Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης
Πολυτεχνική Σχολή
Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών

ΝΕΥΡΩΝΙΚΑ ΔΙΚΤΥΑ ΒΑΘΙΑ ΜΑΘΗΣΗ

Εργασία στα CNN και MLP

Κουκουλέτσου Αικατερίνη 10218

Νοέμβριος 2024

Εισαγωγή

Η παρούσα εργασία παρουσιάζει τη δημιουργία ενός μοντέλου νευρωνικού δικτύου από το μηδέν αλλά και με την χρήση builtin συναρτήσεων της TensorFlow. Δημιουργήθηκαν δύο διαφορετικά μοντέλα, ένα **MLP** και ένα **CNN**, προκειμένου να αντιμετωπιστεί το πρόβλημα της ταξινόμησης εικόνας για το CIFAR-10 dataset. Ωστόσο, η ανάλυση επικεντρώθηκε τελικά στο συνελικτικό νευρωνικό δίκτυο (CNN), καθώς, όπως αναμενόταν, αποδείχθηκε να είναι πιο κατάλληλο για την επίλυση προβλημάτων **image classification**.

Στο συνελικτικό δίκτυο, αναπτύχθηκαν δύο διαφορετικές υλοποιήσεις: η πρώτη είναι υλοποίηση από το μηδέν, η οποία έδωσε την ευκαιρία για μια εις βάθος κατανόηση της λειτουργίας του CNN, πράγμα απαραίτητο για την προσπάθεια βελτιστοποίησης του, που είναι άλλωστε και ο στόχος. Η δεύτερη είναι η υλοποίηση με χρήση της TensorFlow η οποία επέτρεψε την ταχύτερη εκπαίδευση του μοντέλου δίνωντας έτσι την ευκαιρία για πολλές περισσότερες πειραματικές δοκιμές και συνδυασμούς μεθόδων, τόσο στο κομμάτι της προεπεξεργασίας των δεδομένων όσο και στο κομμάτι της ρύθμισης των υπερπαραμέτρων του δικτύου.

Προεπεξεργασία Δεδομένων

Για την προεπεξεργασία των δεδομένων εφαρμόστηκε κανονικοποίηση και reshaping. Δεν χρειάστηκε να γίνει διαχωρισμός του dataset, καθώς το CIFAR-10 παρέχει έτοιμα τα σύνολα εκπαίδευσης και ελέγχου. Αναλυτικότερες πληροφορίες για τα προεπεξεργαστικά βήματα παρατίθενται στην προηγούμενη **αναφορά της ενδιάμεσης εργασίας** και παραλείπονται εδώ για αποφυγή επαναληψιμότητας.

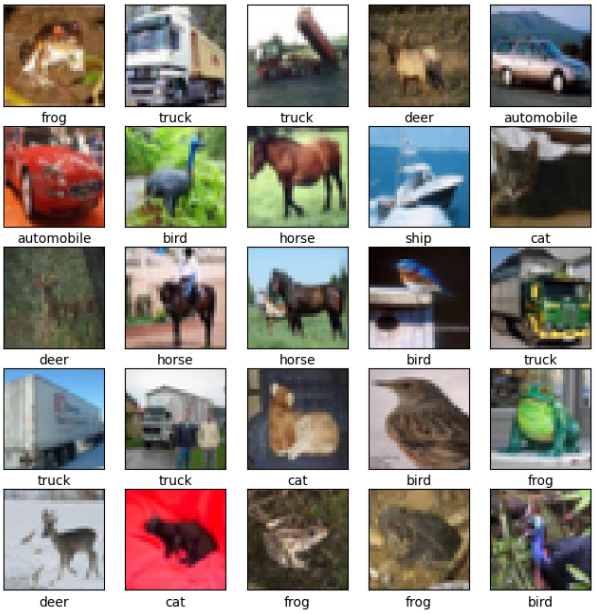
One - Hot Encoding

Σημαντική προσθήκη ωστόσο αυτήν την φορά είναι η χρήση της συνάρτησης to_categorical που μετατρέπει τις κατηγορίες σε μορφή one-hot encoding. Κάθε κατηγορία αναπαρίσταται πλέον από έναν διάνυσμα με 0 και 1, όπου το στοιχείο που αντιστοιχεί στην κατηγορία είναι 1 και όλα τα υπόλοιπα είναι 0.

Η χρήση του one-hot encoding διευκολύνει τον μαθηματικό χειρισμό των ετικετών (labels), καθώς αυτές μετατρέπονται σε διανύσματα. Επιπλέον, εξασφαλίζεται η αποσύζευξη των κατηγοριών καθώς με το one-hot encoding, κάθε κατηγορία αντιμετωπίζεται ως ανεξάρτητη, χωρίς να υπάρχει κάποια ιεραρχική ή αριθμητική σχέση μεταξύ τους. Σε περιπτώσεις όπως αυτή του CIFAR-10 αυτό είναι ένα σημαντικό προεπεξεργαστικό βήμα καθώς οι κατηγορίες δεν έχουν καμία εννοιολογική ιεραρχία (ordinal relationship).

Data Visualization

Αρχικά, πραγματοποιείται μια οπτικοποίηση του dataset με στόχο την κατανόηση των δεδομένων και άρα τον καλύτερο σχεδιασμό του μοντέλου, πιο στοχευμένο data preprocessing και καλύτερο hyperparameter tuning.



Προς αυτή την κατεύθυνση, πραγματοποιείται έλεγχος για να επιβεβαιωθεί ότι το dataset είναι ισορροπημένο (balanced). Αν και είναι γνωστό ότι το CIFAR-10 είναι εγγενώς ισορροπημένο, ο έλεγχος πραγματοποιείται για λόγους πληρότητας και αυστηρότητας της ανάλυσης.

Διόρθωση για την ενδιάμεση εργασία

Συνειδητοποιώ τώρα, που σκέφτηκα να κάνω για πρώτη φορά οπτικοποίηση του dataset, ότι η παράλειψη αυτού του βήματος στην πρώτη εργασία ήταν ένα σημαντικό λάθος. Με την οπτικοποίηση που έκανα, κατάλαβα αμέσως γιατί η προσπάθεια για edge detection με Sobel και Canny δεν είχε κανένα θετικό αποτέλεσμα.

Από τις παρακάτω εικόνες είναι ξεκάθαρο ότι τα δεδομένα στο CIFAR-10, είναι εικόνες χαμηλής ανάλυσης. Η χαμηλή ανάλυση αυτή περιορίζει ήδη τις λεπτομέρειες στις ακμές, ενώ η εφαρμογή edge detection αφαιρεί περαιτέρω πληροφορίες που είναι κρίσιμες για τη διάκριση των κατηγοριών. Από την άλλη, το CIFAR-10 βασίζεται κατά κύριο λόγο στις πληροφορίες χρώματος και στα γενικά μοτίβα των αντικειμένων, στοιχεία που χάνονται κατά την ανίχνευση ακμών.

Μετρικές Αξιολόγησης

1. Confusion Matrix

Confusion Matrix$[i, j]$ = ο αριθμός των δειγμάτων της κατηγορίας i
που ταξινομήθηκαν στην κατηγορία j

1. **Διαγώνια στοιχεία**: Τα διαγώνια στοιχεία του πίνακα δείχνουν τον αριθμό των σωστών προβλέψεων του μοντέλου για κάθε κατηγορία.
2. **Off-diagonal elements**: Οι τιμές που δεν βρίσκονται στη διαγώνιο δείχνουν λανθασμένες προβλέψεις.

		Predicted Condition	
	Total Population = P + N	Positive(PP)	Negative(PN)
Actual Condition	Positive(P)	True Positive(TP)	False Negative(FN)
	Negative(N)	False Positive(FP)	True Negative(TN)

2. Accuracy

Υπολογίζεται ως η αναλογία των σωστών προβλέψεων προς τον συνολικό αριθμό των παραδειγμάτων. Στην εργασία υπολογίζεται το training accuracy και το validation accuracy ξεχωριστά.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

3. Learning Curves

Learning Curves ή Καμπύλες Εκμάθησης, χρησιμοποιούνται για την παρακολούθηση της απόδοσης του μοντέλου κατά τη διάρκεια της εκπαίδευσης. Παρέχουν μια εικόνα για την πρόοδο της ακρίβειας και του σφάλματος τόσο στο training όσο και στο validation set. Χρήσιμες για τον εντοπισμό προβλημάτων overfitting και underfitting που θα φανούν παρακάτω. Για κάθε μοντέλο παραντίθεται τα ακόλουθα διαγράμματα

1. Training Accuracy vs. Epochs
2. Validation Accuracy vs. Epochs
3. Training Loss vs. Epochs
4. Validation Loss vs. Epochs

Convolutional Neural Network - CNN

Αυτή η ενότητα χωρίζεται σε δύο μέρη. Στο πρώτο μέρος, παρουσιάζεται η υλοποίηση ενός συνελικτικού νευρωνικού δικτύου (CNN) από το μηδέν, αναλύοντας όλους τους αλγόριθμους που απαιτούνται για τη λειτουργία του δικτύου.

Στο δεύτερο μέρος, χρησιμοποιείται η TensorFlow για τον ορισμό του CNN. Εδώ, το κύριο ενδιαφέρον μετατοπίζεται από τους αλγορίθμους που χτίζουν ένα CNN, στις βελτιστοποιήσεις όσον αφορά την αρχιτεκτονική των hidden layers, την γενικότερη ρύθμιση των υπερπαραμέτρων και την χρήση άλλων τεχνικών, με στόχο τη βελτίωση της απόδοσης του μοντέλου. Η ανάλυση εστιάζει πλέον στο πώς πραγματοποιείται ένα σωστό tuning των υπερπαραμέτρων, ποια είναι τα προεπεξεργαστικά βήματα που παίζουν καθοριστικό ρόλο και ποιές ήταν τελικά οι επιλογές που έδωσαν το βέλτιστο accuracy.

Υλοποίηση του CNN

Συνάρτηση Convolve

Πρόκειται για μια συνάρτηση η οποία εφαρμόζει το zero padding και το kernel σε ολη την εικονα. Το zero padding αναφέρεται στην επέκταση της εικόνας με προσθήκη μηδενικών στις άκρες, έτσι ώστε να είναι δυνατή η εφαρμογή του kernel και στις άκρες της εικόνας, εξασφαλίζοντας μια ομοιόμορφη εφαρμογή του σε όλη την περιοχή της εικόνας. Η παράμετρος stride, δηλαδή το βήμα μετατόπισης του, ορίστηκε ίσο με την μονάδα. Το αποτέλεσμα είναι τελικά το Feature Map το οποίο και επιστρέφεται από την συνάρτηση

$$\text{Feature Map Height} = \left\lfloor \frac{\text{Image Height} + 2 \cdot \text{Padding} - \text{Kernel Height}}{\text{Stride}} \right\rfloor + 1 = \left\lfloor \frac{32 - 3}{1} \right\rfloor + 1 = 30$$

$$\text{Feature Map Width} = \left\lfloor \frac{\text{Image Width} + 2 \cdot \text{Padding} - \text{Kernel Width}}{\text{Stride}} \right\rfloor + 1 = \left\lfloor \frac{32 - 3}{1} \right\rfloor + 1 = 30$$

όπου το padding έχει την τιμή μηδέν και το μέγεθος του υπολογίζεται σύμφωνα με τον τύπο

$$\text{Padding} = \left\lfloor \frac{\text{Kernel Size} - 1}{2} \right\rfloor = \left\lfloor \frac{3 - 1}{2} \right\rfloor = 1$$

Activation Functions

Ως συναρτήσεις ενεργοποίησης χρησιμοποιούνται οι ReLU στα hidden layers και η softmax στο output layer.

1. Rectified Linear Unit - ReLU:

$$\text{ReLU}(x) = \max(0, x)$$

2. softmax:

$$\text{softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{n} e^{z_j}}$$

Υλοποίηση τεχνικής Max Pooling

Το MaxPooling μειώνει τις διαστάσεις του feature map και εξάγει τα σημαντικότερα χαρακτηριστικά. Στην περίπτωση ενός feature map διαστάσεων 30×30, εφαρμόζεται max pooling με region 4×4, όπου από κάθε περιοχή κρατείται η μέγιστη τιμή. Το αποτέλεσμα είναι ένας πίνακας διαστάσεων 7×7, σύμφωνα με τον σχετικό τύπο.

$$\text{Output Size} = \left\lfloor \frac{\text{feature map height}}{\text{pool size}} \times \frac{\text{feature map width}}{\text{pool size}} \right\rfloor = \left\lfloor \frac{30}{4} \right\rfloor \times \left\lfloor \frac{30}{4} \right\rfloor = \lfloor 7.5 \rfloor \times \lfloor 7.5 \rfloor = 7 \times 7$$

Dense Layers

Ακολούθησε η υλοποίηση του Dense -πλήρως συνδεδεμένου- Layer. Η συνάρτηση υπολογίζει το weighted sum των εισόδων για κάθε νευρώνα σύμφωνα με τον τύπο:

$$\text{z} = \text{weights} \cdot \text{inputs} + \text{bias}$$

και επιλέγει την ReLU σαν συνάρτηση ενεργοποίησης αν πρόκειται για hidden layer ή την softmax αν πρόκειται για το τελευταίο, output layer.

Categorical Cross - Entropy

Υπολογίζει την απώλεια μεταξύ των πραγματικών τιμών (y_{true}) και των προβλεπόμενων πιθανοτήτων (y_{pred}). Η συνάρτηση βασίζεται στην ιδέα του logarithmic scoring, με σκοπό να τιμωρήσει περισσότερο τις λανθασμένες προβλέψεις για τις κατηγορίες με υψηλή πραγματική πιθανότητα.

$$\text{Loss} = -\sum_{i=1}^{C} y_i \cdot \log(\hat{y}_i + \epsilon)$$

Backpropagation

Ηταν με διαφορά η πιο δύσκολη υλοποίηση από όλο το νευρωνικό. Χωρίζεται σε δύο συναρτήσεις στην **backprop_dense** και στην backprop_conv. Στην πρώτη, γίνεται ο υπολογισμός των τεσσάρων gradients που φαίνονται παρακάτω.

1. 1. Gradient του Output Layer (με softmax και Cross-Entropy loss):

$$\mathbf{g}_{\text{output}} = \frac{\partial \mathcal{L}}{\partial \mathbf{y}_{\text{pred}}} = \mathbf{y}_{\text{pred}} - \mathbf{y}_{\text{true}}$$

   όπου $\mathcal{L}$ είναι το loss function, και με $\mathbf{y}_{\text{pred}}$ συμβολίζεται η έξοδος μετά την εφαρμογή της softmax.

2. 2. Gradient ως προς τα βάρη: υπολογίζεται από το εξωτερικό γινόμενο του gradient του output layer και της τιμής που δίνει η συνάρτηση ενεργοποίησης του προηγούμενου layer, δηλαδή της εξόδου μετά την εφαρμογή της ReLU.

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}} = \mathbf{g}_{\text{output}} \cdot \mathbf{a}^T$$

   όπου το $\mathbf{a}^T$ είναι το διάνυσμα με τις τιμές από το προηγούμενο layer μετά την εφαρμογή της ReLU.

3. Gradient ως προς το Bias: Ίδιο με το gradient ως προς το output:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{b}} = \mathbf{g}_{\text{output}}$$

4. Gradient ως προς την ενεργοποίηση του προηγούμενου layer:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{a}} = \mathbf{W}^T \cdot \mathbf{g}_{\text{output}}$$

όπου $\mathbf{W}^T$ είναι το διάνυσμα των βαρών.

Ενώ στην bacprop_conv γίνονται οι ακόλουθοι υπολογισμοί.

1. Gradient του Kernel: για κάθε περιοχή του feature map, υπολογίζεται το πώς το αντίστοιχο μέρος του kernel πρέπει να προσαρμοστεί ώστε να μειώσει την απώλεια. Αυτό γίνεται με το άθροισμα των gradients των εξόδων του μοντέλου σε σχέση με το kernel και την περιοχή του εισερχόμενου σήματος (input region).

$$\frac{\partial L}{\partial K} = \sum_{i,j,k} \left(\delta_{ij} \cdot \text{region}_{ij} \right)$$

2. Gradient για το image: Ο υπολογισμός του gradient για την εικόνα γίνεται από το error του feature map που προκύπτει από την εφαρμογή του kernel και την ανάλυση των συντελεστών του gradient του kernel.

$$\frac{\partial L}{\partial I} = \sum_{i,j} K_{ij} \cdot \delta_{ij}$$

Υλοποίηση της train

Η συνάρτηση train_cnn υλοποιεί τη διαδικασία εκπαίδευσης. Η εκπαίδευση γίνεται μέσω μιας διαδικασίας πολλαπλών εποχών (epochs), όπου για κάθε εικόνα στο σύνολο εκπαίδευσης (x_train), πραγματοποιούνται τα εξής βήματα: πρώτα εκτελείται η διαδικασία εφαρμογής του kernel, ακολουθεί η εφαρμογή της συνάρτησης ενεργοποίησης ReLU, και έπειτα γίνεται μείωση διάστασης με την max pooling. Στη συνέχεια, τα δεδομένα περνούν μέσα από το dense layer και τελικά μέσα από το output layer όπου η συνάρτηση ενεργοποίσης είναι πλέον η softmax. Κατά τη διάρκεια της εκπαίδευσης, υπολογίζεται η απώλεια (loss) με τη μέθοδο cross-entropy και ακολουθεί η διαδικασία του backpropagation για την ενημέρωση των βαρών και των παραμέτρων του δικτύου, χρησιμοποιώντας τη μέθοδο του gradient descent. Οι παράμετροι του μοντέλου (όπως τα βάρη του φίλτρου και τα βάρη του πλήρους συνδεδεμένου στρώματος) ενημερώνονται σε κάθε εποχή με βάση την τιμή του learning rate.

Μερικές σημειώσεις, το νευρωνικό που υλοποιήθηκε χρησιμοποιεί gradient descent και όχι κάποιον άλλον αλγόριθμο όπως Adam, SGD. Επίσης έχει ένα μόναδικό hidden layer και όχι μεταβλητό learning rate. Όλες αυτές οι επιλογές έγιναν με σκοπό την μείωση του υπολογιστικού κόστους καθώς θέλει ήδη.

Υλοποίηση του CNN με χρήση TensorFlow

Καθ' όλη τη διάρκεια της διεκπεραίωσης της εργασίας, καταβλήθηκε προσπάθεια να εφαρμοστούν όσες περισσότερες τεχνικές και μέθοδοι ήταν δυνατόν. Συγκεκριμένα, δοκιμάστηκαν οι παρακάτω τεχνικές:

1. Η AveragePooling απορρίφθηκε: Η τεχνική του AveragePooling απορρίφθηκε γιατί δεν βοήθησε στην επίλυση του συγκεκριμένου προβλήματος με το CIFAR-10 dataset. Αντιθέτως, προτιμήθηκε και χρησιμοποιήθηκε εκτενώς η MaxPooling η οποία λειτούργησε σαν ένας τρόπος feature selection.

2. Data Augmentation: Η τεχνική αυτή απέδωσε πολύ καλά αποτελέσματα, καθώς βοήθησε να γίνει το μοντέλο πιο ανθεκτικό στις μεταβολές των δεδομένων, βοηθώντας στη βελτίωση της γενίκευσης. Αν και η βελτίωση στην ακρίβεια ήταν μικρή, το augmentation βοήθησε στην εξομάλυνση των learning curves.

3. Learning Rate Scheduling: Το learning rate scheduling επίσης απέφερε θετικά αποτελέσματα. Η σταδιακή μείωση του learning rate έκανε τα learning curves πιο ομαλά και βοήθησε στη βελτίωση της ακρίβειας, αν και η βελτίωση ήταν και πάλι περιορισμένη. Παρόλα αυτά, το learning rate scheduling αποδείχθηκε χρήσιμο για τη σταθερότητα κατά τη διάρκεια της εκπαίδευσης.

4. Stochastic Gradient Descent (SGD): Ο Stochastic Gradient Descent, αν και χρησιμοποιείται συχνά σε deep learning μοντέλα, χειροτέρεψε τα αποτελέσματα. Γι' αυτό το λόγο, επιλέχθηκε το Adam optimizer.

5. Color Space Transformation σε LAB: Η μετατροπή του χρωματικού χώρου σε LAB χρησιμοποιήθηκε λόγω της καλής απόδοσης που είχε προσφέρει σε προηγούμενη εργασία με ταξινομητές. Ωστόσο, σε αυτή την περίπτωση, η μετατροπή αυτή δεν είχε ιδιαίτερη επίδραση, αν και βελτίωσε ελαφρώς το accuracy.

6. Relu και Softmax ως activation functions και categorical crossentropy για το Loss: δεν πειράχτηκαν, καθώς είναι οι ιδανικές επιλογές για προβλήματα πολυκατηγοριών στην αναγνώριση εικόνας. Η ReLU επιτρέπει το μοντέλο να μάθει μη γραμμικές σχέσεις χωρίς να αντιμετωπίζει προβλήματα με την εξαφάνιση του gradient, ενώ η Softmax είναι κατάλληλη για την κατηγοριοποίηση σε πολλαπλές κλάσεις. Επίσης, η χρήση του categorical crossentropy ως συνάρτησης απώλειας ήταν η κατάλληλη για το συγκεκριμένο πρόβλημα multiclassification.

7. Dropout: Η χρήση του Dropout ήταν ιδιαίτερα χρήσιμη σε περιπτώσεις όπου τα hidden layers ήταν πυκνά και περιείχαν πολλούς νευρώνες. Αυτή η τεχνική βοήθησε στην αποφυγή υπερεκπαίδευσης (overfitting) και βελτίωσε τη γενίκευση του μοντέλου.

8. Early Stopping: Το Early Stopping χρησιμοποιήθηκε για να αποφευχθεί η υπερεκπαίδευση, παρακολουθώντας την απόδοση του μοντέλου σε ένα validation set. Εάν η απόδοση δεν βελτιωνόταν για αρκετές συνεχόμενες εποχές, η εκπαίδευση σταματούσε πρόωρα, αποτρέποντας την υπερβολική προσαρμογή στο training set.

9. Batch Size: Ο αριθμός του batch size επίσης είχε σημαντική επίδραση στα δεδομένα. Αλλαγές στο μέγεθος του batch επηρέασαν τη σταθερότητα των gradients και την απόδοση του μοντέλου. Χρησιμοποιήθηκαν διάφοροι πειραματικοί αριθμοί για να επιτευχθεί η καλύτερη απόδοση.

10. Batch Normalization: Αν και το Batch Normalization γενικά βοηθά στη βελτίωση της απόδοσης των μοντέλων, προκαλούσε απότομες κορυφές (spikes) στα learning curves για το συγκεκριμένο μοντέλο και dataset. Επομένως, αποφασίστηκε να μην χρησιμοποιηθεί, καθώς δεν παρείχε και εμφανή πλεονεκτήματα.

11. Epoch Number: Ο αριθμός των εποχών χρησιμοποιήθηκε ως εργαλείο για την παρακολούθηση του overfitting και του underfitting. Ο ιδανικός αριθμός των εποχών επιλέχθηκε έτσι ώστε το μοντέλο να εκπαιδευτεί επαρκώς, αλλά να μην υπερεκπαιδευτεί.

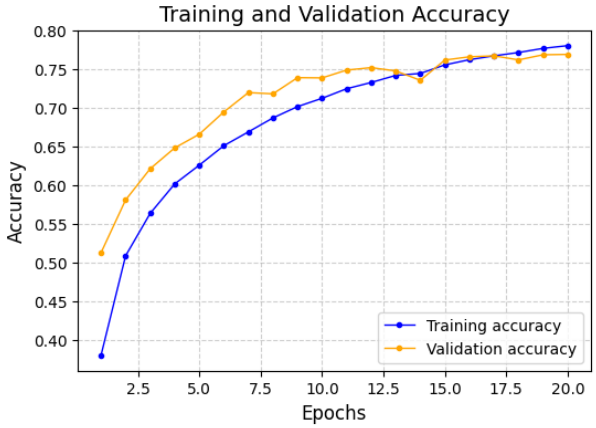
Ωστόσο, σύντομα έγινε κατανοητό ότι, αν και τεχνικές όπως η προεπεξεργασία δεδομένων, η χρήση μεθόδων όπως το SGD, η εφαρμογή learning scheduling, Batch Normalization και άλλες, μπορούν να συμβάλλουν στη βελτίωση της απόδοσης του μοντέλου, οι βελτιώσεις που επιφέρουν είναι περιορισμένες όταν η αρχιτεκτονική του δικτύου παραμένει αμετάβλητη. Με άλλα λόγια, οι παραπάνω τεχνικές μπορούν να ενισχύσουν την αποτελεσματικότητα του μοντέλου, αλλά δεν είναι ικανές να επιφέρουν θεμελιώδεις αλλαγές στην απόδοση από μόνες τους. Για το λόγο αυτό, η προσοχή στράφηκε στην ανασχεδίαση και βελτίωση της ίδιας της αρχιτεκτονικής του νευρωνικού δικτύου, με έμφαση στη δομή και τη διαμόρφωση των hidden layers.

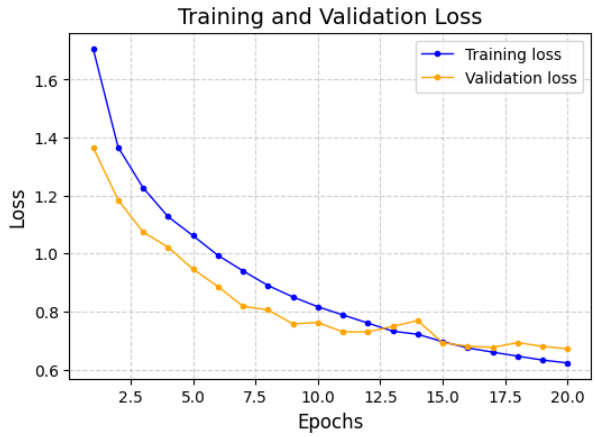
Η δομή των νευρωνικών δικτύων, βασίστηκε στην ιδέα του **hierarchical feature learning**, η οποία προέρχεται από τον τρόπο που ο ανθρώπινος εγκέφαλος επεξεργάζεται τις πληροφορίες. Στην αρχή, τα χαμηλότερα επίπεδα του δικτύου ανιχνεύουν βασικά, τοπικά χαρακτηριστικά των εικόνων, τα οποία έχουν μικρότερη διάσταση και πιο απλή μορφή. Καθώς προχωράει στα hidden layers, ο αριθμός των νευρώνων αυξάνεται, επιτρέποντας στο δίκτυο να επεξεργαστεί και να συνδυάσει αυτές τις τοπικές πληροφορίες για να εξάγει πιο αφηρημένα και σύνθετα χαρακτηριστικά. Αυτή η στρατηγική προσομοιώσει αρκετά τη διαδικασία μάθησης του ανθρώπινου εγκεφάλου.

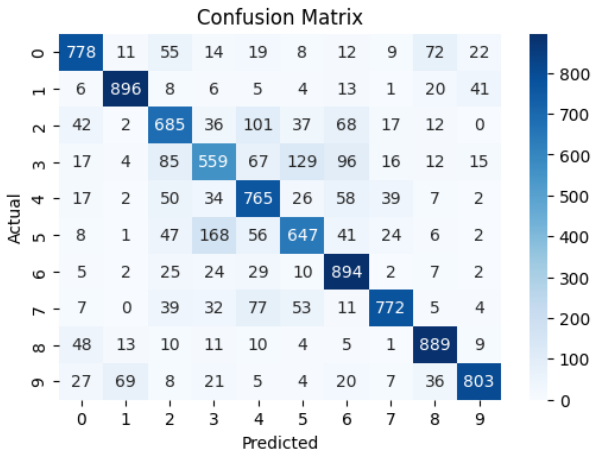
Επομένως, δοκιμάστηκε η σταδιακή αύξηση του αριθμού των νευρώνων στα hidden layers, φτάνοντας μέχρι το μέσο ή το τέλος του νευρωνικού δικτύου, όπου και οι νευρώνες πρέπει να είναι ίσοι με 10, αντίστοιχοι με τον αριθμό των κλάσεων. Αυτή η προσέγγιση απέδωσε πραγματικά πολύ καλά αποτελέσματα.

Μοντέλο CNN

Για το μοντέλο CNN, το test accuracy είναι ίσο με 76.88%. Και ο συνολικός χρόνος για 20 epochs είναι 643.2099080085754 seconds. Παρατίθενται οι καμπύλες εκμάθησης, το confusion matrix και το συνολικό ποσοστό σωστής κατηγοριοποίησης ανά κλάση.



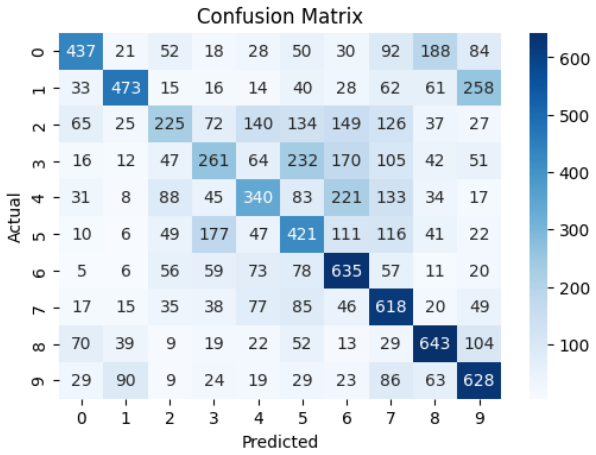




Class	Classification Success Rate (%)
Class 0	77.80
Class 1	89.60
Class 2	68.50
Class 3	55.90
Class 4	76.50
Class 5	64.70
Class 6	89.40
Class 7	77.20
Class 8	88.90
Class 9	80.30

Μοντέλο MLP

Για το μοντέλο MLP, το test accuracy είναι 46.81%. Και ο συνολικός χρόνος εκτέλεσης είναι 12 λεπτά για 20 εποχές. Παρατίθενται τα ζητούμενα διαγράμματα.



Class	**Classification Success Rate (%)**
0	77.80
1	89.60
2	68.50
3	55.90
4	76.50
5	64.70
6	89.40
7	77.20
8	88.90
9	80.30

Σύγκριση με αλγορίθμους Ταξινόμησης

Ο συγκριτικός πίνακας απόδοσης των διάφορων μεθόδων ταξινόμησης δείχνει σαφείς διαφορές στην αποδοτικότητα τους. Συγκεκριμένα, το k-Nearest Neighbors (k-NN) με $k = 1$ παρουσίασε απόδοση 35%, το $k = 3$ είχε ελαφρώς χαμηλότερη απόδοση 31%, ενώ η μέθοδος "Nearest Centroid" παρουσίασε την πιο χαμηλή απόδοση με 27%. Αυτές οι μέθοδοι ταξινόμησης είχαν σημαντικά χαμηλότερες επιδόσεις, πιθανόν λόγω της απλότητας τους και της έλλειψης ικανότητας να επεξεργάζονται πιο περίπλοκες και μη γραμμικές σχέσεις μεταξύ των χαρακτηριστικών των δεδομένων. Αντίθετα, τα πιο σύγχρονα μοντέλα όπως το Multilayer Perceptron (MLP) και το Convolutional Neural Network (CNN) παρουσίασαν πολύ υψηλότερες επιδόσεις, με το MLP να φτάνει το 46.81% και το CNN να σημειώνει εντυπωσιακό ποσοστό 76.88%. Η υψηλότερη απόδοση του CNN οφείλεται στη ικανότητά του να εξάγει και να αναγνωρίζει χαρακτηριστικά από εικόνες μέσω των convolutional layers, ενώ το MLP κατάφερε να εκμεταλλευτεί την πολυπλοκότητα των δεδομένων μέσω των πολλαπλών layers του, επιτυγχάνοντας επίσης πολύ καλύτερα αποτελέσματα σε σχέση με τις παραδοσιακές μεθόδους.