YÜKSEK DÜZEY PROGRAMLAMA DERSİ

DÖNEM PROJE ÖDEVİ

DİGİT RECOGNİZER PROJESİ SUNUM RAPORU

Öğrencinin Adı Soyadı: Efsun Peynirci

Öğretim: Normal Öğretim

Okul Numarası: 202113171004

Öğretim Üyesinin Adı Soyadı: Doç. Dr. Hasan Temurtaş

Digit Recognizer Projesi Sunum Raporu

1.Proje Tanıtımı

Bu proje, Kaggle platformunda bulunan Digit Recognizer veri setini kullanarak, el yazısı rakamlarını sınıflandırabilen bir Convolutional Neural Network (CNN) modeli geliştirmeyi amaçlamaktadır. Projede, 28x28 piksel boyutunda gri tonlamalı el yazısı rakam görüntüleri hem eğitim hem de test sürecinde kullanılmıştır. Veri seti, 0-9 arası rakamları temsil eden sınıflardan oluşmakta olup, modelin bu sınıfları doğru bir şekilde ayırt edebilmesi için tasarlanmıştır.

Model, TensorFlow kütüphanesi kullanılarak oluşturulmuş ve farklı katmanlarla özelleştirilmiştir. Eğitim sırasında, doğruluk (accuracy) ve kayıp (loss) değerleri takip edilerek modelin performansı optimize edilmiştir. Doğrulama seti üzerindeki başarısı analiz edilmiş ve sonuçlar bir Confusion Matrix ile görselleştirilmiştir. Bunun yanı sıra, kullanıcı tarafından çizilen özel resimler modele verilerek genelleştirme kapasitesi değerlendirilmiştir.

Bu proje, el yazısı rakamların otomatik tanımlanmasına yönelik bir sistem geliştirmek için temel oluşturmakta ve özellikle OCR (Optical Character Recognition), el yazısı tanıma, bankacılıkta çek işleme ve dijital form değerlendirme gibi alanlarda uygulanabilir çözümler sunmaktadır. Ayrıca, makine öğrenimi ve derin öğrenme süreçlerine yönelik pratik bir yaklaşım sunarak bu alandaki yetkinlikleri artırmayı hedeflemektedir.

2. Kullanılan Teknolojiler ve Araçlar

Bu projede, el yazısı rakamların sınıflandırılması için makine öğrenimi ve derin öğrenme tekniklerini destekleyen çeşitli araçlar ve teknolojiler kullanılmıştır:

1. **Python:** Veri işleme, model geliştirme ve analiz süreçlerinde kullanılan temel programlama dili.
2. **TensorFlow:** Convolutional Neural Network (CNN) modelinin geliştirilmesi ve eğitilmesi için kullanılan popüler bir derin öğrenme kütüphanesi.
3. **Scikit-learn (Sklearn):** Model performansını değerlendirmek için Confusion Matrix gibi metriklerin hesaplanmasında kullanılmıştır.
4. **Matplotlib:** Eğitim ve doğrulama süreçlerinden elde edilen doğruluk (accuracy) ve kayıp (loss) metriklerinin görselleştirilmesinde kullanılmıştır.
5. **OpenCV:** Kullanıcı tarafından çizilen özel resimlerin işlenmesi (gri tonlamaya dönüştürme, yeniden boyutlandırma) ve modele uygun hale getirilmesi için kullanılmıştır.

Bu araçlar, model geliştirme sürecinin her aşamasında kolaylık sağlamış ve projenin sonuçlarını analiz etmek için güçlü bir altyapı sunmuştur. TensorFlow ile model eğitimi optimize edilirken, görselleştirme ve veri işleme aşamaları Matplotlib ve OpenCV ile desteklenmiştir.

3. Veri Seti ve Ön İşleme

3.1. Veri Setinin Tanıtımı

Proje kapsamında Kaggle'dan alınan Digit Recognizer veri seti kullanılmıştır. Bu veri seti, 0'dan 9'a kadar olan el yazısı rakamlarını temsil eden görsellerden oluşmaktadır. Görsellerin her biri 28x28 piksel boyutundadır ve gri tonlamalıdır. Veri seti iki ana parçaya ayrılmıştır:

- **Eğitim Veri Seti:** 42.000 örnek içerir ve modelin eğitimi için kullanılmıştır.
- **Test Veri Seti:** 28.000 örnek içerir ve modelin tahmin doğruluğunu değerlendirmek için kullanılmıştır.

3.2. Veri Ön İşleme

Veriler, modelin uygun bir şekilde eğitilmesi için aşağıdaki adımlardan geçirilmiştir:

A.Normalizasyon: Veri setindeki piksel değerleri başlangıçta 0 ile 255 arasında değişmektedir. Bu değerler, [0, 1] aralığına ölçeklendirilmiş ve normalizasyon işlemi gerçekleştirilmiştir. Normalizasyon, modelin daha hızlı ve etkili bir şekilde öğrenmesini sağlar.

Kod:

```
# X (giriş verisi) ve Y (etiketler) olarak ayrıldı.
X = train.drop(columns=['label']).to_numpy()
Y = train['label'].to_numpy()
X_test = test.to_numpy()

# Normalizasyon ve Şekillendirme
X = X / 255.0  # Normalizasyon, piksel değerlerini [0-255] aralığından [0-1] aralığına normalleştirildi
X_test = X_test / 255.0
```

B. Şekillendirme Her bir görüntü, modelin gereksinimlerine uygun olacak şekilde 28x28x1 boyutunda düzenlenmiştir. Bu, CNN modelinin giriş katmanına uygun bir format sağlamıştır.

Kod:

```
X = X.reshape(-1, 28, 28, 1)  # 28x28 boyutunda tek kanal (gri tonlamalı) olarak duzenlendi
X_test = X_test.reshape(-1, 28, 28, 1)
```

C. Eğitim ve Doğrulama Setine Bölme Eğitim veri seti, %80 eğitim ve %20 doğrulama olacak şekilde ayrılmıştır. Bu sayede modelin performansı doğrulama seti üzerinde test edilmiştir.

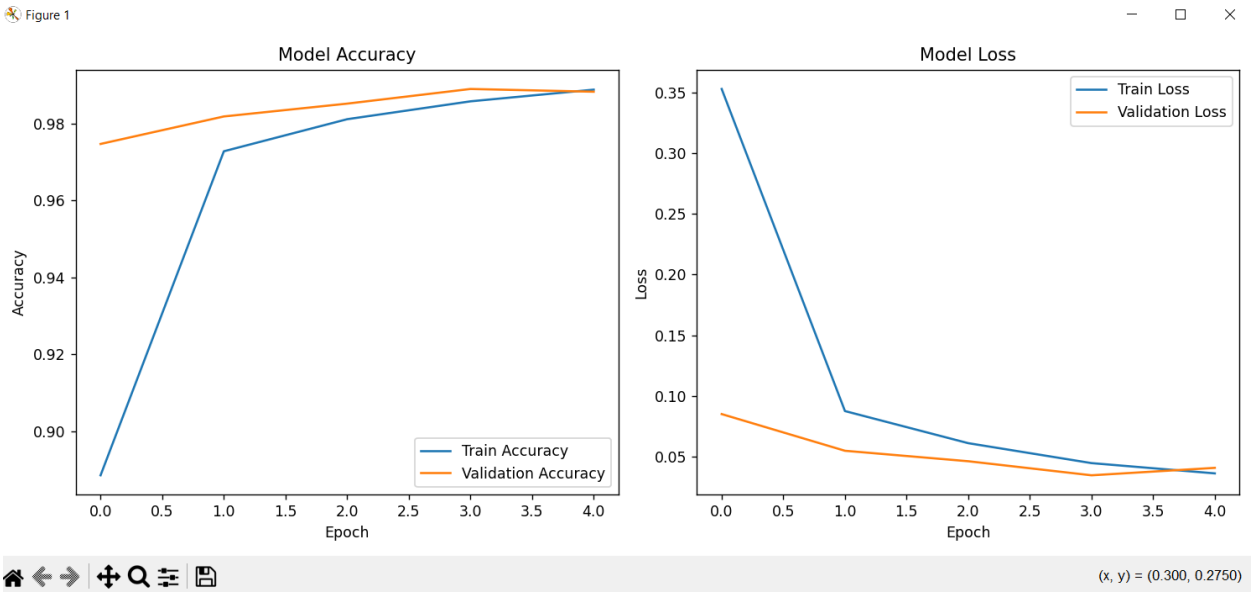
Kod:

```
# Veriler, %80 eğitim ve %20 doğrulama seti olacak şekilde bölündü
X_train, X_val, Y_train, Y_val = train_test_split( *arrays: X, Y, test_size=0.2, random_state=42)
```

3.3. Eğitim ve Doğrulama Sürecinde Elde Edilen Sonuçlar

Veri ön işleme işlemleri tamamlandıktan sonra model eğitilmiş ve eğitim sürecinde doğruluk (accuracy) ve kayıp (loss) değerleri aşağıdaki grafiklerde görselleştirilmiştir:

Eğitim ve Doğrulama Sonuçları:



Accuracy Grafiği: Modelin eğitim doğruluğu (mavi çizgi) her epoch boyunca artmıştır ve doğrulama doğruluğu (turuncu çizgi) ile paralel bir şekilde ilerlemiştir.

Loss Grafiği: Eğitim ve doğrulama kayıp değerleri azalmıştır, bu da modelin başarılı bir öğrenme gerçekleştirdiğini göstermektedir.

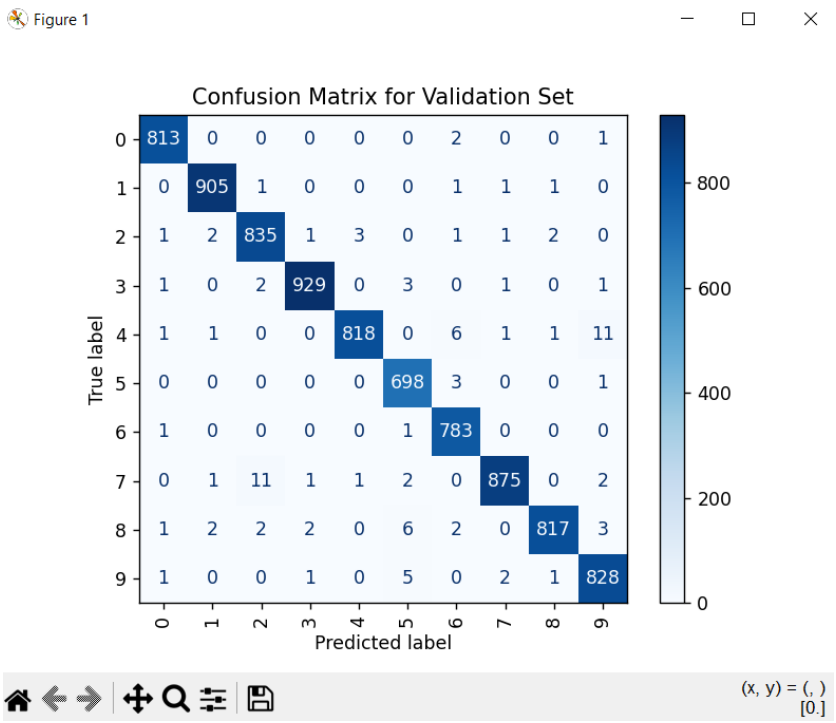
Kod: Accuracy ve Loss Grafiklerinin Çizimi

```
# Eğitim sonuçlarını görselleştirdim. pt.figure ile grafik figürünün boyutunu ayarladım. Genişlik 12 inç, yükseklik 5 inç
plt.figure(figsize=(12, 5))

# Doğruluk grafiği çizildi
plt.subplot( *args: 1, 2, 1) #Grafigin yerlesimi ayarlanir. Bir sayfa üzerinde 1 satır, 2 sütun grafik olmasini ve ilkinin doğruluk grafigi olaca
plt.plot( *args: history.history['accuracy'], label='Train Accuracy') #Train Accuracy cizilir
plt.plot( *args: history.history['val_accuracy'], label='Validation Accuracy') #Validation Accuracy cizilir
plt.title('Model Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()

# Kayıp grafiği cizildi
plt.subplot( *args: 1, 2, 2) #1 satir, 2 sutun grafikten 2. sutun olacagini soyler
plt.plot( *args: history.history['loss'], label='Train Loss') #Train Loss cizilir
plt.plot( *args: history.history['val_loss'], label='Validation Loss') #Validation Loss cizilir
plt.title('Model Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()

plt.tight_layout() #Grafik düzenini otomatik olarak optimize eder, birden fazla alt grafik olduğunda çakışmaları önler.
plt.show() #Cizilen grafikleri ekrana görüntüler.
```

3.4. Confusion Matrix ile Doğrulama Performansı

Validation set üzerindeki tahminler, Confusion Matrix kullanılarak değerlendirilmiştir. Aşağıdaki grafik, modelin doğrulama setinde hangi sınıfları doğru tahmin ettiğini ve hangi sınıflarda hata yaptığını göstermektedir:

Confusion Matrix:



Kod:

```
conf_matrix = confusion_matrix(Y_val, val_predicted_labels, labels=np.arange(10))
disp = ConfusionMatrixDisplay(conf_matrix, display_labels=np.arange(10))
disp.plot(cmap='Blues', xticks_rotation='vertical')
plt.title("Confusion Matrix for Validation Set")
plt.show()
```

3.5. Sonuç

Veri seti ve ön işleme adımları, modelin eğitim performansını optimize etmek için başarılı bir şekilde uygulanmıştır. Eğitim ve doğrulama süreçlerinde elde edilen sonuçlar, modelin öğrenme kapasitesinin yüksek olduğunu ve genelleştirme başarısının iyi bir seviyede olduğunu göstermektedir. Ancak, Confusion Matrix ile analiz edilen sonuçlar, bazı sınıflar arasında karışıklıkların olduğunu ve daha fazla veri ile bu karışıklıkların giderilebileceğini ortaya koymaktadır.

4. Model Mimarisi

Bu proje kapsamında, el yazısı rakamların sınıflandırılması için bir Convolutional Neural Network (CNN) modeli geliştirilmiştir. CNN, özellikle görüntü işleme problemlerinde yaygın olarak kullanılan ve görsellerin özniteliklerini otomatik olarak öğrenen bir sinir ağı mimarisidir. Model, aşağıdaki katmanlardan oluşmaktadır:

4.1. Giriş Katmanı

Modelin ilk katmanı, 28x28 piksel boyutundaki gri tonlamalı görüntüleri giriş olarak kabul eder. Bu giriş formatı, modelin veriyle uyumlu bir şekilde çalışmasını sağlar.

Kod:

```
model = tf.keras.Sequential([
    tf.keras.layers.Input(shape=(28, 28, 1)), # Giriş katmanı
```

4.2. Konvolüsyon Katmanları

Modelde üç adet konvolüsyon katmanı bulunmaktadır. Bu katmanlar, görüntülerdeki kenar, köşe ve diğer öznitelikleri öğrenerek modelin veriyi anlamasını sağlar. Her bir konvolüsyon katmanı, ReLU aktivasyon fonksiyonu kullanılarak negatif değerleri sıfırlar ve modelin doğrusal olmayan öznitelikleri öğrenmesini destekler.

Kod:

```
tf.keras.layers.Conv2D(32, (3, 3), activation='relu'), # Konvolüsyon katmanı
```

```
tf.keras.layers.Conv2D(64, (3, 3), activation='relu'),
```

```
tf.keras.layers.Conv2D(128, (3, 3), activation='relu'),
```

4.3. MaxPooling Katmanları

Her konvolüsyon katmanından sonra bir MaxPooling katmanı eklenmiştir. Bu katmanlar, görüntü boyutunu küçülterek işlem maliyetini azaltır ve en önemli özniteliklerin korunmasını sağlar.

Kod:

```
tf.keras.layers.MaxPooling2D(2, 2),
```

4.4. Flatten Katmanı

Konvolüsyon ve pooling işlemleri sonrasında, elde edilen çok boyutlu veriler tek bir boyuta indirgenir. Bu işlem, verilerin tam bağlı katmanlara aktarılması için gereklidir.

Kod:

```
tf.keras.layers.Flatten(), # Veriyi düzleştirir
```

4.5. Tam Bağlı (Dense) Katmanlar

Modelde bir adet tam bağlı katman (dense layer) ve bir adet çıkış katmanı bulunmaktadır:

- **256 nöronlu dense katman:** Görüntüden elde edilen öznitelikleri öğrenir ve modelin karmaşık ilişkileri anlamasını sağlar. Dropout tekniği ile aşırı öğrenme (overfitting) engellenir.
- **Softmax çıkış katmanı:** 10 sınıf (0-9 rakamları) için olasılık değerleri üreterek, modelin sınıflandırma yapmasını sağlar.

Kod:

```
tf.keras.layers.Dense(256, activation='relu'), # Tam bağlı katman, ne
tf.keras.layers.Dropout(0.5), # Eğitim sırasında rastgele nöronları (
tf.keras.layers.Dense(10, activation='softmax') # Çıkış katmanı
```

4.6. Modelin Yapısı

Modelin genel yapısı, aşağıdaki kod ile oluşturulmuştur:

```
# CNN yani konvolüsyonel sinir agi modeli oluşturuldu.
model = tf.keras.Sequential([
    tf.keras.layers.Input(shape=(28, 28, 1)), # Giriş katmanı
    tf.keras.layers.Conv2D(32, (3, 3), activation='relu'), # Konvolüs
    tf.keras.layers.MaxPooling2D(2, 2), # Maksimum havuzlama, pooling
    tf.keras.layers.Conv2D(64, (3, 3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2, 2),
    tf.keras.layers.Conv2D(128, (3, 3), activation='relu'),
    tf.keras.layers.Flatten(), # Veriyi düzleştirir
    tf.keras.layers.Dense(256, activation='relu'), # Tam bağlı katman
    tf.keras.layers.Dropout(0.5), # Eğitim sırasında rastgele nöronla
    tf.keras.layers.Dense(10, activation='softmax') # Çıkış katmanı
])
```

4.7. Sonuç

Modelin tasarımı, el yazısı rakamların sınıflandırılmasında etkili bir mimari sunmuştur. Konvolüsyon katmanları ile özniteliklerin öğrenilmesi ve dense katmanlar ile sınıflandırma işleminin gerçekleştirilmesi, proje hedeflerine ulaşılmasını sağlamıştır.

5. Model Eğitimi

Model eğitimi, verilerin model tarafından işlenerek sınıflandırmayı öğrenmesi sürecidir. Bu projede, Kaggle'dan alınan Digit Recognizer veri seti kullanılarak 5 epoch boyunca bir Convolutional Neural Network (CNN) modeli eğitilmiştir. Eğitim süreci boyunca aşağıdaki adımlar gerçekleştirilmiştir:

5.1. Eğitim Parametreleri

Model, aşağıdaki parametreler ile eğitilmiştir:

- **Epoch Sayısı:** Modelin veri üzerinde kaç kez öğrenme gerçekleştireceği belirlenmiştir (5 epoch).
- **Batch Size:** Veriler 128'lik mini gruplara ayrılarak işlem yapılmıştır.
- **Kayıp Fonksiyonu:** sparse_categorical_crossentropy kullanılmış ve modelin tahmin sonuçları ile gerçek değerler arasındaki fark hesaplanmıştır.
- **Optimizasyon Algoritması:** adam optimizer, modelin ağırlıklarını optimize etmek için kullanılmıştır.

Kod:

```
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
```

5.2. Eğitim Süreci

Model, eğitim seti kullanılarak eğitilmiş ve doğrulama seti ile performansı değerlendirilmiştir. Her epoch sonunda, eğitim doğruluğu (accuracy) ve kayıp (loss) değerleri ölçülerek modelin öğrenme durumu gözlemlenmiştir.

Kod:

```
history = model.fit(X_train, Y_train, epochs=5, batch_size=128, validation_data=(X_val, Y_val))
```

5.3. Eğitim Süreci Çıktıları

Her epoch sonunda modelin performansı kaydedilmiş ve aşağıdaki gibi doğruluk ve kayıp değerleri hesaplanmıştır:

```
Epoch 1/5
263/263 ━━━━━━━━━━━━━━━━━━━━ 6s 21ms/step - accuracy: 0.7628 - loss: 0.7407 - val_accuracy: 0.9746 - val_loss: 0.0853
Epoch 2/5
263/263 ━━━━━━━━━━━━━━━━━━━━ 5s 20ms/step - accuracy: 0.9708 - loss: 0.0946 - val_accuracy: 0.9818 - val_loss: 0.0551
Epoch 3/5
263/263 ━━━━━━━━━━━━━━━━━━━━ 5s 20ms/step - accuracy: 0.9802 - loss: 0.0619 - val_accuracy: 0.9851 - val_loss: 0.0465
Epoch 4/5
263/263 ━━━━━━━━━━━━━━━━━━━━ 5s 19ms/step - accuracy: 0.9862 - loss: 0.0456 - val_accuracy: 0.9889 - val_loss: 0.0349
Epoch 5/5
263/263 ━━━━━━━━━━━━━━━━━━━━ 5s 19ms/step - accuracy: 0.9900 - loss: 0.0348 - val_accuracy: 0.9882 - val_loss: 0.0410
```

Bu sonuçlar, modelin her epoch boyunca daha doğru tahminler yapmayı öğrendiğini ve doğrulama seti üzerinde genelleştirme başarısının yüksek olduğunu göstermektedir.

6. Validation Set Performansı

Validation set, modelin genelleştirme yeteneğini değerlendirmek için eğitim sırasında kullanılmamış verilerdir. Bu set üzerindeki analiz, modelin gerçek dünyadaki performansını tahmin etmeye yardımcı olur.

Modelin validation set üzerindeki doğruluk oranı son epoch itibarıyla %98.82 olarak ölçülmüştür. Bu sonuç, modelin sadece eğitim verisine değil, aynı zamanda daha önce görmediği verilere de başarılı bir şekilde uyum sağladığını göstermektedir.

Hataların dağılımını ve modelin hangi sınıflarda zorlandığını incelemek için Confusion Matrix kullanılmıştır. Bu analiz, modelin sınıflandırmada özellikle benzer görsellere sahip rakamlar (örneğin, "8" ve "3") arasında hata yapabileceğini ortaya koymuştur.

Validation performansı, modelin genelleştirme kapasitesinin yüksek olduğunu ve aşırı öğrenmeden kaçınıldığını kanıtlamaktadır.

7. Özel Resimlerle Test

Modelin gerçek dünya senaryolarında performansını değerlendirmek için özel olarak çizilmiş el yazısı rakamlar üzerinde test gerçekleştirilmiştir. Bu test, modelin daha önce eğitim setinde bulunmayan verilerle nasıl bir performans sergilediğini anlamayı amaçlamaktadır.

7.1. Özel Resimlerin Hazırlanması

Kullanıcı tarafından elle çizilen toplam 5 rakam, model üzerinde test edilmiştir. Bu resimler:

- Gri tonlamaya dönüştürülmüş,
- 28x28 piksel boyutuna küçültülmüş,

- 0-1 aralığında normalize edilmiştir.

Kod:

```
if img is not None:
    img = cv.resize(img,  dsize: (28, 28))  # 28x28 boyutuna küçült
    img = img / 255.0  # Normalize et
    img = img.reshape(28, 28, 1)
    custom_images.append(img)
```

7.2. Model Tahmini:

Model, işlenmiş her resim için tahmin edilen sınıfı döndürmüştür. Tahminler, modelin çıktı olarak döndürdüğü olasılık değerlerinden en yüksek değerin seçilmesiyle yapılmıştır.

Kod:

```
# Tahmin yap
custom_predictions = model.predict(X_custom)
custom_predicted_labels = np.argmax(custom_predictions, axis=1)
```

7.3. Sonuçların Görselleştirilmesi:

Her bir resim için, modelin tahmini ve gerçek etiket ekran üzerinde gösterilmiştir. Bu işlem, modelin hangi resimlerde doğru veya yanlış tahmin yaptığını hızlıca anlamak için kullanılmıştır.

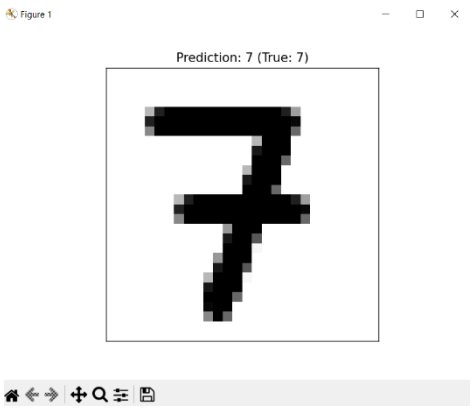
Kod:

```
# Ozel resimlerin tahmin sonuclarını görsellestirildi
for idx, pred in enumerate(custom_predictions):
    predicted_label = np.argmax(pred) #Modelin verdiği tahmin olasılıkları arasından en yüksek değ
    print(f"Image {idx + 1}: Predicted Label: {predicted_label}, True Label: {true_labels[idx]}")
    plt.imshow(X_custom[idx].reshape(28, 28), cmap='gray')
    plt.title(f"Prediction: {predicted_label} (True: {true_labels[idx]})")
    plt.xticks([])  # X ekseninde degerleri kaldırır
    plt.yticks([])  # Y ekseninde degerleri kaldırır
    plt.show()
```

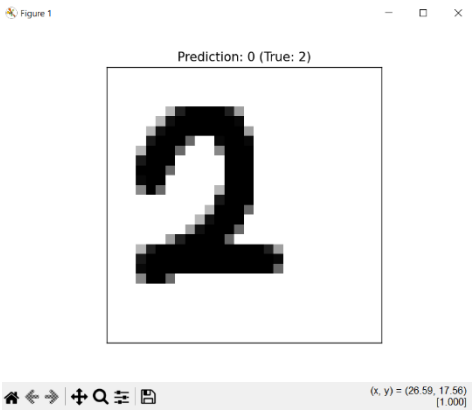
7.4. Sonuç Görselleri

Model, özel resimlere ait tahminlerini gerçekleştirmiş ve her resim için tahmin edilen etiketler ile gerçek etiketler karşılaştırılmıştır. Test sonuçları, modelin çoğu özel resmi doğru bir şekilde sınıflandırdığını, ancak bazı durumlarda hata yaptığını göstermiştir.

- **Doğru Tahmin Örneği:** Çizilen "7" rakamı model tarafından doğru şekilde "7" olarak tahmin edilmiştir.



- **Yanlış Tahmin Örneği:** Çizilen "2" rakamı model tarafından "0" olarak tahmin edilmiştir.



7.5. Confusion Matrix ile Analiz

Test edilen özel resimlerin sonuçları, modelin doğruluğunu ve hata yaptığı sınıfları daha iyi anlamak için bir Confusion Matrix ile görselleştirilmiştir. Confusion Matrix, modelin doğru tahmin ettiği sınıflarla yanlış sınıflandırmaları net bir şekilde göstermektedir.

Confusion Matrix Oluşturma Süreci:

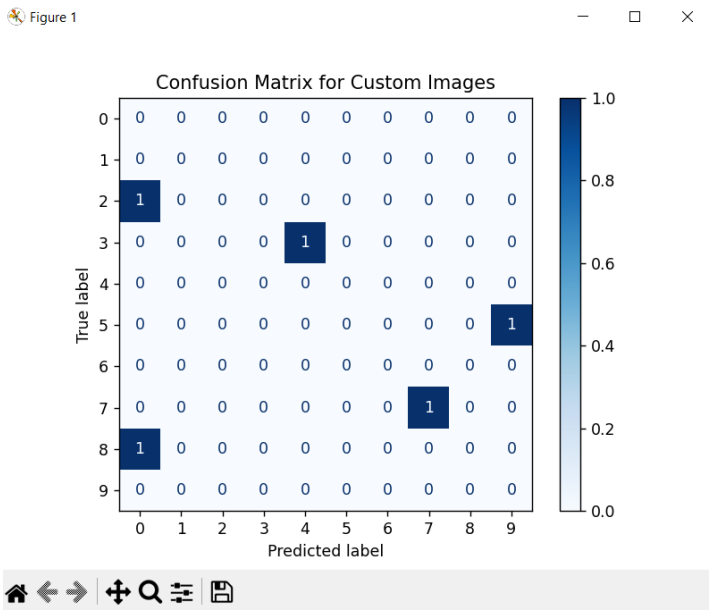
- Özel resimlerin gerçek etiketleri (örneğin: 2, 3, 5, 7, 8) belirlenmiştir.
- Model, her resim için tahmin edilen sınıfı döndürmüştür.
- Gerçek etiketler ve tahmin edilen etiketler karşılaştırılarak Confusion Matrix oluşturulmuştur.

Kod:

```
custom_conf_matrix = confusion_matrix(true_labels, custom_predicted_labels, labels=np.arange(10))
disp_custom = ConfusionMatrixDisplay(custom_conf_matrix, display_labels=np.arange(10))
disp_custom.plot(cmap='Blues')
plt.title("Confusion Matrix for Custom Images")
plt.show()
```

Confusion Matrix, modelin doğru tahmin ettiği sınıfları (diyagonal elemanlar) ve yanlış sınıflandırmaları (diyagonal dışındaki elemanlar) açıkça göstermektedir. Aşağıda, özel resimlerle yapılan testin Confusion Matrix'i sunulmuştur:

Confusion Matrix:



7.6. Sonuç

Özel resimlerle yapılan test, modelin genelleştirme yeteneğini bir kez daha doğrulamıştır. Model, çizimlerin çoğunu başarılı bir şekilde sınıflandırmış, ancak bazı durumlarda hatalar yapmıştır. Bu sonuçlar, modelin doğruluğunun çizim kalitesi ve verilerin eğitim setine ne kadar benzer olduğu gibi faktörlere bağlı olduğunu göstermektedir. Özel resimlerle test, modelin gerçek dünya uygulamalarında kullanılabilirliğini analiz etmek için önemli bir adım olmuştur.

8. Sonuç ve Gelecekteki Çalışmalar

Bu proje kapsamında geliştirilen derin öğrenme tabanlı model, doğrulama setinde elde ettiği %98.82 doğruluk oranı ile el yazısı rakam sınıflandırma görevinde oldukça başarılı bir performans sergilemiştir. Eğitim ve doğrulama süreci boyunca modelin kayıp ve doğruluk metrikleri tutarlı bir şekilde ilerlemiş, overfitting gibi istenmeyen durumların önüne geçilmiştir. Özel resimler üzerinde yapılan testler, modelin daha önce görmediği veriler üzerindeki genelleştirme yeteneğini ortaya koymuş, ancak özellikle benzer rakamlar arasında (örneğin "2" ve "0") karışıklık yaşanabileceği gözlemlenmiştir.

Modelin genel başarısı, kullanılan yöntemlerin ve model mimarisinin bu problem için uygun olduğunu göstermektedir. Ancak, modelin hatalarını analiz ettiğimizde, bazı durumlarda sınıflandırma performansını artırmak için ek iyileştirmelerin yapılabileceği fark edilmiştir. Örneğin, daha fazla ve çeşitli

el yazısı örneğiyle veri seti genişletilebilir veya modelin hassasiyetini artırmak için farklı veri artırma teknikleri uygulanabilir.

Gelecekteki çalışmalar kapsamında, modelin genelleştirme yeteneğini daha da geliştirmek ve performansını artırmak adına aşağıdaki adımlar önerilebilir:

- **Daha Zengin Veri Setleri:** Daha fazla el yazısı örneği içeren, farklı dillerden ve yazı tiplerinden oluşan veri setleri eklenerek modelin farklı yazı stillerine karşı daha sağlam hale getirilmesi sağlanabilir.
- **Veri Artırma:** Veri setinde çeşitlilik oluşturmak için dönüşüm (rotation), ölçekleme (scaling) ve bozulma (distortion) gibi veri artırma teknikleri uygulanabilir.
- **Model Optimizasyonu:** Modelin daha hafif ve verimli hale getirilmesi için optimizasyon yöntemleri uygulanabilir, bu da mobil ve gömülü sistemlerde gerçek zamanlı kullanım için faydalı olacaktır.
- **Hata Analizi ve İyileştirme:** Confusion matrix'ten elde edilen sonuçlar doğrultusunda modelin sınıflandırmada hata yaptığı durumlar detaylıca incelenip iyileştirme yöntemleri geliştirilebilir.

Sonuç olarak, bu proje, el yazısı rakam sınıflandırma problemini çözme yolunda başarılı bir adım olmuş ve gelecekte benzer projeler için bir temel oluşturmuştur. Ulaşılan sonuçlar, modelin geniş bir uygulama yelpazesinde kullanılabileceğini göstermekte ve özellikle gerçek zamanlı sistemlerde kullanım için yeni fırsatlar sunmaktadır.