Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης
Πολυτεχνική Σχολή
Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών

# ΝΕΥΡΩΝΙΚΑ ΔΙΚΤΥΑ ΒΑΘΙΑ ΜΑΘΗΣΗ

## Radial Basis Functions Neural Networks and Autoencoders

Κουκουλέτσου Αικατερίνη 10218

Ιανουάριος 2025

# Εισαγωγή

Η παρούσα εργασία αποτελείται από δύο μέρη. Στο πρώτο μέρος, υλοποιείται ένα Radial Basis Function Neural Network, το οποίο χρησιμοποιείται για την επίλυση του προβλήματος του multiclass classification στο σύνολο δεδομένων CIFAR-10. Παρουσιάζονται δύο μοντέλα, το βέλτιστο RBFNN που προέκυψε έπειτα από ρύθμιση των υπερπαραμέτρων και ένας συνδυασμός του CNN της πρώτης εργασίας και του RBFNN, όπου το CNN χρησιμοποιείται ως μηχανισμός feature extraction.

Με τις δύο αυτές υλοποιήσεις ολοκληρώνεται και η μελέτη της ταξινόμησης των εικόνων στο σύνολο δεδομένων CIFAR-10. Το RBFNN αποτελεί την τελευταία δοκιμή, μετά το CNN, MLP, SVM, SVM με CNN πάλι ως μηχανισμό feature extraction καθώς και οι αλγόριθμοι kNN και NCC. Στην εργασία παρουσιάζονται τα αποτελέσματα που προέκυψαν από την μελέτη καθώς και τα συμπεράσματα που εξάγονται για κάθε τύπο νευρωνικού δικτύου από την απόδοση τους στο CIFAR10.

Στο δεύτερο και τελευταίο μέρος της εργασίας, υλοποιείται ένας autoencoder στο σύνολο δεδομένων MNIST. Στόχος είναι να μειώσει τη διάσταση των δεδομένων μέσω ενός encoder και στη συνέχεια να ανακατασκευάσει τα δεδομένα χρησιμοποιώντας έναν decoder. Το δίκτυο αξιολογείται με μετρικές όπως το PSNR (Peak Signal-to-Noise Ratio) και το SSIM (Structural Similarity Index), ενώ παρέχονται παραδείγματα των πρωτότυπων και ανακατασκευασμένων εικόνων για οπτική σύγκριση. Τέλος χρησιμοποιείται ένα CNN για digit recognition των ανακατασκευασμένων εικόνων.

# 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. Χρησιμοποιήθηκε ο `StandardScaler` για την κανονικοποίηση των δεδομένων, εφαρμόστηκε `PCA` με 90% και οι ετικέτες μετασχηματίστηκαν με την τεχνική `OneHotEncoder`.

# Μετρικές Αξιολόγησης

Οι Μετρικές Αξιολόγησης που χρησιμοποιούνται στην παρούσα εργασία περιλαμβάνουν το Confusion Matrix, το Accuracy Score και Classification Report. Αυτές οι μετρικές έχουν ήδη αναλυθεί λεπτομερώς στις εργασίες των δύο προηγούμενων αναφορών. Για λόγους αποφυγής επανάληψης, η αναλυτική παρουσίασή τους παραλείπεται από την παρούσα αναφορά.

# Radial Basis Function Neural Network

## Εύρεση των κέντρων των Clusters

Χρησιμοποιείται το K-Means για να ομαδοποιηθούν τα δεδομένα και να δημιουργηθούν τα κέντρα των RBF νευρώνων στον κρυφό επίπεδο

## Επιλογή RBF Kernel

Καθοριστικό ρόλο έχει και η επιλογή του kernel function που θα χρησιμοποιηθεί για την ενεργοποίηση. Παρουσιάζονται οι διάφοροι τύποι RBF που χρησιμοποιήθηκαν στην εργασία:

1. **Γκαουσιανή (Gaussian)**: Η πιο κοινή RBF, χρησιμοποιεί την εκθετική συνάρτηση με αρνητική τετραγωνική απόσταση.

$$\phi(\alpha) = \exp(-\alpha^2)$$

$$K(x, c_i, \sigma) = \exp\left(-\frac{\|x - c_i\|^2}{\sigma^2}\right)$$

   όπου $\alpha$ είναι η απόσταση μεταξύ των δεδομένων εισόδου και του πυρήνα.

2. **Γραμμική (Linear)**: Είναι χρήσιμη σε περιπτώσεις που οι διακυμάνσεις των δεδομένων είναι μικρές.

$$\phi(\alpha) = \alpha$$

$$K(x, c_i) = \|x - c_i\|$$

3. **Τετραγωνική (Quadratic)**: Η τετραγωνική συνάρτηση είναι μια εκδοχή της γραμμικής όμως πιο ευαίσθητη σε μεγάλες αποστάσεις λόγω της ύψωσης στο τετράγωνο.

$$\phi(\alpha) = \alpha^2$$

$$K(x, c_i) = \|x - c_i\|^2$$

4. **Αντίστροφη Τετραγωνική (Inverse Quadratic)**: Η αντίστροφη τετραγωνική συνάρτηση μειώνει την απόκριση ανάλογα με το τετράγωνο της απόστασης και άρα παρέχει ισχυρότερη επίδραση για κοντινά σημεία και πιο εξασθενημένη για τα απομακρυσμένα σημεία.

$$\phi(\alpha) = \frac{1}{1 + \alpha^2}$$

$$K(x, c_i) = \frac{1}{1 + \|x - c_i\|^2}$$

5. **Πολυτετραγωνική (Multiquadric)**: Ισχυρή εκθετική αύξηση στην αρχή και στη συνέχεια μειώνεται αργά. Είναι χρήσιμη σε περιπτώσεις που είναι επιθυμητή μια ομαλή προσαρμογή σε πιο απομακρυσμένα σημεία.

$$\phi(\alpha) = \sqrt{1 + \alpha^2}$$

$$K(x, c_i) = \sqrt{1 + \|x - c_i\|^2}$$

6. **Αντίστροφη Πολυτετραγωνική (Inverse Multiquadric)**: Η αντίστροφη πολυτετραγωνική είναι παρόμοια με την πολυτετραγωνική, αλλά η απόκριση της μειώνεται αργά με την αύξηση της απόστασης.

$$\phi(\alpha) = \frac{1}{\sqrt{1+\alpha^2}}$$
$$K(x, c_i) = \frac{1}{\sqrt{1+\|x-c_i\|^2}}$$

7. **Matern-32** ($Matern_{3/2}$): Η συνάρτηση Matern για το μοντέλο $\nu = 3/2$ προσφέρει ένα ενδιάμεσο αποτέλεσμα μεταξύ της εκθετικής και της Γκαουσιανής συνάρτησης. Χρησιμοποιείται συχνά σε εφαρμογές αναγνώρισης προτύπων.

$$\phi(\alpha) = \left(1+\sqrt{3}\alpha\right)\exp\left(-\sqrt{3}\alpha\right)$$
$$K(x, c_i) = \left(1+\sqrt{3}\|x-c_i\|\right)\exp\left(-\sqrt{3}\|x-c_i\|\right)$$

8. **Matern-52** ($Matern_{5/2}$): Παρόμοια με το $Matern_{3/2}$, αλλά πιο ευαίσθητη στην απόσταση. Χρησιμοποιείται για πιο ακριβείς προσαρμογές δεδομένων σε περιπτώσεις που η γεωμετρία των δεδομένων είναι πιο πολύπλοκη.

$$\phi(\alpha) = \left(1+\sqrt{5}\alpha+\frac{5}{3}\alpha^2\right)\exp\left(-\sqrt{5}\alpha\right)$$
$$K(x, c_i) = \left(1+\sqrt{5}\|x-c_i\|+\frac{5}{3}\|x-c_i\|^2\right)\exp\left(-\sqrt{5}\|x-c_i\|\right)$$

Στην συνέχεια, τα βήματα προς υλοποίηση που ακολουθούνται ορίζονται ως εξής. Για κάθε δείγμα υπολογίζεται η ευκλείδια απόσταση από τα κέντρα των clusters και έπειτα εφαρμόζεται η επιλεγμένη συνάρτηση του RBF kernel. Αποθηκεύεται η έξοδος του κρυφού επιπέδου (hidden layer output).

## Υπολογισμός του Adaptive Sigma

Σε μια προσπάθεια αύξησης της απόδοσης, χρησιμοποιήθηκε η τεχνική `adaptive_sigma` που υπολογίζει τις τιμές του $\sigma$ και προσαρμόζεται κατά την διάρκεια της εκπαίδευσης. Η συνάρτηση `compute_adaptive_sigma` υπολογίζει για κάθε κέντρο των clusters τη μέση απόσταση από τα $k$ πιο κοντινά δείγματα εκπαίδευσης:

$$\sigma_i = \frac{1}{k}\sum_{j=1}^{k}\|x_j - c_i\|$$

όπου $x_j$ είναι τα δείγματα εκπαίδευσης και $c_i$ τα κέντρα.

### Υλοποίηση του RBF Kernel

Επειδή τελικά έπειτα από δοκιμές, παρατηρήθηκε ότι το kernel που έδινε τα βέλτιστα αποτελέσματα ήταν το Gaussian, παρουσιάζεται η ανάλυση λειτουργίας του kernel συγκεκριμένα για την περίπτωση που είναι τύπου Gaussian. Η συνάρτηση `rbf_kernel` υπολογίζει για κάθε δείγμα εισόδου $x$ και κάθε κέντρο του K-Means centers:

$$K(x, c_i, \sigma) = \exp\left(-\frac{\|x - c_i\|^2}{\sigma^2}\right)$$

όπου $\|x - c_i\|$ είναι η Ευκλείδεια απόσταση μεταξύ του δείγματος $x$ και του κέντρου $c_i$, και $\sigma$ είναι είτε σταθερό και ελεύθερο προς επιλογή είτε προσαρμόζεται κατά την εκπαίδευση όπως αναφέρθηκε παραπάνω.

### Υπολογισμός της εξόδου του Hidden Layer

Η συνάρτηση `hiddenlayer_output` υπολογίζει τις εξόδους του hidden layer για όλα το training set. Η έξοδος για κάθε δείγμα υπολογίζεται ως εξής:

$$h(x) = [K(x, c_1, \sigma), K(x, c_2, \sigma), \ldots, K(x, c_n, \sigma)]$$

όπου κάθε $K(x, c_i, \sigma)$ είναι η τιμή του Kernel για το δείγμα $x$ και το κέντρο $c_i$.

### Εκπαίδευση του Output Layer

Η εκπαίδευση του output layer γίνεται με την εφαρμογή της μεθόδου των ελαχίστων τετραγώνων. Για κάθε κλάση $i$, υπολογίζονται τα βάρη $W$ ως εξής:

$$W = \left(X^T X\right)^{-1} X^T y$$

Η μέθοδος αυτή χρησιμοποιεί τον ψευδοαντίστροφο του $X^T X$, ο οποίος υπολογίζεται στην πράξη με την χρήση της συνάρτησης `np.linalg.pinv()` στον κώδικα:

$$W = \texttt{np.linalg.pinv}(X).dot(y)$$

## Αποτελέσματα RBFNN

Για σταθερό sigma, Training Accuracy: 66.98

Testing Accuracy: 54.23
Και ο χρόνος εκτέλεσης ήταν 20.56s

Επίσης δοκιμάστηκε και το adaptive sigma. Σημειώνεται πως αυτός ο τρόπος έδωσε λίγο χειρότερα αποτελέσματα από ότι το σιγμα σταθερό. Επομένως δεν χρησιμοποιήθηκε τελικα. Για 5500 νευρώνες τα αποτελέσματα ήταν τα εξης

Training Accuracy: 63.15
Testing Accuracy: 53.01

| Class | Precision | Recall | F1-Score | Support |
|---|---|---|---|---|
| 0 | 0.62 | 0.63 | 0.63 | 1000 |
| 1 | 0.62 | 0.68 | 0.65 | 1000 |
| 2 | 0.46 | 0.38 | 0.41 | 1000 |
| 3 | 0.40 | 0.32 | 0.36 | 1000 |
| 4 | 0.47 | 0.42 | 0.44 | 1000 |
| 5 | 0.49 | 0.43 | 0.46 | 1000 |
| 6 | 0.51 | 0.68 | 0.58 | 1000 |
| 7 | 0.59 | 0.61 | 0.60 | 1000 |
| 8 | 0.60 | 0.69 | 0.65 | 1000 |
| 9 | 0.59 | 0.58 | 0.58 | 1000 |
| **Accuracy** | | 0.54 | | 10000 |
| **Macro avg** | 0.54 | 0.54 | 0.54 | 10000 |
| **Weighted avg** | 0.54 | 0.54 | 0.54 | 10000 |

Table 1: Classification Report

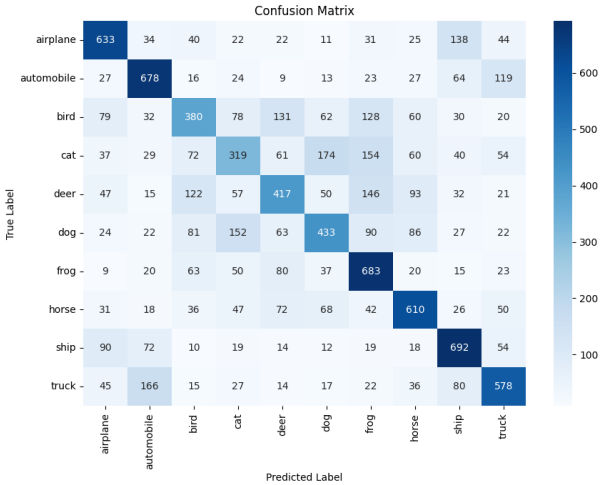
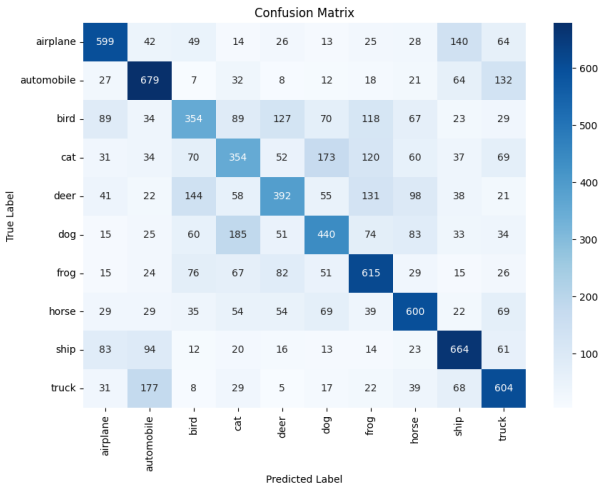



Figure 1: Caption




Και ο χρόνος εκτέλεσης 25.417 s.

| Class | Precision | Recall | F1-Score | Support |
|---|---|---|---|---|
| 0 | 0.62 | 0.60 | 0.61 | 1000 |
| 1 | 0.59 | 0.68 | 0.63 | 1000 |
| 2 | 0.43 | 0.35 | 0.39 | 1000 |
| 3 | 0.39 | 0.35 | 0.37 | 1000 |
| 4 | 0.48 | 0.39 | 0.43 | 1000 |
| 5 | 0.48 | 0.44 | 0.46 | 1000 |
| 6 | 0.52 | 0.61 | 0.57 | 1000 |
| 7 | 0.57 | 0.60 | 0.59 | 1000 |
| 8 | 0.60 | 0.66 | 0.63 | 1000 |
| 9 | 0.54 | 0.60 | 0.57 | 1000 |
| **Accuracy** | | 0.53 | | 10000 |
| **Macro avg** | 0.52 | 0.53 | 0.52 | 10000 |
| **Weighted avg** | 0.52 | 0.53 | 0.52 | 10000 |

# Ποσοστά Accuracy για πολλούς διαφορετικούς αριθμούς νευρώνων

| $n_{hidden}$ | Training Accuracy (%) | Testing Accuracy (%) |
|---|---|---|
| 100 | 45.23 | 41.50 |
| 500 | 55.12 | 49.80 |
| 1000 | 60.75 | 51.40 |
| 3000 | 63.10 | 53.00 |
| 5500 | 66.98 | 54.23 |

# Παραδείγματα Σωστής και Εσφαλμένης Κατηγοριοποίησης

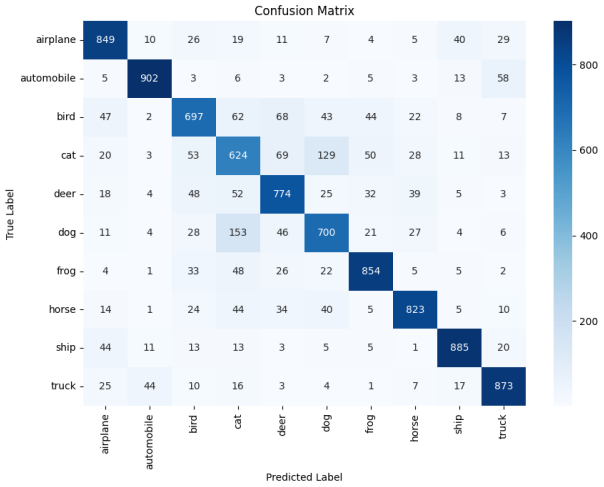
## CNN and RBFNN

Για την τελευταία προσπάθεια επίλυσης του προβλήματος ταξινόμησης εικόνων του CIFAR-10, επιλέχθηκε το βέλτιστο μοντέλο CNN που προέκυψε στην πρώτη εργασία και συνδυάστηκε με το βέλτιστο RBFNN που αναπτύχθηκε στην παρούσα εργασία. Συγκεκριμένα, το CNN χρησιμοποιήθηκε για να εκτελέσει τον ρόλο του feature extraction, εξάγοντας τα χαρακτηριστικά των εικόνων, ενώ το RBFNN ανέλαβε την ταξινόμηση των εξαγόμενων χαρακτηριστικών. Τα αποτελέσματα που προέκυψαν από αυτή τη σύνθεση ήταν εξαιρετικά, δείχνοντας σημαντική βελτίωση στην απόδοση του μοντέλου.

# Αποτελέσματα

Ο χρόνος εκτέλεσης ήταν 68 λεπτά. Πρόκειται για την καλύτερη ταξινόμηση από όλες τις εργασίες συνολικά. Training Accuracy: 96.31

Test Accuracy: 80.18



| Class | Precision | Recall | F1-Score | Support |
|---|---|---|---|---|
| Airplane (0) | 0.82 | 0.85 | 0.83 | 1000 |
| Automobile (1) | 0.92 | 0.90 | 0.91 | 1000 |
| Bird (2) | 0.75 | 0.70 | 0.72 | 1000 |
| Cat (3) | 0.60 | 0.62 | 0.61 | 1000 |
| Deer (4) | 0.75 | 0.77 | 0.76 | 1000 |
| Dog (5) | 0.72 | 0.70 | 0.71 | 1000 |
| Frog (6) | 0.84 | 0.85 | 0.85 | 1000 |
| Horse (7) | 0.86 | 0.82 | 0.84 | 1000 |
| Ship (8) | 0.89 | 0.89 | 0.89 | 1000 |
| Truck (9) | 0.86 | 0.87 | 0.86 | 1000 |
| **Accuracy** | | **0.7981** | | 10000 |
| **Macro Avg** | 0.80 | 0.80 | 0.80 | 10000 |
| **Weighted Avg** | 0.80 | 0.80 | 0.80 | 10000 |

# Παραδείγματα Σωστής και Εσφαλμένης Κατηγοριοποίησης


# Σύγκριση RBFNN με kNN και NCC και Συμπεράσματα

Η απόδοση του RBFNN στο CIFAR-10 με ποσοστό ταξινόμησης 54.23%, υπερέχει του kNN και του NCC. Αυτό είναι άλλωστε και το αναμενόμενο αποτέλεσμα αφού τα δεδομένα είναι υψηλής διάστασης κάτι το οποίο δεν βοηθάει σε αλγορίθμους ταξινόμησης όπως αυτός του $k$-NN αφού δυσχεραίνει τη διαφοροποίηση των αποστάσεων ανάμεσα σε σημεία

Αντίστοιχα, το NCC περιορίζεται από την απλοποιημένη υπόθεση ότι κάθε κατηγορία μπορεί να αναπαρασταθεί πλήρως από το μέσο σημείο της, κάτι που είναι ανεπαρκές για δεδομένα με μεγάλο intra-class variance όπως αυτό της cifar10.

Το RBFNN υπερτερεί, επειδή χρησιμοποιεί κεντρικά σημεία (clusters) που καθορίζονται μέσω K-Means, τα οποία αποτυπώνουν καλύτερα την κατανομή των δεδομένων σε κάθε κατηγορία. Επιπλέον, η χρήση RBF kernels επιτρέπει την κατασκευή μη γραμμικών ορίων απόφασης. Ειδικά με την προσαρμογή του $\sigma$ ανά cluster, το RBFNN προσαρμόζεται τοπικά στη δομή των δεδομένων, ενώ το NCC και το $k$-NN βασίζονται αποκλειστικά σε γραμμικά distance metrics.

# Autoencoder on MNIST

Αυτός ο κώδικας υλοποιεί έναν Autoencoder για image reconstruction χρησιμοποιώντας το σύνολο δεδομένων MNIST. Κατά τα προεπεξεργαστικά βήματα, τα δεδομένα κανονικοποιούνται και μετασχηματίζονται στις απαιτούμενες διαστάσεις ενώ εφαρμόζεται και data augmentation αφού έπειτα από δοκιμές παρατηρήθηκε ότι συνεισφέρει σημαντικά στην βελτίωση της απόδοσης του μοντέλου. Ο διαχωρισμός του dataset δίνεται έτοιμος από την MNIST.

Όσον αφορά τον κώδικα, αρχικά υλοποιείται ο Encoder που συμπιέζει τις εικόνες σε ένα latent space και στην συνέχεια υλοποιείται ο Decoder που ανασυνθέτει τις εικόνες πίσω στην αρχική τους μορφή. Το μοντέλο εκπαιδεύεται με απώλεια μέσου τετραγώνου σφάλματος (MSE) και αξιολογείται με την ανασύνθεση εικόνας από το test set.

Σε μια προσπάθεια βελτίωσης της απόδοσης έγινε αντιληπτό ότι χρειάζονται convolutional layers στον autoencoder επομένως κατασκευάστηκε ενα 2ο μοντέλο, ένας autoencoder με layers τέτοιου τύπου και έγινε νέο reconstruction. Πάνω σε αυτό το νέο reconstructed σετ εικόνων, χρησιμοποιήθηκε ένα CNN για digit recognition. Για τον έλεγχο και σύγκριση της απόδοσης, εξετάστηκε η αναγνώριση τόσο στο original οσο και στο reconstructed σετ εικόνων. Παρατίθενται τα αποτελέσματα.

# Υλοποίηση

Κατά τη διάρκεια της διαδικασίας ρύθμισης των υπερπαραμέτρων, παρατηρήθηκε ότι το μοντέλο αποδίδει πολύ καλά, γεγονός που είναι λογικό και συνηθισμένο για το σύνολο δεδομένων MNIST. Οι παράμετροι και οι αλλαγές που επηρέασαν σημαντικά την ρύθμιση του μοντέλου ήταν η επιλογή των layers και των νευρώνων ανά layer, καθώς και το batch size. Επίσης, ο αριθμός των εποχών είχε καθοριστική σημασία για την εκπαίδευση.

Για κάθε layer του encoder και decoder, ισχύουν οι παρακάτω κανόνες για τις διαστάσεις:

## Encoder:

1. Το Input layer έχει διαστάσεις $(batch\_size, height, width, channels)$.
   Δηλαδή $28 \cdot 28 \cdot 1 = 784$.

2. Για κάθε Dense Layer ισχύουν οι διαστάσεις $(batch\_size, neurons$, όπου το $neurons$ είναι ο αριθμός των νευρώνων του layer.

3. Ενώ για το Latent Layer ισχύουν οι διαστάσεις $(batch\_size, latent\_dim)$, όπου $latent\_dim$, η διάσταση που έχει προαποφασιστεί για το Latent Layer.

## Decoder:

1. Το Input του decoder έχει διαστάσεις $(batch\_size, latent\_dim)$.

2. Για κάθε Dense Layer του decoder έχει διαστάσεις $(batch\_size, neurons)$.

3. Το Output Layer αναδιαμορφώνει την έξοδο στις διαστάσεις του αρχικού input, δηλαδή $(batch\_size, height, width, channels)$.

### Συνάρτησεις Ενεργοποίησης

Ως συναρτήσεις ενεργοποίησης χρησιμοποιούνται οι Rectified Linear Unit και η Σιγμοειδής. Για την **ReLU**, ο μαθηματικός τύπος είναι:

$$f(x) = \max(0, x)$$

Η χρήση της ReLU σε ένα autoencoder εξυπηρετεί τους εξής σκοπούς:

1. Αποφυγή του προβλήματος του vanishing gradient
2. Δημιουργία ισχυρών μη γραμμικών μοντέλων, επιτρέποντας στον autoencoder να μάθει πιο σύνθετες αναπαραστάσεις των δεδομένων.

Για την **Sigmoid**, ο μαθηματικός τύπος της είναι:

$$f(x) = \frac{1}{1 + e^{-x}}$$

Χρησιμοποιείται στο τελικό layer του decoder για να περιορίσει τις τιμές της ανακατασκευασμένης εικόνας στο εύρος [0, 1], όπως και η αρχική εικόνα. Η χρήση της Sigmoid εξασφαλίζει ότι η έξοδος είναι κατάλληλη για την ανακατασκευή εικόνας, καθώς κάθε pixel είναι μια τιμή πιθανότητας μεταξύ 0 και 1.

### Ο Αλγόριθμος Βελτιστοποίησης Adam

Η ενημέρωση της παραμέτρου σε κάθε βήμα είναι δίνεται από τον τύπο:

$$\theta_t = \theta_{t-1} - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \cdot \hat{m}_t$$

### Μέσο Τετραγωνικό Σφάλμα (MSE)

Το Μέσο Τετραγωνικό Σφάλμα (MSE) αντιπροσωπεύει τη μέση τετραγωνική απόκλιση μεταξύ των πραγματικών και των ανασυντεθειμένων τιμών. Ο τύπος του είναι:

$$\text{MSE} = \frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2$$

όπου $y_i$ είναι η πραγματική τιμή και $\hat{y}_i$ η προβλεπόμενη τιμή για την $i$-οστή είσοδο.

### Δείκτης Δομικής Ομοιότητας (SSIM)

Ο δείκτης SSIM χρησιμοποιείται για την εκτίμηση της ομοιότητας ανάμεσα σε δύο εικόνες, λαμβάνοντας υπόψη όχι μόνο τη φωτεινότητα, αλλά και τη δομή και την υφή:

$$\text{SSIM}(x, y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)}$$

όπου $\mu_x, \mu_y$ είναι οι μέσες τιμές των εικόνων $x$ και $y$, $\sigma_x^2, \sigma_y^2$ είναι οι διακυμάνσεις και $\sigma_{xy}$ η συνδιακύμανση.

### Σχέση Σήματος προς Θόρυβο Κορυφής (PSNR)

Η Σχέση Σήματος προς Θόρυβο Κορυφής (PSNR) χρησιμοποιείται για την εκτίμηση της ποιότητας της εικόνας, συγκρίνοντας το σήμα της εικόνας με το θόρυβο. Υπολογίζεται από τον τύπο:

$$\text{PSNR} = 10 \cdot \log_{10}\left(\frac{R^2}{\text{MSE}}\right)$$

όπου $R$ είναι η μέγιστη δυνατή τιμή του pixel και MSE είναι το Μέσο Τετραγωνικό Σφάλμα.

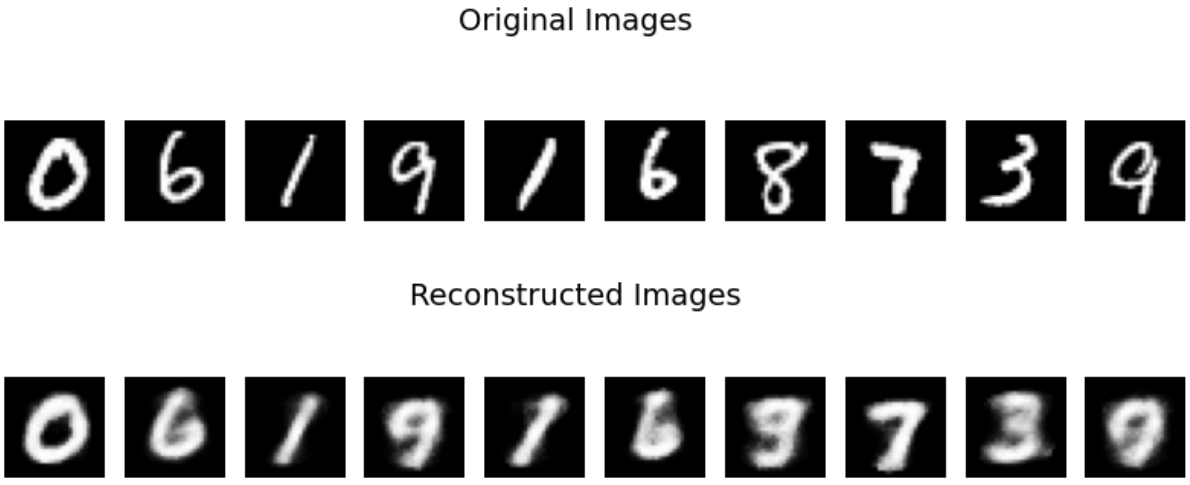
### Αποτελέσματα Πρώτου Autoencoder

Έπειτα από αρκετό tuning, παρατίθεται η βέλτιστη αρχιτεκτονική που βρέθηκε για τον Autoencoder και τα αποτελέσματα ανασύνθεσης.

Η αρχιτεκτονική του Encoder αποτελείται από πλήρως συνδεδεμένα (dense) στρώματα. Ξεκινά με την είσοδο που είναι εικόνα διαστάσεων 28x28x1 αφού το ύψος και το πλάτος των εικόνων της MNIST είναι 28x28 και το κανάλι χρώματος είναι 1 (grayscale). Η είσοδος μετατρέπεται σε μονοδιάστατο διανύσμα μέσω της συνάρτησης Flatten. Στη συνέχεια, η έξοδος του Flattened διανύσματος περνά από τρία πλήρως συνδεδεμένα στρώματα με 512, 256 και 128 νευρώνες αντίστοιχα, όλα με ενεργοποίηση ReLU. Το τελευταίο στρώμα παράγει το latent representation, δηλαδή τη συμπιεσμένη αναπαράσταση της εικόνας, με διάσταση 256.

Αυτή η συμπιεσμένη αναπαράσταση είναι η έξοδος του Encoder, η οποία στη συνέχεια χρησιμοποιείται από τον Decoder για την ανακατασκευή της αρχικής εικόνας.

Τα αποτελέσματα φαίνονται παρακάτω. Σε μια προσπάθεια βελτίωσης της ποιότητας της εικόνας χρησιμοποιήθηκαν Conv Layers και δημιουργήθηκε το 2ο μοντέλο του Autoencoder το οποιο παρουσιάζεται στην επόμενη ενότητα.



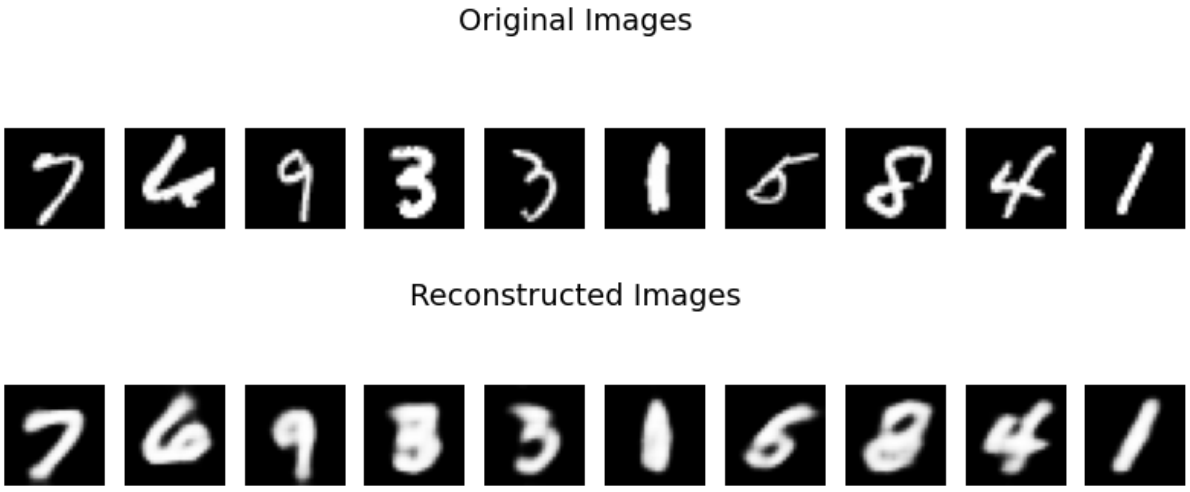

## Autoencoder με Convolutional Layers

Η αρχιτεκτονική του περιλαμβάνει δύο στρώματα convolutional layers με 32 και 64 αντίστοιχα, χρησιμοποιήθηκε και πάλι η τεχνική του max pooling για μείωση της διάστασης. Ως συνάρτηση ενεργοποίησης ορίστηκε να είναι και πάλι η ReLU.

Τα δύο τελικά πυκνά στρώματα εκτελούν την ταξινόμηση με την softmax ενεργοποίηση

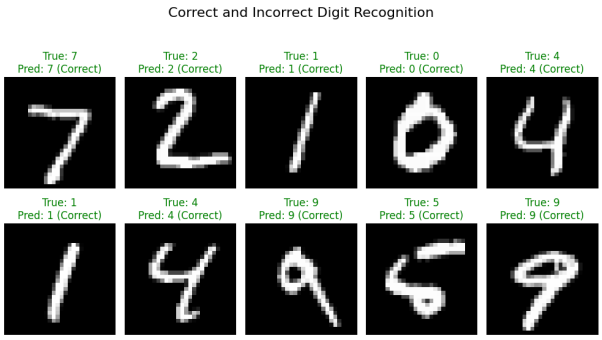
για τις 10 κατηγορίες.

Με αυτόν τον τρόπο βελτιώνεται κατά πάρα πολύ η ποιότητα των ανακατασκευασμένων εικόνων. Πλέον και περισσότερα νούμερα "μαντεύονται" σωστά και οι ανακατασκευασμένες εικόνες που προκύπτουν είναι λιγότερο "θολές".




## Digit Recognition στο Reconstructed Set

Ως τελευταίο βήμα χρησιμοποιείται ένα CNN για digit recognition πάνω στις ανακατασκευασμένες εικόνες του Autoencoder με τα Convolutional Layers. Τα αποτελέσματα είναι




Τα ποσοστά απόδοσης για αναγνώριση ψηφίου στο original και στο reconstructed dataset ειναι 99.40% και 82.62% αντίστοιχα.

Χρόνοι 20 λεπτά και 50 λεπτά για το 1ο και 2ο μοντέλο autoencoder αντίστοιχα.