Feladat: Neurális háló tanítása a Keras CIFAR-10 adatbázisán

Ebben a feladatban a Keras által biztosított CIFAR-10 adatbázist fogjuk használni, amely színes képek gyűjteménye, tíz különböző kategóriával (például repülőgép, autó, madár stb.). A cél, hogy egy neurális hálót építsünk és tanítsunk az osztályozási feladatra.

Feladat lépései:

1. Adatok betöltése és előfeldolgozás

- Töltsd be a CIFAR-10 adatbázist a `keras.datasets.cifar10` modul segítségével.
- Skálázd a képek pixelértékeit 0 és 1 közé.
- Alakítsd át a címkéket (labels) one-hot kódolásúvá.

2. Hálózat felépítése

- Építs egy *deep feedforward* neurális hálót a Keras `Sequential` API-jával.
- Használj legalább:
    - Egy bemeneti réteget a képek dimenziójának megfelelően.
    - Két rejtett réteget 256 és 128 neuronnal, **ReLU** aktivációs függvénnyel.
    - Egy kimeneti réteget 10 neuronnal (az osztályok száma), **softmax** aktivációs függvénnyel.

3. Modell tanítása

- Kompilezd a modellt a következő paraméterekkel:
    - Optimalizáló: **Adam**
    - Veszteségfüggvény: **categorical_crossentropy**
    - Kiértékelési metrika: **accuracy**
- Tanítsd a modellt 15 epoch-on keresztül, minibatch mérete legyen 64.

4. Modell kiértékelése

- Értékeld ki a modell teljesítményét a teszthalmazon.
- Vizsgáld meg, hogy milyen pontosságot ér el a hálózat.

```python
# Szükséges csomagok importálása
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Flatten
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.datasets import cifar10
import matplotlib.pyplot as plt

# 1. Adatok betöltése és előfeldolgozás
(x_train, y_train), (x_test, y_test) = cifar10.load_data()

# Adatok normalizálása (0 és 1 közé)
x_train = x_train.astype('float32') / 255.0
x_test = x_test.astype('float32') / 255.0
```

Építsd fel a hálózatot!

```python
# 2. Hálózat felépítése
model = Sequential([
    Flatten(input_shape=(32, 32, 3)),  # Bemeneti réteg (2D képek vektorizálása)
    Dense(256, activation='relu'),
    Dense(128, activation='relu'),
    Dense(10, activation='softmax')  # Kimeneti réteg 10 osztályra
])

# 3. Modell konfigurálása
model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])
```

Modell tanítása:

4. Modell tanítása

history = model.fit(x_train, y_train, validation_split=0.2, epochs=15, batch_size=64, verbose=2)

5. Modell kiértékelése

test_loss, test_accuracy = model.evaluate(x_test, y_test, verbose=0)

print(f"Teszt pontosság: {test_accuracy:.2f}")

```
# 6. Pontosság és veszteség ábrázolása
plt.plot(history.history['accuracy'], label='Train Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```

Kimenet

A kód futtatása után a következőket kapod:

1. A teszthalmazon elért pontosság kiírása.
2. Az epoch-ok során a pontosság (accuracy) és veszteség (loss) görbéi, amelyek segítenek megérteni, hogyan tanult a modell.