Θάνος Γκουτής 4129

1Η ΥΠΟΧΡΕΩΤΙΚΗ ΕΡΓΑΣΙΑ

Έκθεση Αποτελεσμάτων

Φόρτωση των δεδομένων CIFAR-10 και εφαρμογή preprocessing (scaling)

```python
def unpickle(file):
    with open(file, 'rb') as fo:
        dict = pickle.load(fo, encoding='bytes')
    return dict

# Συνάρτηση για τη φόρτωση της CIFAR-10
1 usage
def load_cifar10_data(folder_path):
    train_data = None
    train_labels = []
    for i in range(1, 6):
        batch = unpickle(f"{folder_path}/data_batch_{i}")
        batch_data = batch[b'data']
        if train_data is None:
            train_data = batch_data
        else:
            train_data = np.concatenate( arrays: (train_data, batch_data), axis=0)
        train_labels.extend(batch[b'labels'])

    test_batch = unpickle(f"{folder_path}/test_batch")
    test_data = test_batch[b'data']
    test_labels = test_batch[b'labels']

    # Φόρτωση των metadata για τα labels
    meta_data = unpickle(f"{folder_path}/batches.meta")
    label_names = meta_data[b'label_names']
    label_names = [label.decode('utf-8') for label in label_names]  # Μετατροπή των ετικετών από bytes σε string

    # Τυποποίηση δεδομένων
    sc = StandardScaler()
    train_data = sc.fit_transform(train_data)
    test_data = sc.transform(test_data)
```

MLP Νευρωνικό Δίκτυο

```python
# Ορισμός του MLP μοντέλου
2 usages
class MLP(nn.Module):
    def __init__(self):
        super(MLP, self).__init__()
        self.fc1 = nn.Linear( in_features: 3072,  out_features: 3072)
        self.fc2 = nn.Linear( in_features: 3072,  out_features: 1024)
        self.fc4 = nn.Linear( in_features: 1024,  out_features: 10)

    def forward(self, x):
        x = F.relu(self.fc1(x))
        x = F.relu(self.fc2(x))
        x = self.fc4(x)  # Το τελικό επίπεδο χωρίς ReLU
        return x


model = MLP()

criterion = nn.CrossEntropyLoss()
optimizer = optim.SGD(model.parameters(), lr=0.01)
```

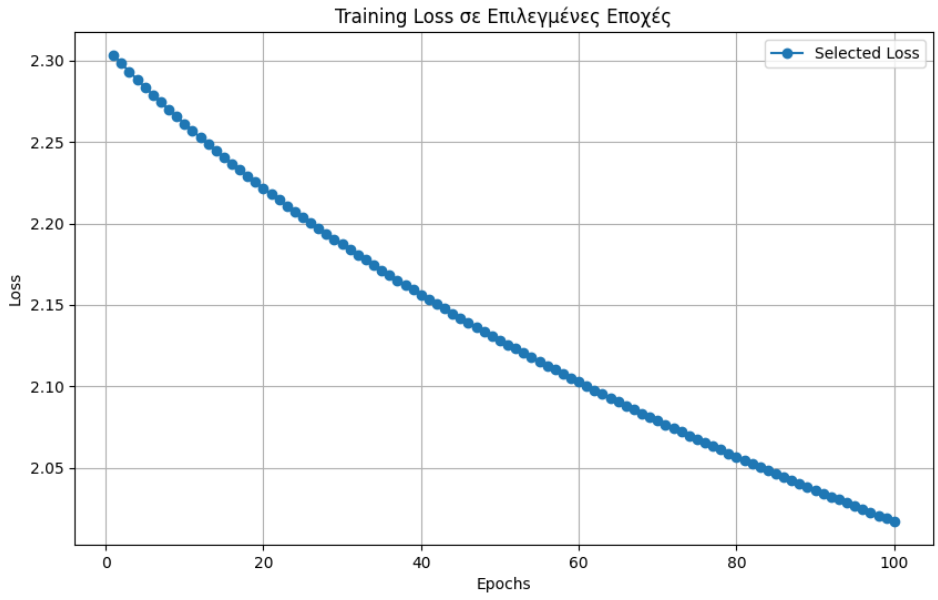
Εκπαίδευση Δικτύου

```python
# Λίστα για αποθήκευση του loss σε επιλεγμένες εποχές
selected_epochs = []
selected_losses = []

# Λούπα εκπαίδευσης
num_epochs = 100
start_time = time.time()
for epoch in range(num_epochs):
    optimizer.zero_grad()
    outputs = model(train_data)
    loss = criterion(outputs, train_labels)
    loss.backward()
    optimizer.step()

    selected_epochs.append(epoch + 1)
    selected_losses.append(loss.item())
    print(f"Epoch [{epoch + 1}/{num_epochs}], Loss: {loss.item():.4f}")

end_time = time.time()
print(f"Χρόνος εκπαίδευσης: {(end_time - start_time) / 60:.2f} λεπτά")
```

Αξιολόγηση Μοντέλου

```python
# Αξιολόγηση του μοντέλου
model.eval()

2 usages
def predict_acc(data, labels, name):
    with torch.no_grad():
        outputs = model(data)
        _, predicted = torch.max(outputs, 1)
        correct = (predicted == labels).sum().item()
        accuracy = 100 * correct / labels.size(0)

    print(f'Ακρίβεια στο {name} set: {accuracy:.2f}%')

    return predicted

predict_acc(train_data, train_labels, name: "Training")
predicted = predict_acc(test_data, test_labels, name: "Test")
```

Πρώτα Αποτελέσματα



1. Επεξήγηση του διαγράμματος

- **Training Loss:**
    - Το διάγραμμα δείχνει τη μείωση του **training loss** κατά τη διάρκεια των 100 εποχών εκπαίδευσης.
    - Στην αρχή το loss είναι περίπου **2.30**, και μειώνεται σταδιακά σε **2.01**.
    - Η αργή μείωση υποδεικνύει ότι το μοντέλο μαθαίνει, αλλά πολύ αργά, πιθανώς λόγω:
        - Περιορισμένης πολυπλοκότητας του μοντέλου.
        - Υποεκπαίδευσης (underfitting), καθώς το τελικό loss παραμένει υψηλό.

2. Επεξήγηση των αποτελεσμάτων

- **Χαμηλή ακρίβεια (Training και Test set):**
    - Η ακρίβεια στο training set είναι **30.64%**, ενώ στο test set **30.79%**.
    - Αυτό δείχνει ότι το μοντέλο δεν μπορεί να μάθει επαρκώς για να διαχωρίσει τις κλάσεις.
    - Η παρόμοια απόδοση μεταξύ training και test set υποδεικνύει ότι δεν υπάρχει overfitting, αλλά το μοντέλο έχει περιορισμένη ικανότητα γενίκευσης.
- **Απόδοση ανά κατηγορία:**
    - Υψηλότερες ακρίβειες: Ship (**52.10%**) και Truck (**51.30%**).
    - Χαμηλότερες ακρίβειες: Bird (**3.50%**) και Cat (**12.20%**).
    - Οι καλύτερες κατηγορίες είναι πιθανώς πιο εύκολα διακριτές λόγω συγκεκριμένων χαρακτηριστικών (χρώμα, σχήμα), ενώ οι χαμηλές ακρίβειες προέρχονται από κατηγορίες που μπερδεύονται λόγω ομοιότητας ή ανεπαρκούς εκπαίδευσης.
- **Χρόνος εκπαίδευσης:** 26.75 λεπτά. Δείχνει ότι ίσως ο αριθμός των χαρακτηριστικών ή το μέγεθος του μοντέλου χρειάζεται περαιτέρω βελτιστοποίηση καθώς είναι πολύ αργό.

---

4. Συμπεράσματα από τα αποτελέσματα

- Το μοντέλο δεν κατάφερε να μάθει αποτελεσματικά, πιθανώς επειδή:
    - Η χρήση μεγάλου αριθμού χαρακτηριστικών (3072) χωρίς μείωση διαστάσεων καθιστά την εκπαίδευση δύσκολη.
    - Η απουσία τεχνικών όπως dropout δεν βοηθά στη σταθερότητα της εκπαίδευσης.
    - Το σταθερό learning rate μπορεί να περιορίζει την προσαρμοστικότητα του optimizer.

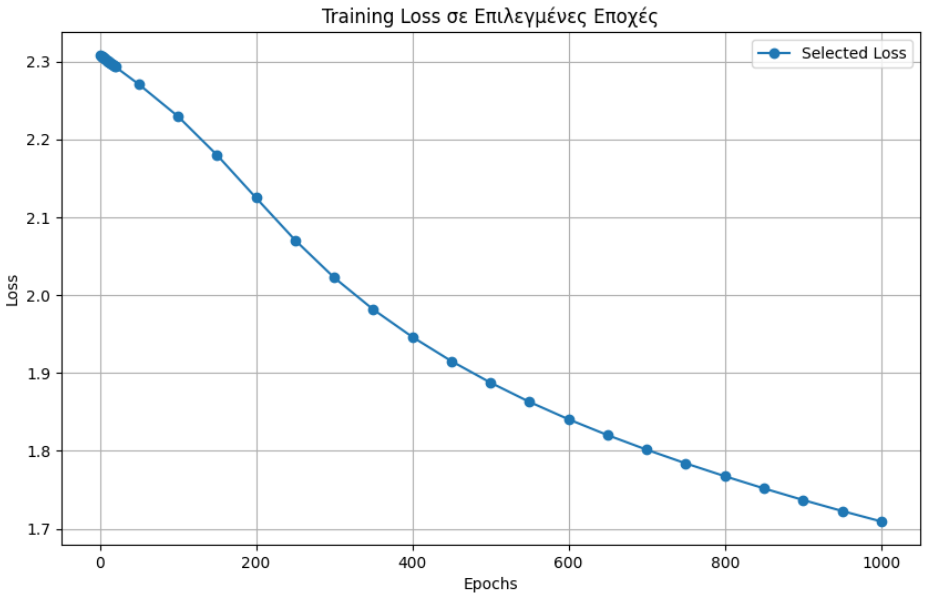
2ο Στάδιο

Προσθήκη PCA και αύξηση εποχών σε 1000

```
# Μείωση διαστάσεων με PCA
pca = PCA(n_components=300)
train_data = pca.fit_transform(train_data)
test_data = pca.transform(test_data)
```

```
# Ορισμός του MLP μοντέλου
2 usages
class MLP(nn.Module):
    def __init__(self):
        super(MLP, self).__init__()
        self.fc1 = nn.Linear( in_features: 300,  out_features: 300)
        self.fc2 = nn.Linear( in_features: 300,  out_features: 128)
        self.fc3 = nn.Linear( in_features: 128,  out_features: 64)
        self.fc4 = nn.Linear( in_features: 64,  out_features: 10)

    def forward(self, x):
        x = F.relu(self.fc1(x))
        x = F.relu(self.fc2(x))
        x = F.relu(self.fc3(x))
        x = self.fc4(x)  # Το τελικό επίπεδο χωρίς ReLU
        return x
```

Αποτελέσματα



Μείωση Διαστάσεων με PCA

Σκοπός: Η εφαρμογή του PCA έγινε με στόχο τη μείωση των διαστάσεων των δεδομένων από 3072 σε 300, προκειμένου:

1. Να επιταχυνθεί ο χρόνος εκπαίδευσης.
2. Να αφαιρεθεί ο θόρυβος ή οι λιγότερο σημαντικές πληροφορίες, επιτρέποντας στο μοντέλο να επικεντρωθεί στα πιο χρήσιμα χαρακτηριστικά.

Διαδικασία:

1. **Προεπεξεργασία Δεδομένων:**
    - Τα δεδομένα τυποποιήθηκαν χρησιμοποιώντας **StandardScaler**, ώστε όλες οι τιμές να έχουν μέση τιμή 0 και διακύμανση 1.
2. **Εφαρμογή PCA:**
    - Οι διαστάσεις μειώθηκαν από 3072 (αρχικές διαστάσεις εικόνας CIFAR-10) σε 300.
    - Το PCA διατήρησε τη μέγιστη δυνατή πληροφορία που επηρεάζει τα χαρακτηριστικά της εικόνας.

Χρόνος Εκπαίδευσης:

- Ο χρόνος εκπαίδευσης μειώθηκε σημαντικά από **26.75 λεπτά** (χωρίς PCA) σε **3.59 λεπτά** (με PCA).
- Αυτή η βελτίωση οφείλεται στη μείωση του μεγέθους εισόδου από 3072 σε 300.

Ακρίβεια:

- **Training Set:** 39.25% (σε σύγκριση με 30.64% χωρίς PCA).
- **Test Set:** 39.19% (σε σύγκριση με 30.79% χωρίς PCA).
- Η ακρίβεια βελτιώθηκε, καθώς η μείωση διαστάσεων απέβαλε αχρείαστο θόρυβο.

Ακρίβεια Ανά Κατηγορία:

Οι κατηγορίες που ταξινομήθηκαν πιο σωστά ήταν:

- **Ship:** 58.20%
- **Frog:** 46.30%

Οι κατηγορίες με τη χαμηλότερη ακρίβεια ήταν:

- **Bird:** 21.80%
- **Cat:** 20.60%

Παρατηρήσεις:

- Κατηγορίες με έντονα οπτικά χαρακτηριστικά (π.χ., **ship**, **frog**) ταξινομούνται πιο σωστά.
- Κατηγορίες με πιο περίπλοκα ή παρόμοια χαρακτηριστικά (π.χ., **cat**, **dog**) εμφανίζουν μεγαλύτερη δυσκολία.

Γράφημα Loss:

- Η εκπαίδευση συνέχιζε να βελτιώνει το **training loss** ακόμα και μετά από 1000 εποχές, με τελική τιμή **1.7094**.
- Αυτό δείχνει ότι το μοντέλο δεν έχει υπερεκπαιδευτεί, αλλά συνεχίζει να μαθαίνει.

Συμπεράσματα:

- Η εφαρμογή του PCA βελτίωσε την ακρίβεια του μοντέλου και μείωσε σημαντικά τον χρόνο εκπαίδευσης.
- Ωστόσο, παρατηρείται ακόμα περιορισμένη ακρίβεια σε ορισμένες κατηγορίες, γεγονός που δείχνει την ανάγκη για πιο σύνθετες τεχνικές.

3ο Στάδιο: Μείωση Νευρώνων

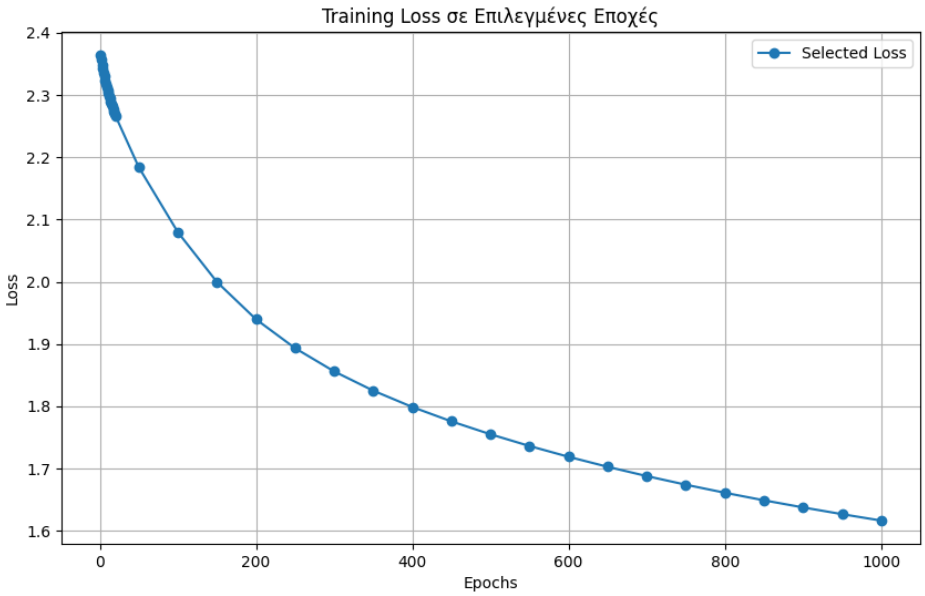
```python
class MLP(nn.Module):
    def __init__(self):
        super(MLP, self).__init__()
        self.fc1 = nn.Linear( in_features: 300,  out_features: 300)
        self.fc2 = nn.Linear( in_features: 300,  out_features: 64)
        self.fc3 = nn.Linear( in_features: 64,  out_features: 10)

    def forward(self, x):
        x = F.relu(self.fc1(x))
        x = F.relu(self.fc2(x))
        x = self.fc3(x)  # Το τελικό επίπεδο χωρίς ReLU
        return x
```

Αλλαγές στο Μοντέλο

Στο συγκεκριμένο στάδιο διατηρήθηκε η μείωση των διαστάσεων των δεδομένων μέσω PCA (σε 300 διαστάσεις). Ωστόσο, το νευρωνικό δίκτυο απλοποιήθηκε περαιτέρω:

- Ο αριθμός των επιπέδων μειώθηκε σε τρία (300 -> 300 -> 64 -> 10).
- Οι παράμετροι του μοντέλου μειώθηκαν σημαντικά.

Αποτελέσματα



Χρόνος Εκπαίδευσης: Το μοντέλο εκπαιδεύτηκε σε **2.62 λεπτά**, παρουσιάζοντας βελτίωση σε χρόνο εκπαίδευσης, λόγω της απλοποίησης της αρχιτεκτονικής.

Ακρίβεια:

- **Training Set**: 43.26%
- **Test Set**: 42.61%

Ακρίβεια Ανά Κατηγορία:

Κατηγορία: airplane | Σωστά: 499 / 1000 | Ακρίβεια: 49.90%

Κατηγορία: automobile | Σωστά: 502 / 1000 | Ακρίβεια: 50.20%

Κατηγορία: bird | Σωστά: 255 / 1000 | Ακρίβεια: 25.50%

Κατηγορία: cat | Σωστά: 262 / 1000 | Ακρίβεια: 26.20%

Κατηγορία: deer | Σωστά: 302 / 1000 | Ακρίβεια: 30.20%

Κατηγορία: dog | Σωστά: 345 / 1000 | Ακρίβεια: 34.50%

Κατηγορία: frog | Σωστά: 530 / 1000 | Ακρίβεια: 53.00%

Κατηγορία: horse | Σωστά: 449 / 1000 | Ακρίβεια: 44.90%

Κατηγορία: ship | Σωστά: 608 / 1000 | Ακρίβεια: 60.80%

Κατηγορία: truck | Σωστά: 509 / 1000 | Ακρίβεια: 50.90%

Παραδείγματα Σωστών Κατηγοριοποιήσεων:

Δείγμα 0: Πραγματική Ετικέτα = cat, Προβλεπόμενη Ετικέτα = cat

Δείγμα 1: Πραγματική Ετικέτα = ship, Προβλεπόμενη Ετικέτα = ship

Δείγμα 2: Πραγματική Ετικέτα = ship, Προβλεπόμενη Ετικέτα = ship

Δείγμα 5: Πραγματική Ετικέτα = frog, Προβλεπόμενη Ετικέτα = frog

Δείγμα 7: Πραγματική Ετικέτα = frog, Προβλεπόμενη Ετικέτα = frog

Παραδείγματα Εσφαλμένων Κατηγοριοποιήσεων:

Δείγμα 3: Πραγματική Ετικέτα = airplane, Προβλεπόμενη Ετικέτα = ship

Δείγμα 4: Πραγματική Ετικέτα = frog, Προβλεπόμενη Ετικέτα = deer

Δείγμα 6: Πραγματική Ετικέτα = automobile, Προβλεπόμενη Ετικέτα = cat

Δείγμα 8: Πραγματική Ετικέτα = cat, Προβλεπόμενη Ετικέτα = bird

Δείγμα 10: Πραγματική Ετικέτα = airplane, Προβλεπόμενη Ετικέτα = ship

Συμπεράσματα

- **Χρόνος Εκπαίδευσης**: Το μοντέλο με λιγότερους νευρώνες είναι ταχύτερο στην εκπαίδευση σε σχέση με πιο πολύπλοκα δίκτυα.
- **Ακρίβεια**: Η ακρίβεια στο test set αυξήθηκε στο **42.61%**, υπερβαίνοντας τα αποτελέσματα της προηγούμενης αρχιτεκτονικής με περισσότερα επίπεδα (39.19%).
- **Γενίκευση**: Το απλοποιημένο δίκτυο παρουσίασε καλύτερη γενίκευση, πιθανότατα λόγω της μείωσης των παραμέτρων που απέτρεψαν το overfitting.

Εξήγηση της Βελτίωσης:

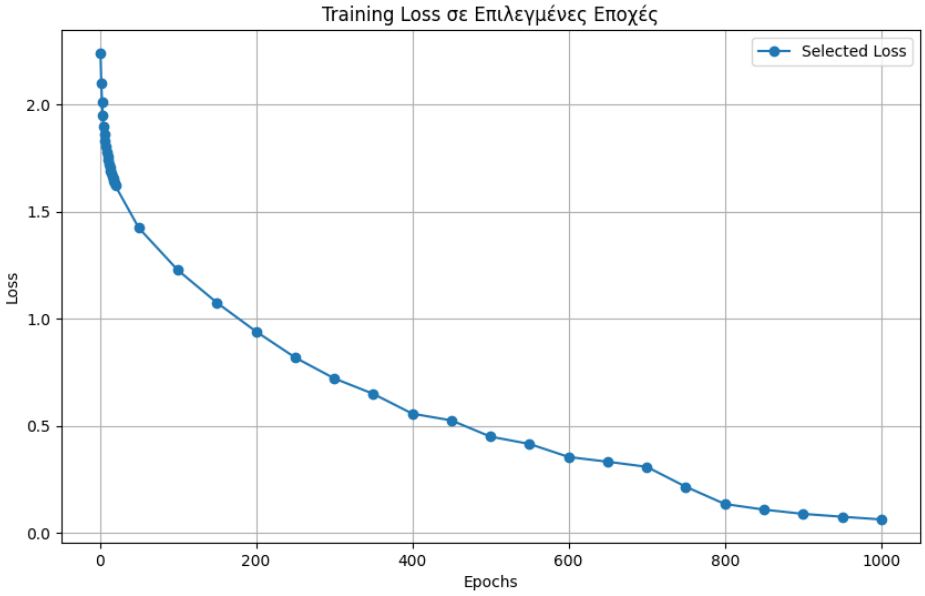
- **Αποφυγή Overfitting**: Η μείωση του αριθμού των παραμέτρων συνέβαλε στη μείωση της πολυπλοκότητας του μοντέλου. Ένα πιο λιτό μοντέλο είναι λιγότερο πιθανό να υπερεκπαιδευτεί στα δεδομένα εκπαίδευσης και έχει καλύτερη δυνατότητα γενίκευσης.
- **Εξισορρόπηση Διαστάσεων και Παραμέτρων**: Η μείωση των διαστάσεων μέσω PCA απομάκρυνε τις περιττές πληροφορίες. Έτσι, ένα μικρότερο δίκτυο ήταν αρκετό για να εκπαιδευτεί αποδοτικά χωρίς να "μπερδεύεται" από την πλεονάζουσα πληροφορία.

4ο Στάδιο: DataLoader με batch size = 1024

```python
# Δημιουργία DataLoader με batch size
batch_size = 1024  # Μεγαλύτερο batch size για σταθερά gradients
train_dataset = TensorDataset( *tensors: train_data, train_labels)
train_loader = DataLoader(train_dataset, batch_size=batch_size, shuffle=True)
```

```python
for epoch in range(num_epochs):
    model.train()
    epoch_loss = 0.0
    for batch_data, batch_labels in train_loader:
        outputs = model(batch_data)
        loss = criterion(outputs, batch_labels)
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()
        epoch_loss += loss.item()
    avg_loss = epoch_loss / len(train_loader)

    if (epoch + 1) <= 20 or (epoch + 1) % 50 == 0:
        selected_epochs.append(epoch + 1)
        selected_losses.append(avg_loss)
        print(f"Epoch [{epoch + 1}/{num_epochs}], Loss: {avg_loss:.4f}")
```

Αποτελέσματα



Χρόνος Εκπαίδευσης:

- Συνολικός χρόνος: **11.74 λεπτά**.

Αυξήθηκε σε σχέση με προηγούμενες εκτελέσεις, καθώς τα μεγαλύτερα batches απαιτούν περισσότερο χρόνο για υπολογισμούς.

Τελικό Loss:

- **Training Loss**: **0.0623** στην 1000ή εποχή.

Διάγραμμα Loss:

Το διάγραμμα δείχνει:

- Σταθερή και ομαλή μείωση του loss καθ' όλη τη διάρκεια των εποχών.
- Αξιοσημείωτη μείωση στις πρώτες 200-300 εποχές, που σταθεροποιείται αργότερα, επιδεικνύοντας επιτυχημένη σύγκλιση.

Ακρίβεια στο Training Set: 99.60%
Ακρίβεια στο Test Set: 50.63%
Βελτίωση στην Ακρίβεια: Η ακρίβεια στο Test Set αυξήθηκε από το προηγούμενο μοντέλο, κάτι που υποδεικνύει ότι η αλλαγή στο batch size βοήθησε στη γενίκευση.

Γιατί Αυξήθηκε η Ακρίβεια στο Test Set;

1. **Σταθερά Gradients**:
    - Με αλλαγή στο batch size, τα gradients είναι πιο ακριβή, καθώς βασίζονται σε περισσότερα δείγματα ανά βήμα εκπαίδευσης.
    - Αυτό επιτρέπει στο μοντέλο να μαθαίνει πιο σταθερά και αποφεύγει τις τυχαίες αποκλίσεις από τα θορυβώδη δεδομένα.
2. **Ομαλή Σύγκλιση**:
    - Όπως φαίνεται στο διάγραμμα, το loss μειώνεται πιο σταθερά χωρίς απότομες αλλαγές, κάτι που υποδηλώνει σταθερή εκπαίδευση

Ακρίβεια Ανά Κατηγορία:
Κατηγορία: airplane | Σωστά: 611 / 1000 | Ακρίβεια: 61.10%
Κατηγορία: automobile | Σωστά: 585 / 1000 | Ακρίβεια: 58.50%
Κατηγορία: bird | Σωστά: 404 / 1000 | Ακρίβεια: 40.40%
Κατηγορία: cat | Σωστά: 354 / 1000 | Ακρίβεια: 35.40%
Κατηγορία: deer | Σωστά: 412 / 1000 | Ακρίβεια: 41.20%
Κατηγορία: dog | Σωστά: 411 / 1000 | Ακρίβεια: 41.10%
Κατηγορία: frog | Σωστά: 558 / 1000 | Ακρίβεια: 55.80%
Κατηγορία: horse | Σωστά: 549 / 1000 | Ακρίβεια: 54.90%
Κατηγορία: ship | Σωστά: 647 / 1000 | Ακρίβεια: 64.70%
Κατηγορία: truck | Σωστά: 532 / 1000 | Ακρίβεια: 53.20%

Χαρακτηριστικά Σωστά Κατηγοριοποιημένα Δείγματα:
Δείγμα 0: Πραγματική Ετικέτα = cat, Προβλεπόμενη Ετικέτα = cat
Δείγμα 1: Πραγματική Ετικέτα = ship, Προβλεπόμενη Ετικέτα = ship
Δείγμα 3: Πραγματική Ετικέτα = airplane, Προβλεπόμενη Ετικέτα = airplane
Δείγμα 4: Πραγματική Ετικέτα = frog, Προβλεπόμενη Ετικέτα = frog
Δείγμα 6: Πραγματική Ετικέτα = automobile, Προβλεπόμενη Ετικέτα = automobile
Δείγμα 7: Πραγματική Ετικέτα = frog, Προβλεπόμενη Ετικέτα = frog
Δείγμα 9: Πραγματική Ετικέτα = automobile, Προβλεπόμενη Ετικέτα = automobile
Δείγμα 10: Πραγματική Ετικέτα = airplane, Προβλεπόμενη Ετικέτα = airplane
Δείγμα 11: Πραγματική Ετικέτα = truck, Προβλεπόμενη Ετικέτα = truck
Δείγμα 18: Πραγματική Ετικέτα = ship, Προβλεπόμενη Ετικέτα = ship

Χαρακτηριστικά Εσφαλμένα Κατηγοριοποιημένα Δείγματα:
Δείγμα 2: Πραγματική Ετικέτα = ship, Προβλεπόμενη Ετικέτα = airplane
Δείγμα 5: Πραγματική Ετικέτα = frog, Προβλεπόμενη Ετικέτα = cat
Δείγμα 8: Πραγματική Ετικέτα = cat, Προβλεπόμενη Ετικέτα = deer
Δείγμα 12: Πραγματική Ετικέτα = dog, Προβλεπόμενη Ετικέτα = bird
Δείγμα 13: Πραγματική Ετικέτα = horse, Προβλεπόμενη Ετικέτα = dog
Δείγμα 14: Πραγματική Ετικέτα = truck, Προβλεπόμενη Ετικέτα = automobile
Δείγμα 15: Πραγματική Ετικέτα = ship, Προβλεπόμενη Ετικέτα = dog
Δείγμα 16: Πραγματική Ετικέτα = dog, Προβλεπόμενη Ετικέτα = bird
Δείγμα 17: Πραγματική Ετικέτα = horse, Προβλεπόμενη Ετικέτα = dog
Δείγμα 20: Πραγματική Ετικέτα = horse, Προβλεπόμενη Ετικέτα = cat

Overfitting

Το μοντέλο δείχνει σημάδια **overfitting**, και αυτό φαίνεται από:

1. **Υψηλή Ακρίβεια στο Training Set**:
    - Η ακρίβεια στο training set φτάνει το **99.60%**, που υποδηλώνει ότι το μοντέλο μαθαίνει πολύ καλά τα δεδομένα εκπαίδευσης.
2. **Χαμηλότερη Ακρίβεια στο Test Set**:
    - Η ακρίβεια στο test set είναι **50.63%**, που υποδεικνύει ότι το μοντέλο δεν γενικεύει καλά σε δεδομένα που δεν έχει δει.
3. **Τελικό Loss**:
    - Το **training loss** είναι πολύ χαμηλό (**0.0623**), δείχνοντας ότι το μοντέλο έχει "παραμάθει" τα δεδομένα εκπαίδευσης, αλλά το test loss (αν το υπολόγιζες) πιθανότατα θα ήταν σημαντικά υψηλότερο.

Τι προκαλεί το overfitting εδώ;

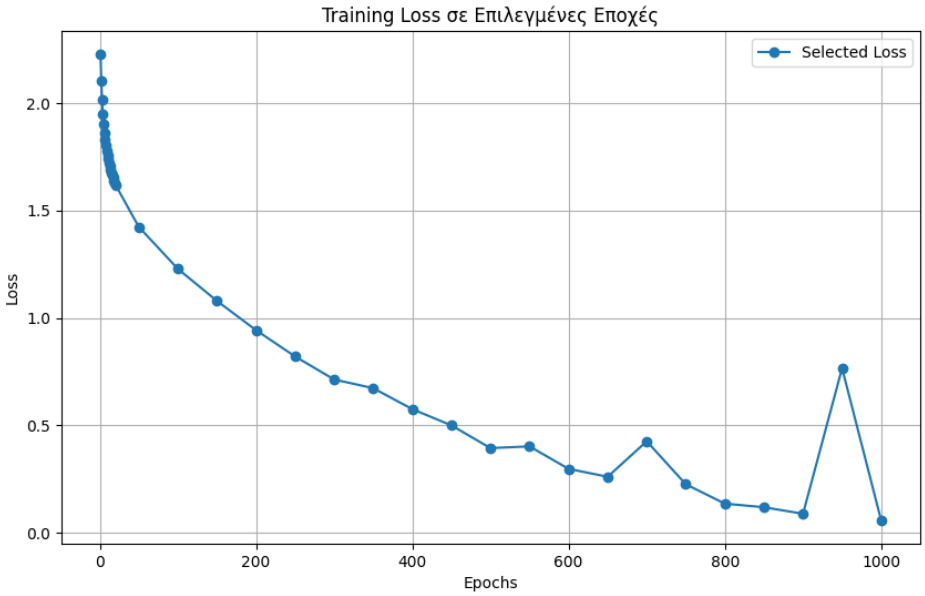
- **Απουσία Regularization** (π.χ. L2 regularization ή Dropout), το οποίο βοηθάει στον περιορισμό του overfitting.
- **Πολύ μεγάλος χρόνος εκπαίδευσης** (1000 εποχές), που επιτρέπει στο μοντέλο να μάθει ακόμα και το "θόρυβο" των δεδομένων.

5ο Στάδιο: Προσθήκη Early Stopping

```python
# Early Stopping parameters
patience = 20
best_loss = float('inf')
counter = 0
```

```python
    # Early Stopping Check
    if avg_loss < best_loss:
        best_loss = avg_loss
        counter = 0
    else:
        counter += 1
        if counter >= patience:
            if (epoch + 1) not in selected_epochs:
                selected_epochs.append(epoch + 1)
                selected_losses.append(avg_loss)
            print(f"Early stopping at epoch {epoch + 1}, Loss: {avg_loss:.4f}")
            break
```

Αποτελέσματα



Στο διάγραμμα παρατηρούνται **αιχμές** στο loss περίπου στις εποχές **700** και **950**.
Αυτό μπορεί να εξηγηθεί από τα εξής:

Πιθανές Αιτίες:

1. **Ευαισθησία στο Learning Rate**:
    - Αν το learning rate είναι σταθερό και το μοντέλο πλησιάζει την **ελάχιστη τιμή του loss**, τότε οι ενημερώσεις των weights μπορεί να είναι υπερβολικές, οδηγώντας σε προσωρινή "απόκλιση" από το minimum.
2. **Overfitting**:
    - Πιθανόν το μοντέλο να "προσαρμόζεται" υπερβολικά σε δεδομένα εκπαίδευσης που είναι θορυβώδη ή ασυνήθιστα, κάτι που επηρεάζει αρνητικά το loss σε συγκεκριμένες εποχές.
    - Το φαινόμενο αυτό ενισχύεται καθώς το training loss γίνεται εξαιρετικά χαμηλό (π.χ. <0.1), κάτι που σημαίνει ότι το μοντέλο ίσως μαθαίνει χαρακτηριστικά των δεδομένων εκπαίδευσης που δεν γενικεύουν καλά.

Χρόνος Εκπαίδευσης:

- Ο χρόνος εκπαίδευσης ήταν **11.93 λεπτά** άρα το μοντέλο έφτασε τον μέγιστο αριθμό εποχών (1000), καθώς το **Early Stopping** δεν ενεργοποιήθηκε λόγω συνεχόμενης βελτίωσης στο loss.

Τελικό Loss:

- Το **training loss** έφτασε στο **0.0581**.

Ακρίβεια:

- **Training set**: **99.69%**, που είναι ενδεικτικό **overfitting**.
- **Test set**: **50.05%**, παρατηρείται ελαφριά πτώση της ακρίβειας σε σχέση με το προηγούμενο στάδιο. Αυτό μπορεί να οφείλεται στο ότι το μοντέλο συνεχίζει να εκπαιδεύεται για πολλές εποχές χωρίς να βελτιώνει τη γενίκευση.

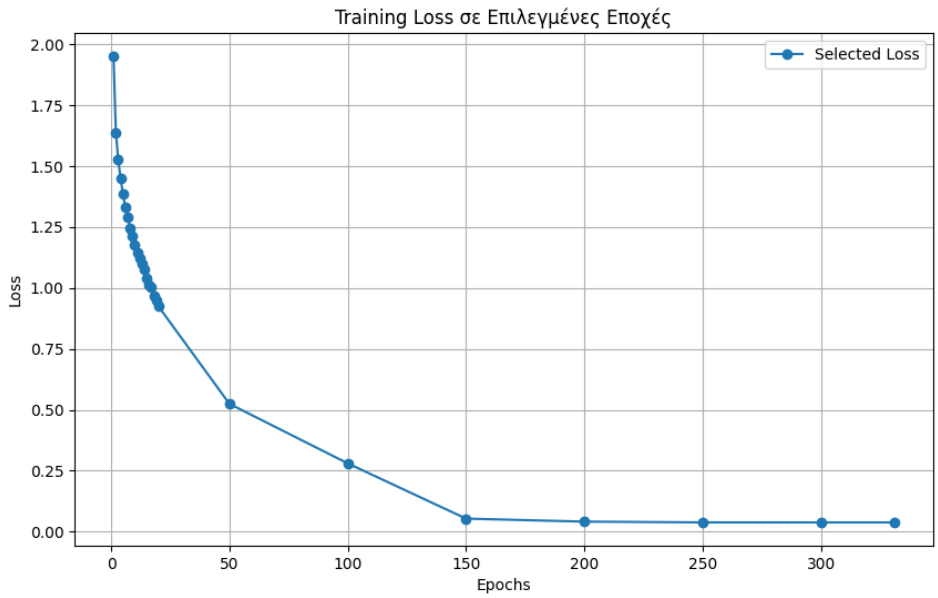
6ο Στάδιο: Προσθήκη Momentum, Weight Decay και Scheduler

```
optimizer = optim.SGD(model.parameters(), lr=0.02, momentum=0.9, weight_decay=1e-3)
scheduler = ReduceLROnPlateau(optimizer, mode='min', factor=0.5, patience=10)
scheduler.step(avg_loss)
```

Περιγραφή Προσθήκης:

- Προστέθηκε **Scheduler** με ReduceLROnPlateau για να μειώνεται το learning rate κατά 50% όταν το loss σταματήσει να βελτιώνεται για **10 συνεχόμενες εποχές**.
- Ενεργοποιήθηκε **Weight Decay** (L2 Regularization) με τιμή 1e-3 για να αποφεύγεται η υπερπροσαρμογή.
- Προστέθηκε **momentum** με τιμή 0.9

Αποτελέσματα



Το διάγραμμα δείχνει μια πολύ καλή συνολική συμπεριφορά του μοντέλου όσον αφορά το loss. Παρατηρούμε:

1. **Σημαντική Μείωση Loss στις Πρώτες Εποχές:**
   - Το loss μειώνεται γρήγορα μέχρι περίπου την **50ή εποχή**, κάτι που σημαίνει ότι το μοντέλο μαθαίνει γρήγορα τα βασικά χαρακτηριστικά των δεδομένων.
2. **Σταθεροποίηση του Loss:**
   - Μετά την **100ή εποχή**, η μείωση του loss γίνεται σταδιακά μικρότερη, με το μοντέλο να βελτιώνεται πιο αργά.
   - Το **ReduceLROnPlateau** πιθανότατα έχει μειώσει το learning rate σε αυτό το σημείο, κάτι που εξηγεί τη σταθεροποίηση.

3. **Early Stopping:**
    - Το early stopping ενεργοποιήθηκε στην **331η εποχή**, όταν το loss δεν μειωνόταν πλέον σημαντικά. Αυτό φαίνεται στο διάγραμμα, καθώς το loss παραμένει σταθερό. Θα μπορούσε το patience να είναι μικρότερο.

4. **Χαμηλό Τελικό Loss:**
    - Το τελικό loss στο training set είναι **0.0379**, δείχνοντας ότι το μοντέλο έχει πολύ καλή απόδοση στο training set.

5. **Επιδράσεις του Scheduler και του Weight Decay:**
    - Η προσθήκη του scheduler βοήθησε στη σταθεροποίηση και τη βελτίωση του loss στις τελικές εποχές.
    - Το weight decay περιόρισε την υπερπροσαρμογή, κάτι που φαίνεται από την ελεγχόμενη μείωση του training loss χωρίς ακραία overfitting.

Momentum

Το **momentum** βοήθησε να μειωθεί το loss πιο γρήγορα στις πρώτες εποχές, δίνοντας ώθηση στη διαδικασία εκπαίδευσης.

Σε συνδυασμό με το weight decay και τον scheduler, οδήγησε σε σταθερότερη εκπαίδευση, μειώνοντας τις ταλαντώσεις και τις απότομες αλλαγές στα βάρη.

Συμπέρασμα: Ο συνδυασμός momentum (**0.9**) και weight decay έπαιξε σημαντικό ρόλο στην αποτελεσματική εκμάθηση των χαρακτηριστικών και τη συγκράτηση του overfitting.

Βελτίωση Επιδόσεων:

- Η εκπαίδευση σταμάτησε στην **331η εποχή** λόγω του **early stopping**, μειώνοντας τον χρόνο εκπαίδευσης σε **3.99 λεπτά**.
- Το **training loss** κατέληξε στο **0.0379**, ενώ η ακρίβεια στο training set έφτασε στο **99.98%**, επιβεβαιώνοντας την πλήρη προσαρμογή στο training set (**Overfitting**).
- Η ακρίβεια στο test set αυξήθηκε στο **52.75%**, που αποτελεί βελτίωση σε σχέση με το προηγούμενο στάδιο (**52.61%**).

Κατηγορίες με Καλύτερη Απόδοση:

- Οι καλύτερες κατηγορίες ήταν τα **ship (66.60%)**, **airplane (62.20%)**, και **horse (57.90%)**.

- Οι χαμηλότερες επιδόσεις εμφανίστηκαν στις κατηγορίες **cat (35.60%)** και **dog (43.00%)**, που είναι συνηθισμένο, καθώς αυτές οι κατηγορίες έχουν παρόμοια χαρακτηριστικά.

Ακρίβεια Ανά Κατηγορία:
Κατηγορία: airplane | Σωστά: 622 / 1000 | Ακρίβεια: 62.20%

Κατηγορία: automobile | Σωστά: 602 / 1000 | Ακρίβεια: 60.20%

Κατηγορία: bird | Σωστά: 438 / 1000 | Ακρίβεια: 43.80%

Κατηγορία: cat | Σωστά: 356 / 1000 | Ακρίβεια: 35.60%

Κατηγορία: deer | Σωστά: 447 / 1000 | Ακρίβεια: 44.70%

Κατηγορία: dog | Σωστά: 430 / 1000 | Ακρίβεια: 43.00%

Κατηγορία: frog | Σωστά: 574 / 1000 | Ακρίβεια: 57.40%

Κατηγορία: horse | Σωστά: 579 / 1000 | Ακρίβεια: 57.90%

Κατηγορία: ship | Σωστά: 666 / 1000 | Ακρίβεια: 66.60%

Κατηγορία: truck | Σωστά: 561 / 1000 | Ακρίβεια: 56.10%

Χαρακτηριστικά Σωστά Κατηγοριοποιημένα Δείγματα:

Δείγμα 0: Πραγματική Ετικέτα = cat, Προβλεπόμενη Ετικέτα = cat

Δείγμα 1: Πραγματική Ετικέτα = ship, Προβλεπόμενη Ετικέτα = ship

Δείγμα 3: Πραγματική Ετικέτα = airplane, Προβλεπόμενη Ετικέτα = airplane

Δείγμα 5: Πραγματική Ετικέτα = frog, Προβλεπόμενη Ετικέτα = frog

Δείγμα 7: Πραγματική Ετικέτα = frog, Προβλεπόμενη Ετικέτα = frog

Δείγμα 9: Πραγματική Ετικέτα = automobile, Προβλεπόμενη Ετικέτα = automobile

Δείγμα 11: Πραγματική Ετικέτα = truck, Προβλεπόμενη Ετικέτα = truck

Δείγμα 13: Πραγματική Ετικέτα = horse, Προβλεπόμενη Ετικέτα = horse

Δείγμα 17: Πραγματική Ετικέτα = horse, Προβλεπόμενη Ετικέτα = horse

Δείγμα 18: Πραγματική Ετικέτα = ship, Προβλεπόμενη Ετικέτα = ship

Χαρακτηριστικά Εσφαλμένα Κατηγοριοποιημένα Δείγματα:

Δείγμα 2: Πραγματική Ετικέτα = ship, Προβλεπόμενη Ετικέτα = airplane

Δείγμα 4: Πραγματική Ετικέτα = frog, Προβλεπόμενη Ετικέτα = cat

Δείγμα 6: Πραγματική Ετικέτα = automobile, Προβλεπόμενη Ετικέτα = frog

Δείγμα 8: Πραγματική Ετικέτα = cat, Προβλεπόμενη Ετικέτα = deer

Δείγμα 10: Πραγματική Ετικέτα = airplane, Προβλεπόμενη Ετικέτα = ship

Δείγμα 12: Πραγματική Ετικέτα = dog, Προβλεπόμενη Ετικέτα = bird

Δείγμα 14: Πραγματική Ετικέτα = truck, Προβλεπόμενη Ετικέτα = automobile

Δείγμα 15: Πραγματική Ετικέτα = ship, Προβλεπόμενη Ετικέτα = frog

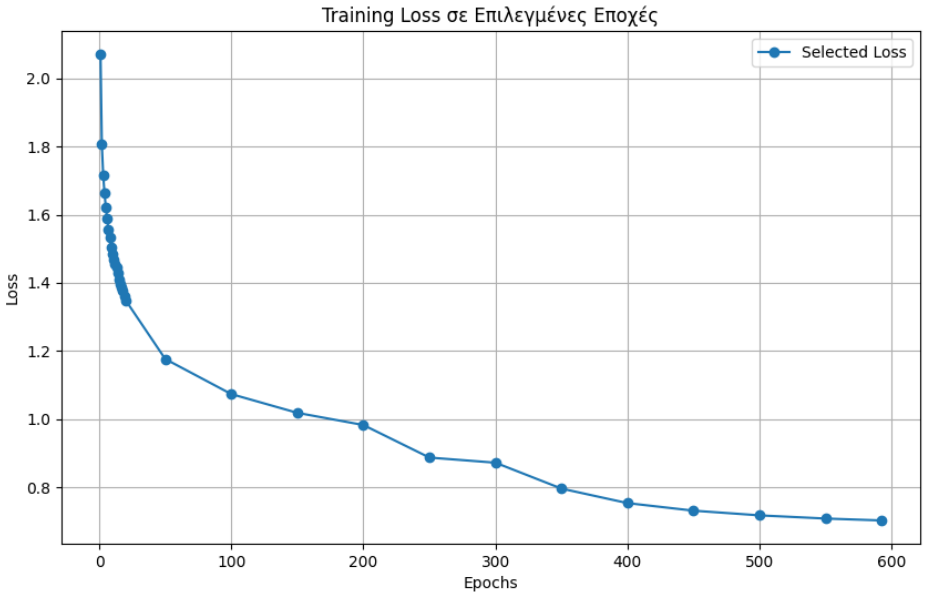
Δείγμα 16: Πραγματική Ετικέτα = dog, Προβλεπόμενη Ετικέτα = bird

Δείγμα 20: Πραγματική Ετικέτα = horse, Προβλεπόμενη Ετικέτα = dog

Τελευταίο Στάδιο: Προσθήκη Dropout

```python
# Ορισμός του MLP μοντέλου
2 usages
class MLP(nn.Module):
    def __init__(self):
        super(MLP, self).__init__()
        self.fc1 = nn.Linear( in_features: 300,  out_features: 300)
        self.fc2 = nn.Linear( in_features: 300,  out_features: 64)
        self.fc3 = nn.Linear( in_features: 64,  out_features: 10)
        self.dropout = nn.Dropout(p=0.3)

    def forward(self, x):
        x = F.relu(self.fc1(x))
        x = self.dropout(x)
        x = F.relu(self.fc2(x))
        x = self.dropout(x)
        x = self.fc3(x)  # Το τελικό επίπεδο χωρίς ReLU
        return x
```

Αποτελέσματα



Διάγραμμα Training Loss

Το διάγραμμα δείχνει σταθερή μείωση του training loss, με πιο αργή πτώση στις τελευταίες εποχές.

1. **Χρόνος Εκπαίδευσης:**

   - Η εκπαίδευση ολοκληρώθηκε σε περίπου **7.52 λεπτά**, με **early stopping** στην εποχή **592**.

2. **Τελικό Loss:**

   - Το loss σταθεροποιήθηκε κοντά στο **0.7028**, με το training loss να δείχνει ότι το μοντέλο εκπαιδεύτηκε αποτελεσματικά.

3. **Ακρίβεια:**

   - **Training Set Accuracy: 89.35%**
     - Το dropout περιόρισε την υπερβολική προσαρμογή στο training set, χωρίς να επιτρέπει στο μοντέλο να φτάσει το 99% που είδαμε προηγουμένως (δείκτης overfitting).
   - **Test Set Accuracy: 59.19%**
     - Εμφανής βελτίωση σε σχέση με τα προηγούμενα στάδια, με το test accuracy να φτάνει τη **μέγιστη τιμή σε αυτό το project**.

Επεξήγηση για το Dropout

1. **Αποφυγή Overfitting:**
   - Το dropout βοήθησε στη μείωση του overfitting, αναγκάζοντας το δίκτυο να μάθει πιο γενικεύσιμα χαρακτηριστικά.
2. **Καλύτερη Απόδοση στο Test Set:**
   - Η αφαίρεση τυχαίων νευρώνων κατά την εκπαίδευση ενίσχυσε την ανθεκτικότητα του μοντέλου σε νέα δεδομένα.
3. **Περιορισμός Συνεργασιών Νευρώνων:**
   - Η απενεργοποίηση νευρώνων αποτρέπει την εξάρτηση του δικτύου από συγκεκριμένες ομάδες νευρώνων.

4. **Κατηγορίες:**

   - Η **κατηγορία "ship"** είχε την υψηλότερη ακρίβεια (**72.50%**), ενώ η **"cat"** παρέμεινε μία από τις πιο δύσκολες κατηγορίες (**41.70%).**

Ακρίβεια Ανά Κατηγορία:

Κατηγορία: airplane | Σωστά: 653 / 1000 | Ακρίβεια: 65.30%

Κατηγορία: automobile | Σωστά: 703 / 1000 | Ακρίβεια: 70.30%

Κατηγορία: bird | Σωστά: 439 / 1000 | Ακρίβεια: 43.90%

Κατηγορία: cat | Σωστά: 417 / 1000 | Ακρίβεια: 41.70%

Κατηγορία: deer | Σωστά: 522 / 1000 | Ακρίβεια: 52.20%

Κατηγορία: dog | Σωστά: 482 / 1000 | Ακρίβεια: 48.20%

Κατηγορία: frog | Σωστά: 670 / 1000 | Ακρίβεια: 67.00%

Κατηγορία: horse | Σωστά: 659 / 1000 | Ακρίβεια: 65.90%

Κατηγορία: ship | Σωστά: 725 / 1000 | Ακρίβεια: 72.50%

Κατηγορία: truck | Σωστά: 649 / 1000 | Ακρίβεια: 64.90%

Χαρακτηριστικά Σωστά Κατηγοριοποιημένα Δείγματα:

Δείγμα 0: Πραγματική Ετικέτα = cat, Προβλεπόμενη Ετικέτα = cat

Δείγμα 1: Πραγματική Ετικέτα = ship, Προβλεπόμενη Ετικέτα = ship

Δείγμα 3: Πραγματική Ετικέτα = airplane, Προβλεπόμενη Ετικέτα = airplane

Δείγμα 4: Πραγματική Ετικέτα = frog, Προβλεπόμενη Ετικέτα = frog

Δείγμα 5: Πραγματική Ετικέτα = frog, Προβλεπόμενη Ετικέτα = frog

Δείγμα 6: Πραγματική Ετικέτα = automobile, Προβλεπόμενη Ετικέτα = automobile

Δείγμα 7: Πραγματική Ετικέτα = frog, Προβλεπόμενη Ετικέτα = frog

Δείγμα 9: Πραγματική Ετικέτα = automobile, Προβλεπόμενη Ετικέτα = automobile

Δείγμα 10: Πραγματική Ετικέτα = airplane, Προβλεπόμενη Ετικέτα = airplane

Δείγμα 11: Πραγματική Ετικέτα = truck, Προβλεπόμενη Ετικέτα = truck

Χαρακτηριστικά Εσφαλμένα Κατηγοριοποιημένα Δείγματα:

Δείγμα 2: Πραγματική Ετικέτα = ship, Προβλεπόμενη Ετικέτα = airplane

Δείγμα 8: Πραγματική Ετικέτα = cat, Προβλεπόμενη Ετικέτα = dog

Δείγμα 15: Πραγματική Ετικέτα = ship, Προβλεπόμενη Ετικέτα = dog

Δείγμα 16: Πραγματική Ετικέτα = dog, Προβλεπόμενη Ετικέτα = cat

Δείγμα 17: Πραγματική Ετικέτα = horse, Προβλεπόμενη Ετικέτα = cat

Δείγμα 20: Πραγματική Ετικέτα = horse, Προβλεπόμενη Ετικέτα = deer

Δείγμα 22: Πραγματική Ετικέτα = deer, Προβλεπόμενη Ετικέτα = airplane

Δείγμα 24: Πραγματική Ετικέτα = dog, Προβλεπόμενη Ετικέτα = deer

Δείγμα 25: Πραγματική Ετικέτα = bird, Προβλεπόμενη Ετικέτα = frog

Δείγμα 26: Πραγματική Ετικέτα = deer, Προβλεπόμενη Ετικέτα = bird

Σύγκριση CrossEntropyLoss και MSELoss

MSELoss

- **Χρόνος εκπαίδευσης**: 5.38 λεπτά
- **Ακρίβεια στο Training set**: 51.64%
- **Ακρίβεια στο Test set**: 49.13%
- **Early stopping:** Εποχή 447, **Loss:** 0.0664
- **Καλύτερη Κατηγορία**: ship με ακρίβεια 67.40%.
- **Χειρότερη Κατηγορία**: bird με ακρίβεια 25.60%.

Παρατηρήσεις και Συμπεράσματα:

1. **Χρόνος Εκπαίδευσης**:
    - Το MSELoss απαιτεί λιγότερο χρόνο για εκπαίδευση, πιθανώς λόγω της μικρότερης πολυπλοκότητας στη διαδικασία υπολογισμού του loss.
2. **Ακρίβεια**:
    - Το CrossEntropyLoss ξεπερνά το MSELoss τόσο στο training set όσο και στο test set.

- Η γενική απόδοση του μοντέλου με CrossEntropyLoss είναι σημαντικά καλύτερη (59.19% έναντι 49.13% στο test set).

3. **Overfitting**:
   - Το CrossEntropyLoss φαίνεται να μαθαίνει καλύτερα τα μοτίβα χωρίς να υπερεκπαιδεύεται υπερβολικά. Με το MSELoss, η ακρίβεια στο training set είναι πιο χαμηλή, υποδηλώνοντας ότι το μοντέλο δεν μαθαίνει καλά τα δεδομένα.
4. **Κατηγορίες**:
   - Στο MSELoss, η απόδοση σε δύσκολες κατηγορίες όπως **bird** και **cat** είναι πολύ χαμηλότερη, κάτι που δείχνει ότι δεν καταφέρνει να διαχωρίσει επαρκώς ορισμένα classes.
5. **Μικρότερο Loss με MSELoss:** Παρατηρήθηκε ότι το **MSELoss** έχει μικρότερη τιμή σε σχέση με το **CrossEntropyLoss**. Αυτό οφείλεται στο ότι το MSELoss υπολογίζει τη μέση τετραγωνική απόκλιση, η οποία μειώνεται γρηγορότερα καθώς το μοντέλο προσαρμόζεται στα δεδομένα. Ωστόσο, η απόδοση του μοντέλου με **CrossEntropyLoss** είναι σαφώς καλύτερη, καθώς η συνάρτηση αυτή είναι ειδικά σχεδιασμένη για προβλήματα ταξινόμησης. Χρησιμοποιεί τις πιθανότητες των εξόδων και "τιμωρεί" περισσότερο τις λανθασμένες προβλέψεις, οδηγώντας έτσι σε υψηλότερη ακρίβεια. Συμπερασματικά, αν και το **MSELoss** εμφανίζει μικρότερη τιμή, το **CrossEntropyLoss** προτιμάται καθώς αποδίδει καλύτερα στις ταξινομήσεις.
6. **Εκλογή Loss Function**:
   - Το CrossEntropyLoss είναι γενικά πιο κατάλληλο για προβλήματα ταξινόμησης πολλαπλών κατηγοριών, επειδή εκμεταλλεύεται καλύτερα τη δομή των πιθανών κλάσεων.

Σύγκριση Απόδοσης Νευρωνικού Δικτύου, k-NN και NCC

1. Χρόνος Εκπαίδευσης

- **Νευρωνικό Δίκτυο (NN):** Ο χρόνος εκπαίδευσης για το νευρωνικό δίκτυο ήταν **7.52 λεπτά** (452.2 δευτερόλεπτα). Παρότι είναι σημαντικά μεγαλύτερος σε σύγκριση με τις άλλες μεθόδους, αυτή η διαφορά είναι αναμενόμενη λόγω της πολυπλοκότητας.

- **k-NN:** Ο χρόνος εκπαίδευσης ήταν πρακτικά αμελητέος (**0.16 δευτερόλεπτα**), καθώς η μέθοδος δεν απαιτεί εκπαίδευση αλλά βασίζεται στις αποστάσεις μεταξύ δειγμάτων κατά την πρόβλεψη.
- **Nearest Centroid Classifier (NCC):** Χρειάστηκαν **0.49 δευτερόλεπτα**, λόγω του υπολογισμού του κέντρου για κάθε κατηγορία.

2. Χρόνος Πρόβλεψης

- **NN:** Η πρόβλεψη στο test set έγινε εξαιρετικά γρήγορα λόγω της κατασκευής του μοντέλου.
- **k-NN:** Ο χρόνος πρόβλεψης για το test set με k=1 και k=3 ήταν **24.85** και **26.88 δευτερόλεπτα**, αντίστοιχα, ενώ για το training set ήταν **137.32** και **153.45 δευτερόλεπτα**. Η μεγάλη διάρκεια πρόβλεψης οφείλεται στον υπολογισμό των αποστάσεων μεταξύ όλων των δειγμάτων.
- **NCC:** Ήταν ο ταχύτερος για το test set (**0.11 δευτερόλεπτα**) και το training set (**0.58 δευτερόλεπτα**), καθώς υπολογίζει μόνο την απόσταση από τα κέντρα.

3. Ακρίβεια

Μέθοδος	**Ακρίβεια (Training)**	**Ακρίβεια (Test)**
Νευρωνικό Δίκτυο	89.35%	59.19%
k-NN (k=1)	100.00%	35.67%
k-NN (k=3)	57.99%	33.08%
NCC	27.18%	28.12%

- Το **νευρωνικό δίκτυο** υπερτερεί σημαντικά σε ακρίβεια στο test set (**59.19%**), επιτυγχάνοντας καλύτερη γενίκευση.
- Το **k-NN με k=1** παρουσιάζει υπερεκπαίδευση (overfitting) με **100.00%** ακρίβεια στο training set, αλλά χαμηλή απόδοση στο test set (**35.67%**). Με k=3, το overfitting μειώνεται, αλλά η ακρίβεια παραμένει χαμηλή (**33.08%**).
- Το **NCC** είναι η λιγότερο αποδοτική μέθοδος, με χαμηλή ακρίβεια τόσο στο training όσο και στο test set.

4. Επεξήγηση Διαφορών

- **Νευρωνικό Δίκτυο:** Η καλύτερη απόδοση του νευρωνικού οφείλεται στη δυνατότητά του να μαθαίνει πολύπλοκα μοτίβα στα δεδομένα. Ωστόσο, απαιτεί σημαντικό χρόνο για εκπαίδευση.
- **k-NN:** Είναι απλή μέθοδος, αλλά η απόδοση της εξαρτάται από την πυκνότητα των δεδομένων και είναι υπολογιστικά δαπανηρή κατά την πρόβλεψη.
- **NCC:** Είναι γρήγορη, αλλά βασίζεται μόνο στις μέσες αποστάσεις, κάτι που δεν επαρκεί για τη διαχείριση σύνθετων συνόλων δεδομένων, όπως το CIFAR-10.

Συμπέρασμα

Το νευρωνικό δίκτυο είναι η καλύτερη επιλογή για το CIFAR-10, προσφέροντας την υψηλότερη ακρίβεια στο test set και τη δυνατότητα γενίκευσης. Ωστόσο, απαιτεί περισσότερους πόρους και χρόνο σε σχέση με τις απλούστερες μεθόδους. Οι μέθοδοι k-NN και NCC μπορούν να χρησιμοποιηθούν για μικρότερα datasets ή σε περιπτώσεις όπου η ταχύτητα της πρόβλεψης είναι κρίσιμη.