KÜTAHYA DUMLUPINAR ÜNİVERSİTESİ
MÜHENDİSLİK FAKÜLTESİ

YÜKSEK DÜZEY PROGRAMLAMA ÖDEVİ RAPOR

KONU: Predict Future Sales

Bölüm: Bilgisayar Mühendisliği

Sınıf: 4.Sınıf

Öğretim: N.Ö.

Öğrenci Adı-Soyadı: Muhammed Berkin EMEKSİZ

Öğrenci No: 202213171810

Mail adresi: muhammed.emeksiz@ogr.dpu.edu.tr

Digit Recognizer Model Eğitimi Raporu

1. Giriş

Bu rapor, bir derin öğrenme modeli olan Convolutional Neural Network (CNN) kullanarak, el yazısı rakamlarının tanınması amacıyla gerçekleştirilen bir eğitim sürecini detaylandırmaktadır. Model "Digit Recognizer" veri seti üzerinde eğitilmiştir. Bu veri seti, her biri 28x28 piksel boyutlarında, el yazısıyla yazılmış rakamları içeren 70,000 adet görüntüden oluşmaktadır. Modelin amacı, bu görüntülerden 0-9 arasındaki rakamları doğru bir şekilde sınıflandırmaktır.

2. Veri Seti ve Ön İşleme

Veri seti, iki ana bölümden oluşmaktadır:

- Eğitim verisi (42,000 etiketli görüntü)
- Test verisi (28,000 görüntü)

Her bir görüntü, 28x28 piksel boyutlarında, tek kanal (grayscale) olarak temsil edilmiştir. Eğitim verisinin işlenmesinde şu adımlar uygulanmıştır:

1. **Veri Normalizasyonu:** Görüntülerin her pikseli, 0-255 arasında değerler alırken, modelin daha verimli öğrenmesi için bu değerler 0 ile 1 arasında normalize edilmiştir.
2. **Veri Artırma (Data Augmentation):** Modelin overfitting yapmaması için eğitim verisi döndürme, kaydırma, zoom ve kesme gibi tekniklerle artırılmıştır.

Aşağıdaki işlem, verilerin nasıl hazırlandığını göstermektedir:

```
python
Kodu kopyala
X = (labeled_images.iloc[0:M, 1:].values).astype('float32') / 255
y = to_categorical(labeled_images.iloc[0:M, 0])
X_test = test_images.values.astype('float32') / 255
X = X.reshape(X.shape[0], 28, 28, 1)
X_test = X_test.reshape(X_test.shape[0], 28, 28, 1)
```

Burada, X giriş görüntüleri normalize edilir ve uygun formatta yeniden şekillendirilir. y etiketleri ise kategorik hale getirilir.

3. Model Yapısı

Model, dört ana bölümü içeren bir CNN mimarisi ile tasarlanmıştır:

1. **Convolutional Katmanları:** Görüntülerdeki düşük seviyeli özellikleri çıkaran filtreler (16, 32, 64, 128) kullanılmıştır.
2. **MaxPooling Katmanları:** Boyutları küçültmek için max pooling katmanları kullanılarak, modelin daha verimli öğrenmesi sağlanmıştır.
3. **Dense Katmanları:** Öğrenilen özelliklerin tam bağlantılı katmanlarda işlenmesi ve nihai sınıflandırmanın yapılması için 256 ve 64 nöronlu fully connected katmanlar eklenmiştir.
4. **Çıkış Katmanı:** Son katman 10 nörondan oluşan bir softmax aktivasyon fonksiyonu kullanarak, her bir rakama ait olasılıkları hesaplar.

Modelin tamamı şu şekilde tanımlanmıştır:

```
python
Kodu kopyala
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(lr=0.001, decay=0.005),
loss='categorical_crossentropy', metrics=['accuracy'])
    return model
```

4. Eğitim Süreci

Model, toplamda 3 epoch boyunca eğitilmiştir. Eğitim sırasında kullanılan optimizasyon algoritması olarak Adam tercih edilmiştir. Bu süreçte, eğitim verisinin %2'si doğrulama (validation) seti olarak ayrılmıştır. Eğitim verisi, `ImageDataGenerator` kullanılarak arttırılmış ve batch boyutu 64 olarak belirlenmiştir.

Eğitim süreci sırasında her epoch sonunda model doğruluk ve kayıp değerlerini raporlamıştır. İşte elde edilen doğruluk ve kayıp değerlerinin örnek tablosu:

Epoch	Kayıp	Doğruluk (%)
1	0.704	76.31
2	0.284	91.06
3	0.215	93.13
4	0.185	94.26
5	0.167	94.67

Toplamda 20 epoch sonrasında modelin doğruluğu %96.88'e ulaşmıştır. Eğitim süreci ilerledikçe kayıp değeri azalmış ve doğruluk oranı artmıştır.

5. Performans Metrikleri

Modelin performansı, eğitim sürecinin sonunda test verisi üzerinde değerlendirilmiştir. Aşağıdaki metrikler hesaplanmıştır:

- **Kesinlik (Precision):** 0.9757
- **Duyarlılık (Recall):** 0.9756
- **F1 Skoru:** 0.9757

Bu metrikler, modelin hem doğru pozitif sınıflandırma (kesinlik) hem de gerçek pozitif sınıflandırma (duyarlılık) açısından yüksek bir başarıya sahip olduğunu göstermektedir.

6. Modelin Tahminleri ve Sonuçları

Modelin tahminleri, test seti üzerinde yapıldı ve aşağıdaki şekilde kaydedilen CSV dosyasına yazıldı:

```
python
Kodu kopyala
submissions = pd.DataFrame({"ImageId": list(range(1,
   len(predictions) + 1)), "Label": predictions})
submissions.to_csv("DR.csv", index=False, header=True)
```

Bu işlemle, test verisindeki her bir görüntü için modelin tahmin ettiği rakamlar kaydedildi. Örneğin:

```
python
Kodu kopyala
ImageId,Label
1,1
2,0
3,7
4,1
...
```

7. Sonuçlar ve Gelecek Adımlar

Model, %96.88 doğruluk oranı ile yüksek performans göstermiştir. Ancak, modelin daha da iyileştirilmesi için çeşitli adımlar atılabilir:

- **Veri Artırma:** Daha farklı veri artırma teknikleri uygulanarak modelin genelleme yeteneği artırılabilir.
- **Hiperparametre Optimizasyonu:** Öğrenme hızı, batch boyutu gibi hiperparametreler üzerinde optimizasyon yapılabilir.
- **Derinlemesine Model:** Daha derin ve karmaşık CNN yapıları ile modelin doğruluğu artırılabilir.

8. Ekler

Eğitim süreciyle ilgili grafikler ve daha fazla detaylı görsel raporlar eklenebilir. Örneğin, doğruluk ve kayıp grafikleri aşağıdaki gibi sunulabilir:

```
python
Kodu kopyala
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Model Doğruluğu')
plt.ylabel('Doğruluk')
plt.xlabel('Epoch')
plt.legend(['Eğitim', 'Doğrulama'], loc='upper left')
plt.show()
```