MNIST Rakam Tanıma Model Eğitimi

1. Giriş

Bu rapor, MNIST veri seti üzerinde gerçekleştirilen iki farklı rakam tanıma çalışmasının sonuçlarını ve yönremlerini açıklamaktadır.

İlk çalışma, NumPy kütüphanesi kullanılarak sıfırdan bir yapay sinir ağı (YSA) implementasyonunu ve analizini içermektedir.

İkinci çalışma ise TensorFlow ve TensorFlow Datasets kullanılarak daha gelişmiş bir sinir ağı pipeline modelinin oluşturulması ve eğitilmesi üzerine odaklanmaktadır. Rapor, her iki çalışmanın metodolojilerini, sonuçlarını ve karşılaştırmalı analizlerini sunarak, MNIST veri seti üzerinde farklı yaklaşımların etkinliğini değerlendirmeyi amaçlamaktadır.

2. Bölüm 1: NumPy ile Sıfırdan Yapay Sinir Ağı Implementasyonu

Colab da görüntüle  github da görüntüle 

2.1. Metodoloji

Bu bölümde, MNIST veri seti üzerinde temel bir YSA modeli sıfırdan implemente edilmiştir. Model, bir girdi katmanı, bir gizli katman ve bir çıktı katmanından oluşmaktadır. Aktivasyon fonksiyonları olarak ReLU (gizli katman) ve Softmax (çıktı katmanı) kullanılmıştır. Modelin eğitimi, mini-batch gradyan inişi (mini-batch gradient descent) algoritması ile gerçekleştirilmiştir. Performans değerlendirmesi için doğruluk, precision, recall ve F1 skorları hesaplanmıştır.

2.2. Veri Ön İşleme

MNIST veri seti, TensorFlow Keras kütüphanesi kullanılarak yüklenmiş ve normalize edilmiştir. Veri seti, eğitim, doğrulama ve test setlerine ayrılmıştır.

2.3. Model Mimarisi ve Eğitimi

Model, 784 girdi nöronu (28x28 piksel), 128 gizli nöron ve 10 çıktı nöronu (0-9 rakamları) içermektedir. Ağırlıklar, küçük rastgele değerlerle başlatılmış ve bias'lar sıfır olarak ayarlanmıştır. Model, 10 epoch boyunca eğitilmiş ve her epoch sonunda doğrulama seti üzerinde performansı izlenmiştir.

2.4. Sonuçlar

Model, test seti üzerinde %90'ın üzerinde bir doğruluk oranına ulaşmıştır. Ancak, daha gelişmiş modellere kıyasla performansı sınırlı kalmıştır.

3. Bölüm 2: TensorFlow ile İleri Düzey Model Eğitimi

Colab da görüntüle  github da görüntüle 

3.1. Metodoloji

Bu bölümde, TensorFlow kullanılarak daha gelişmiş bir sinir ağı pipeline modeli oluşturulmuştur. Model, farklı optimizasyon (Adam, SGD) ve regularizasyon (Dropout, Batch Normalization) teknikleri kullanılarak eğitilmiştir. Modelin performansı, test seti üzerinde doğruluk ve kayıp değerleri ile değerlendirilmiştir.

3.2. Veri Ön İşleme

MNIST veri seti, TensorFlow Datasets kullanılarak yüklenmiş ve normalize edilmiştir. Veri seti, eğitim ve test setlerine ayrılmıştır. Veri seti, batch'lere bölünmüş ve prefetch yöntemi ile optimize edilmiştir.

3.3. Model Mimarileri ve Eğitimi

Farklı model mimarileri denenmiştir:

- **Temel Model (Adam)**: Tek gizli katman (128 nöron), Adam optimizasyonu.
- **Temel Model (SGD)**: Tek gizli katman (128 nöron), SGD optimizasyonu.
- **Dropout Modeli**: Tek gizli katman (128 nöron), Dropout regularizasyonu.
- **Batch Normalization Modeli**: Tek gizli katman (128 nöron), Batch Normalization regularizasyonu.
- **Gelişmiş Model (Batch Normalization + Dropout)**: İki gizli katman (128 ve 64 nöron), Batch Normalization ve Dropout regularizasyonu.

3.4. Sonuçlar

Gelişmiş modeller, özellikle Batch Normalization kullanılan modeller, %97'nin üzerinde doğruluk oranlarına ulaşmıştır. Adam optimizasyonu, SGD'ye göre daha hızlı yakınsama ve daha iyi sonuçlar sağlamıştır. Regularizasyon teknikleri, overfitting'i azaltmada etkili olmuştur.

4. Karşılaştırmalı Analiz

4.1. Performans Karşılaştırması

Model	Doğruluk (%)	Optimizasyon	Regularizasyon
NumPy YSA	90+	Mini-batch gradyan inişi	Yok
TensorFlow Temel (Adam)	97+	Adam	Yok
TensorFlow Temel (SGD)	91	SGD	Yok
TensorFlow Dropout	97	Adam	Dropout
TensorFlow Batch Normalization	97+	Adam	Batch Normalization
TensorFlow Gelişmiş (BatchNorm + Dropout)	97+	Adam	Batch Normalization + Dropout

4.2. Model Karmaşıklığı ve Eğitim Süresi

TensorFlow modelleri, NumPy YSA modeline göre daha karmaşık ve eğitim süreleri daha uzundur. Ancak, daha yüksek doğruluk oranları sağlamışlardır.

4.3. Hiperparametre Optimizasyonu

TensorFlow modellerinde, hiperparametre optimizasyonu (öğrenme oranı, batch boyutu, dropout oranı vb.) performansı önemli ölçüde etkilemektedir. NumPy YSA modelinde ise hiperparametre optimizasyonu yapılmamıştır

5. İyileştirme Önerileri

- **Evrişimli Sinir Ağları (CNN)**: MNIST gibi görüntü sınıflandırma problemlerinde CNN mimarisi kullanmak performansı önemli ölçüde artırabilir.
- **Hiperparametre Optimizasyonu**: Grid search veya random search yöntemleri ile hiperparametre optimizasyonu yapılabilir.
- **Veri Artırma (Data Augmentation)**: Eğitim veri setini zenginleştirmek için veri artırma teknikleri kullanılabilir.
- **Öğrenme Oranı Ayarlama (Learning Rate Scheduling)**: Eğitim sırasında öğrenme oranını dinamik olarak ayarlamak, daha iyi yakınsama sağlayabilir.
- **Erken Durdurma (Early Stopping)**: Overfitting'i önlemek için erken durdurma yöntemi kullanılabilir.
- **Model Entegrasyonu (Model Ensembling)**: Farklı modellerin sonuçlarını birleştirerek(Voting) daha iyi bir model oluşturulabilir.

6. Sonuç

Bu rapor, MNIST veri seti üzerinde farklı yaklaşımların karşılaştırmalı bir analizini sunmaktadır. TensorFlow ile daha gelişmiş modeller, özellikle Batch Normalization kullanıldığında, daha yüksek doğruluk oranlarına ulaşmıştır. Ancak, NumPy ile sıfırdan YSA implementasyonu da temel bir anlayış sağlamaktadır.