Transformer Tabanlı Hayvan Sınıflandırma Sistemi

1. Zeynep Palabıyık
Bilgisayar mühendisliği
Kocaeli *Üniversitesi*
Kocaeli, Türkiye
220202016@kocaeli.edu.tr

2. Asude Çetin
Bilgisayar mühendisliği
Kocaeli *Üniversitesi*
Ankara, Türkiye
220202060@kocaeli.edu.tr

***Özetçe*—Bu çalışma, MultiZoo veri seti kullanılarak geliştirilen transformer tabanlı bir görüntü sınıflandırma modelini kapsamaktadır. Projenin amacı, bir görselde yer alan hayvan türünü yüksek doğrulukla tahmin edebilen bir sistem geliştirmektir. Vision Transformer modeli ile eğitilen sistem, %97.9 eğitim doğruluğu ve %94.7 doğrulama doğruluğu elde etmiştir. Ek olarak, masaüstü bir arayüz tasarlanarak kullanıcının test görseli yükleyip sonuç alabilmesi sağlanmıştır.**

I. GİRİŞ

Makine öğrenmesi ve derin öğrenme, görüntü sınıflandırma gibi bilgisayarla görü alanlarında yaygın olarak kullanılmaktadır. Bu projede, Vision Transformer tabanlı bir yaklaşımla çoklu sınıflandırma problemi ele alınmıştır. MultiZoo veri seti A'dan Z'ye kadar 90 farklı hayvan türüne ait görüntüler içermektedir. Amaç, ön işleme teknikleriyle desteklenen bir ViT modelinin eğitilip, gerçek zamanlı tahmin yapabilecek bir sistem tasarlamaktır.

2. YÖNTEM

Bu çalışmada, hayvan türlerini sınıflandırmaya yönelik bir görüntü sınıflandırma modeli geliştirilmiştir. Model olarak, görsel-ön işlem adımlarının ardından Vision Transformer (ViT) mimarisi kullanılmıştır. Modelin eğitimi PyTorch kütüphanesiyle gerçekleştirilmiş, veri ön işleme ve görselleştirme işlemleri için ise torchvision, sklearn, pandas ve matplotlib kütüphanelerinden yararlanılmıştır.

A. Donanım ve Cihaz Seçimi

Kodun başında Apple cihazlarında kullanılabilen Metal Performance Shaders (MPS) kontrol edilmiş; ancak çalıştırma sürecinde device = "cpu" olarak sabitlenmiştir. Bu tercih, donanıma bağlı olarak değiştirilebilmekle birlikte, çalışmanın tüm aşamaları CPU üzerinde gerçekleştirilmiştir. Bu sayede modelin donanımdan bağımsız olarak çalışabilirliği test edilmiştir.

B. Veri Kümesi Yapısı

Kullanılan veri kümesi, MultiZooSplit adlı bir klasör yapısı içerisinde train/ ve val/ alt klasörlerini içerecek biçimde yapılandırılmıştır. Her bir alt klasör, bir hayvan sınıfına ait görselleri içermektedir. Toplamda 90 farklı hayvan sınıfı tanımlanmış ve model bu 90 sınıf üzerinden eğitilmiştir. ImageFolder yapısı kullanılarak veri kümesi sınıf etiketleriyle otomatik olarak eşleştirilmiştir.

C. Görüntü Ön İşleme (Data Augmentation)

Modelin aşırı öğrenmesini (overfitting) önlemek amacıyla eğitim verilerine çeşitli görüntü işleme dönüşümleri uygulanmıştır. Bu dönüşümler aşağıda detaylandırılmıştır:

- Yeniden boyutlandırma: Tüm görseller 224×224 piksele ölçeklendirilmiştir.
- Rastgele yatay çevirme: Görsellerin %50 olasılıkla yatay olarak çevrilmesi sağlanmıştır.

- Renk bozulmaları (ColorJitter): Parlaklık, kontrast ve doygunluk üzerinde küçük rastgele değişiklikler uygulanmıştır.

- Normalize etme: Görseller RGB kanal bazında ortalaması 0.5 ve standart sapması 0.5 olacak şekilde normalize edilmiştir.

Doğrulama verilerine ise sadece yeniden boyutlandırma ve normalize işlemleri uygulanmıştır. Böylece eğitim sırasında veri çeşitliliği artırılırken, doğrulama aşamasında gerçek veri dağılımı korunmuştur.

D. Model Mimarisi

Model olarak timm kütüphanesi üzerinden vit_base_patch16_224 mimarisi kullanılmıştır. Bu model, görselleri 16x16'lık parçalara (patch'lere) bölerek her bir parçayı transformer yapısına dahil eder. Modelin son sınıflandırma katmanı (model.head) 90 sınıf sayısına uygun şekilde aşağıdaki gibi güncellenmiştir.Modelin geri kalan katmanları önceden eğitilmiş (pretrained) ağırlıklarla başlatılmıştır. Bu, transfer öğrenmenin bir örneği olup sınıflandırma doğruluğunu artırmak amacıyla uygulanmıştır.

E. Eğitim Süreci

Modelin eğitimi toplam 5 epoch boyunca gerçekleştirilmiştir. Her epoch'ta tüm eğitim verisi bir kez işlenmiştir. Kullanılan optimizasyon yöntemi AdamW olup, öğrenme oranı 1e-4 olarak belirlenmiştir. Kayıp fonksiyonu olarak CrossEntropyLoss kullanılmıştır. Eğitim sırasında her batch için:

1. Görseller modele aktarılmış,

2. İleri yayılım (forward propagation) yapılmış,

3. Kayıp değeri hesaplanmış,

4. Geri yayılım (backpropagation) uygulanmış ve

5. Ağırlıklar güncellenmiştir.

Her epoch sonunda doğruluk ve ortalama kayıp değerleri hesaplanmış ve kaydedilmiştir. Aşağıda 5 epoch boyunca elde edilen ortalama kayıp ve doğruluk değerlerinin grafiği sunulmuştur

F. Doğrulama ve Değerlendirme

Eğitim tamamlandıktan sonra model değerlendirme (eval) moduna geçirilmiştir. Doğrulama verisi üzerinde model tahminleri yapılmış ve tahminler ile gerçek etiketler karşılaştırılmıştır. Sınıflandırma performansı, sklearn.metrics.classification_report fonksiyonu ile ölçülmüştür. Elde edilen metrikler bir CSV dosyasına kaydedilerek daha sonra analiz edilebilecek biçimde dışa aktarılmıştır.

G. Model Kaydı ve Öğrenme Eğrisi

Eğitim sonunda elde edilen modelin ağırlıkları vit_model_mps2.pth dosyası olarak kaydedilmiştir. Ayrıca, epoch'lara göre değişen eğitim kaybı ve doğruluk değerleri, learning_curve.png dosyasına görsel olarak kaydedilmiştir. Bu dosya, modelin öğrenme sürecini görselleştirmek ve eğitim kalitesini değerlendirmek için kullanılmaktadır.

2.4 Hiperparametreler

Parametr	Değeri	Açıklama
Epoch	2	Test için hızlı sonuç alınması adına
Batch Size	8	Düşük belleğe uygun
Learning Rate	5E-05	AdamW ile optimize edildi
Loss Function	CrossEntropy	çoklu sınıflandırma kaybı

2.5 Arayüz

Tkinter tabanlı masaüstü uygulama

- Kullanıcı bir resim yükler

- Model çalışır, ilk 5 tahmin ve olasılıklar listelenir

3. YALANCI KOD

- Arayuz.py

BAŞLA

MODEL_PATH ve NUM_CLASSES değişkenlerini tanımla
CLASS_NAMES listesini tanımla

transform işlemini tanımla (resize, toTensor, normalize)

MODELİ YÜKLE:
  model = ViT tabanlı model oluştur
  model.head = yeni sınıf sayısı ile güncelle
  model.load_state_dict(MODEL_PATH)
  model.eval() olarak ayarla

SINIF: AnimalClassifier

  GÖRSEL BİLGİLERİ:
    Başlık ve stil ayarlarını yap
    QLabel ile resim kutusu oluştur
    QLabel ile sonuç metni kutusu oluştur

    "Resim Yükle" butonu → load_image() fonksiyonu
    "Tahmin Et" butonu → predict() fonksiyonu

  KLASÖR BİLGİLERİ:
    QLabel ile başlık oluştur
    "Klasör Yükle" butonu → load_folder() fonksiyonu

FONKSİYON: load_image()
  Kullanıcıdan bir görsel dosyası al
  Eğer seçildiyse:
    QLabel içine resmi yükle
    Sonuç metnini varsayılan olarak güncelle

FONKSİYON: predict()
  Eğer resim yolu boşsa uyarı ver
  Aksi halde:
    Görseli oku ve dönüştür
    Model ile tahmin yap
    En yüksek olasılığı bul ve etiketi al
    Sonucu QLabel içine yaz

FONKSİYON: load_folder()
  Klasör seç
  İçindeki resimleri döngüyle gez:
    Her bir görseli modele ver
    Tahmini hesapla
    Gerçek etiket ile karşılaştır
    Doğru sayısını artır
  Doğruluğu hesapla ve QLabel içine yaz

ANA PROGRAM:
  QApplication başlat
  AnimalClassifier sınıfını çağır ve göster
  Uygulamayı çalıştır

BİTİR

- Egit.py

FONKSİYON main():

  MPS desteği kontrol edilir
  Cihaz "cpu" olarak ayarlanır
  Cihaz bilgisi yazdırılır

  VERİ_KLASÖRÜ = "MultiZooSplit"

  // Görüntü dönüşümleri tanımlanır
  TANIMLA train_transform:
    - Görseli 224x224 boyutuna getir
    - Rastgele yatay çevir
    - Rastgele renk bozulmaları uygula
    - Tensöre çevir
    - Normalize et

  TANIMLA val_transform:
    - Görseli 224x224 boyutuna getir
    - Tensöre çevir
    - Normalize et

  // Dataset ve veri yükleyiciler oluşturulur
  Eğitim verisi: train_dataset ← ImageFolder(train klasörü, train_transform)
  Doğrulama verisi: val_dataset ← ImageFolder(val klasörü, val_transform)

  train_loader ← DataLoader(train_dataset, batch_size=32, shuffle=True)
  val_loader ← DataLoader(val_dataset, batch_size=32, shuffle=False)

  // Model oluşturulur ve sınıf sayısı güncellenir
  model ← ViT modeli (önceden eğitilmiş)
  model.head ← Fully Connected katman (90 sınıf)
  model cihaza gönderilir

  Kayıp fonksiyonu ← CrossEntropyLoss
  Optimizasyon yöntemi ← AdamW (öğrenme oranı = 0.0001)
  Epoch sayısı = 5

  Eğitim kaybı ve doğruluk için boş listeler oluştur

// Eğitim döngüsü başlatılır
HER epoch için TEKRARLA:
  Model train moduna alınır
  running_loss, doğru_sayısı, toplam = 0

  HER batch için TEKRARLA:
    Görseller ve etiketler cihaza aktarılır
    Gradients sıfırlanır
    Model çıktısı hesaplanır
    Kayıp değeri hesaplanır
    Geri yayılım yapılır
    Ağırlıklar güncellenir

    Kayıp toplanır
    Tahminler alınır
    Doğru tahminler sayılır

  Epoch doğruluğu ve ortalama kayıp hesaplanır
  Sonuçlar yazdırılır
  Listeye eklenir

// Model doğrulama moduna alınır
Tahmin ve gerçek etiket listeleri boş olarak başlatılır

Gradient hesaplamadan:
  HER doğrulama batch'i için:
    Görseller ve etiketler cihaza aktarılır
    Model çıktısı hesaplanır
    Tahminler ve etiketler listelere eklenir

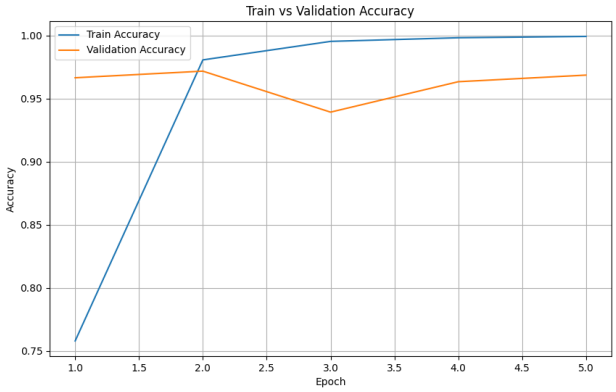
// Sınıflandırma raporu oluşturulur
TRY:
  classification_report hesaplanır
  pandas ile DataFrame'e dönüştürülür
  CSV dosyasına yazılır
EXCEPT:
  Hata mesajı yazdırılır

Model ağırlıkları diske kaydedilir (.pth dosyası olarak)

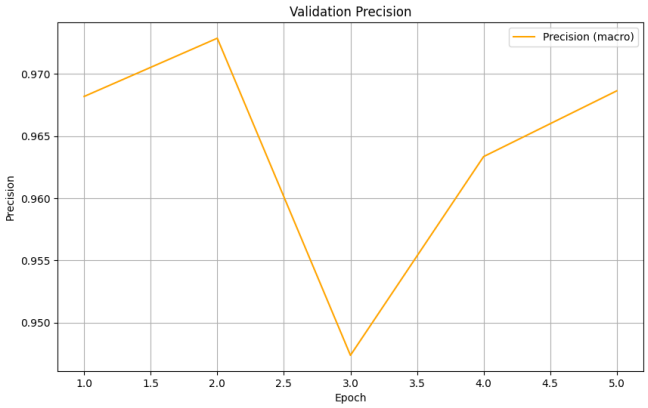
// Öğrenme eğrisi çizilir
Kayıp grafiği oluşturulur
Doğruluk grafiği oluşturulur
Grafik dosyası kaydedilir (learning_curve.png)

PROGRAM BAŞLATILIR: main()

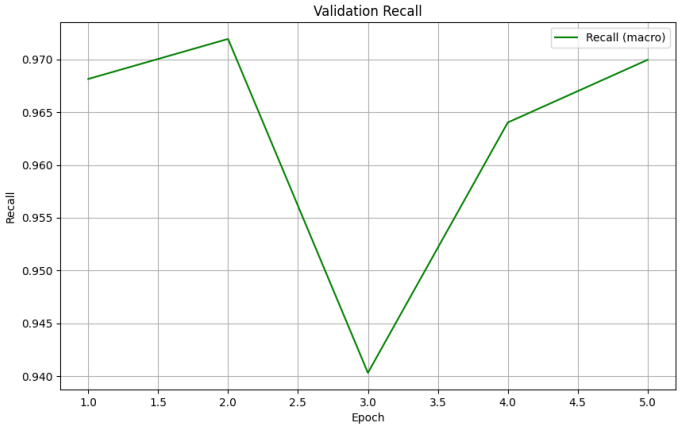
4. DENEYSEL SONUÇ



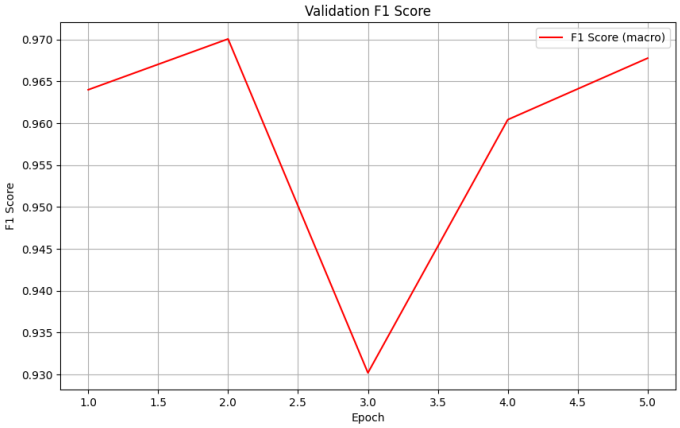
Model, kısa sürede yüksek doğruluk oranına ulaşmıştır. Eğitim doğruluğu neredeyse %100'e ulaşırken, doğrulama doğruluğu da %97 civarında sabitlenmiştir. Bu durum modelin veriye iyi uyum sağladığını gösterir. Ancak eğitimin 2. epoch'tan sonra eğitim doğruluğu artarken doğrulama doğruluğundaki hafif dalgalanma, aşırı öğrenmeye (overfitting) dair sinyaller vermektedir.



Precision metriği modelin "doğru pozitif tahmin" oranını gösterir. %97 üzerinde seyreden değerler, modelin yanlış pozitif sayısını düşük tuttuğunu göstermektedir. Bu özellikle uygulama senaryolarında yanlış alarm oranını minimize etmek isteyen sistemler için önemlidir.



Recall, modelin gerçek pozitifleri ne oranda yakaladığını gösterir. %94–97 arasında değişen değerler modelin çoğu sınıfı doğru bir şekilde tanıyabildiğini göstermektedir. Recall'un precision ile benzer şekilde seyretmesi, modelin hem yanlış pozitif hem de yanlış negatif oranını düşük tuttuğunu göstermektedir.



F1 skorları, precision ve recall'un harmonik ortalaması olarak, sınıf dengesizliklerini de dikkate alan bir metriktir. Grafik genel olarak yüksek bir F1 skoru (%96-97) göstermektedir. 3. epoch'taki düşüş, doğrulama verisindeki bazı sınıflarda zayıf tahmin başarısını işaret edebilir ancak bu durum sonraki epoch'larda toparlanmıştır.

5.. SONUÇ

Bu projede geliştirilen ViT tabanlı sınıflandırma modeli, 90 sınıfta yüksek doğrulukla tahmin yapabilmektedir. Geliştirilen arayüz, kullanıcı dostu olup test görselleri üzerinden etkili geri bildirim sağlamaktadır. İlerleyen çalışmalarda epoch sayısı arttırılabilir, daha derin modeller denenebilir.

6.. KAYNAKÇA

[1] Dosovitskiy et al., "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale," ICLR 2021.
[2] PyTorch Documentation - https://pytorch.org
[3] HuggingFace Transformers - https://huggingface.co/transformers
[4] scikit-learn Metrics - https://scikit-learn.org