ΤΕΧΝΗΤΗ ΝΟΗΜΟΣΥΝΗ

Μάθημα 2: Προχωρημένες Τεχνικές με τη Βιβλιοθήκη NumPy

Ενότητα 1: Πολυδιάστατοι Πίνακες (Tensors)

Δημιουργία Πίνακα Εικόνας

Οι **πολυδιάστατοι πίνακες (tensors)** είναι βασικοί στη διαχείριση δεδομένων σε τεχνητή νοημοσύνη. Παράδειγμα: Μια έγχρωμη εικόνα μπορεί να αναπαρασταθεί ως tensor 3 διαστάσεων (πλάτος x ύψος x αριθμός καναλιών).

```
import numpy as np

# Δημιουργία τρισδιάστατου πίνακα (π.χ., εικόνα 28x28 pixels με 3 κανάλια χρώματος)
# Οι τιμές είναι ακέραιοι από 0 έως 255
tensor = np.random.randint(0, 256, size=(28, 28, 3), dtype=np.uint8)
print("Διάσταση του πίνακα (tensor):", tensor.shape)

# Πρόσβαση σε ένα pixel της εικόνας
pixel = tensor[10, 15, :]
print("Τιμές του pixel (RGB):", pixel)

# Μέση τιμή για κάθε κανάλι χρώματος
mean_per_channel = np.mean(tensor, axis=(0, 1))
print("Μέση τιμή για κάθε κανάλι (R, G, B):", mean_per_channel)
```

- Οι τρισδιάστατοι πίνακες μπορούν να χρησιμοποιηθούν για την αναπαράσταση εικόνων, δεδομένων αισθητήρων ή δεδομένων χρονικών σειρών.
- Ο άξονας **axis=(0, 1)** υποδηλώνει ότι υπολογίζουμε τη μέση τιμή κατά μήκος των δύο πρώτων διαστάσεων (πλάτος και ύψος).

Φόρτωση Εικόνας και μετατροπή σε πίνακα

```
# 1. Φόρτωση εικόνας
image = Image.open('your_image.jpg')
image_np = np.array(image)
print("Διαστάσεις εικόνας:", image_np.shape)
```

Βασικές Τεχνικές Numpy για Επεξεργασία Εικόνας

1. Κανονικοποίηση Τιμών

Στις περισσότερες εφαρμογές υπολογιστικής όρασης (Computer Vision), οι τιμές RGB των pixel κανονικοποιούνται στο εύρος [0, 1] για καλύτερη απόδοση στα μοντέλα.

```
# Κανονικοποίηση στο εύρος [0, 1]
normalized_tensor = tensor / 255.0
print("Κανονικοποιημένος πίνακας (0-1):")
print(normalized_tensor)
```

2. Υπολογισμός Μέγιστης και Ελάχιστης Τιμής ανά Κανάλι

Μπορούμε να βρούμε τη μέγιστη και την ελάχιστη τιμή για κάθε χρωματικό κανάλι (R, G, B).

```
# Μέγιστη τιμή για κάθε κανάλι
max_per_channel = np.max(tensor, axis=(0, 1))
print("Μέγιστη τιμή για κάθε κανάλι (R, G, B):", max_per_channel)

# Ελάχιστη τιμή για κάθε κανάλι
min_per_channel = np.min(tensor, axis=(0, 1))
print("Ελάχιστη τιμή για κάθε κανάλι (R, G, B):", min_per_channel)
```

3. Εξαγωγή Συνολικής Φωτεινότητας (Grayscale)

Μπορούμε να μετατρέψουμε την εικόνα σε grayscale χρησιμοποιώντας έναν σταθμισμένο μέσο όρο των καναλιών RGB

```
# Συντελεστές για τη μετατροπή σε grayscale
weights = np.array([0.2989, 0.5870, 0.1140])  # Στάθμιση για R, G, B
grayscale_tensor = np.dot(tensor, weights) / 255.0  # Κανονικοποίηση στο [0, 1]
print("Grayscale εικόνα (28x28):")
print(grayscale_tensor.shape)
```

4. Υπολογισμός Διαφοράς Φωτεινότητας Μεταξύ Περιοχών

Μπορούμε να υπολογίσουμε τη μέση φωτεινότητα διαφορετικών περιοχών της εικόνας. Αυτό είναι χρήσιμο για την ανίχνευση ανωμαλιών ή χαρακτηριστικών.

```
# Υπολογισμός μέσης φωτεινότητας της άνω και κάτω μισής εικόνας
upper_half_mean = np.mean(tensor[:14, :, :])
lower_half_mean = np.mean(tensor[14:, :, :])
print("Μέση φωτεινότητα άνω μισής εικόνας:", upper_half_mean)
print("Μέση φωτεινότητα κάτω μισής εικόνας:", lower_half_mean)
```

5. Δημιουργία Ενός Υποσυνόλου Pixel

Μπορούμε να πάρουμε τυχαία pixels για ανάλυση ή προβολή.

```
# Επιλογή 10 τυχαίων pixel
random_pixels = np.random.randint(0, 28, size=(10, 2))  # 10 συντεταγμένες (x, y)
for i, (x, y) in enumerate(random_pixels):
    print(f"Pixel {i+1} ({x}, {y}): RGB={tensor[x, y, :]}")
```

6. Οπτικοποίηση της "Εικόνας"

Μπορούμε να απεικονίσουμε την εικόνα χρησιμοποιώντας τη βιβλιοθήκη Matplotlib.

```
import matplotlib.pyplot as plt

# Εμφάνιση της εικόνας
plt.imshow(tensor)
plt.title("Τυχαία Εικόνα")
plt.axis("off")
plt.show()
```

7. Αλλαγή Ανάλυσης (Downscaling/Upscaling)

Μπορούμε να αλλάξουμε το μέγεθος του tensor, κάτι που είναι συχνό για προεπεξεργασία εικόνων.

```
from skimage.transform import resize

# Μείωση ανάλυσης της εικόνας σε 14x14 pixels
downscaled_tensor = resize(tensor, (14, 14, 3), anti_aliasing=True)
print("Νέα διάσταση του tensor (downscaled):", downscaled_tensor.shape)
```

8. Σύγκριση Μεταξύ Καναλιών

Μπορούμε να υπολογίσουμε τη διαφορά μέσων τιμών ή άλλων χαρακτηριστικών μεταξύ των καναλιών.

```
# Διαφορά μέσης τιμής μεταξύ R και G καναλιών
diff_rg = np.mean(tensor[:, :, 0]) - np.mean(tensor[:, :, 1])
print("Διαφορά μέσης τιμής μεταξύ R και G καναλιών:", diff_rg)
```

9. Αποθήκευση του Tensor ως Εικόνα

Μπορούμε να αποθηκεύσουμε το tensor ως εικόνα για περαιτέρω χρήση.

```
from PIL import Image

# Αποθήκευση ως PNG εικόνα
image = Image.fromarray(tensor.astype('uint8'))
image.save("random_image.png")
print("Η εικόνα αποθηκεύτηκε ως 'random_image.png'.")
```

10. Υπολογισμός Histogram ανά Κανάλι

Μπορούμε να δημιουργήσουμε ένα ιστόγραμμα για κάθε χρωματικό κανάλι, ώστε να κατανοήσουμε την κατανομή των τιμών στα RGB.

```
import matplotlib.pyplot as plt

# Υπολογισμός histogram για κάθε κανάλι
colors = ['Red', 'Green', 'Blue']
for i in range(3):  # 3 κανάλια
    plt.hist(tensor[:, :, i].flatten(), bins=256, alpha=0.6, label=colors[i])

plt.title("Κατανομή Χρωματικών Καναλιών")
plt.xlabel("Τιμή Pixel")
plt.ylabel("Συχνότητα")
plt.legend()
plt.show()
```

12. Υπολογισμός Συνδιασποράς (Covariance Matrix)

Μπορούμε να υπολογίσουμε τον πίνακα συνδιασποράς μεταξύ των RGB καναλιών για να δούμε πώς σχετίζονται.

```
# Αναδιάταξη των δεδομένων σε (n_pixels, 3) για υπολογισμό συνδιασποράς
reshaped_tensor = tensor.reshape(-1, 3)
cov_matrix = np.cov(reshaped_tensor, rowvar=False)

print("Πίνακας Συνδιασποράς RGB:")
print(cov_matrix)
```

13. Εξαγωγή Ενδιαφέρουσας Περιοχής (Region of Interest - ROI)

Μπορούμε να επιλέξουμε μια συγκεκριμένη περιοχή της εικόνας για ανάλυση.

```
# Επιλογή μιας περιοχής 10x10 (ROI)
roi = tensor[10:20, 10:20, :]
print("Region of Interest (ROI):")
print(roi)

# Εμφάνιση της ROI
plt.imshow(roi)
plt.title("Ενδιαφέρουσα Περιοχή (ROI)")
plt.axis("off")
plt.show()
```

ΑΣΚΗΣΕΙΣ

Άσκηση 1: Βασική Επεξεργασία Εικόνας

1. Δημιουργήστε έναν τρισδιάστατο πίνακα 28x28x3, όπου η τρίτη διάσταση αντιπροσωπεύει τα χρωματικά κανάλια RGB, με τιμές από 0 έως 255.
2. Κανονικοποιήστε τον πίνακα ώστε οι τιμές να βρίσκονται στο εύρος [0, 1].
3. Υπολογίστε τη μέγιστη και την ελάχιστη τιμή για κάθε κανάλι (R, G, B).
4. Υπολογίστε τη συνολική φωτεινότητα της εικόνας (χρησιμοποιήστε το μέσο όρο των RGB τιμών).

Άσκηση 2: Ανάλυση Περιοχών και Pixel

1. Υπολογίστε τη διαφορά φωτεινότητας μεταξύ της άνω μισής και της κάτω μισής εικόνας.
2. Εξάγετε 10 τυχαία pixel και εκτυπώστε τις RGB τιμές τους.
3. Δημιουργήστε ένα υποσύνολο της εικόνας (ROI) που περιλαμβάνει τα pixels από [10:20] και [15:25] και υπολογίστε τη μέση τιμή φωτεινότητας σε αυτό το υποσύνολο.

Άσκηση 3: Ανάλυση Καναλιών και Οπτικοποίηση

1. Υπολογίστε τη μέση τιμή και τη διαφορά μέσων τιμών μεταξύ των καναλιών R και G.
2. Δημιουργήστε το ιστόγραμμα της κατανομής τιμών για κάθε κανάλι (R, G, B).
3. Οπτικοποιήστε την εικόνα χρησιμοποιώντας τη βιβλιοθήκη `matplotlib`.

Άσκηση 4: Συνδιασπορά και Αλλαγή Ανάλυσης

1. Υπολογίστε τον πίνακα συνδιασποράς (covariance matrix) μεταξύ των καναλιών RGB.
2. Μειώστε την ανάλυση της εικόνας σε 14x14 pixels και οπτικοποιήστε τη νέα εικόνα.
3. Αποθηκεύστε τη νέα εικόνα σε αρχείο `.png`.