MultiZoo - Görüntü Sınıflandırma ve Test Arayüzü

1. Cihangir Emre ER
Bilgisayar Mühendisliği
Kocaeli Üniversitesi
Kocaeli, Türkiye
220202036@kocaeli.edu.tr

2. Berker Yiğit
Bilgisayar Mühendisliği
Kocaeli Üniversitesi
Kocaeli, Türkiye
220202046@kocaeli.edu.tr

I. Özet

Bu projede, hayvan sınıflarını yüksek doğrulukla tanıyabilen bir görüntü sınıflandırma modeli eğitilmiş ve bu modeli gerçek zamanlı test edebileceğimiz bir masaüstü arayüz geliştirilmiştir. Görsel sınıflandırma için Vision Transformer (ViT) tabanlı bir model kullanılmış, eğitim süreci Google Colab ortamında T4 GPU kullanılarak gerçekleştirilmiştir. Eğitilen modelin doğruluğu %97 üzerinde tutulmuş, sonuçlar doğruluk (accuracy), kesinlik (precision), duyarlılık (recall) ve F1-skoru (F1-score) metrikleri ile detaylandırılmıştır. Arayüz kullanıcı dostu olacak şekilde customtkinter kullanılarak geliştirilmiş ve test sırasında görüntü seçimi veya sürükle-bırak desteğiyle kullanım imkanı sunulmuştur.

II. Giriş

Mobil Görüntü sınıflandırma problemleri, bilgisayarla görü alanında en temel yapay zeka problemlerindendir. Bu projede, farklı hayvan türlerini temsil eden görüntülerden oluşan bir veri kümesi kullanılarak, her bir görselin en doğru sınıfa atanması amaçlanmıştır. Bu amaçla güncel bir transformer mimarisi olan ViT (Vision Transformer) tercih edilmiştir. Sonuçların doğruluğu ve modelin genelleme yeteneği, eğitim sürecinde takip edilmiştir. Ayrıca modelin kullanımını kolaylaştırmak için modern ve sade bir masaüstü uygulaması da geliştirilmiştir.

III. Yöntem

3.1 Model Mimarisi

Kullanılan model timm kütüphanesi üzerinden alınan vit_base_patch16_224 mimarisidir. Bu model, görselleri 16x16 piksellik parçalara bölerek işlemesi ve dikkat (attention) mekanizması ile çalışması sayesinde yüksek doğruluk elde etmektedir. Modelin son katmanı aşağıdaki gibi özelleştirilmiştir:

```
model.head = nn.Linear(model.head.in_features, num_classes)
```

3.2 Eğitim Süreci

- **Veri Ayrımı**: Eğitim ve doğrulama verileri %80 - %20 oranında ayrılmıştır.
- **Dönüşümler**: Görseller 224x224 boyutuna getirilip normalize edilmiştir.
- **Loss Fonksiyonu**: CrossEntropyLoss()
- **Optimizer**: AdamW, öğrenme oranı 3e-5
- **Epoch Sayısı**: 10
- **Early Stopping**: 3 epoch boyunca gelişme olmazsa durdurulmaktadır.
- **3.3 Arayüz**

Arayüz customtkinter kütüphanesi ile masaüstü ortamında tasarlanmış, kullanıcıların test görselleri yükleyip sınıf tahmini alabilecekleri bir yapı sunulmuştur. Özellikler:

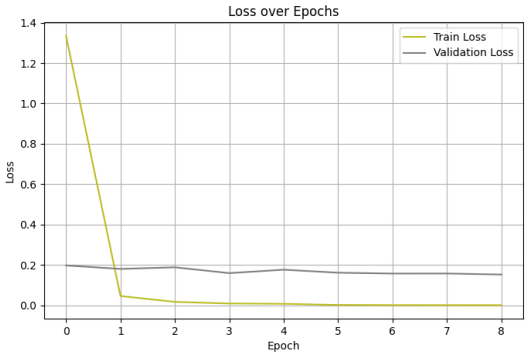
- Görsel seçme veya sürükle-bırak desteği
- İlk 3 tahmini ve güven skorlarını gösterme
- Tıklanabilir görsel önizleme (büyütme)

IV. Deneysel Sonuç

Modelin eğitimi sürecinde doğruluk (Accuracy), kesinlik (Precision), duyarlılık (Recall), F1-Skoru ve eğitim kaybı (Loss) metrikleri epoch bazında takip edilmiştir. Eğitim işlemi, Google Colab ortamında NVIDIA T4 GPU üzerinde gerçekleştirilmiştir ve toplamda 10 epoch boyunca model eğitilmiştir. Aşağıda, her bir metrik için elde edilen grafiksel sonuçlara ve bu sonuçlara ilişkin analizlere yer verilmiştir
.

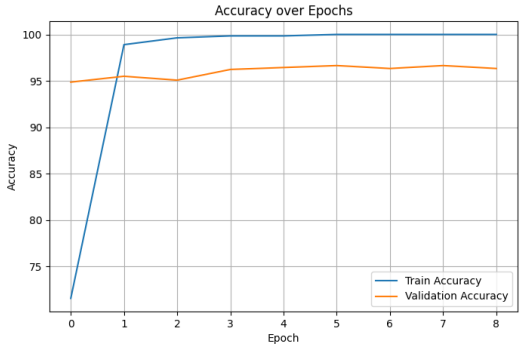
4.1 Eğitim Kayıp (Loss) Eğrisi

Eğitim ve doğrulama kaybı eğrisi, modelin öğrenme süreci boyunca hem eğitim verisine hem de doğrulama verisine ne kadar iyi uyum sağladığını göstermektedir. Eğitim kaybı, ilk epoch'ta yaklaşık 1.3 gibi yüksek bir seviyede başlamış, fakat sonraki epoch'larda hızla düşerek neredeyse sıfıra yaklaşmıştır. Bu durum modelin eğitim verisi üzerindeki hataları hızla minimize ettiğini göstermektedir. Validation (doğrulama) kaybı ise eğitim boyunca düşük seviyelerde kalmış ve sabit bir eğilim izlemiştir. Eğitim ve doğrulama eğrilerinin birbirine yakın seyretmesi, modelin overfitting yapmadığını ve genelleme başarısının yüksek olduğunu göstermektedir.



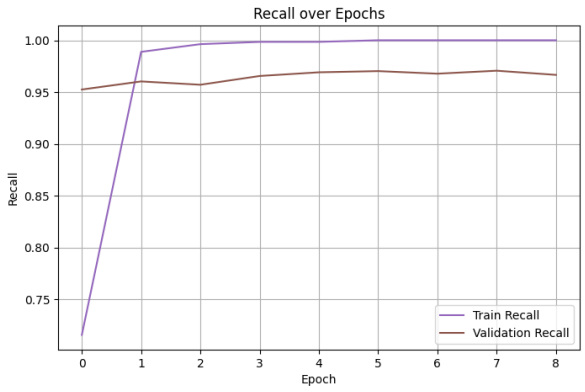
4.2 Doğruluk (Accuracy)

Accuracy eğrisi, modelin genel sınıflandırma performansını ortaya koymaktadır. Eğitim doğruluğu, ilk epoch'tan itibaren hızlı bir artış göstermiş ve 3. epoch itibarıyla %99 seviyelerine ulaşmıştır. Bu yüksek doğruluk, modelin eğitim verisindeki örnekleri çok başarılı bir şekilde sınıflandırabildiğini göstermektedir. Doğrulama doğruluğu ise eğitim süresi boyunca %95'in üzerinde kalmış ve 6. epoch itibarıyla maksimum %97 seviyelerine kadar çıkmıştır. Eğitim ve doğrulama doğrulukları arasında ciddi bir fark bulunmaması, modelin genelleme kabiliyetinin güçlü olduğunun bir başka göstergesidir.



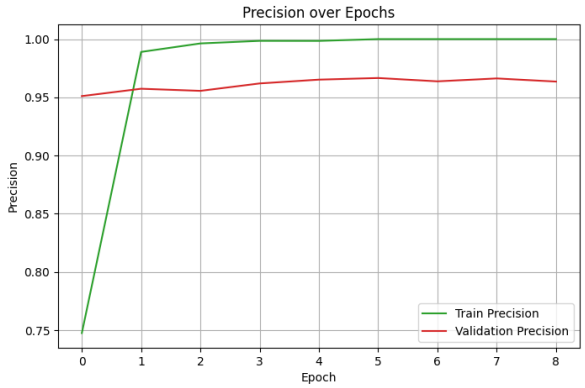
4.3 Duyarlılık (Recall)

Recall metriği, modelin pozitif sınıfları ne kadar doğru tanıdığını ölçer.Eğitim setinde recall değeri neredeyse %100'e ulaşarak modelin pozitif örnekleri neredeyse hiç kaçırmadığını göstermektedir.Validation tarafında ise recall %95 seviyelerinde başlayıp kademeli olarak %97'ye kadar artmıştır.Bu durum, modelin sınıfları ayırt etme konusunda oldukça başarılı olduğunu ve sınıf atlama(false negative)problemini büyük ölçüde engellediğini göstermektedir.



4.4 Kesinlik (Precision)

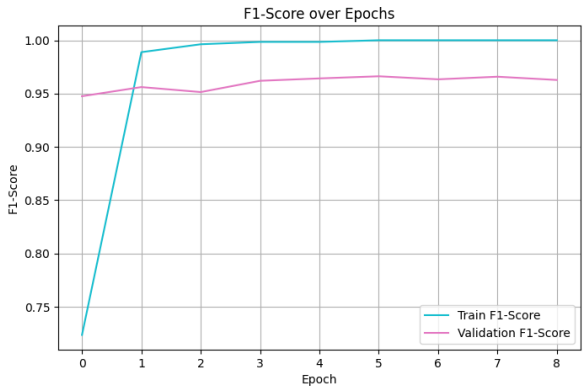
Precision grafiği, modelin yaptığı pozitif tahminlerin ne kadarının gerçekten doğru olduğunu ölçer. Eğitim sırasında precision neredeyse mükemmele yakın seyretmiş, %100'e ulaşmıştır. Validation setinde ise precision %95 seviyelerinde seyrederek eğitim verisine göre biraz daha düşük ama yine oldukça başarılı sonuçlar vermiştir. Bu sonuç, modelin hatalı pozitif tahminlerinin çok düşük olduğunu, yani yanlış sınıf atamaktan büyük ölçüde kaçındığını göstermektedir.



4.5 F1-Skoru

F1-skoru grafiği, duyarlılık ve kesinliğin dengeli bir şekilde ele alındığı metrik olarak genel performansı özetlemektedir. Eğitim setinde %100'e yaklaşan F1-skoru, modelin eğitim verisinde oldukça dengeli ve başarılı bir şekilde çalıştığını göstermektedir. Validation F1-skoru ise %95'ten başlayarak %97 seviyelerine ulaşmıştır. Bu da modelin doğrulama verilerinde de hem yüksek kesinlik hem de yüksek duyarlılıkgösterdiğini ve dengeli tahminler yaptığını

göstermektedir.



4.6 Genel Değerlendirme

Elde edilen tüm bu grafikler ışığında, modelin Vision Transformer (ViT) tabanlı yapısı ile oldukça başarılı bir şekilde eğitildiği görülmektedir. Eğitim sürecinde hem loss değerlerinin düşmesi hem de doğruluk, duyarlılık, kesinlik ve F1-skoru gibi metriklerin yüksek değerlerde seyretmesi, modelin hem öğrenme hem de genelleme kabiliyetinin oldukça güçlü olduğunu göstermektedir. Validation verisindeki istikrarlı ve yüksek metrikler, modelin sadece eğitim verisine değil, daha önce görmediği örneklere karşı da etkili performans gösterdiğini ortaya koymaktadır. Bu bağlamda geliştirilen sistem, görsel sınıflandırma görevlerinde güçlü ve güvenilir bir yapay zeka çözümü olarak değerlendirilebilir.

V. SONUÇ

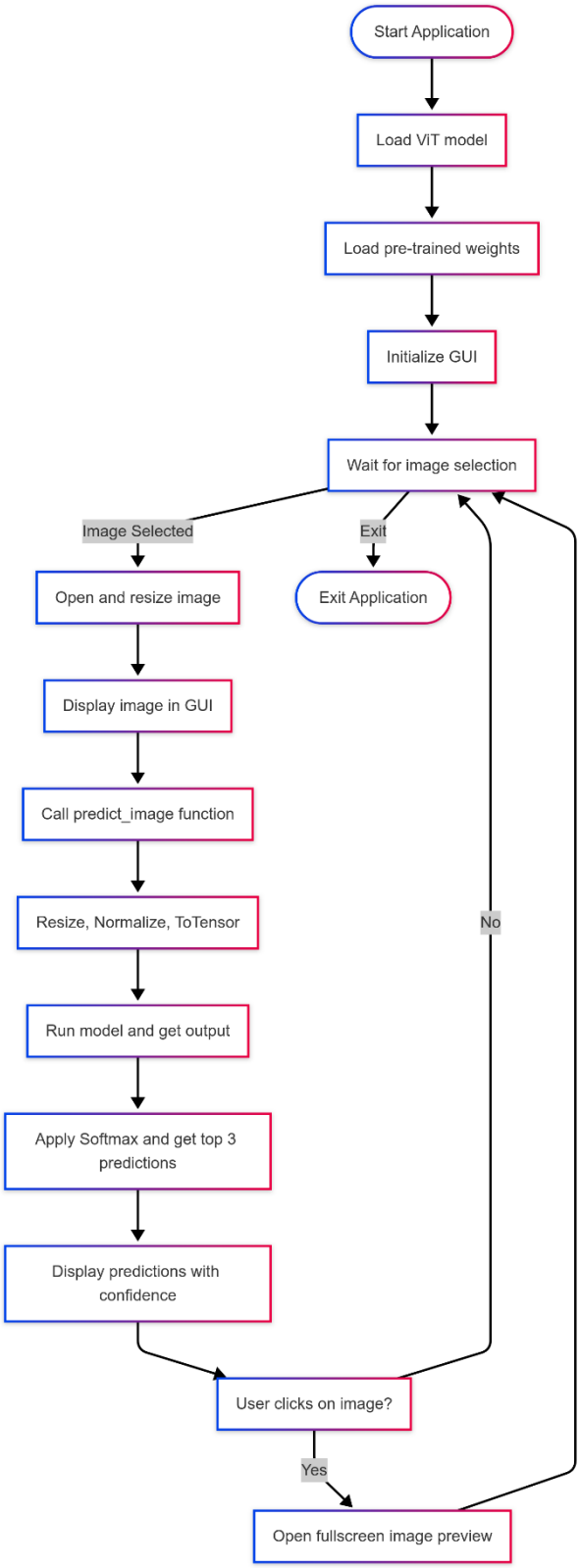
Bu projede Gerçekleştirilen geniş çaplı testlerde (4770 görüntü), modelin yalnızca %2.5 oranında "sorunlu tahmin" yaptığı gözlemlenmiştir. Bu sorunlar iki ana kategoriye ayrılmıştır:

- **Low Confidence:** Modelin doğru sınıfı tahmin ettiği ancak güven oranının %90'ın altında olduğu durumlar. Bu tür örneklerde modelin görsel benzerlikten dolayı emin olamaması dikkat çekicidir. Örneğin: bee sınıfı için %32.97 gibi düşük bir güven oranı.
- **Wrong Prediction:** Modelin ilk sırada tamamen yanlış bir sınıf tahmin ettiği vakalar. Genellikle sınıflar görsel olarak birbirine çok benzerdi (örneğin: rat → mouse, octopus → squid, lion → coyote).

Bu analizler, modelin kararlarında dikkatli davrandığını ve benzer türler arasında ayırt etmenin doğası gereği zor olduğunu göstermektedir.

Genel olarak, elde edilen sonuçlar Vision Transformer mimarisinin görsel sınıflandırmadaki gücünü bir kez daha kanıtlamış, düşük güvenli tahminlerde bile doğru sınıfa yakın sonuçlar verdiğini göstermiştir.

Akış Diyagramı



Yalancı Kod

```
BEGIN

IMPORT required libraries:
   - customtkinter (GUI)
   - PIL (image processing)
   - torch, torchvision, timm (deep learning)
   - os (file system)

SET GUI appearance to dark mode and blue theme

// ------------------------------
// MODEL SETUP
// ------------------------------
IF CUDA is available THEN
   device ← 'cuda'
ELSE
   device ← 'cpu'

READ class names from 'class_names.txt' into class_names_list
num_classes ← length of class_names_list

LOAD pretrained ViT model:
   model ← vit_base_patch16_224 (from timm)
   REPLACE final layer with Linear(in_features, num_classes)
   LOAD weights from 'best_vit_model.pt'
   SET model to evaluation mode on device

// ------------------------------
// IMAGE TRANSFORMATION
// ------------------------------
DEFINE transform:
   - Resize to 224x224
   - Convert to tensor
   - Normalize with mean=0.5, std=0.5 for each channel

// ------------------------------
// PREDICTION FUNCTION
// ------------------------------
FUNCTION predict_image(image_path):
   image ← open image from image_path and convert to RGB
   input_tensor ← apply transform and add batch dimension
   MOVE input_tensor to device
   WITH torch.no_grad():
      output ← model(input_tensor)
      probabilities ← softmax(output)
      top3_probs, top3_indices ← top 3 values and indices
   FOR each i in 0 to 2:
      label ← class_names_list[top3_indices[i]]
      score ← top3_probs[i]
      ADD (label, score) to result_list
   RETURN result_list

// ------------------------------
// GUI APPLICATION CLASS
// ------------------------------
CLASS App inherits from CTk:
   METHOD __init__():
      SET window title and size (500x600)
      CALL center_window()

      CREATE image_label with default text "No image loaded"
      CREATE button "Select Image" → calls load_image()
      CREATE result_label for predictions

   METHOD center_window(width, height):
      GET screen width and height
      COMPUTE center coordinates
      SET geometry to center position

   METHOD load_image():
      file_path ← open file dialog for image selection
      IF file_path exists THEN
         current_image_path ← file_path
         image ← resize image to 224x224
         DISPLAY image on image_label using CTkImage
         BIND left-click event on image_label to
open_fullscreen_image()
         predictions ← predict_image(file_path)
         FORMAT and display top 3 predictions in result_label

   METHOD open_fullscreen_image(event):
      IF current_image_path is not set OR file does not exist THEN
RETURN
      CREATE new CTkToplevel window (800x800)
      LOAD and resize image to 700x700
      DISPLAY image in new window

// ------------------------------
// MAIN
// ------------------------------
IF __name__ == "__main__":
   app ← App()
   app.mainloop()

END
```

KAYNAKLAR

[1] Dosovitskiy, A. et al. (2020). *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*. arXiv:2010.11929

[2] timm - PyTorch Image Models Library: https://github.com/huggingface/pytorch-image-models

[3] PyTorch Docs: https://pytorch.org/docs/stable/index.html

[4] Torchvision Docs: https://pytorch.org/vision/stable/index.html

[5] CustomTkinter GUI Library: https://github.com/TomSchimansky/CustomTkinter

[6] Vaswani, A. et al. (2017). **Attention is All You Need**. Advances in Neural Information Processing Systems, NIPS

[7] Tkinter Official Reference (Python GUI). https://docs.python.org/3/library/tkinter.html

[8] Goodfellow, I., Bengio, Y., & Courville, A. (2016). **Deep Learning**. MIT Press. https://www.deeplearningbook.org/

[9] Howard, A. G. et al. (2017). **MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications**. arXiv preprint https://arxiv.org/abs/1704.04861

[10] Kingma, D. P., & Ba, J. (2014). **Adam: A Method for Stochastic Optimization**. arXiv preprint https://arxiv.org/abs/1412.6980