KODLAR

```
import os

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt


from keras.models import Sequential

from keras.layers import Dense, Dropout, Flatten, Convolution2D, MaxPooling2D,
BatchNormalization

from keras.optimizers import Adam

from sklearn.model_selection import train_test_split

from keras.preprocessing.image import ImageDataGenerator

from keras.utils.np_utils import to_categorical

import warnings

warnings.filterwarnings("ignore", category=FutureWarning)


# Verilerin yüklenmesi

print("Verileri yüklüyorum...")

TRAIN_PATH = '../input/digit-recognizer/train.csv'

TEST_PATH = '../input/digit-recognizer/test.csv'


# Eğitim verileri

labeled_images = pd.read_csv(TRAIN_PATH)

X = (labeled_images.iloc[:, 1:].values).astype('float32') / 255  # Normalize

y = to_categorical(labeled_images.iloc[:, 0])  # One-hot encoding
```

```
# Test verileri
test_images = pd.read_csv(TEST_PATH)
X_test = test_images.values.astype('float32') / 255  # Normalize

# Verilerin yeniden şekillendirilmesi
X = X.reshape(X.shape[0], 28, 28, 1)
X_test = X_test.reshape(X_test.shape[0], 28, 28, 1)

# Eğitim ve doğrulama verilerinin ayrıştırılması
X_train, X_dev, y_train, y_dev = train_test_split(X, y, test_size=0.02, random_state=42)

# Data augmentation
gen = ImageDataGenerator(rotation_range=7, width_shift_range=0.05,
shear_range=0.1,
                         height_shift_range=0.05, zoom_range=0.05)
batches = gen.flow(X_train, y_train, batch_size=64)
dev_batches = ImageDataGenerator().flow(X_dev, y_dev, batch_size=64)

# CNN Modelinin Tanımlanması
def get_cnn_model():
   model = Sequential([
      Convolution2D(16, (3, 3), padding='same', activation='relu', input_shape=(28, 28,
1)),
      BatchNormalization(axis=-1),
      Convolution2D(32, (3, 3), padding='same', activation='relu'),
      MaxPooling2D(),
      BatchNormalization(axis=-1),
      Convolution2D(64, (3, 3), activation='relu'),
```

```
        BatchNormalization(axis=-1),
        Convolution2D(128, (3, 3), activation='relu'),
        MaxPooling2D(),
        Flatten(),
        BatchNormalization(),
        Dense(256, activation='relu'),
        BatchNormalization(),
        Dense(64, activation='relu'),
        BatchNormalization(),
        Dense(10, activation='softmax')
    ])
    model.compile(Adam(learning_rate=0.001, decay=0.005),
loss='categorical_crossentropy', metrics=['accuracy'])
    return model

print("Modeli eğitiyorum...")
model = get_cnn_model()
history = model.fit(batches, steps_per_epoch=len(batches), epochs=3,
                    validation_data=dev_batches, validation_steps=len(dev_batches))

# Doğrulama verisi üzerindeki performansın değerlendirilmesi
accuracy = model.evaluate(X_dev, y_dev, verbose=0)
print(f"Doğrulama verisindeki kayıp: {accuracy[0]:.4f}, Doğruluk: {accuracy[1]:.4f}")

# Tahminler
predictions = model.predict(X_test, verbose=0)
predictions = np.argmax(predictions, axis=1)
```

```
# Tahmin sonuçlarını CSV dosyasına kaydetme
submissions = pd.DataFrame({"ImageId": list(range(1, len(predictions) + 1)), "Label": predictions})
submissions.to_csv("digit_recognizer_submission.csv", index=False, header=True)
print("Tahmin sonuçları 'digit_recognizer_submission.csv' dosyasına kaydedildi.")
```