T.C. KÜTAHYA DUMLUPINAR ÜNİVERSİTESİ

MÜHENDİSLİK FAKÜLTESİ YÜKSEK DÜZEY PROGRAMLAMA DERSİ

DÖNEM PROJE ÖDEVİ

BERKANT BAŞOĞLU 202213172810 İKİNCİ ÖĞRETİM

DANIŞMAN DOÇ. DR. HASAN TEMURTAŞ

KÜTAHYA, 2024

Digit Recognizer Projesi:

Veri Seti:

Bu veri seti, Kaggle'daki Digit Recognizer yarışmasından alınan, el yazısı ile yazılmış rakamları içeren bir kümedir. Her bir görüntü 28x28 piksel boyutundadır ve piksel değerleri 0 ile 255 arasında değişen gri tonlamalı değerler içerir. Veri seti, modelin eğitimi için 42.000 etiketli örnek (train.csv) ve test için 28.000 etiketsiz örnek (test.csv) sunmaktadır.

Model:

Model, bir yapay sinir ağı (ANN) kullanılarak geliştirilmiştir. Bu modelde 128 nöronlu bir Dense katman ve 10 nöronlu çıkış katmanı (softmax aktivasyonu ile) bulunur. Aşırı öğrenmeyi engellemek için Dropout katmanı eklenmiş ve eğitimde Adam optimizasyon algoritması ile sparse categorical crossentropy kayıp fonksiyonu kullanılmıştır. Model, eğitim verileri ile 5 epoch boyunca eğitilmiştir.

Amaç:

Bu projenin amacı, el yazısı rakamlarını doğru şekilde sınıflandırabilen bir model geliştirmektir. Model, eğitim verisi ile eğitildikten sonra test verisi üzerinde tahminlerde bulunarak doğruluğu değerlendirmektedir. Derin öğrenme teknikleri ile el yazısı tanıma görevini başarıyla gerçekleştirmeyi hedeflemektedir.

kütüphane içeri yükleme

```
import matplotlib.pyplot as plt
import numpy as np
import struct
from array import array
import os
from os.path  import join
import random
```

MNIST Veri Yükleyici Sınıfı

Bu sınıf, eğitim ve test verilerini yükler.

```
class MnistDataloader(object):
    def __init__(self, training_images_filepath,training_labels_filepath,
                 test_images_filepath, test_labels_filepath):
        self.training_images_filepath = training_images_filepath
        self.training_labels_filepath = training_labels_filepath
        self.test_images_filepath = test_images_filepath
        self.test_labels_filepath = test_labels_filepath
```

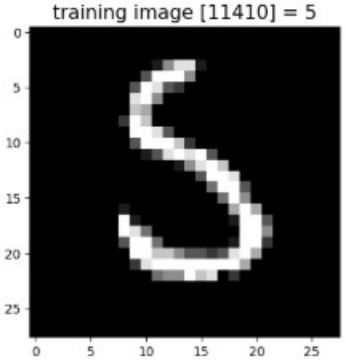
```
    def read_images_labels(self, images_filepath, labels_filepath):
        labels = []
        with open(labels_filepath, 'rb') as file:
            magic, size = struct.unpack(">II", file.read(8))
            if magic != 2049:
                raise ValueError('Magic number mismatch, expected 2049, got {}'.format(magic))
            labels = array("B", file.read())
        with open(images_filepath, 'rb') as file:
            magic, size, rows, cols = struct.unpack(">IIII", file.read(16))
            if magic != 2051:
                raise ValueError('Magic number mismatch, expected 2051, got {}'.format(magic))
            image_data = array("B", file.read())
        images = []
        for i in range(size):
            images.append([0] * rows * cols)
        for i in range(size):
            img = np.array(image_data[i * rows * cols:(i + 1) * rows * cols])
            img = img.reshape(28, 28)
            images[i][:] = img
        return images, labels

    def load_data(self):
        x_train, y_train = self.read_images_labels(self.training_images_filepath, self.training_l
abels_filepath)
        x_test, y_test = self.read_images_labels(self.test_images_filepath, self.test_labels_file
path)
        return (x_train, y_train),(x_test, y_test)
```

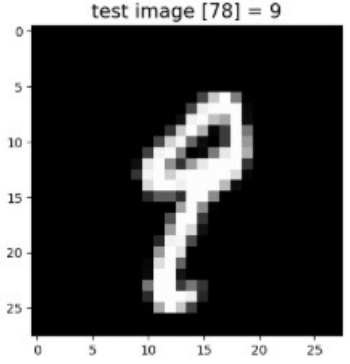
read_images_labels fonksiyonu, dosyaları okur ve resimlerle etiketleri döndürür.
load_data, eğitim ve test verilerini yükler.

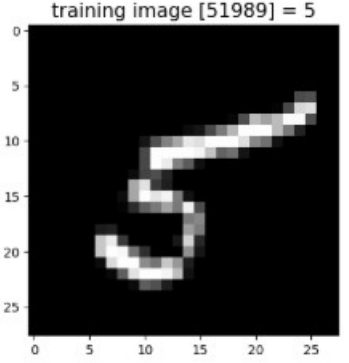
Bazı örnekleri renderlayalım

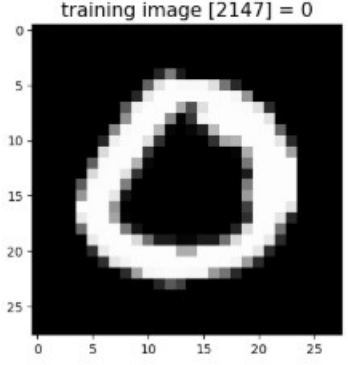
```
def show_images(images, title_texts):
    cols = 5
    rows = int(len(images)/cols) + 1
    plt.figure(figsize=(30,20))
    index = 1
    for x in zip(images, title_texts):
        image = x[0]
        title_text = x[1]
        plt.subplot(rows, cols, index)
        plt.imshow(image, cmap=plt.cm.gray)
        if (title_text != ''):
            plt.title(title_text, fontsize = 15);
        index += 1
```

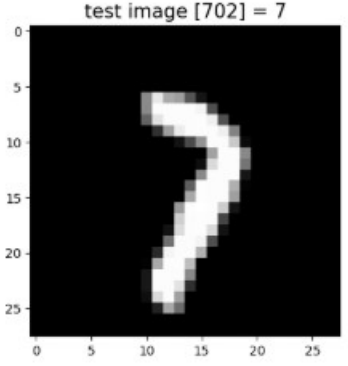
```
images_2_show = []
titles_2_show = []
for i in range(0, 10):
    r = random.randint(1, 60000)
    images_2_show.append(x_train[r])
    titles_2_show.append('training image [' + str(r) + '] = ' + str(y_train[r]))

for i in range(0, 5):
    r = random.randint(1, 10000)
    images_2_show.append(x_test[r])
    titles_2_show.append('test image [' + str(r) + '] = ' + str(y_test[r]))

show_images(images_2_show, titles_2_show)
```

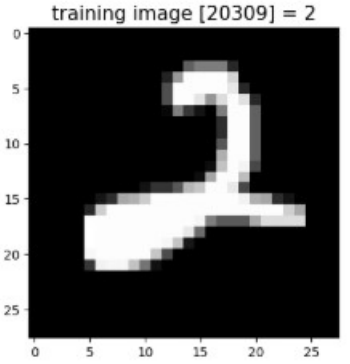

training image [11410] = 5


test image [78] = 9


training image [51989] = 5


training image [2147] = 0


test image [702] = 7


training image [20309] = 2

CNN Modeli

PyTorch kullanılarak bir evrişimli sinir ağı (CNN) modeli oluşturuluyor.

```
import torch
from torch import nn
from torch.utils.data import DataLoader, TensorDataset

model = nn.Sequential(
    nn.Conv2d(1, 28, 5, stride=1),
    nn.MaxPool2d(2, 2),
    nn.Conv2d(28, 26, 5, stride=1),
    nn.ReLU(),
    nn.Flatten(),
    nn.Linear(26 * 8 * 8, 128),
    nn.ReLU(),
    nn.Linear(128, 10),
    nn.Softmax(dim=1)
)
```

Eğitim ve Test

Model, PyTorch'ta CrossEntropyLoss ve SGD kullanılarak eğitilir.

Eğitim sonuçları her epoch için kaydedilir.

Modelin test doğruluğu hesaplanır (%88.85).

```
correct = 0
total = 0

with torch.no_grad():
    for inputs, labels in test_loader:
        outputs = model(inputs)
        _, predicted = torch.max(outputs, 1)
        correct += (predicted == labels).sum().item()
        total += labels.size(0)

accuracy = 100 * correct / total
print(f'Test Doğruluğu: {accuracy:.2f}%')
```

Test Sonucu: Doğruluk oranı yaklaşık %88.85.