποίηση της αρχιτεκτονικής του μοντέλου. Έτσι, γίνεται ενθυλάκωση όλων των επιπέδων του μοντέλου σε ένα object, μπορώντας εύκολα να το επανεκκινήσουμε και να επαναφέρουμε τις παραμέτρους του

## Αποτελέσματα με τις νέες παραμέτρους του OneCycleLR

### Υπερπαραμέτροι

- learning\_rate: όπως αναφέρθηκε
- num\_epochs: 70
- batch\_size: 100
- reg\_lambda = 0.0005

### Αποτελέσματα

- Ακρίβεια Εκπαίδευσης (Training Accuracy): 97.78%
- Ακρίβεια Δοκιμών (Test Accuracy): 84.21%
- Runtime: 78 min



Παρατηρείται overfitting, Αυξάνω το reg\_lambda = 0.005

### Αποτελέσματα

- Ακρίβεια Εκπαίδευσης (Training Accuracy): 87.67%
- Ακρίβεια Δοκιμών (Test Accuracy): 81.96%
- Runtime: 73 min



## ΣΥΜΠΕΡΑΣΜΑΤΑ

Επέλεξα το καλύτερο CNN με βάση την υψηλότερη ακρίβεια δοκιμών, ελαχιστοποιώντας το overfitting.

### **Νευρωνικό Δίκτυο (CNN):**

- **Ακρίβεια Εκπαίδευσης:** 86.62%
- **Ακρίβεια Δοκιμών:** 82.44%
- Το Νευρωνικό Δίκτυο παρουσιάζει την καλύτερη απόδοση σε σχέση με τις μεθόδους Nearest Neighbor και Nearest Class Centroid. Αυτό είναι αναμενόμενο, καθώς τα νευρωνικά δίκτυα έχουν τη δυνατότητα να μαθαίνουν πολύπλοκα μη γραμμικά πρότυπα από τα δεδομένα.

### **Πλησιέστερος Γείτονας (k=1 και k=3):**

- **Ακρίβεια για k=1:** 35.39%
- **Ακρίβεια για k=3:** 33.03%
- Η απόδοση είναι αρκετά χαμηλότερη σε σύγκριση με το Νευρωνικό Δίκτυο. Η μέθοδος Nearest Neighbor βασίζεται αποκλειστικά στις αποστάσεις και δεν γενικεύει καλά σε δεδομένα υψηλής διάστασης, όπως οι εικόνες του CIFAR-10.

### **Πλησιέστερο Κέντρο Κλάσης (NCC):**

- **Ακρίβεια:** 27.74%
- Η μέθοδος Nearest Class Centroid υπολείπεται ακόμα περισσότερο. Επειδή βασίζεται στον μέσο όρο των χαρακτηριστικών κάθε κλάσης για την πρόβλεψη.