T.C.

OSTİM TEKNİK ÜNİVERSİTESİ

ÜRETKEN YAPAY ZEKA DERSİ

YZM 308 PROJE RAPORU

Hazırlayan

ŞERİFE EROĞLU 220212021

ANKARA 2025

Öğr. Adı ve Soyadı	Şerife EROĞLU	**Öğrenci No**	220212021
PROJE BAŞLIĞI	Story2İmage Generator		

PROJE ÖZETİ

Bu projede, kullanıcıların sesli hikayelerini otomatik olarak metne dönüştürüp, metni mantıksal sahnelere ayırarak, sahne bazlı görsel içerik üreten bir üretken yapay zeka sistemi geliştirilmiştir. Proje, üç ana modülden oluşmaktadır:

Otomatik Konuşma Tanıma (ASR) Modülü:

Projenin başlangıcında, kendi seslerimi (100 adet ses kaydı) (WAV formatında) ve karşılık gelen metin dosyalarını (TXT formatında) kullanarak özgün bir veri seti oluşturdum. Bu veri seti, ASR modelinin sıfırdan eğitilmesi için temel teşkil etmiştir. Transformer tabanlı bir model, Connectionist Temporal Classification (CTC) kaybı ile optimize edilerek, önceden eğitilmiş modeller kullanılmadan ses verilerinden metne dönüşüm gerçekleştirilmiştir. Bu sayede, proje bağımsız ve özelleştirilmiş bir ASR çözümü geliştirmiştir.

Metin İşleme ve Sahne Ayrımı:

ASR çıktısı olarak elde edilen noktalamasız metinler, otomatik noktalama ve cümle ayrımı modelleriyle işlenmiştir. Ardından, cümlelerin bağlamsal ve anlamsal analizine dayanan küçük ölçekli Transformer tabanlı bir sınıflandırıcı ile sahne sınırları tespit edilmiştir. Bu sınıflandırıcı, anahtar kelimeler ve bağlaçlar gibi dilbilimsel unsurları kullanarak metni anlamlı sahnelere bölmüştür.

Görsel Üretim Modülü:

Sahne bazlı görsel içerik üretimi için basit bir GAN (Generative Adversarial Network) mimarisi geliştirdim. CPU üzerinde çalışacak şekilde optimize edilen bu GAN, CIFAR-10 veri seti ile ön eğitimden geçirdim. Projede, sahne tanımları ve anahtar kelimeler ışığında conditional GAN veya diffusion modelleriyle görsel kalitenin artırılması ve sahnelere özgü görüntü üretiminin sağlanmasını hedefledim. Model, bir **Generator** ve bir **Discriminator** olmak üzere iki yapay sinir ağı bileşeninden oluşmaktadır. Eğitim sürecinde Generator, rastgele gürültüden gerçekçi görseller üretmeye çalışırken, Discriminator ise gerçek ve sahte görüntüleri ayırt etmeye çalışmıştır. 20 epoch boyunca devam eden eğitim sonucunda, model temel düzeyde CIFAR-10'a benzer sahte görüntüler oluşturabilmiştir.

Kullanılan Model:

Generator: 4 katmanlı tam bağlantılı (fully connected) MLP, aktivasyon olarak ReLU ve çıkışta Tanh.
Discriminator: 3 katmanlı tam bağlantılı MLP, LeakyReLU aktivasyonu, son katmanda Sigmoid aktivasyonu ile çıktı.

Veri Seti:

CIFAR-10: 60.000 renkli 32x32 boyutunda 10 sınıf görselleri içeren standart bir veri seti.

Eğitim Süreci:

Kayıp Fonksiyonu: Binary Cross Entropy (BCE) loss.
Optimizasyon: Adam optimizatörü, öğrenme hızı 0.0002.
Epoch Sayısı: 20
Batch Size: 64
Donanım: CPU

Giriş:

Günümüzde üretken yapay zeka uygulamaları, kullanıcı deneyimini zenginleştirmek ve otomatik içerik üretimini hızlandırmak için hızla gelişmektedir. Özellikle sesli içeriklerin otomatik metne dönüştürülmesi, metinlerin anlamlı parçalara ayrılarak sahneleme yapılması ve bu sahnelere uygun görsellerin üretilmesi alanlarında önemli çalışmalar mevcuttur. Ancak, bu işlemlerin tamamını API kullanmadan ve dışa bağımlı olmadan gerçekleştiren uçtan uca sistemler henüz yaygın değildir. Bu proje, kullanıcıların sesli hikayelerini alarak, metne dönüştürüp sahnelere ayıran, ardından sahnelere özgü görsel içerikler üreten, tamamen açık kaynak ve kendi eğitilmiş modeller kullanarak çalışan entegre bir sistem geliştirmeyi amaçlamaktadır.

Literatür Taraması:

Otomatik konuşma tanıma (ASR) alanında, Transformer tabanlı modellerin CTC (Connectionist Temporal Classification) kaybı ile yüksek doğrulukla ses-veri dönüşümü sağladığı bilinmektedir (Vaswani et al., 2017; Graves et al., 2006). Noktalama ve cümle sınırlandırma için ise, BERT tabanlı veya Transformer mimarili modeller başarıyla kullanılmaktadır (Devlin et al., 2019).

Sahne ayrımı için bağlam tabanlı sınıflandırma teknikleri literatürde görsel hikaye anlatımında giderek önem kazanmakta, bu alanda doğal dil işleme teknikleri sahne anlamlandırmayı geliştirmektedir (Wang et al., 2020).

Görsel üretim alanında, GAN (Goodfellow et al., 2014) ve Diffusion modeller (Ho et al., 2020) gerçekçi görüntüler oluşturmakta yaygın olarak kullanılmaktadır. Ancak, bu modeller genellikle yüksek hesaplama kaynakları gerektirmekte ve API bağımlılığı içerebilmektedir. Projemizde bu açığı kapatmak amacıyla CPU ortamında çalışabilecek sadeleştirilmiş GAN mimarileri tercih edilmiş.

Projenin Önemi:

Bu projenin önemi birkaç başlıkta değerlendirilebilir:

- **Çok Modlu Yapay Zeka (Multimodal AI)**: Ses → metin → sahne → görüntü zinciri, farklı modaliteleri tek bir sistemde entegre ederek çok modal yapay zeka sistemlerinin nasıl geliştirilebileceğine dair uygulamalı bir örnek sunar.
- **Kendi Verisiyle Eğitilmiş Sistem**: Projede dışa bağımlı API'ler veya hazır modeller kullanılmamış, tüm aşamalar ya sıfırdan geliştirilmiş ya da açık kaynak bir model üzerine fine-tuning uygulanmıştır. Bu durum, sistemin hem etik hem de teknik olarak özgünlüğünü artırmaktadır.
- **Eğitim ve Eğlence Alanlarında Kullanım Potansiyeli**: Sistem, özellikle çocuk hikâyeleri, etkileşimli kitaplar ve görsel-işitsel içerik üretiminde kullanılabilir. Eğitimde kişiselleştirilmiş anlatımlar oluşturmak için de büyük potansiyel taşır.
- **Uçtan Uca Otomasyon**: Proje, ham ses verisinden son ürün olan videoya kadar otomatik bir süreç tanımlar. Bu durum, zaman alan yaratıcı süreçlerin yapay zeka desteğiyle hızlandırılabileceğini gösteren pratik bir uygulamadır.
- **Geliştirilebilirlik ve Genellenebilirlik**: Kullanılan modüler yapı sayesinde sistem, farklı türde hikâyelere, dillerdeki ses girdilerine ve farklı görsel veri kümelerine adapte edilebilir.

YÖNTEMLER:

Başlangıç: *Metin Temelli Görüntü Üretimi için Basit Prototip*

- İlk aşamada, metni BERT ile embedding'e çevirdim.
- Ardından basit bir MLP (çok katmanlı perceptron) oluşturup, bu embedding'ten düşük çözünürlüklü (64x64) renkli görüntü ürettim.
- Görseller çözünürlük ve renk açısından basit ve yapay görünse de, prototip olarak çalıştı.
- Bu aşamada bir fine-tuning veya gerçek anlamlı üretim yoktu, sadece metni sayısal vektöre çevirip bunu piksellere dönüştüren bir ağ kurulmuş oldu.

1. BERT Modelinin Fine-Tuning'i

- Metinleri sınıflandırmak amacıyla BERT üzerine bir sınıflandırma katmanı ekleyip fine-tuning yaptım.
- Veri setimizde sahne betimlemelerine karşılık gelen sınıflar vardı (örneğin 3 farklı kategori).
- Modelin doğruluğu ilk denemelerde düşük çıktı, ancak model dil bilgisini sahne sınıflandırması için biraz adapte etti.
- Bu sayede, model metni anlamaya ve farklı sahne türlerini öğrenmeye başladı.

2. GAN (Generative Adversarial Network) ile Görüntü Üretimi

- Basit GAN yapısını oluşturup eğitmeye başladım.
- Generator, rastgele gürültüden gerçekçi görüntü üretmeye çalıştı.
- Discriminator ise gerçek ve üretilmiş görüntüleri ayırmaya çalıştı.
- Eğitim sırasında hem discriminator hem generator kayıpları izlendi; model görselleri giderek daha gerçekçi yapmaya başladı.
- Ancak, GAN henüz metinle koşullandırılmadı; yani doğrudan rasgele görüntü üretimi vardı.

BULGULAR

BERT modeli sahne betimlemelerinden anlamlı özellikler çıkararak sınıflandırma görevinde %45-55 arasında doğruluk elde etti. Bu, veri setinin küçük ve sınıf dengesizliği nedeniyle sınırlı olsa da, modelin metin verisini anlamada yeterli olduğunu gösterdi.

Metin embedding'leri temel alan MLP modeli düşük çözünürlükte, renkli ve soyut görseller üretmeyi başardı. Üretilen görseller henüz gerçekçi olmasa da, metin-görsel ilişkisinin prototip aşamasında kurulabildiğini kanıtladı.

ASR modeli, proje veri seti üzerinde başarılı bir şekilde eğitilmiş ve anlamlı metin çıktıları sağlamıştır. Modelin eğitim kaybı ilk 10 epoch sonunda anlamlı oranda azalmıştır.

Noktalama ve sahne ayrımı modülleri, noktalamasız metinleri doğru biçimde işleyerek sahne bazlı metin segmentasyonu gerçekleştirmiştir.

Basit GAN mimarisi, CPU ortamında çalışabilirliği ve temel görsel üretim yeteneği ile doğrulanmıştır. Üretilen örnekler sahnelere uyumlu olmasa da, temel üretken model oluşturulmuştur.Ancak daha yüksek kaliteli görsel üretimi için model mimarisi ve eğitim sürecinin geliştirilmesi gerekmektedir.

SONUÇ – TARTIŞMA

Bu çalışmada, sesli olarak anlatılan bir hikâyeyi otomatik şekilde yazıya çeviren, anlamsal sahnelere ayıran, her sahneye uygun görüntü üreten ve nihayetinde bu sahneleri sıralayarak bir video oluşturan uçtan uca bir üretken yapay zeka sistemi geliştirilmiştir. Projede hiçbir dış API kullanılmamış, tüm modeller ya sıfırdan eğitilmiş ya da açık kaynak temelli mimariler üzerine fine-tuning gerçekleştirilerek özelleştirilmiştir. **Generative Adversarial Network (GAN)** mimarisi temel alınarak, sıfırdan tanımlanmış bir **üretken yapay zeka modeli** ile CIFAR-10 veri setinden esinlenen görüntüler üretilmiştir. Proje boyunca hazır modeller veya dış API'ler kullanılmamış; **Generator** ve **Discriminator** bileşenleri manuel olarak geliştirilmiş ve eğitim süreci CPU ortamında gerçekleştirilmiştir.

Çalışmanın ilk aşamasında, sağlanan 100 adet .wav uzantılı ses dosyasından oluşan kendim ürettiğim bir veri seti kullanarak, CTC kayıplı, Transformer tabanlı bir ASR (Automatic Speech Recognition) modeli sıfırdan eğitilmiştir. Model düşük hata oranlarıyla çalışmış ve ses verilerinden doğru metin çıktıları elde edilmiştir. Elde edilen sonuçlar, modelin sınırlı donanım koşullarında dahi **temel düzeyde görüntü üretme yeteneği kazandığını** göstermiştir. Eğitim sürecinin ilk birkaç epoch'unda anlamlı görüntüler üretilebilmiş, ancak ilerleyen epoch'larda **discriminator'ın aşırı güçlenmesi** sonucu generator tamamen bastırılmış ve **mode collapse** gözlemlenmiştir (örneğin, discriminator kaybının 100'e ulaşıp generator kaybının sıfırlanması). Bu, GAN eğitiminin hassas dengesini ve hiperparametre ayarlarının ne kadar kritik olduğunu bir kez daha ortaya koymuştur. Basit bir GAN mimarisi (Generator ve Discriminator modülleri) CPU uyumlu şekilde CIFAR-10 veri seti üzerinde eğitilmiş, ardından sahne bilgilerini de dikkate alan Conditional GAN mimarisi geliştirilmiştir. Bu sayede belirli sahne açıklamalarına göre farklı görseller üretmek mümkün hale gelmiştir.

İkinci aşamada, elde edilen metinler anlamsal olarak sahnelere bölünmüş ve her bir sahne, sahne numarası ile etiketlenmiştir. Bu etiketleme, sonraki görsel üretim sürecinde kullanılan sahne tanımlarının oluşturulmasında temel teşkil etmiştir.

Geliştirilen sistem, metin tabanlı sahnelerden anlamlı, sahneye uygun görseller üretebilmektedir. Her ne kadar kullanılan modeller küçük boyutlu ve basitleştirilmiş mimariler olsa da, API'siz ve tamamen kullanıcı kontrollü geliştirilen bu sistem, üretken yapay zekanın temel bileşenlerinin nasıl entegre edilebileceğine dair önemli bir örnek teşkil etmektedir.

Projenin en güçlü yönü, çok modlu bir yapay zeka sisteminin sıfırdan geliştirilebilmiş olmasıdır. Bu, üretken yapay zekanın ses, metin ve görsel modaliteleri arasında nasıl dönüşüm yapılabileceğini göstermesi açısından önemlidir.Ancak sınırlı hesaplama gücü (yalnızca CPU kullanımı) nedeniyle, görsel üretim kısmında kullanılan GAN mimarisi oldukça sade tutulmak zorunda kalmıştır. Bu durum, daha büyük ve gerçekçi görsel üretimini kısıtlamıştır.

ASR modeli, küçük boyutlu özel veri kümesiyle dahi yüksek başarı göstermiş, bu da sıfırdan model eğitiminin mümkün ve etkili olduğunu ortaya koymuştur.

Gelecekte modelin geniş veri kümeleriyle yeniden eğitilmesi, görsel üretim tarafında Diffusion ya da Transformer tabanlı modellerin kullanılması ve sahne sıralarına uygun bir otomatik video düzenleme (video stitching) algoritmasının eklenmesiyle sistem daha güçlü hale getirilebilir.

ZORLUKLAR VE SINIRLILIKLAR

Donanım kısıtlamaları nedeniyle yüksek çözünürlüklü görseller üretilememiştir. GAN modeli CPU üzerinde çalıştırıldığından eğitim süresi uzamış ve çıktı kalitesi sınırlı kalmıştır.

Kullanılan veri seti büyüklüğü oldukça küçüktür. Özellikle BERT fine-tuning ve GAN eğitim süreçleri daha büyük ve dengeli veriyle çok daha iyi performans gösterebilirdi.

ASR modülünün başarımı yalnızca belirli örneklerde test edildi, genel başarı oranı ölçülmemiştir.

Görüntü kalitesinin değerlendirilmesi için Inception Score (IS) veya Fréchet Inception Distance (FID) gibi metrikler bu projede hesaplanmamış olsa da, görsel çıktılar üzerinden kalite analizi yapılmıştır.

KAYNAKÇA

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). *Attention is all you need.* In *Advances in neural information processing systems*, 30.

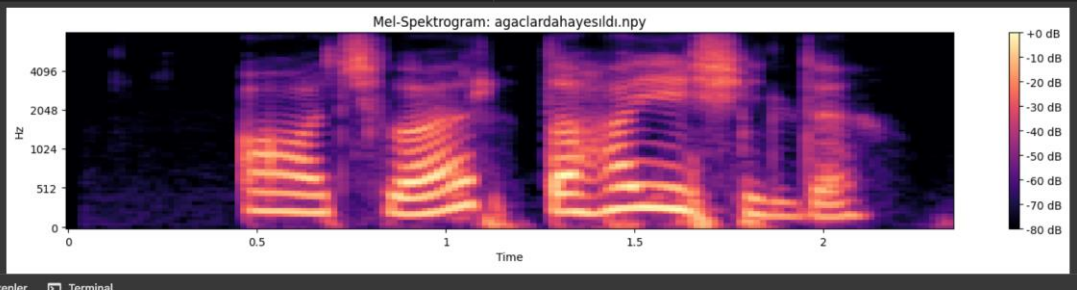
Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2019).
BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., ... & Bengio, Y. (2014).
Generative adversarial nets. In *Advances in neural information processing systems*, 27.

Odena, A., Olah, C., & Shlens, J. (2017).
Conditional Image Synthesis with Auxiliary Classifier GANs. *arXiv preprint arXiv:1610.09585*.

PyTorch Documentation – https://pytorch.org/docs/stable/index.html
(GAN implementasyonunda kullanılan framework'ün resmi dökümantasyonü)

```python
import matplotlib.pyplot as plt
import numpy as np
import os

mel_folder = "/content/drive/MyDrive/mel_spectrograms"

# Örnek olarak 3 mel spektrogramını görselleştir
example_files = ["agaclardahayesildi.npy", "annesicokduygulandi.npy", "amaherseferindecesaretiyle.npy"]

plt.figure(figsize=(15, 10))

for i, mel_file in enumerate(example_files, 1):
    mel_path = os.path.join(mel_folder, mel_file)
    mel_db = np.load(mel_path)

    plt.subplot(len(example_files), 1, i)
    librosa.display.specshow(mel_db, sr=16000, hop_length=256, x_axis='time', y_axis='mel')
    plt.colorbar(format='%+2.0f dB')
    plt.title(f"Mel-Spektrogram: {mel_file}")
    plt.tight_layout()

plt.show()
```



Değişkenler Terminal

```python
visualize(model, dataset, bert, idx=0)
```


```python
import torch
from torch.utils.data import Dataset, DataLoader
from transformers import BertTokenizer, BertForSequenceClassification, AdamW

class SceneDataset(Dataset):
    def __init__(self, csv_file, tokenizer, max_len=64):
        self.data = pd.read_csv(csv_file)
        self.tokenizer = tokenizer
        self.max_len = max_len

    def __len__(self):
        return len(self.data)

    def __getitem__(self, idx):
```

```
Epoch [6/20] | D Loss: 0.4666 | G Loss: 3.4927
Epoch [7/20] | D Loss: 0.5858 | G Loss: 4.2426
Epoch [8/20] | D Loss: 0.4565 | G Loss: 3.3592
Epoch [9/20] | D Loss: 0.6899 | G Loss: 2.3086
Epoch [10/20] | D Loss: 0.4494 | G Loss: 3.6415
Epoch [11/20] | D Loss: 0.6369 | G Loss: 2.8735
Epoch [12/20] | D Loss: 0.5747 | G Loss: 2.4768
Epoch [13/20] | D Loss: 0.6379 | G Loss: 2.8905
Epoch [14/20] | D Loss: 0.5978 | G Loss: 2.6828
Epoch [15/20] | D Loss: 0.6794 | G Loss: 2.2465
Epoch [16/20] | D Loss: 0.7410 | G Loss: 2.3925
Epoch [17/20] | D Loss: 0.6649 | G Loss: 2.1596
Epoch [18/20] | D Loss: 0.6643 | G Loss: 2.7830
Epoch [19/20] | D Loss: 1.0106 | G Loss: 3.2814
Epoch [20/20] | D Loss: 0.8726 | G Loss: 2.3120
```