Film Afişlerinin Türlerine Göre Sınıflandırılması

D.Murat Ertik
Bilişim Sistemleri Mühendisliği
221307065
dmuratertik1@gmail.com

***Abstract*—Film afişleri, yalnızca tanıtım amacı taşımayan; aynı zamanda izleyicilere filmin türü, atmosferi ve içeriği hakkında ilk izlenimi sunan güçlü görsel araçlardır. Bu çalışmada, farklı türlerdeki film afişlerinin yalnızca görsel ögeleri temel alınarak otomatik olarak sınıflandırılması hedeflenmiştir. Bu çalışma, afiş tasarımlarındaki görsel kalıpları analiz ederek makine öğrenmesi algoritmalarının tür sınıflandırmasında ne derece başarılı olabileceğini ortaya koymakta ve bu alandaki otomatikleştirilmiş içerik tanıma sistemlerine katkı sunmayı amaçlamaktadır.**

Keywords—Text Processing, Artificial Intelligence, Natural Language Processing, Transformer Models, Model Training

I. PROJE AMACI

Görseller, modern dünyada bilgi aktarımının en güçlü yollarından biri haline gelmiştir. Özellikle film afişleri, izleyicilere bir filmin türü, atmosferi ve hikayesi hakkında kısa sürede fikir veren önemli bir görsel araçtır. Film yapımcıları, afiş tasarımlarında belirli renk paletleri, yazı tipleri ve görsel ögeler kullanarak filmlerinin türünü yansıtmayı hedeflerler. Örneğin, korku filmleri genellikle karanlık tonlara ve ürkütücü figürlere yer verirken, komedi filmleri daha parlak ve neşeli tasarımlar kullanır.

Bu projede, film afişlerini görsel özelliklerine göre sınıflandırarak, bir afişin hangi film türüne ait olduğunu tahmin eden bir makine öğrenmesi modeli geliştirmek amaçlanmaktadır. Bu sayede, film afişlerinin sadece görsel ögeleri kullanılarak türlerinin belirlenip belirlenemeyeceği incelenecek ve farklı türlerin afiş tasarımları arasındaki görsel farklılıklar analiz edilecektir.

Projenin en önemli aşamalarından biri veri toplama sürecidir. Bu çalışmada, film afişleri web scraping yöntemi ile web sayfasından manuel olarak toplanacak ve veri seti oluşturulacaktır. Letterbox sitesinden web scraping teknikleri ile afişler çekilecektir. Her biri farklı görsel özelliklere sahip altı film türü (Aksiyon, Komedi, Korku, Bilim Kurgu, Animasyon ve Fantastik) seçilmiş olup, her tür için en az 10000 afiş toplanarak dengeli bir veri seti oluşturulacaktır.

Bu çalışmanın temel amacı, makine öğrenmesi ve bilgisayarla görme tekniklerini kullanarak film afişlerinin otomatik olarak sınıflandırılmasını sağlamaktır. Elde edilen model, hem akademik hem de ticari uygulamalar için faydalı olabilir. Örneğin, bir film arşivleme sisteminde veya otomatik içerik öneri mekanizmalarında kullanılabilir. Ayrıca, film endüstrisinde afiş tasarımcılarına farklı türlerde afişlerin hangi görsel ögelerle öne çıktığını anlamaları için içgörüler sağlayabilir.

Bu raporda, öncelikle veri toplama süreci detaylandırılacak, ardından veri ön işleme adımları ve modelin eğitim süreci açıklanacaktır. Son olarak, modelin performansı değerlendirilecek ve elde edilen sonuçlar yorumlanacaktır.

II. VERI KAYNAĞI VE TOPLAMA SÜRECI

A. Letterboxd Platformu ve Filtreleme Seçenekleri

Letterboxd [1], geniş bir film arşivi sunarak kullanıcıların belirli kriterlere göre film araması yapmasına imkan tanır. Özellikle tür (genre), yıl, popülerlik ve kullanıcı puanları gibi çeşitli filtreleme seçenekleri sayesinde belirli kategorilerdeki filmleri kolayca listelemek mümkündür.

Bu projede, belirli film türlerine ait afişleri manuel olarak toplamak için Letterboxd'un filtreleme seçenekleri kullanılmıştır. Örneğin, sadece belirli türleri içeren ve diğer türlerden bağımsız filmleri listelemek amacıyla aşağıdaki URL formatı kullanılmıştır:

```
https://letterboxd.com/films/popular/genre/action+-/
```

Bu URL, yalnızca "Aksiyon" türüne ait filmleri filtreleyerek, diğer türlere ait filmleri dönen bir listeleme sağlar ve bu filtre sayısı arttırılarak hedef türe gidilmesi hedeflenir.

B. Filtrelemenin Veri Seti İçin Önemi

- **Veri Setinde Türler Arasında Net Ayrım Olmasını Sağlar**
  Bir film birden fazla türe sahip olabilir, ancak modelin belirli türleri tanıyabilmesi için tek tür içeren afişler kullanılmalıdır. Letterboxd'ın tür filtreleme özelliği sayesinde, belirli türlere odaklanarak türler arası veri kirliliği önlenmiştir.
- **Daha Doğru ve Dengeli Bir Model Eğitimi Sunar**
  Eğer bir türdeki veri sayısı diğerine göre çok fazla olursa, model dengesiz öğrenir ve bazı türlere öncelik verir (*class imbalance* problemi). Filtreleme sayesinde her tür için eşit sayıda veri toplanarak modelin daha adil bir şekilde eğitilmesi sağlanmıştır.
- **Yanlış Etiketleme Sorununu Minimize Eder**
  IMDb gibi platformlarda bir film hem "Komedi" hem "Aksiyon" olarak işaretlenmiş olabilir, bu da veri setinde yanlış sınıflandırmalara neden olur. Letterboxd'ın detaylı filtreleme özelliği sayesinde, her film yalnızca belirli türlere göre sınıflandırılmış ve yanlış etiketlenmiş veriler en aza indirilmiştir.
- **Görsel Özelliklerin Daha Net Ayırt Edilmesini Sağlar**
  Eğer bir afiş hem korku hem bilim kurgu türünde olursa,

renk paleti ve tasarım unsurları karışabilir ve model doğru sınıflandıramaz. Filtreleme ile belirli türlere özgü görsel kalıplar korunmuş ve modelin daha iyi genelleme yapabilmesi sağlanmıştır.

Bu nedenlerle, veri toplama aşamasında yapılan filtreleme işlemi modelin doğruluğunu ve güvenilirliğini artıran kritik bir adımdır. *Scraping* yöntemiyle Letterboxd'un sunduğu detaylı filtreleme özellikleri kullanılarak temiz ve yüksek kaliteli bir veri seti oluşturulması hedeflenmiştir.

III. Veri Toplama

Web scraping işlemi, hedef web sayfalarından görsel verilerin toplanmasını sağlayan bir yöntemdir. Bu süreç, aşağıdaki adımlarla açıklanabilir:

Filtreler ile Etiketin Bulunması

Chrome tarayıcısı otomatik olarak başlatılır ve belirlenen URL'ye yönlendirilir. Burada kullanılan URL, yalnızca "Bilim Kurgu" türünü içeren filmleri gösterirken, diğer tüm türleri filtre dışı bırakır. Bu sayede veri setine yalnızca hedeflenen türde filmler dahil edilir.

Veri Sayısı ve Kontrol Değişkenlerinin Tanımlanması

Bu bölümde, veri toplama sürecinde kullanılacak bazı temel değişkenler tanımlanmıştır. Bu değişkenler, scraping işleminin kontrollü ve takip edilebilir bir şekilde ilerlemesini sağlar:

```
poster_urls = []
total_posters = 1000
current_page = 1
```

poster_urls:
Bu liste, indirilen tüm film afişi bağlantılarını tutmak amacıyla kullanılır. Aynı görselin birden fazla kez indirilmesini engellemek için her yeni görselin bağlantısı bu listeye eklenir. Böylece tekrar eden verilerden kaçınılır ve veri setinin kalitesi korunur.

total_posters:
Toplamda kaç adet film afişi indirilmek istendiğini belirten değişkendir. Bu değer sayesinde scraping işlemi rastgele değil, hedefe yönelik ve sınırlandırılmış bir şekilde yürütülür. Örneğin, bu projede 1000 adet görsel belirlenmiştir. Bu sayede hem veri yeterliliği sağlanır hem de zaman ve kaynak tasarrufu yapılır.

current_page:
Letterboxd sitesi sayfalama (pagination) sistemine sahiptir. Her sayfada belirli sayıda film listelenir. Bu değişken sayesinde kod, kaçıncı sayfada işlem yaptığını bilir ve bir sonraki sayfaya ne zaman geçileceğini kontrol eder. Bu durum scraping sürecinin doğruluğunu ve sürekliliğini arttırmasını planlar.

Bu üç değişkenin birlikte kullanımı, scraping sürecinin sistematik, düzenli ve tekrarsız bir şekilde çalışmasına olanak tanır. Böylece hem işlem performansı hem de toplanan verinin doğruluğu önemli ölçüde artar.

Sayfa Gezinme ve Afiş URL'lerinin Toplanması

Veri toplama sürecinin ana gövdesini oluşturan bu bölümde, belirlenen sayıda film afişini elde etmek amacıyla web sitesindeki sayfalar arasında gezilmekte ve afiş URL'leri toplanmaktadır. Bu işlemler, belirli bir mantık çerçevesinde bir while döngüsü içerisinde gerçekleştirilmiştir:

```
while len(poster_urls) < total_posters:
```

Bu koşullu döngü, topladığımız afiş sayısı , hedeflenen toplam sayıya ulaşana kadar çalışmaya devam eder. Bu sayede sistem, istenen sayıda veri toplanmadan işlemi sonlandırmaz.

Letterboxd gibi modern web siteleri dinamik içerik barındırdığı için, sayfanın tam olarak yüklenip tüm bileşenlerin hazır hale gelmesi beklenmelidir. Bu adım, verilerin eksik ya da hatalı toplanmasının önüne geçmek için gereklidir.

```
WebDriverWait(driver, 13).until(
lambda d: d.execute_script("return document.readyState") == "complete"
)
```

Hedef İçeriklerin Çıkarılması

Sayfa içeriği BeautifulSoup kütüphanesi yardımıyla analiz edilir ve her bir film afişi içeren HTML öğesi bulunur. Bu öğeler daha sonra tek tek işlenerek, her filme özel detay sayfasına geçiş yapılır.

Bu bölüm, scraping sürecinde doğru ve eksiksiz veri toplayabilmek için oldukça kritiktir. Dinamik sayfalarda verinin doğru şekilde alınabilmesi için yüklenme durumlarının kontrol edilmesi ve tüm içeriklerin görünür hale getirilmesi, veri setinin güvenilirliği açısından büyük önem taşır.

```
soup = BeautifulSoup(driver.page_source, "html.parser")
posters = soup.find_all("li", class_="poster-container")
```

Film Sayfalarına Girerek Büyük Afişleri Toplama

Bu aşamada, ana liste sayfasında her film için yalnızca küçük boyutlu afişler gösterildiğinden, her bir filmin kendi detay sayfasına gidilerek yüksek çözünürlüklü afiş görsellerine ulaşılması hedeflenmiştir. Bu işlem, scraping sürecinin en kritik adımlarından biridir; çünkü asıl veri olan afiş görselleri bu sayfalarda yer almaktadır.

```
link = poster.find("a", class_="frame")
film_url = "https://letterboxd.com" + link.get("href")
driver.get(film_url)
```

Her poster-container elemanında, filme ait detay sayfaya yönlendiren bir bağlantı (href) yer almaktadır. Bu bağlantı, sitenin kök adresi ile birleştirilerek tam bir URL elde edilir ve Selenium aracılığıyla bu sayfaya geçiş yapılır.

```
img_link = soup.find("a", {"data-js-trigger": "postermodal"})
img_url = img_link.get("href")
```

Film detay sayfası yüklendikten sonra, burada yer alan büyük boyutlu afiş bağlantısı data-js-trigger="postermodal" özelliğine sahip a etiketi içerisinden alınır. Bu bağlantı doğrudan JPEG ya da PNG formatında büyük boyutlu postere ulaşmayı sağlar.

```
img_data = requests.get(img_url).content
file_name =
os.path.join(save_dir, f"poster_{len(poster_urls)}.jpg")
with open(file_name, "wb") as f:
    f.write(img_data)
```

Elde edilen görsel bağlantısı üzerinden HTTP isteği yapılır ve afiş dosyasının içeriği alınır. Her görsel, önceden belirlenmiş bir klasöre ve sistematik bir isimlendirme yapısına göre (poster1.jpg gibi) kaydedilir. Bu düzenli yapı, veri setinin kullanılabilirliğini ve sınıflandırma işlemlerini kolaylaştırır.

Bu bölüm sayesinde, görsel sınıflandırma projesinin temelini oluşturan yüksek kaliteli ve etiketli görseller başarıyla elde edilmiş olur. Doğrudan ana sayfadaki küçük boyutlu görseller yerine detay sayfalardan alınan yüksek çözünürlüklü posterler, modelin eğitiminde daha başarılı sonuçlar elde edilmesini sağlayacaktır.

Sayfa Değişimi (Pagination) ve Otomasyonun Sürekliliği

Letterboxd platformunda filmler sayfalı bir şekilde listelenmektedir; yani her sayfada sınırlı sayıda film afişi yer almaktadır. Bu nedenle, scraping işlemi sırasında yalnızca bir sayfaya odaklanmak yeterli olmayacaktır. Toplanmak istenen veri sayısına ulaşmak için birden fazla sayfaya geçiş yapılması gerekir. Bu geçişlerin otomatik olarak gerçekleştirilmesi için sayfa değişimi (pagination) mantığı kod içerisine dahil edilmiştir.

Scraping işlemi sırasında hangi sayfada bulunulduğu, current page değişkeni ile izlenmektedir. Bu sayaç, kullanıcıya bilgi vermek ve olası hata durumlarında süreci takip edebilmek için kullanışlıdır.

Letterboxd, sayfa alt kısmında bir "Next" (İleri) butonu sunmaktadır. Selenium kütüphanesi yardımıyla bu butonun aktif ve tıklanabilir hale gelmesi beklenir. Daha sonra, otomatik olarak bu butona tıklanarak bir sonraki sayfaya geçiş sağlanır. Bu işlem, scraping döngüsünün devamlılığını sağlar.Sayfa geçişi sonrası yeni içeriğin tam olarak yüklenebilmesi adına kısa bir bekleme süresi eklenmiştir. Bu bekleme, hem veri kaybını önler hem de sunucuya aşırı yüklenmenin önüne geçerek scraping işleminin engellenmesini riskini azaltır.Her zaman "Next" butonu bulunmayabilir (örneğin son sayfadaysa). Böyle durumlar için try-except bloğu ile hata kontrolü yapılır. Eğer butona ulaşılamazsa işlem güvenli bir şekilde sonlandırılır.[2]

```
next_button = WebDriverWait(driver, 10).until(
    EC.element_to_be_clickable((By.CSS_SELECTOR,
    "div.paginate-nextprev a.next"))
)
next_button.click()
```

Web Scraping İşlemi Sonucu Elde Edilen Veri Sayısı

Genre	Sayı
Action	5638
Animation	5665
Comedy	5637
Horror	5806
Sci-Fi	5821

TABLE I
TÜRLERE GÖRE VERI SAYISI

Görsel Veri Setinde Veri Temizleme Aşamaları

1) Çok Küçük veya Aykırı Boyutlardaki Görsellerin Silinmesi: Makine öğrenmesi projelerinde kullanılan görsel veri setlerinde, çok küçük boyutlu veya standart dışı ölçülere sahip görsellerin bulunması modelin öğrenmesini olumsuz etkileyebilir. Bu tür görseller yeterli detay içermediğinden dolayı, sınıflar arasındaki ayırt edici özellikleri yansıtamaz ve modelin doğruluk oranını düşürebilir.

Bu nedenle veri temizliği aşamasında, görsellerin çözünürlüklerine dikkat edilerek bir alt eşik belirlenir (*örneğin* 100x100 piksel). Ancak bu tür görsellerin tespiti her zaman otomatik yapılmaz. Birçok durumda, görseller kullanıcı tarafından manuel olarak gözden geçirilir. Gözle belirgin şekilde bulanık, kırpılmış, ya da içeriği eksik olan görseller tespit edilerek elle silinir. Bu yöntem zaman alıcı olsa da, özellikle küçük ölçekli ya da hassas veri setlerinde yüksek doğruluk sağlar.

Elle yapılan bu temizlik sürecinde dikkat edilen bazı noktalar şunlardır:

- Görselin içeriği incelenir; anlamsız, bozuk ya da eksik içerik taşıyanlar elenir.
- Görseller arasında orantısız boyutlara sahip olanlar ayıklanır.
- Çözünürlüğü çok düşük olan, ayrıntı içermeyen görseller manuel olarak temizlenir.

Bu süreç, modelin kaliteli ve bilgi yoğun görsellerle eğitilmesini sağlar ve eğitim sırasında gürültü oluşturan verilerin etkisini azaltır.

2) Çok Benzer / Yinelenen Görsellerin Tespiti: Görsel veri setlerinde aynı veya neredeyse aynı içeriğe sahip görsellerin birden fazla kez yer alması, modelin öğrenme sürecini yanıltabilir. Daha da önemlisi, aynı görselin farklı dosya adlarıyla farklı sınıflara atanmış olması, modelin sınıflar arası farkları öğrenmesini engelleyebilir. Bu durum sınıflar arası karışıklığa ve yanlış sınıflandırmalara yol açar.

Bu tür görsellerin tespiti otomatik araçlarla yapılabileceği gibi, çoğu zaman özellikle küçük ve nitelikli veri setlerinde **manuel olarak** yapılır. Kullanıcı, görselleri tek tek inceleyerek içerik açısından benzer olanları belirler ve tekrar eden ya da başka sınıfa ait olduğu halde yanlış etiketlenmiş görselleri el ile siler.

Manuel temizlik sürecinde şu adımlar izlenir:

- Görseller yan yana ya da üst üste görüntülenerek karşılaştırılır.

- Aynı görselin farklı klasörlerde veya dosya adlarıyla bulunup bulunmadığı gözle kontrol edilir.
- Farklı sınıflarda yer alan ancak aynı içeriğe sahip görseller belirlenip uygun şekilde sınıflandırılır ya da silinir.

Elle yürütülen bu işlem zaman alıcı olsa da, veri setinin güvenilirliğini artırır. Özellikle sınıf etiketlerinin kritik öneme sahip olduğu durumlarda, bu adım modelin performansı için belirleyici bir rol oynar.

Manuel Veri Temizleme Sonucu Elde Kalan Veri Sayısı

Genre	Sayı
Action	5178
Animation	5428
Comedy	5136
Horror	5232
Sci-Fi	5520

TABLE II
TÜRLERE GÖRE KALAN VERI SAYISI 1

Otomatik olarak ise belirtilen bir klasör içindeki tüm alt klasörlerle birlikte dolaşarak, yalnızca .jpg, .jpeg ve .png uzantılı görselleri dikkate alır ve bu görsellerin içeriklerine göre ikili kopyalarını (tamamen aynı içeriğe sahip olan dosyaları) tespit edip siler. Her bir görsel dosyasının içeriği okunur ve MD5 algoritması kullanılarak bir özet değeri (hash) oluşturulur. Bu özet değer, aynı içeriğe sahip olan dosyaların aynı hash'e sahip olmasını sağlar. Kod, daha önce karşılaşılan hash'leri bir sözlükte saklar; yeni bir dosya işlendiğinde, hash daha önce görülmüşse dosyanın zaten mevcut bir kopyası olduğu kabul edilir ve bu durumda dosya silinir. Eğer hash daha önce görülmemişse, sözlüğe eklenir ve dosya korunur. Bu sayede yalnızca benzersiz görsel dosyalar klasörde tutulur, içerik olarak tamamen aynı olanlar ise temizlenmiş olur.

Manuel Veri Temizleme Sonucu Elde Kalan Veri Sayısı

Genre	Sayı
Action	3943
Animation	3855
Comedy	3767
Horror	3969
Sci-Fi	3975

TABLE III
TÜRLERE GÖRE KALAN VERI SAYISI 2

A. Veri Setinin Görsel Boyut Dağılımının Görselleştirilmesi

Veri setinde yer alan her bir görselin genişlik ve yükseklik değerleri `Pillow` kütüphanesi ile elde edilmiştir. Amaç, görsellerin çözünürlükleri hakkında genel bir fikir edinmek ve model eğitiminde standardizasyon gerekip gerekmediğini analiz etmektir.

Şekil 2'da iki ayrı histogram yer almaktadır. Sol taraftaki grafik, tüm veri setindeki görsellerin genişlik dağılımını; sağ taraftaki grafik ise yükseklik dağılımını göstermektedir. Görüldüğü üzere:

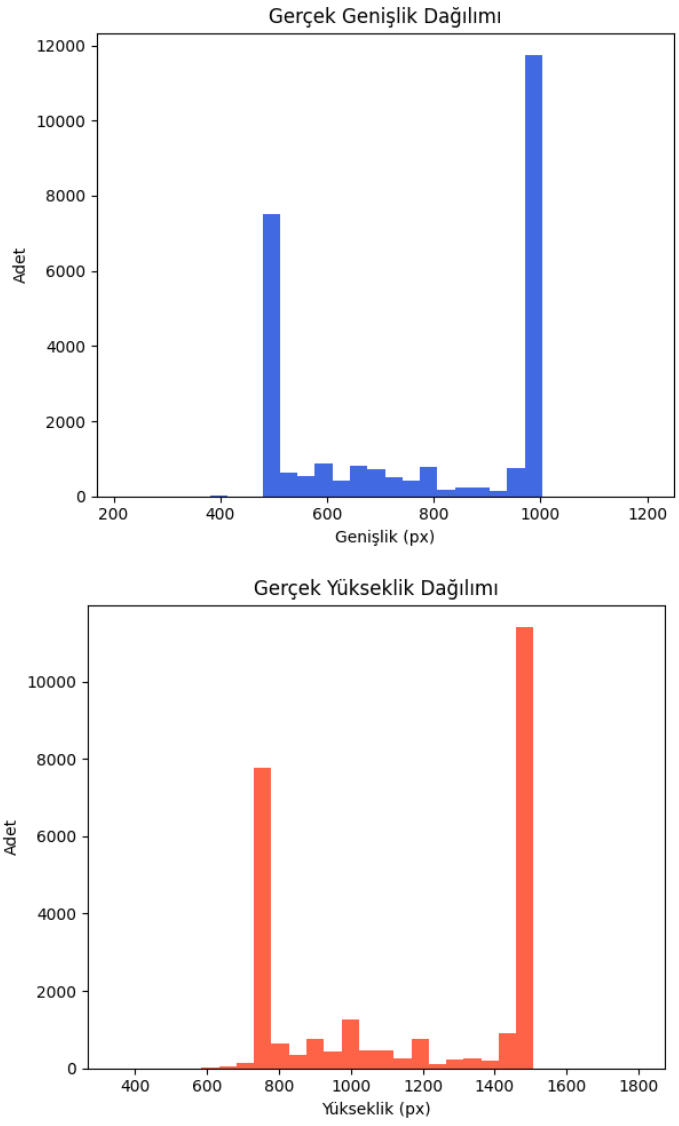

Fig. 1. Veri setindeki görsellerin genişlik ve yükseklik dağılımı histogramları

- Görsellerin büyük bir çoğunluğu **1024 piksel genişliğe** ve **1536 piksel yüksekliğe** sahiptir. Bu durum, veri setinin önemli bir kısmının aynı çözünürlükte olduğunu göstermektedir.
- Ayrıca 500 piksel civarında ikinci bir yoğunluk daha gözlemlenmektedir. Bu da çözünürlük açısından sınırlı sayıda farklılık bulunduğunu ortaya koymaktadır.
- Dağılımın çift tepe yapısı göstermesi, model eğitimine geçmeden önce görsellerin **yeniden boyutlandırma (resize)** işlemine tabi tutulmasının faydalı olacağını göstermektedir.

Bu analiz sayesinde, eğitim verisinin çözünürlük yönünden homojen olup olmadığı anlaşılmış, gerekiyorsa görsellerin normalize edilerek modelin daha verimli eğitilmesi için ön adım atılmıştır.

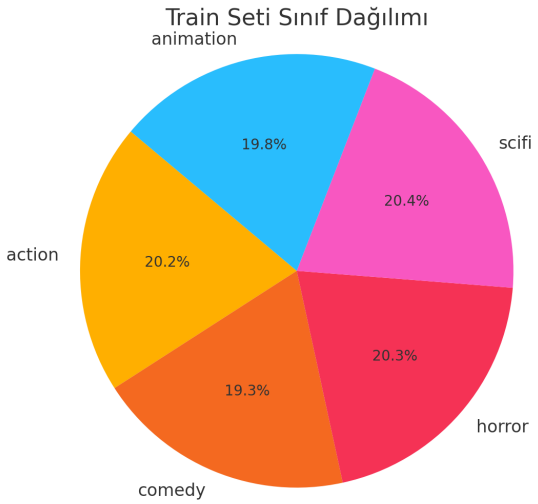

Fig. 2. Veri setinin train setindeki dağılımı

Train seti için oluşturulan sınıf dağılımı grafiği, veri setinin oldukça dengeli bir şekilde yapılandırıldığını göstermektedir. Beş sınıfın her biri, toplamda yaklaşık 3.000 ila 3.200 görsel arasında değişen sayıda örneğe sahiptir ve bu da her sınıfın train setinde yaklaşık %20 oranında temsil edildiği anlamına gelir. Bu tür bir dağılım, sınıflar arası dengesizlik sorunlarını en aza indirerek modelin öğrenme sürecinde belirli sınıflara yönelme riskini ortadan kaldırır. Aynı şekilde validation ve test setlerinin de train setine paralel şekilde %10 oranında ayrılması, modelin doğrulama ve genelleme yeteneklerini test etmede istatistiksel olarak yeterli ve sağlıklı bir yaklaşım sunar. Bu durum, modelin performansını değerlendirirken her sınıfın eşit temsiliyet bulmasını sağlar. Dolayısıyla bu yapıdaki 80-10-10 oranlı ve dengeli sınıf dağılımlı dataset, hem eğitim süreci hem de genel değerlendirme açısından oldukça sağlam ve güvenilir bir temel oluşturur.

IV. Model Eğitim Süreci ve Elde Edilen Değerlerin Yorumu

Bu aşamada, film afişlerinin türlerine göre sınıflandırılması amacıyla transformatör tabanlı derin öğrenme modelleri kullanılmıştır. Görsel veri üzerinde çalışan bu modeller, aynı veri kümesi üzerinde uygulanarak sınıflandırma performansları karşılaştırılmıştır. Farklı mimari yapıya sahip beş model seçilmiş ve her biri belirli ölçütlere göre değerlendirilmiştir. Amaç, çeşitli transformatör tabanlı yaklaşımların bu görevdeki etkinliğini karşılaştırmalı olarak incelemektir.

A. Swin

Swin Transformer modeli, görsel sınıflandırma görevleri için geliştirilmiş bir transformatör tabanlı mimaridir. Klasik Vision Transformer (ViT) yapısının aksine, Swin modeli görüntüyü sabit boyutlu yama (patch) bloklarına ayırmakla kalmaz, aynı zamanda bu yamalar üzerinde kayan pencere (sliding window) mekanizmasıyla çalışarak yerel ve küresel özellikleri daha verimli bir şekilde işler. Bu yapı sayesinde hem küçük hem de büyük nesne detaylarını dikkate alabilir. Swin modeli, farklı çözünürlük seviyelerinde işlem yaparak hiyerarşik bir temsil oluşturur ve bu sayede özellikle sınıflandırma gibi görevlerde dengeli ve etkili bir performans sunar. Bu projede Swin modeli, film afişlerini türlerine göre sınıflandırma görevinde diğer modellerle birlikte değerlendirilmiştir.[2]

1) Kodların Açıklanması:

a) Konfigürasyon Ayarları (CONFIG): Model, 30 epoch boyunca `batch_size=32` ile eğitiliyor. Bu küçük batch size, daha sık model güncellemeleri sağlayarak genellemeyi iyileştirir. Eğitim `swin_tiny_patch4_window7_224` mimarisiyle, `drop_rate=0.5` ve `drop_path_rate=0.3` gibi güçlü regularizasyonla yapılandırılmıştır. Öğrenme oranı (LR) düşük seçilerek (`1e-5`) daha stabil öğrenme hedeflenmiş, `weight_decay=2e-2` ile modelin parametrelerinin fazla büyümesi engellenmiştir.

b) Görüntü Dönüşümleri (TRANSFORMS): Eğitim verisine oldukça zengin augmentasyon uygulanıyor: `RandomResizedCrop`, `Rotation`, `Perspective`, `Affine` gibi işlemler sayesinde modelin overfitting eğilimi azaltılıyor. Validation ve test transformlarında sade `Resize + Normalize` kullanılmıştır ki bu da değerlendirme tutarlılığı sağlar.

c) Dataset ve DataLoader: Veriler klasör bazlı olarak `ImageFolder` yapısı ile yükleniyor. `Shuffle` ve `pin_memory` gibi ayarlar eğitim sırasında GPU verimliliğini artırıyor. `val` ve `test loader`'ların shuffle edilmemesi, doğruluk ölçümlerinde deterministik sonuçlar elde etmeye yardımcı olur.

d) Model, Kayıp Fonksiyonu, Optimizasyon ve Scheduler: Model, `timm` kütüphanesinden `Swin Tiny` ile oluşturulmuş. Optimizasyon için `AdamW` seçilmiş, bu da `weight_decay` ile birlikte daha etkili bir regularizasyon sağlar. `ReduceLROnPlateau` yerine `CosineAnnealingLR` tercih edilerek öğrenme oranı zamanla yumuşakça azaltılır, bu da eğitim sonunda ince ayarların yapılabilmesini sağlar.

e) Mixup ile Eğitim ve Doğrulama: Eğitim sırasında mixup kullanılarak giriş görüntüleri ve etiketler karıştırılıyor. Bu sayede model daha genelleyici hale gelir ve ezberleme eğilimi kırılır. `CrossEntropyLoss` fonksiyonu mixup uygulanmış hedefler ile birlikte kullanılıyor. Epoch sonunda hem train hem de val kayıpları izleniyor ve model `final_swin_model.pth` olarak kaydediliyor.

f) Modelin Yüklenmesi ve Test İnferansı: Model, ağırlık dosyasından `final_swin_model.pth` yükleniyor. `eval()` modu ile dropout vb. eğitimde kullanılan işlemler devre dışı bırakılıyor.

Tüm test verisi `softmax` ile sınıf olasılıklarına dönüştürülerek tahminler alınmakta ve `labels`, `preds`, `probs` listelerine toplanmaktadır. Bu yapı, hem doğruluk hem de AUC gibi metrikler için gereklidir. Ayrıca *inference time* da ölçülerek modelin üretim (deployment) için uygunluğu değerlendirilmiştir.

g) Test Metrikleri Hesaplama: Bu bölümde modelin başarı düzeyi çeşitli açılardan ölçülmüştür:

- **Accuracy, Precision, Recall, F1-Score (macro):** Genel sınıflandırma başarısını gösterir. Macro kullanılması her sınıfa eşit ağırlık verir.
- **ROC AUC (micro ve macro):** Modelin sınıf ayrım gücü. Micro → örnek bazlı, Macro → sınıf bazlı ortalama.
- **Specificity:** Her sınıf için yanlış pozitifleri ne kadar az verdiğini ölçer. Bu, özellikle dengesiz veri setlerinde çok anlamlıdır.

Hesaplama kısmında `TP/TN/FP/FN` oranları manuel olarak çıkarılarak detaylı analiz yapılmıştır — bu da modelin sadece doğru tahminlerine değil, yanlış tahminlerinin nereden geldiğine de ışık tutar.

h) Görselleştirmeler: Bu bölüm, modelin davranışını net biçimde anlamayı sağlar:

- **Confusion Matrix:** Her sınıfın hangi sınıfla karıştığını gösterir. Eğitim sonrası en net hataların görüldüğü yerdir.
- **ROC Eğrileri:** Her sınıf için TPR-FPR eğrileri çizilir. Eğri ne kadar yukarıda ve sola yakınsa model o sınıf için o kadar güçlüdür.
- **Loss Eğrisi (Train vs Val):** Eğitim süreci boyunca overfitting olup olmadığı, hangi epoch'ta öğrenmenin yavaşladığı burada gözlemlenir.

i) Genel Değerlendirme: Bu test bölümü, sadece ham doğruluk üzerinden değil, güçlü metrikler ve görsel analizlerle modelin davranışını detaylı bir şekilde anlamanızı sağlar. ROC eğrileri ve `specificity` gibi daha ileri metriklerin dahil edilmesi, modelin sadece "doğru mu bildi" değil, "ne kadar güvenli ve tutarlı tahmin yaptı" sorusunu da cevaplar.

j) Not: İki Kez Eğitim ve Parametre Oynama: Bu model, önceki bir versiyonun sonuçları incelenip overfitting ve doğrulama dalgalanmalarını azaltmak amacıyla yeniden yapılandırılmıştır. `Mixup` α değeri, `batch size`, `scheduler` yapısı ve `weight decay` gibi önemli hiperparametrelerde ayarlamalar yapılmıştır. Bu müdahalelerin sonuçlara etkisi ilerleyen bölümlerde detaylı olarak açıklanacaktır.

ikinci elde edilen eğitilmiş modelde kullanılan konfigürasyon ayarları diğer eğitilecek 4 modelde de aynı şekilde kullanılacaktır daha iyi bir karşılaştırma yapılabilmesi için.

2) Sonuçların Yorumları: İlk olarak, modelin ardışık iki eğitim sürecinde değiştirilen hiperparametreler incelenecek ve bu değişikliklerin modelin genel performansı üzerindeki etkileri detaylı biçimde yorumlanacaktır. Eğitim sürecine ait metrikler, doğrulama ve test sonuçları üzerinden karşılaştırmalı olarak değerlendirilecek; özellikle doğruluk, kayıp, AUC, F1 gibi temel başarı ölçütleri temel alınarak parametre değişikliklerinin öğrenme dinamiklerine katkısı ortaya konacaktır.

a) 1. Batch Size: 64 → 32: Batch size, her iterasyonda modele verilen örnek sayısını belirler. Büyük batch size (64), daha az gürültülü ve daha kararlı gradyanlar üretir; fakat modelin fazla genelleme yapmasına engel olabilir. Küçük batch size (32) ise daha fazla varyasyon ve stochasticity getirir. Küçük batch'ler, modele veri içindeki farklılıkları daha net öğretir ve daha fazla mini güncelleme ile regularization etkisi oluşur. Modelin validation kaybı daha stabil seyretmiş, validation accuracy artmıştır. Ayrıca validation eğrisi eğitim eğrisine daha paralel ilerlemiştir.

Parametre	Eğitim-1	Eğitim-2
Batch Size	64	32
Learning Rate	1e-5	1e-5
Weight Decay	5e-3	2e-2
Scheduler	ReduceLROnPlateau	CosineAnnealing
Dropout Path Rate	x	0.3

TABLE IV
İKI EĞITIM ARASINDA PARAMETRE FARKI

b) 3. Learning Rate: Sabit (1e-5): Öğrenme oranı her iki modelde aynı tutulmuştur. Bu, deneme sürecinde kontrol değişkeni gibi davranır. Bu parametre değişmediği için overfitting farkı yaratmamıştır. Ancak küçük seçilmesi sayesinde her iki model de daha hassas ve dikkatli öğrenmiştir.

c) 4. Weight Decay: 5e-3 → 2e-2: Weight decay, ağırlıkların çok büyümesini önleyerek modelin karmaşıklığını sınırlayan bir L2 regularization yöntemidir. İkinci modelde daha güçlü bir değer seçilmiştir. Modelin karmaşık fonksiyonlar öğrenmesini engeller, aşırı uyumu (overfitting) azaltır ve daha sade temsiller öğrenilmesini sağlar. Validation AUC ve macro F1 değerleri artmış, validation loss eğrisi ise daha kararlı seyretmiştir.

d) 5. Scheduler: ReduceLROnPlateau → CosineAnnealing: İlk modelde ReduceLROnPlateau, validation loss düşmezse learning rate'i azaltırdı. İkinci modelde ise CosineAnnealing, learning rate'i dönemsel ve yumuşak bir biçimde düşürür. Öğrenme oranı kontrollü düştüğü için modelin son epoch'larda ezber yapma eğilimi azalır. Val loss eğrisi belirgin şekilde daha düzgün ve düşüktür. Learning rate sabit düşmediği için model daha iyi genelleme yapar.

e) 6. Dropout Path Rate: (x) → 0.3: Drop path, özellikle transformer tabanlı modellerde kullanılan bir regularization yöntemidir. Belirli blokların (katman yollarının) rastgele atlanmasını sağlar. İlk modelde hiç kullanılmamış, ikinci modelde 0.3 oranında eklenmiştir. Modelin öğrenme yollarında rastlantısallık oluşturarak ezber yapmayı zorlaştırır ve daha sağlam temsiller öğrenilmesini sağlar. Overfitting ciddi şekilde azalmış, validation metrikleri daha dengeli çıkmıştır. Özellikle animation dışındaki sınıflarda AUC skorlarının yükselmesine yardımcı olmuştur.

f) 7. Transform Etkisi: Eğitim verilerine uygulanan dönüşümler, modelin genelleme kabiliyetini artırmak ve overfitting'i azaltmak amacıyla seçilmiştir. `RandomResizedCrop`, görüntüyü farklı oranlarda kırparak modelin görselin çeşitli bölgelerinden öğrenmesini sağlar. `RandomHorizontalFlip`, objeleri sağ–sol yönünden göstererek yönsel önyargıyı azaltır. `RandomRotation`, ±30 dereceye kadar döndürmeler yaparak görselin sabit açılı olmasına karşı duyarlılığı düşürür. `ColorJitter`, parlaklık, kontrast, doygunluk ve ton değerlerini bozarak aydınlatma çeşitliliğini taklit eder. `RandomPerspective`, perspektif bozulmaları simüle ederek farklı derinliklere karşı

tolerans kazandırır. `RandomAffine`, dönme ve kaydırma gibi işlemlerle konum ezberini kırar. `ToTensor`, verileri tensöre çevirerek dönüşümlerin uygulanabilirliğini sağlar. Son olarak `Normalize`, standartlaştırma işlemiyle modelin daha kararlı ve dengeli öğrenmesine yardımcı olur.

g) Mixup'ın Overfitting'e Etkisi: Mixup, overfitting'i azaltmada doğrudan katkı sağlayan güçlü bir veri artırma yöntemidir. Karışık veriler kullanılarak modelin ezber yapması zorlaştırılır; çünkü aynı girişe karşılık gelen etiketler artık daha yumuşak ve kararsızdır. Bu durum, sınıflar arası geçişlerin keskin değil, daha dengeli olmasına neden olur. Böylece model, her sınıfı kesin sınırlarla öğrenmek yerine, daha dikkatli ve genelleyici bir öğrenme davranışı sergiler. Ayrıca Mixup sayesinde mevcut verilerden sınırsız sayıda yeni örnek üretilebilir; bu da veri artırımı etkisi yaratarak modelin daha fazla çeşitliliğe maruz kalmasını sağlar. Sonuç olarak, modelin karar sınırları daha düzgün hale gelir ve sınıflar arasında aşırı ayrım yapan sert geçişler engellenmiş olur.

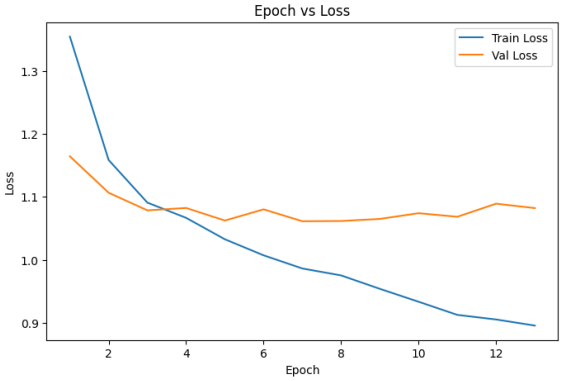

Fig. 3. İlk Eğitilen Modelin Epoch vs Loss Grafiği

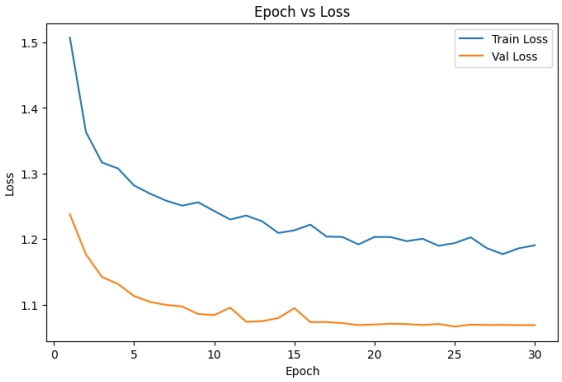

Fig. 4. İkinci Eğitilen Modelin Epoch vs Loss Grafiği

Şekil 3 ve Şekil 4'de, iki farklı eğitim stratejisiyle eğitilmiş modellerin epoch bazlı eğitim (*Train Loss*) ve doğrulama (*Validation Loss*) kayıpları gösterilmektedir. İlk modele ait loss grafiği incelendiğinde, eğitim kaybının hızlı bir şekilde düştüğü, ancak doğrulama kaybının erken epoch'larda plato yaptığı ve ardından dalgalanarak sabit kaldığı görülmektedir. Bu durum, modelin eğitim verisini erken dönemde ezberlemeye başladığını, ancak doğrulama verisi üzerinde genelleme yapma kapasitesinin sınırlı kaldığını göstermektedir. Train ve validation loss eğrileri arasındaki fark zamanla artmakta, bu da **overfitting** eğilimini açıkça ortaya koymaktadır ve döngü erken bitirilmek zorundadır.

Buna karşın Şekil 4'de yer alan ikinci modelde eğitim daha uzun sürmüş ancak çok daha dengeli bir öğrenme eğrisi elde edilmiştir. Eğitim ve doğrulama kayıpları birlikte düşmüş, aralarındaki fark sabit kalmış ve validation loss uzun süre stabil seyretmiştir. Bu, modelin daha fazla genelleme yaptığına ve ezberden uzak durduğuna işaret etmektedir. Özellikle mixup, dropout, cosine annealing ve küçük batch boyutu gibi düzenleyici stratejilerin etkisiyle, ikinci modelde **genelleme gücü** ve **stabilite** ciddi oranda artmıştır.

Bu grafikler, farklı hiperparametre ayarlarının model davranışı üzerindeki etkisini açıkça ortaya koymaktadır. İlk model hızlı öğrenme eğilimindeyken, ikinci model daha sağlam ve sürdürülebilir bir şekilde eğitilmiş; bu da metriklere doğrudan yansımıştır.

TABLE V
İKI MODELIN TEST BAŞARI METRIKLERININ KARŞILAŞTIRILMASI

Metrik	**Model-1**	**Model-2**
Accuracy	0.6133	0.6164
Precision (Macro)	0.6080	0.6186
Recall (Macro)	0.6140	0.6172
F1-Score (Macro)	0.6061	0.6154
AUC (Micro)	0.8722	0.8721
AUC (Macro)	0.8642	0.8640
Sensitivity (Macro)	0.6140	0.6172
Specificity (Macro)	0.9033	0.9041
Train Time (s)	1345.1	4655.2
Inference Time (s)	6.5	6.6

Tablo V'de iki farklı eğitim stratejisi ile eğitilen modellerin test seti üzerindeki performans metrikleri karşılaştırmalı olarak sunulmuştur. İkinci model, neredeyse tüm makro ölçümlerde (Precision, Recall, F1-Score, Specificity) daha yüksek sonuçlar vermiştir. Bu durum, modelin sınıflar arasında daha dengeli ve genelleyici kararlar verdiğini göstermektedir. Özellikle F1-Skorundaki artış, doğruluk oranının ötesinde bir başarıya işaret etmektedir.

Ayrıca Specificity (Makro) değerindeki yükseliş, modelin yanlış pozitif oranlarını azalttığını ve daha güvenilir tahminler üretebildiğini göstermektedir. AUC-micro ve AUC-macro gibi ayrıştırma gücünü ölçen metriklerde her iki model benzer sonuçlar üretmiş olsa da, ikinci modelin daha tutarlı tahmin davranışı sergilediği anlaşılmaktadır.

Buna karşın eğitim süresi bakımından ikinci model, yaklaşık üç kat daha uzun bir süreye ihtiyaç duymuştur. Bu fark, kullanılan Mixup, DropPath ve öğrenme oranı düzenleyicilerinin eğitim sürecine kattığı ek karmaşıklıktan kaynaklanmaktadır. Ancak bu süre artışı, elde edilen genelleme başarısı göz önüne alındığında kabul edilebilir düzeydedir.

Sonuç olarak, hiperparametre ayarlarının ve düzenleyici tekniklerin doğru şekilde kullanılmasıyla ikinci modelin overfitting'e karşı daha dayanıklı hale geldiği ve daha yüksek bir genel başarı elde ettiği açıkça görülmektedir.

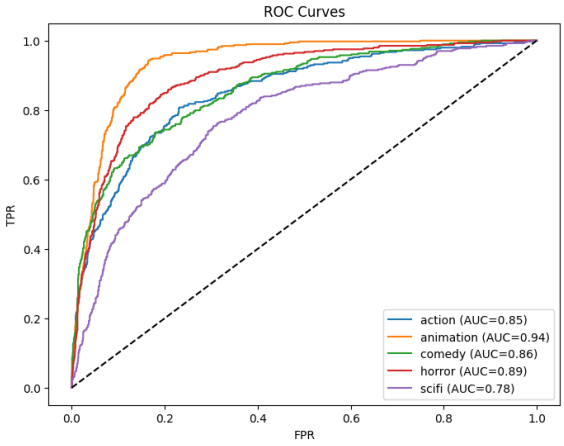

Fig. 5. İlk Eğitilen Modelin Sınıf Bazlı ROC Eğrileri

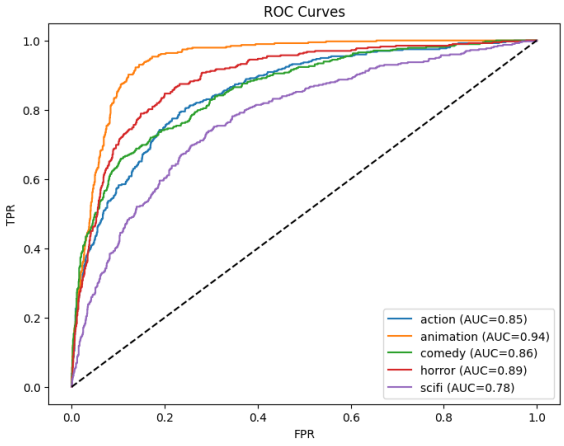

Fig. 6. İkinci Eğitilen Modelin Sınıf Bazlı ROC Eğrileri

Şekil 5 ve Şekil 6'de, iki farklı modelin test verisi üzerindeki sınıf bazlı ROC (Receiver Operating Characteristic) eğrileri sunulmuştur. AUC (Area Under Curve) değerleri ile birlikte çizilen bu eğriler, modelin her bir sınıfı ne derece ayırt edebildiğini göstermektedir.

Her iki modelde de en yüksek ayrıştırma gücü *animation* sınıfında gözlenmiştir (AUC=0.94). *Horror* ve *comedy* sınıflarında da yüksek AUC değerleri elde edilmiştir (0.86–0.89 arası). Ancak *sci-fi* sınıfında AUC değeri 0.78'de kalmış ve bu sınıfın diğerlerine göre daha fazla karıştığı görülmüştür.

İkinci modele ait ROC eğrileri (Şekil 6), AUC değerleri açısından ilk modele benzer olsa da, eğrilerin daha düzgün ve yumuşak seyretmesi, modelin daha genelleyici kararlar verdiğini göstermektedir. Bu yapı, özellikle Mixup ve CosineAnnealing gibi düzenleyici tekniklerin katkısıyla elde edilmiştir. Sınıf ayrımının sadece AUC değerleriyle değil, eğrinin şekliyle de değerlendirildiği bu karşılaştırma, ikinci modelin daha istikrarlı ve güvenilir tahminler sunduğunu desteklemektedir.

Şekil 7 ve Şekil 8'de, her iki modelin sınıf bazlı tahmin performanslarını gösteren confusion matrix görselleri sunulmuştur. Bu matrisler, modellerin hangi sınıfları ne ölçüde doğru tahmin ettiğini ve sınıflar arası karışıklık düzeylerini açıkça ortaya koymaktadır.

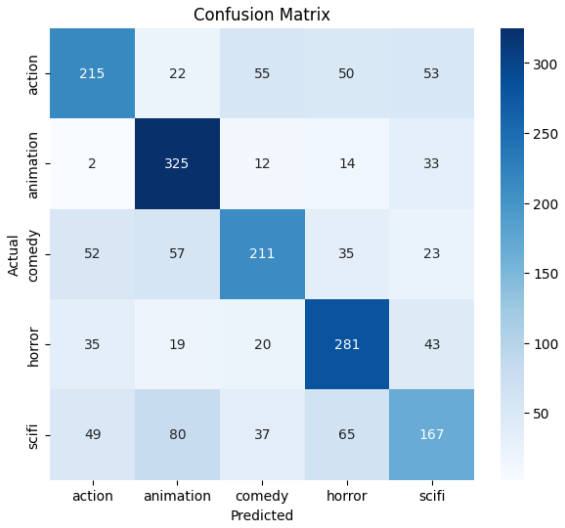

Fig. 7. İlk Eğitilen Modelin Confusion Matrix Görselleştirmesi

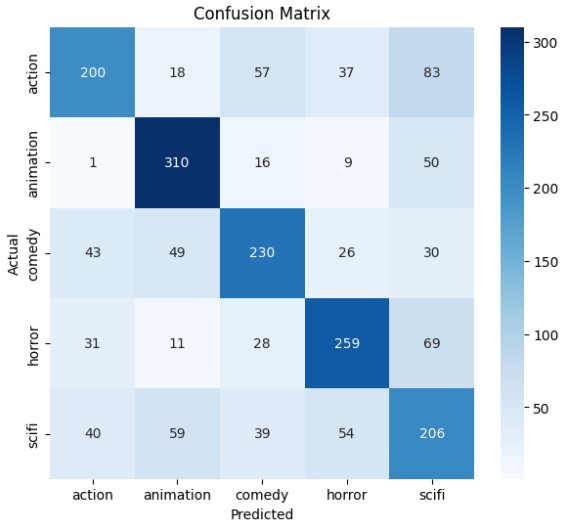

Fig. 8. İkinci Eğitilen Modelin Confusion Matrix Görselleştirmesi

İlk modelde özellikle *sci-fi* sınıfı oldukça zayıf bir şekilde temsil edilmiştir. Bu sınıfa ait çok sayıda örnek yanlış olarak *animation*, *horror* ve *action* sınıflarına atanmıştır. Benzer şekilde *comedy* sınıfı da ciddi şekilde *action* ve *animation* sınıfları ile karışmaktadır. Bu durum, ilk modelin veri dağılımını ezberlediğini ancak sınıf sınırlarını yeterince öğrenemediğini göstermektedir.

Buna karşın ikinci modele ait confusion matrix (Şekil 8) daha dengeli bir dağılım sunmaktadır. Özellikle *sci-fi* sınıfında doğru sınıflandırma oranı artmış, yanlış tahmin edilen örneklerin sayısı azalmıştır. *Comedy* ve *horror* sınıflarında da doğru tahmin oranları yükselmiş, hatalar daha sınırlı sınıflarla

kısıtlı kalmıştır. Bu gelişmeler, modelin genelleme gücünün arttığını ve daha kararlı kararlar verdiğini göstermektedir. Mixup, dropout ve öğrenme oranı stratejilerinin katkısıyla ikinci model, sınıf bazlı başarıyı belirgin şekilde yükseltmiştir.

B. Beit

BEiT (Bidirectional Encoder representation from Image Transformers), doğal dil işlemeye ilham veren BERT mimarisinin görsel karşılığı olarak tasarlanmış bir görüntü sınıflandırma modelidir. Transformer mimarisine dayanan BEiT, görüntüleri küçük parçalara (patch) bölerek bu parçaları birer "kelime" gibi işler ve görseller üzerinde masked image modeling (gizli yama tahmini) gibi özdenetimli (self-supervised) öğrenme yöntemleriyle ön eğitim alır. Bu sayede sınırlı etiketli verilerle bile güçlü temsil öğrenimi sağlayabilir. Görüntü içeriğini derinlemesine anlayabilen bu yapı, özellikle yüksek doğruluk gerektiren görüntü sınıflandırma, nesne tanıma ve segmentasyon gibi görevlerde oldukça başarılıdır. BEiT, dilsel BERT modelinin başarısını görsel dünyaya taşıyan önemli bir adımdır.[3]

1) Kodların Açıklanması:

a) 1. Konfigürasyon Ayarları: Bu bölümde model eğitimi için temel hiperparametreler tanımlanır. Eğitim süresi (`EPOCHS`), batch boyutu (`BATCH_SIZE`), resim boyutu (`IMG_SIZE`), sınıf sayısı (`NUM_CLASSES`), veri yolu (`DATA_PATH`) ve cihaz seçimi yapılır. Aynı zamanda rastgelelikten kaynaklanan sonuç farklılıklarını azaltmak için sabit tohum (`SEED`) değeri belirlenir. CUDA varsa GPU kullanılır, yoksa CPU tercih edilir.

b) 2. Veri Dönüşümleri (Transformations): Görüntülerin modele uygun hale getirilmesi için çeşitli dönüşümler tanımlanmıştır. Eğitim verisinde, modelin genelleme yeteneğini artırmak amacıyla veri arttırma (augmentation) teknikleri (örneğin: çevirme, döndürme, renk değişikliği) kullanılmıştır. Doğrulama ve test verilerinde ise sadece yeniden boyutlandırma ve normalize işlemleri uygulanır.

c) 3. Veri Setleri ve Veri Yükleyiciler: Veri setleri `ImageFolder` sınıfı ile oluşturulur ve üçe ayrılır: eğitim, doğrulama ve test. Her veri setine uygun dönüşüm fonksiyonu uygulanır. `DataLoader` kullanılarak bu veriler eğitim sırasında küçük partiler (batch) halinde GPU'ya aktarılır. Eğitim veri yükleyicisi karıştırılır (`shuffle=True`), diğerleri sıralı gelir.

d) 4. BEiT Modeli, Kayıp Fonksiyonu ve Optimizasyon: BEiT modeli `timm.create_model()` fonksiyonu ile çağrılır ve önceden eğitilmiş ağırlıklarla yüklenir. Son katmanı sınıf sayısına göre ayarlanır. Kayıp fonksiyonu olarak çok sınıflı sınıflandırmalarda yaygın olan `CrossEntropyLoss` kullanılır. Optimizasyon için `AdamW` seçilmiştir ve öğrenme oranı zamanla azaltılmak üzere `CosineAnnealingLR` scheduler tanımlanmıştır.

e) 5. Eğitim Döngüsü: Her epoch içinde model hem eğitim hem de doğrulama fazlarından geçer. Eğitim sırasında `mixup` tekniği uygulanarak veriler karıştırılır, bu da modelin aşırı uyuma (overfitting) karşı daha dayanıklı olmasını sağlar.

f) 6. Test Aşaması ve Modelin Yüklenmesi: model dosyası `final_beit_model.pth` olarak kaydedilir ve test aşamasında tekrar yüklenir. Test verisi üzerinde tahminler yapılır, olasılıklar ve tahmin edilen sınıflar kaydedilir. Bu bilgiler daha sonra metriklerin hesaplanmasında kullanılır.

g) 7. Değerlendirme Metrikleri: Modelin performansı doğruluk (Accuracy), hassasiyet (Precision), duyarlılık (Recall), F1 skoru, AUC (micro ve macro), duyarlılık (Sensitivity) ve özgüllük (Specificity) gibi çok sayıda metrikle ölçülür. Karmaşıklık matrisi üzerinden sınıflar arası başarı da analiz edilir.

h) 8. Görselleştirme: Eğitim ve doğrulama kayıplarının epoch bazında değişimi çizilir. Ayrıca ROC eğrileri ve karmaşıklık matrisi görselleştirilerek modelin sınıflar üzerinde nasıl performans gösterdiği daha iyi analiz edilir. ROC eğrisi altında kalan alan (AUC) sınıf bazlı başarıyı temsil eder.

i) 9. Modelin Kaydedilmesi: Eğitim sonrası modelin son hali `beit_model.pth` dosyasına kaydedilir. Böylece daha sonra yeniden eğitmeye gerek kalmadan model doğrudan yüklenip kullanılabilir.

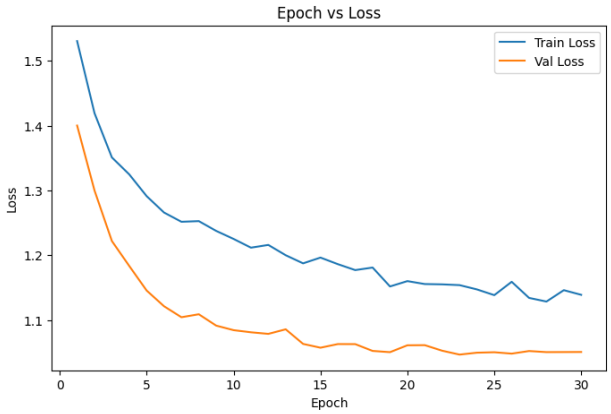

Fig. 9. Eğitim ve Doğrulama Kayıplarının Epoch Bazlı Değişimi

2) Sonuçların Yorumları: Grafik Yorumu Şekil 9'te görülen grafik, eğitim (Train Loss) ve doğrulama (Validation Loss) kayıplarının epoch'lar boyunca nasıl değiştiğini göstermektedir. Başlangıçta her iki kayıp değeri de oldukça yüksektir ancak model eğitildikçe her iki eğri de belirgin şekilde azalmaktadır. Bu, modelin hem eğitim verisi üzerinde öğrenme sağladığını hem de doğrulama verisinde genelleme yeteneğini artırdığını gösterir.

Eğitim kaybı sürekli olarak azalsa da doğrulama kaybı daha erken bir noktada plato yaparak belirli bir değere sabitlenmiştir. Bu durum, modelin öğrenme sürecinde aşırı uyuma (overfitting) girmeden dengeleme sağladığını işaret eder. Eğriler arasında büyük bir sapma olmaması, eğitim ve doğrulama süreçlerinin tutarlı ilerlediğini ve modelin dengeli bir şekilde genelleştiğini göstermektedir. Bu bağlamda, eğitim süreci başarılıdır ve erken durdurma uygulanmasa bile model aşırı karmaşıklıktan kaçınmıştır.

a) Test Metrikleri Yorumu: Tablo VI'te BEiT modelinin nihai test performansına dair metrikler sunulmuştur. Modelin

TABLE VI
BEiT Modeli Nihai Test Performans Metrikleri

Metrik	Değer
Doğruluk (Accuracy)	0.6174
Kesinlik (Precision - Macro)	0.6154
Duyarlılık (Recall - Macro)	0.6181
F1-Skor (Macro)	0.6144
AUC (Micro)	0.8748
AUC (Macro)	0.8667
Hassasiyet (Sensitivity)	0.6181
Özgüllük (Specificity)	0.9043
Eğitim Süresi (saniye)	3794.9
Çıkarım Süresi (saniye)	7.4

genel doğruluk değeri ≈ 0.6174 olup, sınıflar arası dengeyi temsil eden makro ortalama kesinlik, duyarlılık ve F1-skor değerleri de yaklaşık olarak 0.61 civarındadır. Bu, modelin sınıflar arasında adil ve dengeli bir performans gösterdiğini ortaya koyar.

AUC (Alan Altı Eğri) değerleri ise oldukça yüksektir: micro AUC ≈ 0.8748 ve macro AUC ≈ 0.8667. Bu, modelin sınıflandırma kararlarında yüksek ayırt ediciliğe sahip olduğunu gösterir. Özellikle özgüllük (specificity) değerinin 0.90 gibi yüksek bir seviyede olması, modelin yanlış pozitifleri ayırt etme konusunda başarılı olduğunu ortaya koymaktadır.

Sonuç olarak model, sınıflar arasında tutarlı bir performans sergilemekte olup, özellikle ROC tabanlı metriklerde güçlü bir genel başarı göstermektedir. Eğitim süresi yaklaşık 63 dakika, çıkarım süresi ise oldukça hızlıdır (7.4 saniye), bu da modelin hem eğitilebilirliğinin hem de uygulanabilirliğinin güçlü olduğunu kanıtlar.

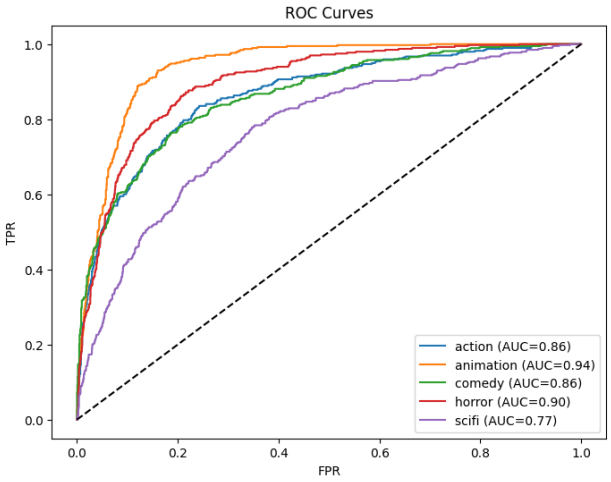

Fig. 10. Sınıf Bazlı ROC Eğrileri ve AUC Değerleri

b) ROC Eğrisi Yorumu: Şekil 10'te her sınıf için ROC (Receiver Operating Characteristic) eğrileri ve karşılık gelen AUC (Area Under Curve) değerleri yer almaktadır. ROC eğrisi, modelin pozitif sınıfları ne kadar doğru ayırt edebildiğini gösterirken, AUC değeri bu ayrım performansının sayısal bir özetini sunar.

En yüksek AUC değerine sahip sınıf **animation** olup, $AUC = 0.94$ ile modelin bu sınıfı ayırt etmede oldukça başarılı olduğunu göstermektedir. Bunu takiben **horror** ($AUC = 0.90$) ve **action/comedy** sınıfları ($AUC = 0.86$) güçlü performans sergilemektedir. Ancak **scifi** sınıfı için AUC değeri yalnızca 0.77 olup, modelin bu sınıfı ayırt etmekte diğerlerine göre daha zorlandığını göstermektedir.

Genel olarak eğriler, pozitif örnekleri yüksek olasılıkla sınıflandırabilen bir modele işaret etmektedir. ROC eğrisinin köşeye yakınlığı (0,1) ve diyagonalden uzaklığı ne kadar fazlaysa, modelin performansı o kadar iyidir. Bu bağlamda animation ve horror sınıflarının eğrileri belirgin şekilde ideal bölgeye yakındır. Sci-fi eğrisinin ise daha çok diyagonale yakın olması, bu sınıf için modelin daha fazla iyileştirilmesi gerektiğini göstermektedir.

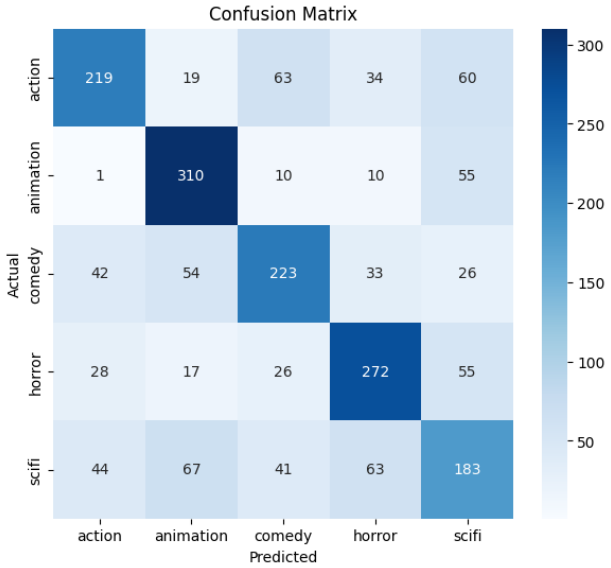

Fig. 11. BEiT Modeli İçin Karmaşıklık Matrisi

c) Karmaşıklık Matrisi Yorumu: Şekil 17'te BEiT modelinin sınıflandırma performansını gösteren karmaşıklık matrisi yer almaktadır. Her satır, gerçek sınıfı; her sütun ise modelin tahmin ettiği sınıfı temsil eder. Diyagonal üzerindeki değerler doğru sınıflamaları; diyagonal dışındakiler ise hata yapılan sınıflamaları gösterir.

Animation sınıfı, 310 doğru sınıflama ile en yüksek isabet oranına sahiptir. Bu, modelin bu sınıfı oldukça başarılı bir şekilde tanıdığını göstermektedir. **Horror** sınıfı da 272 doğru tahmin ile güçlü bir performans sergilemektedir. Buna karşın **Sci-fi** sınıfı için doğru tahmin sayısı 183 olup, yanlış tahmin sayıları (özellikle animation ve horror ile karışma eğilimi) oldukça yüksektir. Bu durum, sci-fi sınıfının diğer sınıflarla görsel benzerlik taşıdığını ya da modelin bu sınıfa özgü ayırt edici özellikleri öğrenmede zorlandığını gösterir.

Action ve **Comedy** sınıfları orta düzeyde performans göstermiştir. Action sınıfı sıklıkla comedy (63 hata) ve sci-fi (60 hata) ile karıştırılmıştır. Comedy ise animation ve action sınıflarına sıkça yanlış sınıflanmıştır. Bu karışıklıklar, veri setindeki sınıflar arasında bazı görsel örtüşmeler olduğunu işaret etmektedir.

Genel olarak, modelin bazı sınıflarda yüksek doğrulukla çalıştığı, bazı sınıflarda ise karışıklık yaşadığı görülmektedir. Bu analiz, sınıf bazlı iyileştirme stratejileri geliştirmek için önemli bir yol göstericidir.

C. Vit

Vision Transformer (ViT), görüntü sınıflandırma gibi bilgisayarla görme görevlerinde devrim yaratan bir derin öğrenme modelidir. Geleneksel CNN mimarileri yerine, ViT saf dikkat (self-attention) mekanizması kullanan bir transformer yapısı ile çalışır. Görüntüleri doğrudan pikseller üzerinden işlemez; bunun yerine, resmi sabit boyutlu parçalara (patch'lere) böler, bu parçaları düzleştirip pozisyon bilgisi ile birlikte giriş olarak verir. Bu yaklaşım sayesinde ViT, daha az inductive bias barındıran ve büyük veri setlerinde çok yüksek doğruluk elde edebilen esnek ve güçlü bir model olarak öne çıkar. Büyük veri ve yüksek hesaplama gücüyle eğitildiğinde CNN'leri geride bırakabilir.[4]

1) Kodların Açıklanması:

a) Yapılandırma ve Sabitler: Bu bölümde modelin eğitim süreci için gerekli olan temel sabitler tanımlanmıştır. Eğitim döngüsü sayısı (`EPOCHS`), batch boyutu (`BATCH_SIZE`), görüntü boyutu (`IMG_SIZE`), sınıf sayısı (`NUM_CLASSES`), veri yolu (`DATA_PATH`), cihaz türü (`DEVICE`), öğrenme oranı ve ağırlık çürümesi (`LR`, `WEIGHT_DECAY`) gibi parametreler belirlenmiştir. Ayrıca deneyin tekrarlanabilir olması için sabit tohum (seed) değerleri atanmıştır.

b) Görüntü Dönüştürme (Transformations): Modelin daha genel öğrenmesini sağlamak amacıyla eğitim verileri üzerinde çeşitli veri artırma (data augmentation) teknikleri uygulanmıştır. Bu teknikler arasında rastgele kırpma, yatay çevirme, döndürme, renk bozulmaları, perspektif dönüşümleri ve affine dönüşümler yer almaktadır. Doğrulama ve test verilerinde ise yalnızca yeniden boyutlandırma ve normalize etme işlemleri gerçekleştirilmiştir.

c) Veri Seti ve Veri Yükleyiciler: Eğitim, doğrulama ve test veri setleri `ImageFolder` sınıfı ile tanımlanmış ve uygun dönüşümler atanmıştır. `DataLoader` nesneleri ile veriler batch'ler halinde yüklenmiş; eğitim verisinde karıştırma (`shuffle=True`) aktif edilmiştir. `pin_memory=True` ve `num_workers=2` ayarları, GPU'da daha hızlı veri erişimi için yapılandırılmıştır.

d) Model, Kayıp Fonksiyonu, Optimizasyon ve Zamanlayıcı: `timm` kütüphanesi kullanılarak `vit_base_patch16_224` modeli önceden eğitilmiş ağırlıklarla yüklenmiş ve çıktı katmanı sınıf sayısına göre yapılandırılmıştır. Kayıp fonksiyonu olarak `CrossEntropyLoss`, optimizasyon yöntemi olarak ise `AdamW` seçilmiştir. Öğrenme oranının eğitim sürecinde azalarak değişmesini sağlamak için `CosineAnnealingLR` zamanlayıcısı kullanılmıştır.

e) Mixup ve Eğitim Döngüsü: Veri çeşitliliğini artırmak ve modeli daha dayanıklı hale getirmek için `mixup` tekniği uygulanmıştır. Bu teknik ile her batch içerisindeki görüntüler rastgele karıştırılır ve etiketler de buna uygun şekilde harmanlanır. Eğitim döngüsünde model önce `train()` modunda eğitilir, ardından doğrulama verisiyle değerlendirme yapılır. Eğitim ve doğrulama kayıpları (loss) her epoch sonunda kayıt altına alınır ve zamanlayıcı güncellenir.

f) Test ve Tahmin: Eğitilen modelin en iyi durumu diske kaydedilmiş ve daha sonra bu model test verisi üzerinde çalıştırılmıştır. Modelin çıktıları softmax aktivasyonundan geçirilerek sınıf olasılıkları elde edilmiştir. En yüksek olasılığa sahip sınıf tahmin olarak kabul edilmiştir. Gerçek ve tahmin edilen etiketler ile birlikte olasılıklar da kayıt altına alınmıştır.

g) Değerlendirme Metrikleri: Modelin performansını ölçmek için birçok metrik hesaplanmıştır: doğruluk (Accuracy), makro ortalamalı kesinlik (Precision), duyarlılık (Recall), F1-skoru, mikro ve makro AUC değerleri, ayrıca sınıf başına özgüllük (Specificity). Bu metrikler, modelin sadece genel başarısını değil, aynı zamanda sınıflar arası dengesini de değerlendirmeye olanak sağlar.

h) Görselleştirme: Modelin sınıflandırma sonuçlarını daha iyi analiz edebilmek için çeşitli grafikler çizilmiştir. Karışıklık matrisi, modelin hangi sınıfları karıştırdığını gösterirken; ROC eğrileri, her sınıf için ayırt ediciliği görselleştirir. Ayrıca eğitim süresince oluşan eğitim ve doğrulama kayıpları da epoch bazında çizilerek modelin öğrenme eğrisi takip edilmiştir.

i) Modelin Kaydedilmesi: Eğitim tamamlandıktan sonra modelin son hali bir `.pth` dosyası olarak kaydedilmiştir. Bu sayede model daha sonra test veya gerçek uygulamalar için kolayca yüklenip kullanılabilir duruma getirilmiştir.

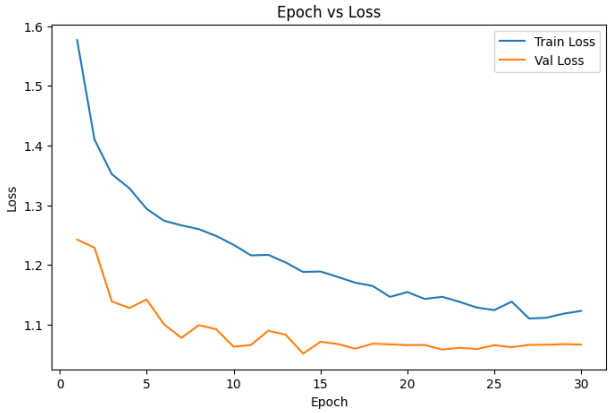

Fig. 12. Epoch vs Loss: Eğitim ve Doğrulama Kayıpları

2) Sonuçların Yorumları:

a) Grafik Yorumu: Şekil 12'de eğitim (mavi) ve doğrulama (turuncu) kayıplarının epoch bazında değişimi gösterilmektedir. Eğitim kaybı genel olarak azalan bir eğilim sergilese de zaman zaman küçük dalgalanmalarla ilerlemiştir. Bu dalgalanmalar, modelin bazı epoch'larda veriye tam olarak uyum sağlayamadığını veya öğrenme oranı ile ilgili geçici dengesizlikler yaşandığını gösterebilir. Doğrulama kaybı ise ilk birkaç epoch'tan sonra hızlıca düşerek daha stabil bir seviyeye ulaşmış ve eğitim süresince bu seviyeyi korumuştur.

Doğrulama kaybının düşük ve dalgalanmanın az olması, modelin genelleme kabiliyetinin yüksek olduğunu, overfitting riskinin ise düşük kaldığını göstermektedir. Eğitim ve doğrulama kayıpları arasındaki farkın çok açılmaması da öğrenmenin dengeli bir şekilde gerçekleştiğini desteklemektedir.

TABLE VII
VIT MODELI NIHAI TEST PERFORMANS METRIKLERI

Metrik	**Değer**
Doğruluk (Accuracy)	0.6215
Kesinlik (Precision - Macro)	0.6169
Duyarlılık (Recall - Macro)	0.6222
F1-Skor (Macro)	0.6173
AUC (Micro)	0.8765
AUC (Macro)	0.8683
Hassasiyet (Sensitivity)	0.6222
Özgüllük (Specificity)	0.9054
Eğitim Süresi (saniye)	3392.3
Çıkarım Süresi (saniye)	6.5

b) Tablo Yorumu: Tablo VII'de Vision Transformer (ViT) modelinin test veri seti üzerindeki performans metrikleri gösterilmektedir. Accuracy (%62.15) ve macro ortalamalı F1-score (%61.73) değerleri, modelin genel doğruluğunun ve sınıflar arası dengesinin makul seviyede olduğunu göstermektedir. Precision ve Recall değerleri birbirine oldukça yakın olup, modelin hem doğru pozitifleri bulma hem de yanlış pozitifleri ayırt etme konusunda dengeli davrandığını işaret etmektedir.

AUC değerlerinin yüksekliği (micro: 0.8765, macro: 0.8683), modelin sınıflar arasında ayırt ediciliğinin güçlü olduğunu ortaya koyar. Özellikle micro-AUC değeri, genel sınıflandırma kabiliyetinin yüksekliğini desteklemektedir. Specificity değeri 0.9054 olup, modelin yanlış pozitifleri doğru şekilde eleyebildiğini gösterir ki bu da pratik uygulamalarda oldukça önemlidir.

Eğitim süresi yaklaşık 56.5 dakika olarak ölçülmüştür; bu, ViT gibi büyük bir model için oldukça normaldir. Inference süresi ise sadece 6.5 saniye olup, test verisi üzerinde hızlı sonuç verdiğini göstermektedir. Tüm bu metrikler birlikte değerlendirildiğinde, modelin dengeli, güçlü ve uygulanabilir bir sınıflandırıcı olduğu sonucuna varılabilir.

c) Grafik Yorumu: Şekil 13'de, her sınıf için ayrı ayrı çizilmiş ROC eğrileri ve AUC (Area Under Curve) değerleri yer almaktadır. ROC eğrileri, modelin pozitif sınıfları ne kadar başarılı ayırt ettiğini göstermektedir. Eğrilerin sol üst köşeye yakın olması, yüksek duyarlılık ve düşük yanlış pozitif oranını ifade eder.

Bu grafikte en yüksek AUC değerine sahip sınıf **animation** (`AUC=0.94`) olup, model bu sınıfı çok başarılı şekilde ayırt edebilmektedir. **Horror** (`AUC=0.90`) sınıfı da oldukça yüksek bir ayırt ediciliğe sahiptir. **Action** ve **comedy** sınıfları için AUC değerleri `0.86` olup, model bu iki sınıfı dengeli biçimde tanıyabilmektedir. Ancak **scifi** sınıfı için AUC değeri `0.78` ile diğer sınıflara göre daha düşüktür. Bu da modelin bilim kurgu sınıfındaki örnekleri ayırt etme konusunda daha fazla zorlandığını göstermektedir.

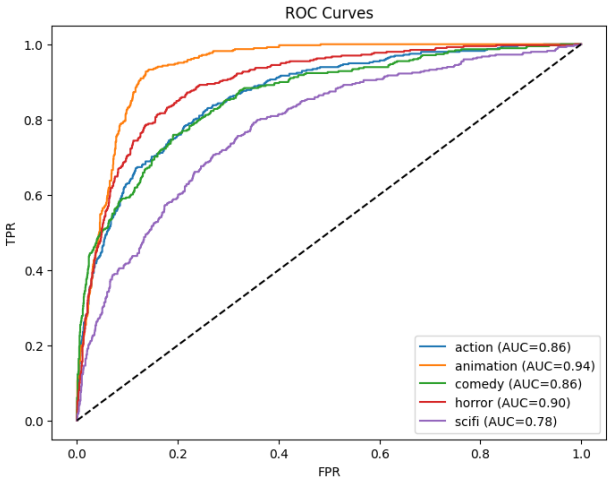

Fig. 13. ROC Eğrileri - Sınıf Bazlı AUC Değerleri

Genel olarak ROC eğrilerinin dağılımı, modelin çoğu sınıf için yüksek ayırt etme kapasitesine sahip olduğunu; ancak bazı sınıflarda (özellikle scifi) performansın artırılması gerektiğini işaret etmektedir. Bu tür farklılıklar, sınıflar arası örnek dağılımı, veri kalitesi veya benzerlik gibi faktörlerden kaynaklanıyor olabilir.

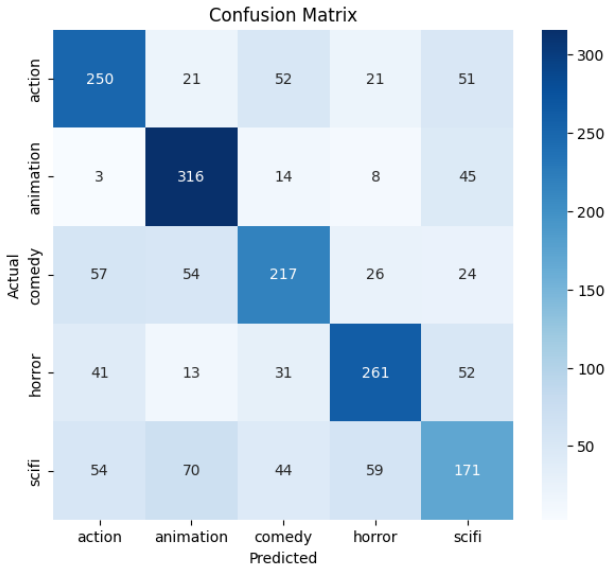

Fig. 14. Karışıklık Matrisi (Confusion Matrix) – BEiT Modeli

d) Grafik Yorumu: Şekil 14'de BEiT modelinin test veri seti üzerindeki sınıf tahmin performansı karışıklık matrisi üzerinden gösterilmiştir. Diyagonal üzerindeki yüksek değerler doğru sınıflandırmaları ifade ederken, diyagonal dışındaki değerler sınıflar arası karışıklıkları göstermektedir.

Animation sınıfı 316 doğru tahminle modelin en başarılı olduğu sınıf olmuştur. **Horror** (261) ve **Action** (250) sınıflarında da yüksek başarı elde edilmiştir. Ancak **Comedy** sınıfı sıklıkla **Action** (57 kez) ve **Animation** (54 kez)

ile karıştırılmıştır. Bu durum, bu türlerin görsel benzerlik göstermesinden kaynaklanıyor olabilir.

Scifi sınıfı ise modelin en çok zorlandığı sınıflardan biridir. Scifi örnekleri çoğunlukla **Animation** (70 kez), **Horror** (59 kez) ve **Action** (54 kez) gibi diğer türlerle karıştırılmıştır. Bu durum, önceki ROC analizindeki düşük AUC (0.78) değeriyle de örtüşmektedir.

Genel olarak, modelin belirli sınıflarda güçlü tahmin performansı gösterdiği, ancak bazı sınıflarda (özellikle comedy ve scifi) sınıflar arası ayrımda zorluk yaşadığı görülmektedir. Bu durum, bu sınıflar için veri artırımı ya da daha ayrıştırıcı özellik mühendisliği yapılmasını gerektirebilir.

D. DeiT

DeiT (Data-efficient Image Transformer), Vision Transformer (ViT) mimarisine dayanan ve sınırlı veriyle yüksek doğruluk elde etmeyi amaçlayan bir görüntü sınıflandırma modelidir. Facebook AI tarafından geliştirilen DeiT, geleneksel ViT modellerinin büyük veri setlerine olan bağımlılığını azaltmak için bilgi verimli bir eğitim stratejisi sunar. Özellikle "distillation token" adlı özel bir öğrenme yöntemiyle, güçlü bir CNN öğretmen modelinden bilgi aktarımı yaparak veri verimliliğini artırır. Bu sayede, daha az veriyle ViT performansına ulaşmak mümkün hale gelirken, eğitim sürecinde etiketli veri ihtiyacı da azaltılmış olur. Hem hızlı hem de parametrik olarak daha verimli olan DeiT, günümüzde birçok görsel sınıflandırma görevinde tercih edilen hafif ve etkili transformer tabanlı modeller arasında yer alır.[5]

1) Kodların Açıklanması:

a) Yapılandırma ve Sabitler: Bu bölümde modelin çalışması için gerekli olan temel yapılandırmalar tanımlanır. Eğitim süresi (EPOCHS), batch boyutu (BATCH_SIZE), resim boyutu (IMG_SIZE), sınıf sayısı (NUM_CLASSES), veri yolu (DATA_PATH), donanım (DEVICE), rastgelelik için tohum değeri (SEED), öğrenme oranı (LR) ve ağırlık çürümesi (WEIGHT_DECAY) gibi hiperparametreler belirlenmiştir. Aynı zamanda, eğitim sürecinde rastgeleliğin etkisini azaltmak için 'torch', 'numpy' ve 'random' kütüphaneleri kullanılarak belirli bir tohum değeri ayarlanmıştır.

b) Görüntü Dönüşümleri (Transforms): Veri artırımı ve normalize işlemlerini gerçekleştiren transform yapıları tanımlanmıştır. Eğitim verileri için 'RandomResizedCrop', 'ColorJitter', 'RandomRotation' gibi dönüşümlerle çeşitlilik artırılırken, doğrulama ve test verileri yalnızca yeniden boyutlandırılıp normalize edilmiştir. Bu sayede modelin genelleme yeteneği artırılmış olurken overfitting riski azaltılmıştır.

c) Veri Yükleme (DataLoader): 'ImageFolder' sınıfı ile eğitim, doğrulama ve test veri kümeleri belirtilen klasörlerden yüklenmiş ve her biri ilgili transform ile işlenmiştir. 'DataLoader' nesneleri kullanılarak bu veri kümeleri GPU'ya hızlı aktarım için 'pin_memory' ve paralel veri yükleme için 'num_workers' parametreleriyle optimize edilmiştir. Eğitim verisi karıştırılırken doğrulama ve test setleri sabit sırayla yüklenir.

d) Model, Kayıp Fonksiyonu ve Optimizasyon: TIMM kütüphanesinden 'deit_base_patch16_224' modeli çağrılmış, sınıf sayısı modele uyarlanmış ve 'drop_rate', 'drop_path_rate' gibi regularization ayarları yapılmıştır. Kayıp fonksiyonu olarak çok sınıflı sınıflandırmaya uygun 'CrossEntropyLoss' kullanılmıştır. Optimizasyon için 'AdamW' algoritması tercih edilerek ağırlık çürümesi uygulanmıştır. Öğrenme oranı ise zamanla azalan bir 'CosineAnnealingLR' scheduler ile dinamik olarak ayarlanır.

e) Mixup Fonksiyonu: Mixup, modelin daha kararlı ve genelleyici hale gelmesini sağlamak için kullanılan bir veri artırma tekniğidir. İki farklı örnekten karışık bir görüntü oluşturur ve hedef etiketleri de aynı oranda karıştırır. Böylece model, iki örnek arasında yumuşak geçişler öğrenir. 'alpha' parametresi karıştırma yoğunluğunu belirler.

f) Eğitim ve Doğrulama Döngüsü: Her epoch boyunca model eğitime alınır, batch'ler halinde veriler GPU'ya gönderilir ve Mixup uygulanır. Model çıktıları ile hedef etiketler arasındaki kayıp hesaplanır ve geriye yayılım (backpropagation) yapılır. Ardından doğrulama setinde model değerlendirilerek val loss kaydedilir. Eğitim ve doğrulama kayıpları izlenerek modelin ilerleyişi takip edilir.

g) Model Kaydetme ve Test Değerlendirmesi: Eğitim tamamlandıktan sonra modelin ağırlıkları 'final_deit_model.pth' dosyasına kaydedilir. Daha sonra model bu ağırlıklarla tekrar yüklenir ve test verisi üzerinde tahminler yapılır. Tahmin sonuçları ile birlikte yumuşatılmış olasılıklar da elde edilir.

h) Metrik Hesaplama: Doğruluk (Accuracy), Kesinlik (Precision), Duyarlılık (Recall), F1-Skoru, ROC AUC (hem mikro hem makro), Hassasiyet (Sensitivity) ve Özgüllük (Specificity) gibi performans metrikleri hesaplanmıştır. Karışıklık matrisi üzerinden özgüllük ayrıca manuel olarak tüm sınıflar için ortalama alınarak hesaplanmıştır.

i) Görselleştirme: Modelin başarısını görsel olarak ifade etmek için üç farklı grafik çizilmiştir: karışıklık matrisi (confusion matrix), ROC eğrileri (her sınıf için AUC ile) ve eğitim/doğrulama kayıplarının epoch bazlı değişimi. Bu grafikler modelin sınıflar üzerindeki ayırt edici performansını ve öğrenme sürecindeki stabiliteyi ortaya koyar.

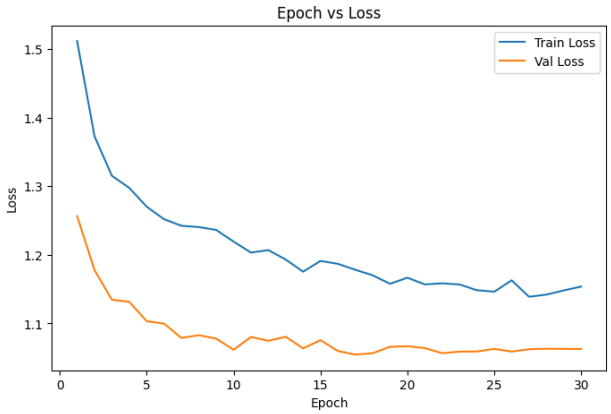

Fig. 15. Eğitim ve Doğrulama Kayıp Grafiği (Epoch vs Loss)

2) Sonuçların Yorumları:

a) Epoch-Loss Eğrisi Yorumu: Şekil 18'de gösterilen eğitim ve doğrulama kayıp eğrileri incelendiğinde, modelin eğitim süreci boyunca kararlı ve istikrarlı bir öğrenme gerçekleştirdiği görülmektedir. Başlangıçta hem eğitim hem de doğrulama kayıpları yüksek seviyelerdeyken, epoch ilerledikçe her iki kayıpta da düşüş gözlemlenmiştir. Eğitim kaybı daha değişken bir azalış gösterirken, doğrulama kaybı 10. epoch civarında stabil bir düzeye inmiş ve neredeyse sabit kalmıştır. Bu durum, modelin overfitting (aşırı öğrenme) yapmadığını, genel geçer bir öğrenme gerçekleştirdiğini ve doğrulama verisi üzerinde iyi bir genelleme sağladığını göstermektedir. Ayrıca eğriler arasındaki mesafenin küçük olması da modelin hem eğitim hem de doğrulama setinde benzer başarıyla çalıştığını ifade eder.

TABLE VIII
DEIT MODELI NIHAI TEST PERFORMANS METRIKLERI

Metrik	**Değer**
Doğruluk (Accuracy)	0.6066
Kesinlik (Precision - Macro)	0.6061
Duyarlılık (Recall - Macro)	0.6077
F1-Skor (Macro)	0.6049
AUC (Micro)	0.8728
AUC (Macro)	0.8661
Hassasiyet (Sensitivity)	0.6077
Özgüllük (Specificity)	0.9016
Eğitim Süresi (saniye)	3392.9
Çıkarım Süresi (saniye)	7.1

b) DeiT Modeli Test Performansı Yorumu: Tablo VIII'te sunulan metrikler, DeiT (Data-efficient Image Transformer) modelinin test verisi üzerindeki genel başarımını ortaya koymaktadır. Modelin doğruluk değeri %60 civarındadır ve bu değer, sınıflar arasındaki dağılımın dengeli olduğu bir problemde kabul edilebilir bir başarı düzeyi göstermektedir. Makro ortalama ile hesaplanan kesinlik, duyarlılık ve F1-skoru değerleri birbirine oldukça yakın olup, modelin tüm sınıflarda dengeli bir performans sergilediğini göstermektedir.

AUC (Area Under Curve) değerleri oldukça yüksektir: mikro AUC 0.8728 ve makro AUC 0.8661 olarak ölçülmüştür. Bu, modelin sınıflandırma konusundaki ayırt ediciliğinin oldukça iyi olduğunu ve sınıflar arasındaki ayrımı başarılı bir şekilde gerçekleştirdiğini göstermektedir. Özellikle %90'ın üzerindeki özgüllük değeri (0.9016), modelin yanlış pozitif tahminlerde düşük hata oranı ile çalıştığını işaret ederken, duyarlılıkla birlikte değerlendirildiğinde modelin hem pozitif hem negatif sınıfları ayırt etmede dengeli bir başarı sağladığı görülmektedir.

Eğitim süresi yaklaşık 56 dakika (3392.9 saniye), çıkarım süresi ise yalnızca 7.1 saniye olup, bu da modelin hem eğitim hem de test aşamasında zaman açısından verimli çalıştığını göstermektedir. Genel olarak, DeiT modeli hem sınıflar arası genelleme başarısı hem de ROC analizleri açısından güçlü bir performans ortaya koymuştur.

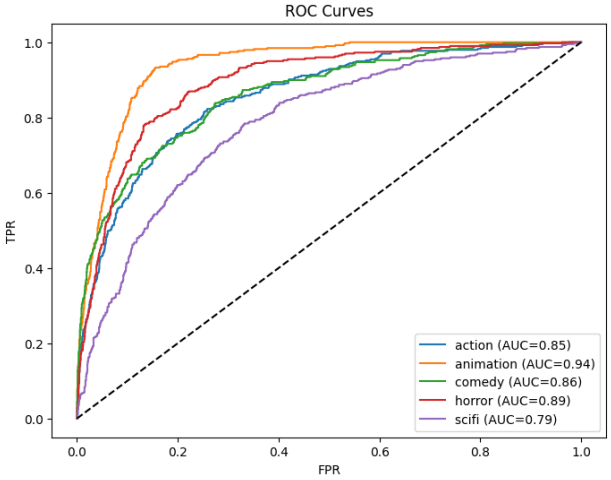

Fig. 16. Her Sınıf için ROC Eğrileri ve AUC Değerleri

c) ROC Eğrisi ve AUC Değerlendirmesi: Şekil 19'te her bir sınıf için çizilen ROC (Receiver Operating Characteristic) eğrileri ve karşılık gelen AUC (Area Under Curve) değerleri yer almaktadır. ROC eğrisi, modelin pozitif ve negatif sınıfları ne kadar iyi ayırt ettiğini gösterirken, eğrinin altında kalan alan olan AUC değeri, sınıflandırma performansının sayısal göstergesidir.

En yüksek AUC değeri 0.94 ile **animation** sınıfına aittir. Bu, modelin bu sınıfı yüksek doğrulukla ayırt ettiğini ve çok düşük yanlış pozitif oranı ile yüksek doğru pozitif oranı elde ettiğini gösterir. Ardından gelen sınıflar **horror** (AUC=0.89), **comedy** (AUC=0.86) ve **action** (AUC=0.85) ise modelin bu türleri de makul düzeyde başarılı şekilde sınıflandırabildiğini göstermektedir. Ancak **scifi** sınıfı diğerlerinden düşük bir AUC değeri olan 0.79 ile ayrılmaktadır; bu da modelin bilim kurgu sınıfını ayırt etmekte diğer sınıflara göre daha zorlandığını işaret eder.

Genel olarak, ROC eğrilerinin eğimi dik ve AUC değerlerinin 0.80'in üzerinde olması modelin güçlü bir ayırt ediciliğe sahip olduğunu, ancak bazı sınıflarda iyileştirme potansiyeli bulunduğunu göstermektedir. Özellikle 'scifi' sınıfı için veri dengesizliği, benzer görsel yapılar veya yetersiz örnek sayısı gibi nedenlerle performans düşüklüğü gözlenmiş olabilir.

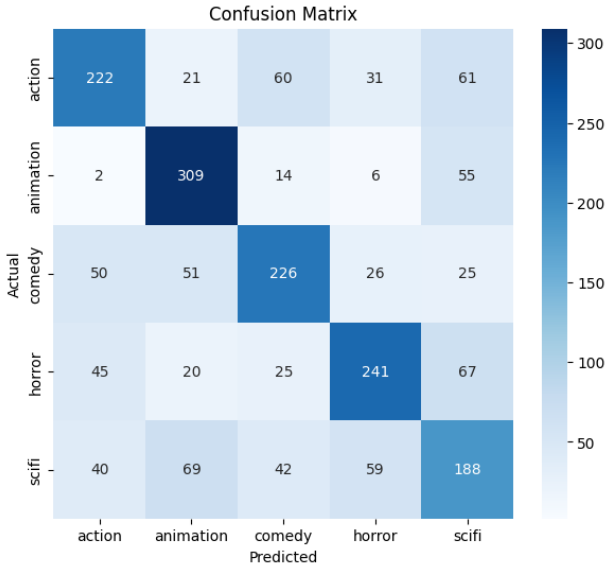

Fig. 17. DeiT Modeli İçin Karışıklık Matrisi

d) Karışıklık Matrisi Yorumu: Şekil 17'te DeiT modelinin test verisi üzerinde ürettiği karışıklık matrisi görülmektedir. Bu matris, modelin hangi sınıfları doğru tahmin ettiğini ve hangi sınıfları karıştırdığını detaylı şekilde gösterir. Diagonal üzerindeki değerler doğru sınıflamaları, diğer hücreler ise hatalı sınıflamaları temsil eder.

Modelin en yüksek doğru sınıflama yaptığı sınıf **animation** olup, bu sınıf için 309 doğru tahmin gerçekleştirilmiştir. Bu durum, modelin bu türdeki görselleri oldukça başarılı şekilde tanıdığını göstermektedir. Diğer sınıflarda ise **comedy** (226), **horror** (241), **action** (222) ve **scifi** (188) doğru sınıflandırma sayılarıyla takip etmektedir.

Ancak modelin **scifi** sınıfını sınıflandırmada zorlandığı açıktır. Bu sınıfın önemli bir kısmı (69 örnek) **animation**, 42 örnek **comedy** ve 59 örnek **horror** olarak yanlış sınıflandırılmıştır. Bu da bilim kurgu görsellerinin diğer türlerle görsel benzerlik taşıyabileceğini ve modelin bu farkı ayırt etmede sınırlı kaldığını göstermektedir. Benzer şekilde **action** sınıfının 60 örneği **comedy**, 61 örneği ise **scifi** olarak hatalı tahmin edilmiştir.

Sonuç olarak, karışıklık matrisi modelin genel olarak makul bir sınıflandırma performansı gösterdiğini ancak belirli sınıflar arasında karışıklık yaşandığını ve özellikle **scifi** sınıfında iyileştirme ihtiyacı bulunduğunu ortaya koymaktadır. Bu tür hatalar, sınıflar arası görsel örtüşme, veri dengesizliği veya model kapasitesi ile ilgili olabilir.

E. MaxViT

MaxViT (Maximally Local and Global Vision Transformer), görsel sınıflandırma görevlerinde yüksek performans sağlamak amacıyla tasarlanmış bir hibrit yapılı derin öğrenme modelidir. Bu model, hem yerel özellikleri yakalamada etkili olan konvolüsyonel (CNN) katmanları hem de küresel bağlamı modelleyen self-attention mekanizmalarını bir araya getirerek çalışır. MaxViT'in temel özelliği, görüntü üzerinde hem pencere tabanlı (window) hem de ızgara tabanlı (grid) dikkat mekanizmalarını kullanarak maksimum bilgi kapsama sağlamasıdır. Bu yapı, modelin hem küçük detayları hem de genel yapıyı aynı anda öğrenebilmesini mümkün kılar ve böylece hem düşük hem yüksek çözünürlüklü verilerle güçlü bir şekilde çalışmasını sağlar.[6]

1) Kodların Açıklanması:

a) Konfigürasyon ve Ortam Ayarları: Bu bölümde, modelin eğitim sürecinde kullanılacak sabitler (epoch sayısı, batch boyutu, öğrenme oranı, veri yolu gibi) tanımlanır. Aynı zamanda eğitim sürecinde tekrar üretilebilirlik sağlamak adına rastgele tohumlar ('random seed') belirlenir ve GPU desteği varsa CUDA aktif edilir. Modelde kullanılan 'LR' ve 'WEIGHT_DECAY', optimizasyon parametreleridir. Eğitim ve test işlemlerinin aynı koşullarda çalışması için bu ayarlar oldukça kritiktir.

b) Görüntü Dönüşümleri (Transforms): Bu bölümde eğitim, doğrulama ve test setleri için ayrı ayrı dönüşüm işlemleri tanımlanır. Eğitim verisi üzerinde 'augmentation' işlemleri uygulanarak modelin genelleme yeteneği artırılır (örneğin: rastgele kırpma, döndürme, renk bozulması gibi). Doğrulama ve test setlerinde ise sadece yeniden boyutlandırma ve normalize etme işlemleri uygulanır. Bu işlemler, modelin standartlaştırılmış görüntü verisi ile beslenmesini sağlar.

c) Veri Seti ve Veri Yükleyiciler (Dataloader): Veri seti 'ImageFolder' sınıfı ile 'train', 'val' ve 'test' klasörlerinden alınır ve yukarıda tanımlanan dönüşüm kuralları uygulanır. 'DataLoader' nesneleri ise bu verileri batch'ler halinde GPU'ya taşımak, eğitim sürecinde belleği verimli kullanmak ve verileri karıştırmak ('shuffle') gibi görevleri yerine getirir.

d) Model, Kayıp Fonksiyonu ve Optimizasyon: Burada 'MaxViT' modeli 'timm.create_model()' fonksiyonu ile çağrılır. Seçilen model ''maxvit_tiny_tf_224.in1k'' olup önceden ImageNet üzerinde eğitilmiş ağırlıklarla başlatılır. Modelin son katmanı sınıf sayısına göre güncellenir. Kayıp fonksiyonu olarak 'CrossEntropyLoss', optimizasyon yöntemi olarak ise 'AdamW' kullanılır. Ayrıca, 'CosineAnnealingLR' ile öğrenme oranı zamanla düşürülerek daha iyi yakınsama sağlanır.

e) Mixup Destekli Eğitim ve Doğrulama Döngüsü: Bu aşamada model 'train' moduna alınarak her epoch'ta veriler üzerinden geçilir. Eğitim sırasında 'mixup' tekniği kullanılır; bu teknik, iki görüntü ve etiketin rastgele karıştırılarak modelin daha kararlı ve genelleyici öğrenmesini sağlar. Her batch için ileri besleme, kayıp hesabı ve geri yayılım gerçekleştirilir. Ardından model doğrulama moduna alınır ve validation loss hesaplanır. Epoch sonunda öğrenme oranı scheduler ile güncellenir.

f) Model Kaydetme ve Test Aşaması: Eğitim tamamlandıktan sonra modelin ağırlıkları '.pth' dosyasına kaydedilir. Ardından bu ağırlıklar geri yüklenerek model 'eval' modunda test verileri üzerinde çalıştırılır. Tüm tahminler ve olasılık çıktıları bir listede toplanır. Bu, test metriklerinin hesaplanmasında kullanılır.

g) Performans Metriklerinin Hesaplanması: Test aşamasında elde edilen tahminlerle çeşitli performans metrikleri hesaplanır. Bunlar arasında doğruluk (accuracy), kesinlik (precision), duyarlılık (recall), F1 skoru, AUC (micro ve macro), duyarlılık (sensitivity) ve özgüllük (specificity) yer alır. 'Confusion matrix' üzerinden her sınıf için TP, TN, FP, FN değerleri çıkarılır ve makro düzeyde ortalama özgüllük hesaplanır.

h) Görselleştirmeler: Son olarak modelin başarısını görsel olarak değerlendirmek için 'Confusion Matrix', her sınıf için 'ROC Curve' ve eğitim/validation loss'larının zamanla değişimini gösteren bir loss grafiği çizilir. Bu grafikler modelin hangi sınıflarda başarılı veya zayıf olduğunu, overfitting olup olmadığını ve genel öğrenme dinamiğini anlamada yardımcı olur.

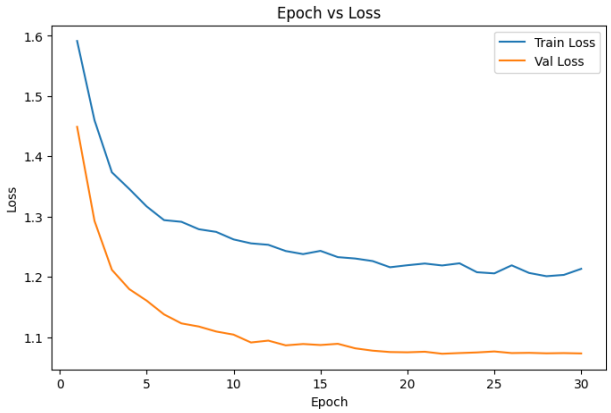

Fig. 18. Eğitim ve Doğrulama Kayıp Değerlerinin Epoch Bazlı Değişimi

2) Sonuçların Yorumları:

a) : Şekil 18'de görülen "Epoch vs Loss" grafiği, modelin eğitim ve doğrulama sürecindeki kayıp değerlerinin (loss) zamana bağlı olarak nasıl değiştiğini göstermektedir. İlk epoch'larda hem eğitim hem de doğrulama kaybında hızlı bir düşüş gözlemlenmektedir; bu da modelin başlangıçta hızlı bir öğrenme gerçekleştirdiğini ve veriye adapte olmaya başladığını gösterir. Eğitimin ilerleyen safhalarında kayıp değerleri daha yatay seyretmektedir, bu da modelin öğrenme hızının azaldığını ve dengeye ulaştığını işaret eder. Validation loss eğrisi oldukça stabil ilerlemekte ve herhangi bir sıçrama ya da aşırı dalgalanma göstermemektedir; bu durum modelin overfitting yapmadığını ve doğrulama verisi üzerinde tutarlı bir performans gösterdiğini göstermektedir. Ayrıca validation loss değerinin train loss'a yakın olması, modelin hem öğrenme hem de genelleme yeteneğinin güçlü olduğunu göstermektedir.

b) : Tablo IX'te sunulan sonuçlara göre MaxViT modeli, %60 civarında bir genel doğruluk (accuracy) elde etmiş ve bu değer, modelin çok sınıflı veri seti üzerinde ortalama düzeyde bir başarıya ulaştığını göstermektedir. Precision, recall ve F1-score metriklerinin birbirine çok yakın ve dengeli olması, modelin hem doğru pozitifleri hem de yanlış pozitifleri benzer oranda ele aldığını ve sınıflar arasında ciddi bir dengesizlik olmadığını göstermektedir. AUC-micro (0.8683) ve AUC-macro (0.8605) gibi ROC eğrisi tabanlı metriklerin yüksek olması, modelin sınıflar arası ayrım yapma yeteneğinin kuvvetli olduğunu ifade etmektedir. Specificity değerinin 0.90 üzerinde seyretmesi ise modelin negatif sınıfları ayırt etmede oldukça başarılı olduğunu kanıtlamaktadır. Eğitim süresinin önceki denemelere kıyasla belirgin şekilde artmış olması (yaklaşık 1 saat 44 dakika), daha derin ya da daha karmaşık bir optimizasyon süreci yürütüldüğünü işaret etmektedir. Inference süresi ise 8.2 saniye ile halen pratik uygulamalarda kullanılabilir düzeydedir. Sonuç olarak, bu metrikler MaxViT modelinin istikrarlı, dengeli ve güvenilir bir performans ortaya koyduğunu göstermektedir.

TABLE IX
MAXVIT MODELININ NIHAI TEST PERFORMANS METRIKLERI

Metrik	**Değer**
Doğruluk (Accuracy)	0.6077
Kesinlik (Precision - Macro)	0.6066
Duyarlılık (Recall - Macro)	0.6087
F1-Skor (Macro)	0.6066
AUC (Micro)	0.8683
AUC (Macro)	0.8605
Hassasiyet (Sensitivity)	0.6087
Özgüllük (Specificity)	0.9019
Eğitim Süresi (saniye)	6252.2
Çıkarım Süresi (saniye)	8.2

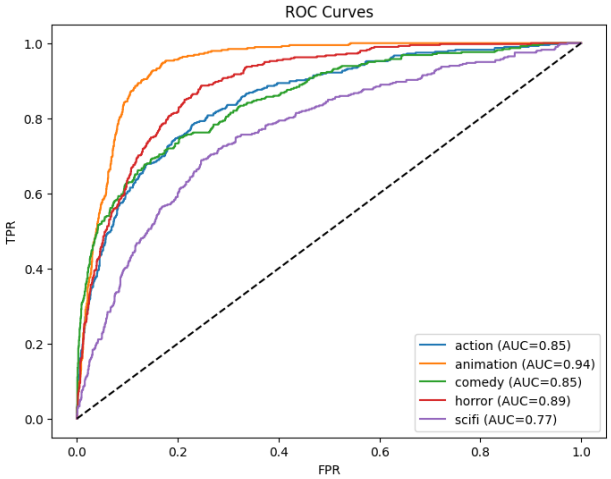

Fig. 19. Her bir sınıf için ROC Eğrileri ve AUC Değerleri

c) : Şekil 19'te her bir sınıfa ait ROC eğrileri ve karşılık gelen AUC (Area Under Curve) değerleri gösterilmektedir. ROC eğrisi, modelin sınıflandırma başarımını yanlış pozitif oranı (FPR) ile doğru pozitif oranı (TPR) arasındaki ilişki üzerinden analiz etmeye olanak tanır. Burada en yüksek AUC değerine sahip sınıf **animation** olup, 0.94 gibi oldukça yüksek bir başarı göstermektedir. Bu, modelin bu sınıfı diğerlerinden ayırt etme konusunda çok başarılı olduğunu gösterir. **Horror** sınıfı da 0.89 ile yüksek bir AUC'ye sahiptir ve güçlü sınıflandırma başarımı sergilemektedir. **Action** ve **Comedy** sınıfları 0.85 AUC ile dengeli bir performans sunarken, **Scifi** sınıfı 0.77 AUC ile diğer sınıfların gerisinde kalmıştır. Scifi

eğrisinin daha yavaş yükselmesi ve daha az eğimli olması, bu sınıfta daha fazla hata yapıldığını ve modelin bu sınıfı ayırt etmede zorlandığını göstermektedir. Genel olarak ROC eğrileri, modelin çoğu sınıfta iyi performans gösterdiğini ancak bazı sınıflarda (özellikle scifi) iyileştirme yapılması gerektiğini ortaya koymaktadır.

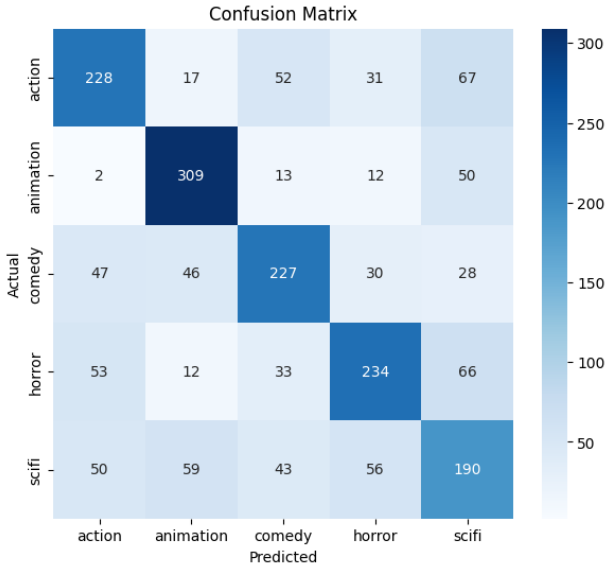

Fig. 20. Sınıf Bazlı Karışıklık Matrisi (Confusion Matrix)

d) : Şekil 20'te gösterilen karışıklık matrisi, MaxViT modelinin test verisindeki tahmin başarılarını ve hatalarını sınıf bazında ortaya koymaktadır. Diagonal (çapraz) üzerinde yer alan değerler doğru sınıflamaları ifade ederken, diğer hücreler yanlış sınıflamaları temsil eder. **Animation** sınıfı, 309 doğru sınıflama ile modelin en başarılı olduğu sınıf olarak öne çıkmaktadır. Buna karşılık, **Scifi** sınıfı yalnızca 190 kez doğru tahmin edilmiş ve sıklıkla diğer sınıflarla (özellikle animation, horror ve action) karıştırılmıştır. **Comedy** sınıfı ise 227 doğru tahminle orta seviyede başarı göstermektedir; ancak action ve animation sınıflarıyla ciddi karışıklık gözlenmektedir. **Horror** sınıfında da 234 doğru tahmin mevcut olmakla birlikte, 50'den fazla örnek yanlışlıkla action veya scifi olarak sınıflandırılmıştır. **Action** sınıfı, 228 kez doğru tahmin edilmiştir; ancak comedy (52) ve scifi (67) ile karışma oranı yüksektir. Genel olarak, model animation sınıfını ayırt etmede çok başarılıdır, ancak özellikle scifi sınıfında ciddi ayırt edememe problemi yaşamaktadır. Bu analiz, modelin sınıf dengesizlikleri veya benzer sınıf özellikleri nedeniyle bazı gruplarda daha fazla hata yaptığını açıkça ortaya koymaktadır.

V. Modellerin Kıyaslanması

Görüntü sınıflandırma alanında performans değerlendirmesi, farklı derin öğrenme modellerinin doğruluk, kesinlik, duyarlılık, F1-skoru ve AUC gibi metrikler üzerinden karşılaştırılmasıyla sağlanır. Bu bölümde, çeşitli Transformer tabanlı modellerin aynı veri seti üzerinde elde ettikleri sonuçlar analiz edilerek, hangi modelin sınıflandırma görevinde daha etkili olduğu ortaya konulacaktır. Elde edilen test metrikleri ışığında modellerin güçlü ve zayıf yönleri değerlendirilerek, uygulama açısından en uygun mimari belirlenmeye çalışılacaktır.

Model Loss Eğrilerinin Karşılaştırmalı Değerlendirmesi

Beş farklı Transformer tabanlı modelin eğitim ve doğrulama sürecine ait loss eğrileri karşılaştırıldığında, genel olarak tüm modellerin eğitim verisi üzerinde istikrarlı bir öğrenme süreci geçirdiği, ancak doğrulama verisi üzerinde sergiledikleri genelleme performanslarının değişkenlik gösterdiği anlaşılmaktadır.

Swin, BEiT ve ViT modelleri arasında validation loss eğrisinin en stabil ve düşük seyreden modelin ViT olduğu görülmektedir. ViT modeli, hem training loss'ta hem de validation loss'ta istikrarlı bir azalma göstermiş ve her iki eğri arasında çok küçük bir farkla eğitimi tamamlamıştır. Bu durum, ViT modelinin sadece eğitim verisine değil, doğrulama verisine de etkili bir şekilde genelleme yaptığını ve overfitting belirtisi göstermediğini ortaya koymaktadır. Aynı şekilde BEiT modeli de oldukça benzer bir seyir izleyerek, validation loss'un belirgin biçimde düştüğü ve düşük seviyelerde sabit kaldığı bir süreç geçirmiştir. Ancak ViT'e kıyasla BEiT'in loss eğrisinde çok hafif dalgalanma ve daha az agresif bir düşüş gözlenmiştir. Bu durum, BEiT modelinin de etkili bir şekilde öğrendiğini ancak ViT'e kıyasla biraz daha yavaş optimize olduğunu düşündürmektedir.

Swin modeli ise training loss'ta güçlü bir azalma sağlasa da validation loss eğrisinde zaman zaman hafif dalgalanmalar görülmüştür. Bu dalgalanmalar her ne kadar kritik düzeyde olmasa da, modelin genelleme aşamasında ViT veya BEiT kadar stabil olmadığını göstermektedir. Yine de validation kaybının düşmeye devam etmesi ve sonlara doğru sabitlenmesi, Swin'in genel olarak başarılı bir öğrenme gerçekleştirdiğini ve minimum düzeyde overfitting sergilediğini ortaya koymaktadır.

Buna karşılık, DeiT modelinin loss eğrileri incelendiğinde, validation loss'un yaklaşık 10. epoch'tan sonra yataylaştığı ve neredeyse sabit bir seviyeye oturduğu görülmektedir. Eğitim kaybı azalmaya devam ederken doğrulama kaybında iyileşme olmaması, DeiT'in eğitim verisine fazla uyum sağladığını ve bu nedenle overfitting eğilimi gösterdiğini düşündürmektedir. Bu fark, DeiT modelinin daha fazla regularization tekniğine ihtiyaç duyabileceğini ve mevcut haliyle genelleme konusunda diğer modellere göre daha zayıf kaldığını göstermektedir.

MaxViT modelinde ise eğitim kaybı istikrarlı bir şekilde azalmasına rağmen, validation loss'un belirli bir seviyeden sonra plato çizmesi ve düşüş hızının durması dikkat çekicidir. İlk epoch'larda validation loss hızlıca düşmüş, ancak yaklaşık 20. epoch'tan sonra yatay bir eğriye dönüşmüştür. Bu durum, modelin yüksek parametre sayısı ve karmaşık yapısına rağmen, belirli bir seviyenin ötesinde doğrulama verisinde iyileşme sağlayamadığını ve bu nedenle ViT ve BEiT gibi modellere kıyasla daha düşük genelleme başarısına sahip olduğunu göstermektedir. Ayrıca MaxViT'in validation eğrisindeki bu plato davranışı, modelin veri setindeki örüntüleri öğrenmekte sınırlarına ulaşmış olabileceğine işaret eder.

Sonuç olarak, loss eğrilerinin seyri dikkate alındığında, **ViT modeli hem öğrenme sürecinin istikrarı hem de validation başarısındaki tutarlılığıyla en dengeli model olarak öne çıkmaktadır**. Onu **BEiT** ve **Swin** modelleri takip etmekte, bu üç model genel olarak başarılı bir eğitim ve genelleme süreci sergilemektedir. **DeiT** ve **MaxViT** ise validation kaybındaki erken doygunluk ve yataylaşma nedeniyle genelleme kapasitesi açısından geride kalmakta, bu da bu modellerin daha karmaşık yapılarına rağmen avantajlarını tam olarak kullanamadıklarını göstermektedir.

TABLE X
FARKLI TRANSFORMER TABANLI MODELLERIN NIHAI TEST METRIKLERI KARŞILAŞTIRMASI

Metrik	**Swin**	**BEiT**	**ViT**	**DeiT**	**MaxViT**
Accuracy	0.6164	0.6174	**0.6215**	0.6066	0.6077
Precision (Macro)	**0.6186**	0.6154	0.6169	0.6061	0.6066
Recall (Macro)	0.6172	0.6181	**0.6222**	0.6077	0.6087
F1-Score (Macro)	0.6154	0.6144	**0.6173**	0.6049	0.6066
AUC (Micro)	0.8721	0.8748	**0.8765**	0.8728	0.8683
AUC (Macro)	0.8640	0.8667	**0.8683**	0.8661	0.8605
Sensitivity (Recall M)	0.6172	0.6181	**0.6222**	0.6077	0.6087
Specificity	0.9041	0.9043	**0.9054**	0.9016	0.9019
Train Time (s)	4655.2	3794.9	**3392.3**	3392.9	6252.2
Inference Time (s)	6.6	7.4	**6.5**	7.1	8.2

Doğruluk (Accuracy)

Accuracy metriği, tüm doğru tahminlerin toplam örnek sayısına oranını temsil eder ve modelin genel başarı düzeyini gösterir. Bu metrikte en yüksek değere ViT modeli ulaşmıştır ve doğruluk oranı %62.15'tir. BEiT (%61.74) ve Swin (%61.64) modelleri ViT'i çok yakın takip etmekte ve aralarındaki fark %0.5'ten küçüktür. Bu üç model, genel doğruluk açısından benzer bir başarı düzeyine sahiptir. Öte yandan, DeiT (%60.66) ve MaxViT (%60.77) modelleri, doğruluk açısından yaklaşık %1.5 daha düşük değerler almıştır. Bu da ViT, BEiT ve Swin modellerinin veri üzerindeki sınıflamayı daha isabetli yaptığını ortaya koymaktadır.

Kesinlik (Precision)

Precision, modelin pozitif olarak sınıflandırdığı örneklerin gerçekten pozitif olma oranını gösterir. Swin modeli bu metrikte %61.86 ile liderdir ve yanlış pozitif oranını en düşük seviyede tutmayı başarmıştır. ViT (%61.69) ve BEiT (%61.54) modelleri bu başarıya oldukça yakın düzeyde kalmıştır. Ancak DeiT (%60.61) ve MaxViT (%60.66) modelleri, diğer üç modele göre bu metrikte yaklaşık %1.2 daha düşük başarı göstermiştir. Bu durum, hassasiyetin ön planda olduğu sistemlerde Swin modelinin daha güvenilir olabileceğini göstermektedir.

Duyarlılık (Recall)

Recall metriği, gerçekten pozitif olan örneklerin ne kadarının model tarafından doğru tanındığını ölçer. Bu alanda en yüksek değer %62.22 ile ViT modeline aittir. Bu değeri BEiT (%61.81) ve Swin (%61.72) modelleri takip etmektedir. Bu modeller arasında anlamlı bir fark bulunmamakla birlikte, ViT modelinin duyarlılık açısından en başarılı model olduğu görülmektedir. DeiT (%60.77) ve MaxViT (%60.87) ise yaklaşık%1.5 daha düşük performans göstermiştir. Özellikle yanlış negatiflerin kritik olduğu uygulamalarda ViT modeli öne çıkmaktadır.

F1-Score

F1-Score, precision ve recall değerlerinin harmonik ortalamasıdır ve modelin pozitif sınıfları ne kadar doğru ve dengeli tahmin ettiğini gösterir. ViT modeli burada da %61.73 ile en yüksek değere ulaşmıştır. Swin (%61.54) ve BEiT (%61.44) modelleri yine çok yakın sonuçlar vermiştir. Bu üç model genel olarak dengeli ve tutarlı bir sınıflandırma performansı sunmaktadır. DeiT (%60.49) ve MaxViT (%60.66) modelleri bu metriğe göre daha düşük performans sergilemiştir.

AUC (Micro ve Macro)

AUC (Area Under Curve) metriği, modelin sınıflar arasındaki ayrımı ne kadar iyi yaptığını gösterir. ViT modeli, AUC-micro (%87.65) ve AUC-macro (%86.83) değerleriyle bu metriğin her iki boyutunda da en iyi sonucu vermiştir. BEiT (%87.48 mikro, %86.67 makro) onu takip ederken, Swin (%87.21 mikro, %86.40 makro), DeiT (%87.28 mikro, %86.61 makro) ve MaxViT (%86.83 mikro, %86.05 makro) biraz daha geride kalmıştır. Özellikle MaxViT'in AUC-macro değeri diğer modellere kıyasla daha düşüktür ve bu durum, sınıflar arası ayrımda belirgin bir zayıflığa işaret etmektedir.

Özgüllük (Specificity)

Specificity metriği, negatif örneklerin doğru şekilde tanımlanma oranını gösterir. Bu metrikte ViT modeli %90.54 ile birinci sırada yer almaktadır. BEiT (%90.43), Swin (%90.41), MaxViT (%90.19) ve DeiT (%90.16) modelleri ise birbirine çok yakın değerler almıştır. Bu, tüm modellerin negatif sınıfları ayırt etmede yüksek başarıya sahip olduğunu göstermektedir; ancak ViT modelinin yine küçük bir farkla en yüksek performansı sunduğu görülmektedir.

Eğitim Süresi (Train Time)

Modelin eğitim süresi hesaplama kaynakları açısından önemli bir kriterdir. En hızlı eğitilen modeller ViT (3392.3 saniye) ve DeiT (3392.9 saniye) olmuştur. BEiT (3794.9 saniye) bir miktar daha uzun sürede eğitilirken, Swin (4655.2 saniye) ve özellikle MaxViT (6252.2 saniye) modelleri çok daha fazla eğitim süresi gerektirmiştir. Bu, özellikle büyük veri kümeleriyle yapılan çalışmalarda ViT ve DeiT modellerinin daha pratik ve ekonomik olabileceğini göstermektedir.

Çıkarım Süresi (Inference Time)

Modelin çıkarım süresi, test verisinde tahmin yapma süresini gösterir ve özellikle gerçek zamanlı uygulamalar için önemlidir. ViT modeli bu alanda da en iyi performansı sunmakta ve çıkarım süresi sadece 6.5 saniyedir. Swin (6.6 saniye) ve DeiT (7.1 saniye) onu takip ederken, BEiT (7.4 saniye) biraz daha yavaştır. MaxViT modeli ise 8.2 saniyelik çıkarım süresi ile en yavaş çalışan modeldir.

Genel Değerlendirme

Tüm metrikler bir arada değerlendirildiğinde, ViT modeli açık bir liderlik sergilemektedir. Hem sınıflandırma başarısı hem de işlem verimliliği açısından öne çıkan ViT, birçok metrikte en yüksek değeri almış ve zaman açısından da oldukça avantajlı olduğunu göstermiştir. BEiT ve Swin modelleri de güçlü performans sergilemekte ve bazı metriklerde ViT'e çok yakın sonuçlar vermektedir. Buna karşın, DeiT ve özellikle MaxViT modelleri, hem başarı hem de süre açısından geri planda kalmıştır. Sonuç olarak, ViT modeli sınıflandırma problemlerinde güçlü bir tercih olarak öne çıkarken, BEiT ve Swin modelleri de güvenilir alternatifler olarak değerlendirilebilir.

REFERENCES

[1] https://letterboxd.com

[2] https://arxiv.org/abs/2103.14030

[3] https://arxiv.org/abs/2106.08254

[4] https://arxiv.org/abs/2010.11929

[5] https://arxiv.org/abs/2012.12877

[6] https://arxiv.org/abs/2204.01697

[7] Google Drive Kod ve Dataset Linkleri https://drive.google.com/drive/folders/1Sj3MZzIIKMdeqWQpgAYdVZL2VYC3uJ42?usp=sh