MNIST El Yazısı Rakam Tanıma Projesi

Hazırlayan: Ediz Savaş
Tarih: Haziran 2025
Süre: 10–14 gün (tam zamanlı bireysel geliştirme)

Giriş

Bu proje, el yazısıyla yazılmış rakamları tanıyabilen bir yapay zeka sistemini baştan sona tamamen kendi yazılmış sinir ağı ile gerçekleştirme amacıyla geliştirilmiştir. MNIST veri seti kullanılarak, herhangi bir dış yapay zeka kütüphanesi kullanmadan, yalnızca NumPy gibi temel bilimsel kütüphanelerle sinir ağı sıfırdan kodlanmıştır.

Projenin temel hedefi, hem teorik olarak derin öğrenme mantığının kavranmasını sağlamak hem de uygulamalı bir arayüz ile son kullanıcının sisteme etkileşimli şekilde veri girişi yapabilmesini mümkün kılmaktır.

Teknik Özellikler

• **Sinir Ağı Mimarisi:**

•Giriş katmanı: 784 nöron (28x28 piksel)
•Gizli katmanlar: 256 ve 128 nöron
•Aktivasyon fonksiyonları: ReLU (gizli katmanlar), Softmax (çıkış katmanı)

•**Eğitim Detayları:**

•Kayıp fonksiyonu: Categorical Cross Entropy
•Erken durdurma mekanizması (patience = 20, delta = 1e-6)
•Öğrenme oranı: 0.05 (zamanla azalan)
•Batch size: 32
•Normalizasyon: -1 ila 1 aralığına getirilen girdi değerleri Özel veri ekleme (>=300 örnek)

•**Eğitim Sonuçları:**

•Test Doğruluğu: ~%98
•GUI tahmin doğruluğu: görsel olarak doğrulanabilir

GUI Arayüz Detayları

• **Kullanılan Teknoloji:** PyQt6

•**Temel Özellikler:**

•Fareyle çizim yapılabilen alan (280x280 piksel)
•"Tahmin Et" ve "Temizle" butonları
•Çıktıyı anlık olarak gösteren tahmin paneli
•Kullanıcının yeni veri etiketleyip kaydedebileceği alan
•**Veri Kaydetme:** Kullanıcıdan alınan çizim, merkezlenip normalize edilerek custom_data/ klasörüne image_.npy ve label_.npy olarak kaydedilir.

Kullanılan Kütüphaneler

• **NumPy:** Tüm matrissel işlemler
•**Matplotlib:** Eğitim sürecindeki doğruluk ve kayıp grafiklerinin çizimi
•**SciPy:** center_of_mass ve shift ile çizimlerin merkezlenmesi
•**PyQt6:** Grafiksel kullanıcı arayüz (GUI)
•**Gzip / OS:** MNIST veri setinin açılması ve dosya yönetimi

Yenilikçi Yönler

• Sinir ağı tamamen elle yazıldı, PyTorch/TensorFlow gibi framework'ler kullanılmadı.
•Erken durdurma ve adaptif öğrenme oranı manuel olarak entegre edildi.
•Kullanıcı arayüzü sadece tahmin için değil, aynı zamanda yeni veri oluşturma ve eğitime dahil etme için tasarlandı.
•Gerçek zamanlı tahmin ve merkezleme sistemiyle düşük kaliteli çizimlerin doğruluğu artırıldı.
•Otomatik olarak en iyi modelin kaydedilmesi ve grafiklerin indeksli olarak saklanması sağlandı.

Projeyi Yaparken Yaşadıklarım

Projeye ilk başladığımda MNIST projeleri hakkında neredeyse hiçbir bilgim yoktu. Daha önce yalnızca birkaç basit yapay sinir ağı örneği hazırlamıştım. Bu nedenle en büyük zorluk başlangıç için yeterli bilgiyi toplamak oldu. Yaklaşık 8–9 gün süresince temel kavramları öğrenmeye çalıştım. Projenin ilerleyişi, sürekli olarak “bir şey yaz → çalışmadı → araştır → düzelt” şeklinde devam etti. Ama sonunda ihtiyaç duyduğum bilgiye ulaşıp sistemi başarıyla kurmayı başardım.

Geri yayılım fonksiyonunu yazarken ilk 30 dakikada hiçbir şey anlamadım. Yazdığım kodun çalışmaması da zaten yeterince açıklayıcıydı. Sonrasında yaklaşık 4 saat boyunca bu fonksiyonu defalarca baştan yazdım. Ya çalışmadı ya da istediğim doğruluk oranına ulaşmadı. Ancak inatla sürdürdüğüm çabalar sonunda başarılı oldum

GUI tarafında PyQt6 deneyimim olduğu için süreci daha kolay yönettim. Arayüz kısmı, projenin diğer bölümlerine kıyasla çok daha sorunsuz ilerledi.

Kullanıcının veri girip bunu veri setine dahil edebilmesi özelliği MNIST projelerinde nadiren görülür. Araştırdığım projelerin hiçbirinde böyle bir sistemle karşılaşmadım. Bu nedenle bu özelliği eklerken veri seti dengesini bozup bozmayacağı konusunda ciddi tereddütlerim oldu. Özellikle yanlışlıkla sadece 3 özel veri ile 60.000 örneklik veri setini birleştirdiğimde oluşan bozulmayı unutamıyorum. Ağırlıklar dağıldı, bias değerleri kaydı. Bu yanlış veri ile elde ettiğim grafik hâlâ “0.png” olarak kayıtlı. Daha sonra “300 veri sınırı” sistemini bu sorunu önlemek için ekledim. Gerçekten en korktuğum anlardan biriydi.

Sonuç ve Değerlendirme

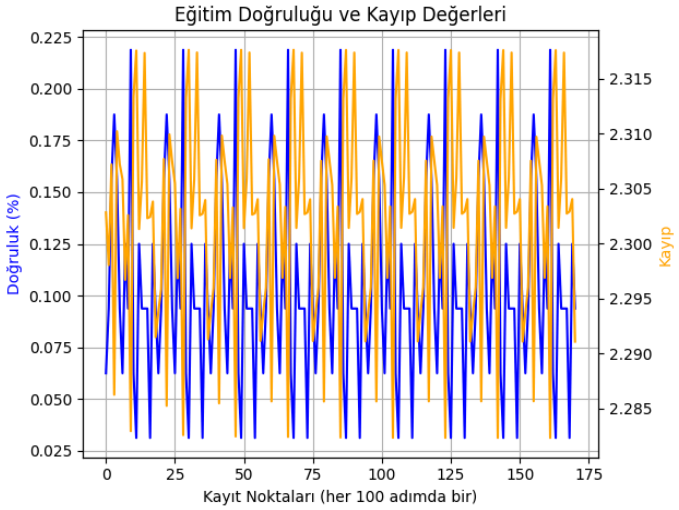
Bu proje, sıfırdan sinir ağı yazılmasından GUI entegrasyonuna kadar tüm aşamalarıyla yapay zeka geliştirme konusundaki pratiğimi derinleştiren bir çalışma oldu. Gerçek dünya uygulamaları için modüler yapı kurmak, veri işleme ve tahmin doğruluğunu optimize etmek gibi birçok temel beceriyi kazandım.

Projeyi geliştirirken teorik bilgilerimi pratik deneyimle birleştirerek yapay zeka sistemlerine dair çok yönlü bir anlayış edindim. Bu projenin, üniversiteye kabul sürecinde hem teknik yeterliliğimi hem de kendi başıma derin öğrenme uygulayabildiğimi gösteren önemli bir referans olacağına inanıyorum.

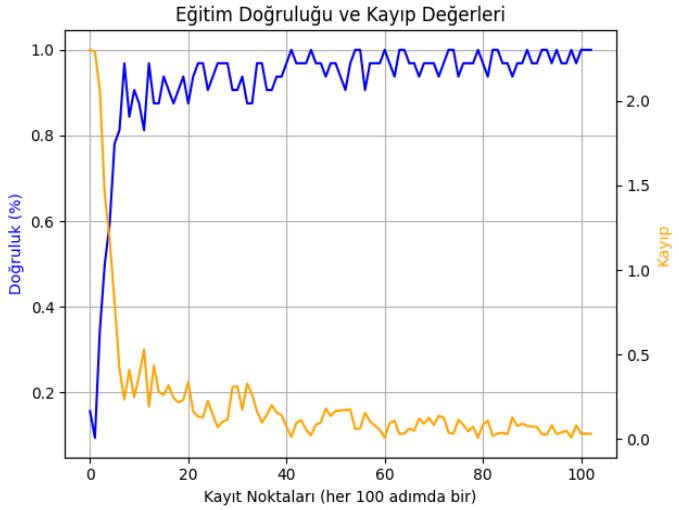
Ekler

• **Eğitim Grafikleri:**
Kayıp ve doğruluk grafikleri:
• **Eğitilen Model:**
graph/ klasörü modelWeights/ ve best_model/ klasörlerinde ağırlık ve bias dosyaları (final_weight_npy, final_bias.npy)
• **Kayıt Edilen Özel Veriler:** custom_data/ klasörü
• **GUI Kodları:** gui.py
• **Model ve Eğitim Kodları:** Wow.py

Grafik Çıktıları



— Yanlışlıkla Custom Data Kullanımı Sonucu Bozulmuş Model Grafiği



— Şu An Kullanılan Eğitimin Sonucu (En Güncel ve Stabil Model Çıktısı)