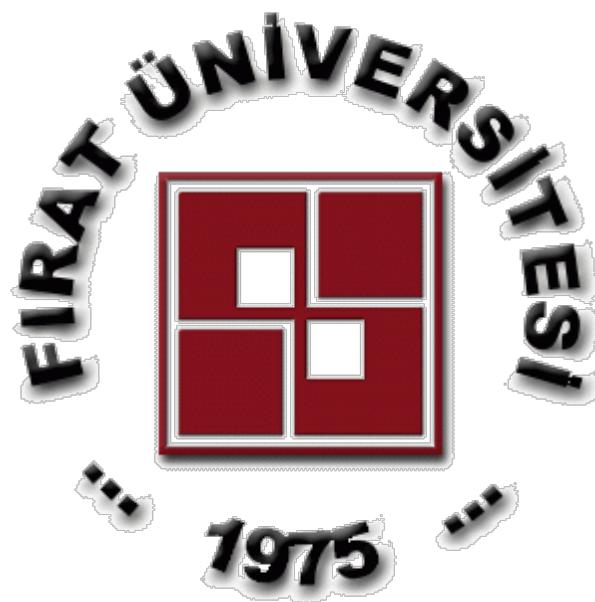


T.C.
FIRAT ÜNİVERSİTESİ



STAJ DEFTERİ

MÜHENDİSLİK FAKÜLTESİ



*"Çalışmak demek, boşuna yorulmak, terlemek
değildir. Zamanın gereklерine göre bilim
teknik ve her türlü uygar buluşlardan azami
derecede istifade etmek zorunludur."*

**T.C.
FIRAT ÜNİVERSİTESİ
MÜHENDİSLİK FAKÜLTESİ
STAJ DEFTERİ**

ÖĞRENCİNİN	BÖLÜMÜ	Yazılım Mühendisliği		
	NUMARASI	200290604		
	ADI VE SOYADI	Ahmed Salih		
	STAJ DÖNEMİ	4.Dönem		
	STAJ BAŞ. TARİHİ	03/09/2024	TOPLAM İŞ GÜNÜ SAYISI	20
	STAJ BİTİŞ TARİHİ	30/09/2024		

FOTOĞRAF

STAJ YAPILAN İŞ YERİ BİLGİLERİ

İŞ YERİ	ADI	Intellium Bilişim Teknolojileri A.Ş.	<p>Yukarıda bilgileri yer alan öğrencinin iş yerimizde 20 iş günü staj yaptığını ve bu defterin öğrenci tarafından tanzim edildiğini beyan ve tasdik ederim.</p> <p style="text-align: right;">Onay</p>
	ADRESİ	Intellium Bilişim Teknolojileri A.Ş. Dijitalpark Teknokent, Çekmeköy Yerleşkesi, Kirazdere Mah. Eski Ankara Cad. İdari Bina A-1 Blok No:4A No:15 Çekmeköy/İstanbul	
	TELEFON-FAKS	+90 216 388 4033	
SORUMLU MÜHENDİS	ADI VE SOYADI		
	ÜNVANI		
	GÖREVİ		
AMİR	ADI VE SOYADI		
	ÜNVANI		
	GÖREVİ		

BÖLÜM STAJ KOMİSYONU DEĞERLENDİRME SONUCU

Yapılan pratik çalışmanın iş günü Dönem stajı olarak kabul edilmiştir /edilmemiştir.
Yapılan pratik çalışmanın iş günü Dönem stajı olarak kabul edilmiştir /edilmemiştir.

STAJ KOMİSYONU

...../...../...../...../...../...../.....
İMZA	İMZA	İMZA
BAŞKAN	ÜYE	ÜYE

İÇİNDEKİLER

<u>KONU</u>	<u>Sayfa No</u>
Gün 1: AI Ekipi ile başlama	1
Gün 2: Super-Resolution Convolutional Neural Networks (SRCNN)	2
Gün 3: Python ile Yapay Zeka Eğitimi	4
Gün 4: PyTorch ile Derin Öğrenme Modelleri	6
Gün 5: Terms and Conditions Ekranı için Görev Uygulaması (Trello Task)	8
Gün 6: <i>Customer</i> ve <i>Ticket</i> Widget Yapılarının Tasarımı ve Uygulanması	10
Gün 7: Ticket Yönetiminde Arama ve Filtreleme Fonksiyonlarının Eklenmesi ve Ticket Detay Ekranının Geliştirilmesi	12
Gün 8: Müşteri Çekmecesi Tasarımı (Customer Drawer)	14
Gün 9: Firebase Entegrasyonu ve Giriş Sayfası Oluşturulması	16
Gün 10: Giriş Ekranı Detayları ve Hata Yönetimi	24
Gün 11: WelcomePage	18
Gün 12: Yapay Zeka Modeli Geliştirme ve Super-Resolution (Süper Çözünürlük) Uygulaması	20
Gün 13: Yapay Zeka Modeli Geliştirme ve Super-Resolution (Süper Çözünürlük) Uygulaması	22
Gün 14: Derin Öğrenme ve Süper Çözünürlük: Google Colab ile Yapay Zeka Modeli Eğitimi	24
Gün 15: Süper Çözünürlük Modelinin Kurulumu ve İlk Aşamaları	26
Gün 16: Eğitim Süreci ve Modelin Performans Analizi	27
Gün 17: Yeni Modelin Geliştirilmesi ve SRCNN Yapısı	28
Gün 18: Modelin Eğitimi ve Performans Analizi	29
Gün 19: Modelin Performansını Artırma ve Kayıp Fonksiyonlarının Değerlendirilmesi	30
Gün 20: Veri Çeşitlendirme ve Augmentasyon Stratejileri	31

ŞEKİL, ÇİZELGE VE EKLER LİSTESİ

<u>Sekil, Çizelge veya Ek No</u>	<u>Sayfa No</u>
Şekil 1 ve 2	1
Şekil 3 ve 4	2
Şekil 5	3
Şekil 6 ve 7	4
Şekil 8	5
Şekil 9, 10.....	6
Şekil 11 ,12, 13 ve14.....	7
Şekil 15 ve 16.....	8
Şekil 17	9
Şekil 18	10
Şekil 19	11
Şekil 20	12
Şekil 21 ve 22	13
Şekil 23	14
Şekil 24 ,25 ve 26.....	15
Şekil 27 ve 28.....	16
Şekil 29	17
Şekil 30 ve 31	18
Şekil 32 ve 33	19
Şekil 34 ,35 ,36 ,37 ve 38	20
Şekil 39 ,40 ve 41	21
Şekil 42ve 43	22
Şekil 44	23
Şekil 45	24
Şekil 46 ve 47.....	25
Şekil 48 ,49 50 ve 51.....	26
Şekil 52 ve 53.....	27
Şekil 54 ve 55.....	28
Şekil 56.	29
Şekil 57.	30
Şekil 58	31

STAJIN YAPILDIĞI KURUM VEYA KURULUŞUN TANIMI

Intellium, İstanbul merkezli bir Türk teknoloji şirketidir. Şirket, deneyimli bir mühendis ekibi tarafından 2015 yılında kurulmuştur. Intellium, şirketlerin tüm büyüklüklerine yapay zeka çözümleri geliştirmeye odaklanmaktadır.

Intellium, şunlar da dahil olmak üzere çeşitli ürünler ve hizmetler sunar:

- Veri işleme ve karar alma için makine öğrenme çözümleri
- Görüntü ve video analizi için görüntü işleme çözümleri
- İnsan dilini anlamak için doğal dil işleme çözümleri
- Akıllı robotlar oluşturmak için robotik çözümler

Intellium, yapay zeka teknolojilerini şirketlerin verimliliğini ve üretkenliğini artırmak için kullanır. Örneğin, Intellium'un çözümleri şirketlere şunlarda yardımcı olabilir:

- Tahmin doğruluğunu artırma
- İşletme verimliliğini artırma
- Müşteri deneyimini iyileştirme

Intellium'un ana müşterileri farklı sektörlerden şirketleri içerir, bu sektörler arasında finansal hizmetler, imalat ve sağlık hizmetleri bulunmaktadır.

Aşağıda Intellium'un bazı projelerine örnekler verilmiştir:

- Intellium, müşteri verilerini analiz etmek için yapay zeka tabanlı bir sistem geliştirdi ve bir havayolu şirketinin tahmin doğruluğunu artırmalarına yardımcı oldu.
- Intellium, bir üretim tesisi verimliliğini artırmak için yapay zeka tabanlı bir kontrol sistemini geliştirdi ve bir imalat şirketine yardımcı oldu.
- Intellium, sağlık hizmeti veren bir şirketin müşteri deneyimini iyileştirmelerine yardımcı olmak için sağlık verilerini analiz etmek için yapay zeka tabanlı bir sistem geliştirdi.

Intellium, Türkiye'de yapay zeka alanında öncü olmayı hedeflemektedir. Şirket dünya genelinde yeni pazarlara açılmayı planlamaktadır.

GİRİŞ

Bu staj sürecinde, farklı teknolojiler ve araçlar kullanarak hem teorik hem de pratik düzeyde bir dizi proje üzerinde çalıştım. Staj boyunca süper çözünürlük (super-resolution) modellemelerinden mobil uygulama geliştirmeye kadar geniş bir yelpazede çeşitli yazılım araçlarını ve tekniklerini öğrendim ve uyguladım. **Python**, **PyTorch**, **Google Colab**, **Flutter** gibi platform ve dillerin entegrasyonu ile gerçek dünyadaki problemleri çözmeye yönelik projeler geliştirdim. Aşağıda, kullandığım ana araçlar ve yöntemler detaylı bir şekilde açıklanmıştır.

1. Python ve PyTorch

Staj sürecinde, derin öğrenme temelli görüntü işleme projeleri için **Python** ve **PyTorch**'u yoğun olarak kullandım. PyTorch, esnek yapısı ve geniş topluluk desteği ile derin öğrenme modellerini geliştirmek ve eğitmek için ideal bir framework sağladı. Özellikle süper çözünürlük modellemelerinde, düşük çözünürlüklü görüntüler daha yüksek çözünürlüklere dönüştürmek amacıyla **SRCNN (Super-Resolution Convolutional Neural Network)** gibi modeller geliştirdim. PyTorch'un sunduğu GPU desteği ile eğitim süreçlerini hızlandıracak büyük veri setleriyle çalışabildim. Ayrıca, PyTorch'un veri işleme modüllerini kullanarak veri setlerini hazırladım, dönüşümler gerçekleştirdim ve modelin performansını izlemek için çeşitli kayıp fonksiyonlarını denedim.

2. Google Colab

Google Colab, derin öğrenme projelerimi bulut tabanlı bir ortamda çalıştmak için kullandığım önemli bir araç oldu. **Colab**, ücretsiz GPU ve TPU kaynakları sunarak büyük boyutlu veri setlerini işleyemem ve karmaşık modelleri daha hızlı eğitebilmemi sağladı. Süper çözünürlük modellemeleri için Colab üzerinde **DIV2K** gibi büyük veri setlerini indirip işledim. Colab'in interaktif çalışma ortamı, kodları hızlı bir şekilde çalıştırıp görselleştirmeler yapmamı, eğitim süreçlerini detaylı bir şekilde izlememi sağladı. Ayrıca, staj süresince Python kodlarını farklı platformlarda çalışma ihtiyacı duyduğumda, Colab üzerinden hızlıca testler yaparak projeleri optimize edebildim.

3. Flutter

Staj süresince üzerinde çalıştığım bir diğer önemli teknoloji ise **Flutter** oldu. Mobil uygulama geliştirme platformu olan Flutter, tek bir kod tabanıyla hem iOS hem de Android uygulamaları geliştirmeye olanak sağladı. Özellikle mobil cihazlar üzerinde görüntü işleme projelerimi test etmek ve kullanıcı dostu arayüzler geliştirmek için Flutter'ı tercih ettim. Flutter'in zengin widget kütüphanesi ve esnek yapısı sayesinde, kullanıcı deneyimini en üst düzeye çıkarmak için modern ve duyarlı arayüzler tasarladım. Mobil uygulama geliştirme sürecinde Dart dilini kullanarak yazdığım kodlar, uygulamanın performansını ve hızını optimize etmemeye yardımcı oldu.

4. Keras ve TensorFlow

Derin öğrenme projelerinde ayrıca **Keras** ve **TensorFlow** framework'lerini de kullandım. **Keras**, PyTorch'a kıyasla daha yüksek seviyeli bir API sunarak daha hızlı prototipleme yapmama olanak sağladı. Özellikle, basit model yapılarını hızlıca test etmek ve sonuçları görselleştirmek için Keras'ı tercih ettim. Keras'ın TensorFlow ile entegrasyonu sayesinde, gelişmiş görüntü işleme tekniklerini ve sınırlı ağı modellerini kullanarak hızlı sonuçlar elde edebildim.

5. Git ve Versiyon Kontrol

Projelerimi geliştirme sürecinde **Git** versiyon kontrol sistemini aktif bir şekilde kullandım. Git, hem bireysel projelerimde hem de ekip çalışmasında proje takibini ve sürüm yönetimini sağladı. **GitHub** üzerindeki projelerimde, kod değişikliklerini takip ettim ve farklı sürümler arasında geçiş yaparak kodun gelişimini izledim. Bu süreç, yazılım projelerinde iş birliği yaparken kodun güncel kalmasını ve geriye dönük hataların tespit edilmesini kolaylaştırdı.

6. Veri Bilimi ve Görselleştirme Araçları

Staj sürecinde ayrıca **Matplotlib**, **Seaborn**, **Pandas** gibi veri bilimi kütüphanelerini kullanarak projelerimdeki verileri analiz ettim ve görselleştirdim. Model eğitim süreçlerinde kayıpların ve doğrulukların izlenmesi, sonuçların analiz edilmesi ve raporlanması aşamalarında bu araçlar oldukça faydalı oldu. Eğitim süreçlerinin performansını izlemek için görselleştirme araçları ile kayıp ve doğrulama egrilerini çıkararak, modelin ne zaman aşırı öğrenmeye başladığını veya yeterli performansı gösterip göstermediğini analiz ettim.

03/09/2024 tarihinden 06/09/2024 tarihine kadar bir haftalık çalışma

Gün	Yapılan İşler	Yapılan İşle İlgili Bilginin Bulunduğu Sayfa	Saat
Pazartesi			
Salı	AI Ekipi ile başlama	1	8
Çarşamba	Super-Resolution Convolutional Neural Networks (SRCNN)	2 ve 3	8
Perşembe	Python ile Yapay Zeka Eğitimi	4 ve 5	8
Cuma	PyTorch ile Derin Öğrenme Modelleri	6 ve 7	8
Cumartesi			
Denetleyenin İmzası		Toplam Saat	32

09/09/2024 tarihinden 13/09/2024 tarihine kadar bir haftalık çalışma

Gün	Yapılan İşler	Yapılan İşle İlgili Bilginin Bulunduğu Sayfa	Saat
Pazartesi	Terms and Conditions Ekranı için Görev Uygulaması (Trello Task)	8 ve 9	8
Salı	Customer ve Ticket Widget Yapılarının Tasarımı ve Uygulanması	10 ve 11	8
Çarşamba	Ticket Yönetiminde Arama ve Filtreleme Fonksiyonlarının Eklenmesi ve Ticket Detay Ekranının Geliştirilmesi	12 ve 13	8
Perşembe	Müşteri Çekmecesi Tasarımı (Customer Drawer)	14 ve 15	8
Cuma	Firebase Entegrasyonu ve Giriş Sayfası Oluşturulması	16 ve 17	8
Cumartesi			
Denetleyenin İmzası		Toplam Saat	40

16/09/2024 tarihinden 20/09/2024 tarihine kadar bir haftalık çalışma

Gün	Yapılan İşler	Yapılan İşle İlgili Bilginin Bulunduğu Sayfa	Saat
Pazartesi	Giriş Ekranı Detayları ve Hata Yönetimi	18 ve 19	8
Salı	WelcomePage	20 ve 21	8
Çarşamba	Yapay Zeka Modeli Geliştirme ve Super-Resolution (Süper Çözünürlük) Uygulaması	22 ve 23	8
Perşembe	Yapay Zeka Modeli Geliştirme ve Super-Resolution (Süper Çözünürlük) Uygulaması	24	8
Cuma	Derin Öğrenme ve Süper Çözünürlük: Google Colab ile Yapay Zeka Modeli Eğitimi	25	8
Cumartesi			
Denetleyenin İmzası		Toplam Saat	40

23/09/2024 tarihinden 27/09/2024 tarihine kadar bir haftalık çalışma

Gün	Yapılan İşler	Yapılan İşle İlgili Bilginin Bulunduğu Sayfa	Saat
Pazartesi	Süper Çözünürlük Modelinin Kurulumu ve İlk Aşamaları	26	8
Salı	Eğitim Süreci ve Modelin Performans Analizi	27	8
Çarşamba	Yeni Modelin Geliştirilmesi ve SRCNN Yapısı	28	8
Perşembe	Modelin Eğitimi ve Performans Analizi	29	8
Cuma	Modelin Performansını Artırma ve Kayıp Fonksiyonlarının Değerlendirilmesi	30	8
Cumartesi			
Denetleyenin İmzası		Toplam Saat	40

30/09/2024 tarihinden 30/09/2024 tarihine kadar bir haftalık çalışma

Gün	Yapılan İşler	Yapılan İşle İlgili Bilginin Bulunduğu Sayfa	Saat
Pazartesi	Veri Çeşitlendirme ve Augmentasyon Stratejileri	31	8
Salı			
Çarşamba			
Perşembe			
Cuma			
Cumartesi			
Denetleyenin İmzası		Toplam Saat	8

...../...../20.... tarihinden/...../20.... tarihine kadar bir haftalık çalışma

Gün	Yapılan İşler	Yapılan İşle İlgili Bilginin Bulunduğu Sayfa	Saat
Pazartesi			
Salı			
Çarşamba			
Perşembe			
Cuma			
Cumartesi			
Denetleyenin İmzası		Toplam Saat	

Gün 1:

AI Ekipi ile başlama

İkinci stajimin ilk günü, üniversitemizin belirlediği staj tarihlerinin farklılığı göstermesi nedeniyle diğer stajyerlerle eş zamanlı başlamadım. Bu durum, projeye biraz daha geç katılmama neden oldu. Staja geç başlamış olsam da, projeye hızlı bir şekilde adapte olmak için gerekli adımları attım. İlk gün, AI ekibine dahil olduğumda, projenin sonuna yaklaşıldığını fark ettim. Bu süreçte bana rehberlik eden Turgut Bey, haftalık bir eğitim ve araştırma programı oluşturmamı önerdi.

Yeni projemizin ana konusu, yapay zeka tabanlı bir model geliştirmek görüntülerin kalitesini artırmak üzerine kurulu. Bu model, görüntülerin çözünürlüğünü yükselterek daha net ve detaylı görseller elde etmeyi hedeflemekte. Söz konusu teknoloji, literatürde "süper çözünürlük" (super resolution) olarak adlandırılmasında ve günümüzde yapay zeka ve derin öğrenme yöntemleri kullanılarak bu alanda önemli gelişmeler sağlanmaktadır. Bu modelin temel amacı, düşük çözünürlüklü görüntülerden yüksek çözünürlüklü, daha keskin ve detaylı görseller elde etmektir.

Projede benim ve ekibimin görevi ise iki ana aşamadan oluşuyor. İlk aşamada, Flutter kullanarak bir mobil uygulama geliştirmemiz gerekiyor. Bu uygulama, kullanıcıların düşük çözünürlüklü bir görsel yükleyerek yapay zeka modelinden faydalananarak yüksek çözünürlüklü versiyonunu elde etmesini sağlayacak. Uygulamanın mimarisi, daha önce GetX ile yaptığımız projeye benzer şekilde tasarılanacak. GetX, Flutter ile geliştirdiğimiz uygulamalarda state management (durum yönetimi) ve dependency injection (bağımlılık yönetimi) işlemlerini kolaylaştırın bir mimari yapı sunmakta, bu da uygulamanın hem geliştirme sürecini hızlandırmakta hem de yönetilebilirliğini artırmaktadır.

Projenin ikinci aşaması, yapay zeka modelinin geliştirilmesi ve bu modelin bir sunucuya entegre edilerek API (Application Programming Interface) üzerinden çalıştırılmasıdır. Modeli oluşturmak için derin öğrenme (deep learning) yöntemlerinden yararlanılacak. Özellikle, görüntü işleme (image processing) ve derin öğrenme tekniklerinin birleşimi olan konvolüsyonel sinir ağları (CNN - Convolutional Neural Networks) gibi yöntemler süper çözünürlük problemlerinde oldukça etkilidir. Bu tür modeller, düşük çözünürlüklü bir görüntüyü giriş olarak alır ve bu görüntüyü daha yüksek çözünürlüklü bir çıktıya dönüştürmeyi öğrenir.

Yapay zeka modelimizi oluşturduktan sonra, bu modeli bir sunucu üzerinde çalıştırarak Flutter uygulaması ile iletişim kurmasını sağlayacağız. Kullanıcılar uygulama aracılığıyla bir görsel yüklediklerinde, bu görsel sunucuya gönderilecek ve model tarafından işlenerek yüksek çözünürlüklü hali tekrar uygulamaya döndürülecek. Projenin bu aşaması, hem modelin performansı hem de sunucu ve mobil uygulama arasındaki iletişim sorunsuz olması açısından kritik bir öneme sahip.

Projemizin adı "SUREM+" olacak ve bu isim, projenin temel amacını yani "Süper Çözünürlük Modeli" (Super Resolution Model) geliştirilmesini yansıtmaktır. İlk gün, projenin genel hatlarını ve ana hedeflerini kavradım. İllerleyen günlerde, yapay zeka modelinin detaylarına inerek modelin nasıl geliştirileceğini araştırmak, sunucu kurulumu üzerine çalışmak ve Flutter uygulamasının geliştirme sürecine başlamak üzere bir yol haritası oluşturacağım.



Şekil 1



Şekil 1

Gün 2 :

Super-Resolution Convolutional Neural Networks (SRCNN)

Super-Resolution Convolutional Neural Networks (SRCNN), görüntü çözünürlüğünü iyileştirmek için kullanılan derin öğrenme tabanlı bir tekniktir. SRCNN, düşük çözünürlüklü bir görüntüyü alır ve bu görüntüyü daha yüksek bir çözünürlüğe çıkararak daha net ve detaylı hale getirir. Bu teknik, özellikle tıbbi görüntüleme, uydu fotoğrafları ve güvenlik kameraları gibi alanlarda oldukça etkilidir.

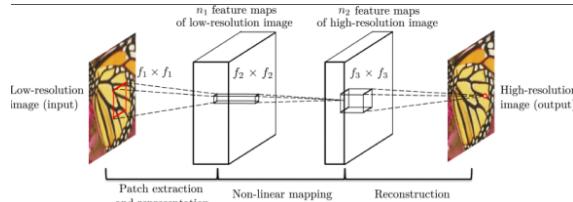
Kullanım Alanları:

- Tıbbi Görüntüleme:** Düşük çözünürlüklü tıbbi görüntülerin iyileştirilmesi, hastalıkların daha net teşhisini için kullanılır.
- Uydu Görüntüleme:** Düşük çözünürlüklü uydu görüntülerinin çözünürlüğünü artırarak, arazi analizi ve çevresel gözlemler yapılır.

Örnek 1:

Bir uydu görüntüsü aldığınızı düşünün. Bu görüntü, atmosferik koşullar nedeniyle düşük çözünürlükte olabilir. SRCNN kullanarak, bu görüntüyü daha net ve yüksek çözünürlükte bir hale getirebilirsizsiniz. Örneğin, tarım alanlarını daha net bir şekilde analiz etmek için bu yöntemi kullanabilirsiniz.

SRCNN ile Görüntü Çözünürlüğü Artırma



Şekil 1

```

1 import cv2
2 import numpy as np
3 from tensorflow.keras.models import Sequential
4 from tensorflow.keras.layers import Conv2D
5
6 # SRCNN Modelini Tanımlama
7 def build_srcnn():
8     model = Sequential()
9     # İlk katman: Giriş katmanı, 64 filtre, 9x9 boyutunda
10    model.add(Conv2D(64, (9, 9), activation='relu', padding='same', input_shape=(None, None, 1)))
11    # İkinci katman: 32 filtre, 1x1 boyutunda
12    model.add(Conv2D(32, (1, 1), activation='relu', padding='same'))
13    # Üçüncü katman: Çıkış katmanı, 1 filtre, 5x5 boyutunda
14    model.add(Conv2D(1, (5, 5), activation='linear', padding='same'))
15
16    return model
17
18 # Modeli Oluşturma
19 model = build_srcnn()
20 model.compile(optimizer='adam', loss='mean_squared_error')
21
22 # Görüntüyü Yükleme ve Ön İşleme
23 low_res_image = cv2.imread('low_res_image.jpg', cv2.IMREAD_GRAYSCALE)
24 low_res_image = low_res_image.astype('float32') / 255.0
25 low_res_image = np.expand_dims(low_res_image, axis=0)
26 low_res_image = np.expand_dims(low_res_image, axis=-1)
27
28 # Model ile Yüksek Çözünürlük Görüntü Oluşturma
29 predicted = model.predict(low_res_image)
30 predicted_image = predicted[0, :, :, 0] * 255.0
31 cv2.imwrite('high_res_image.jpg', predicted_image)

```

Şekil 1

Bu örnekte, SRCNN mimarisini kullanılarak düşük çözünürlüklü bir görüntünün yüksek çözünürlüğe çıkarılması gösterilmiştir. SRCNN, üç katmanlı bir konvolüsyonel sinir ağıdır:

- İlk Katman:** Giriş görüntüsünü alır ve 64 adet 9x9 boyutunda filtre uygular. Bu katman, temel özelliklerin çıkarılmasını sağlar.
- İkinci Katman:** 32 adet 1x1 boyutunda filtre kullanarak, önemli özellikleri daha da özelleştirir.
- Üçüncü Katman:** Son olarak, 1 adet 5x5 boyutunda filtre ile yüksek çözünürlüklü görüntüyü üretir.

Bu model, düşük çözünürlüklü görüntüyü girdi olarak alır, yüksek çözünürlüklü görüntüyü tahmin eder ve çıktıyı kaydeder.

Tarih ve İşyeri Amirinin İmzası

05/09/2024

Örnek 2:

Bir tıbbi görüntüleme sisteminde, MRI veya CT taramaları düşük çözünürlüklü olabilir. SRCNN kullanılarak, bu taramaların çözünürlüğü artırılarak doktorların hastalıkları daha kolay ve doğru bir şekilde teşhis etmesine yardımcı olunur.

SRCNN Kullanarak Medikal Görüntü İyileştirme

```

1 import cv2
2 import numpy as np
3 from tensorflow.keras.models import Sequential
4 from tensorflow.keras.layers import Conv2D
5
6 # SRCNN Modelini Tanımlama
7 def build_srcnn():
8     model = Sequential()
9     model.add(Conv2D(64, (9, 9), activation='relu', padding='same', input_shape=(None, None, 1)))
10    model.add(Conv2D(32, (1, 1), activation='relu', padding='same'))
11    model.add(Conv2D(1, (5, 5), activation='linear', padding='same'))
12    return model
13
14 # Modeli Oluşturma ve YÜKLEME
15 model = build_srcnn()
16 model.load_weights('srcnn_medical_weights.h5') # Önceden eğitilmiş ağırlıklar
17
18 # Medikal Görüntüyü YÜKLEME ve Ôn İşleme
19 low_res_medical = cv2.imread('low_res_medical.jpg', cv2.IMREAD_GRAYSCALE)
20 low_res_medical = low_res_medical.astype('float32') / 255.0
21 low_res_medical = np.expand_dims(low_res_medical, axis=0)
22 low_res_medical = np.expand_dims(low_res_medical, axis=-1)
23
24 # YÜKSEK ÇÖZÜNÜRLÜKLÜ Görüntü Tahmini
25 predicted_medical = model.predict(low_res_medical)
26 predicted_medical_image = predicted_medical[0, :, :, 0] * 255.0
27 cv2.imwrite('high_res_medical.jpg', predicted_medical_image)

```

Şekil 1

Bu örnekte, SRCNN kullanılarak medikal görüntülerin çözünürlüğünün artırılması ele alınmıştır. Medikal görüntüler, düşük çözünürlükteyken bile hastalıkların teşhisinde kritik öneme sahiptir. SRCNN modeli, bu görüntülerin kalitesini artırarak doktorların daha doğru teşhisler koymasına yardımcı olur.

Model, önceden eğitilmiş ağırlıklarla yüklenmiş olup, düşük çözünürlüklü medikal görüntü üzerinde tahmin yaparak yüksek çözünürlüklü görüntüyü oluşturur. Bu süreç, görüntüdeki detayların korunmasını ve netliğin artırılmasını sağlar.

Gün 3:

Python ile Yapay Zeka Eğitimi

Python, yapay zeka ve makine öğrenimi uygulamaları için en popüler programlama dillerinden biridir. Geniş bir kütüphane desteği sunar ve yapay zeka geliştirme sürecini hızlandırır. Python, veri analizi, model geliştirme ve eğitme süreçlerinde kullanılır.

Kullanım Alanları:

- **Doğal Dil İşleme:** Python, metinlerin işlenmesi ve anlamlandırılması için kullanılır. Örneğin, chatbotlar ve dil çevirisi sistemlerinde Python yaygın olarak kullanılır.
- **Veri Analizi:** Büyük veri setlerini işlemek ve analiz etmek için Python kütüphaneleri (NumPy, Pandas, Scikit-learn) kullanılır.

Örnek 1:

Bir veri analisti olarak büyük bir veri kümesi üzerinde çalışıyorsunuz. Python'ın Pandas ve NumPy kütüphaneleri kullanılarak, veriler üzerinde hızlı analizler yapabilir, veriyi temizleyebilir ve anlamlı sonuçlar elde edebilirsiniz.

Python ile Veri Analizi



Şekil 1

```

1 import pandas as pd
2 import numpy as np
3 import matplotlib.pyplot as plt
4
5 # Veri Setini Yükleme
6 data = pd.read_csv('sales_data.csv')
7
8 # Veriyi İnceleme
9 print(data.head())
10
11 # Eksik Verileri Kontrol Etme
12 print(data.isnull().sum())
13
14 # Basit Veri Temizleme
15 data = data.dropna()
16
17 # Satış Verilerinin Yıllara Göre Dağılımını Görselleştirme
18 sales_by_year = data.groupby('Year')['Sales'].sum()
19 sales_by_year.plot(kind='bar')
20 plt.title('Yıllara Göre Toplam Satış')
21 plt.xlabel('Yıl')
22 plt.ylabel('Satış Miktarı')
23 plt.show()

```

Şekil 1

Bu örnekte, Python'un Pandas kütüphanesi kullanılarak satış verilerinin analizi gerçekleştirilmiştir. İlk olarak, sales_data.csv dosyası yüklenir ve veri setinin ilk birkaç satırı incelenir. Eksik veriler kontrol edilerek temizlenir. Daha sonra, satış verileri yıllara göre gruplanarak toplam satış miktarları görselleştirilir. Bu tür analizler, işletmelerin satış trendlerini anlamalarına ve stratejik kararlar almalarına yardımcı olur.

Örnek 2:

Doğal dil işleme projesi yapıyorsunuz ve Python'un NLTK veya spaCy gibi kütüphanelerini kullanarak, dil modelleri eğitebilir ve metinleri analiz edebilirsiniz. Örneğin, metin sınıflandırma projelerinde Python kullanarak bir model geliştirebilirsiniz.

Python ile Doğal Dil İşleme (NLP)

```

● ● ●

1 import nltk
2 from nltk.corpus import stopwords
3 from nltk.tokenize import word_tokenize
4 from collections import Counter
5
6 # Gerekli NLTK Verilerini İndirme
7 nltk.download('punkt')
8 nltk.download('stopwords')
9
10 # Metin Verisini YÜKLEME
11 text = """
12 Yapay zeka, günümüz teknolojisinin en önemli alanlarından biridir. Doğal dil işleme,
13 makine öğrenimi ve derin öğrenme gibi alt dalları içerir.
14 """
15
16 # Metni Tokenize Etme
17 tokens = word_tokenize(text.lower())
18
19 # Durak Kelimeleri Kaldırma
20 stop_words = set(stopwords.words('turkish'))
21 filtered_tokens = [word for word in tokens if word.isalnum() and word not in stop_words]
22
23 # Kelime Frekansını Hesaplama
24 word_freq = Counter(filtered_tokens)
25 print(word_freq)
26
27 # En Yaygın 5 Kelimeyi Görselleştirme
28 common_words = word_freq.most_common(5)
29 words, counts = zip(*common_words)
30 plt.bar(words, counts)
31 plt.title('En Yaygın 5 Kelime')
32 plt.xlabel('Kelime')
33 plt.ylabel('Frekans')
34 plt.show()
35

```

Şekil 1

Bu örnekte, Python'un NLTK kütüphanesi kullanılarak basit bir doğal dil işleme (NLP) işlemi gerçekleştirilmiştir. Metin verisi ilk olarak tokenize edilir, yani kelimeye ayrılır. Daha sonra, Türkçe'deki durak kelimeler (önemsiz kelimeler) filtrelenir. Kalan kelimelerin frekansı hesaplanarak en yaygın kelimeler belirlenir ve görselleştirilir. Bu tür analizler, metin verisindeki anahtar kelimelerin belirlenmesi ve metinlerin daha iyi anlaşılmasına için kullanılır.

Gün 4 :

PyTorch ile Derin Öğrenme Modelleri

Tanım:

PyTorch, Facebook tarafından geliştirilen bir derin öğrenme kütüphanesidir. Esnek yapısı ve dinamik hesap grafiği desteği ile araştırmacılar ve mühendisler tarafından sıkılıkla tercih edilir. Derin öğrenme modellerini kolayca geliştirmek ve test etmek için kullanılır.

Kullanım Alanları:

- Doğal Dil İşleme:** PyTorch, dil modellerini eğitmek ve metin verisi üzerinde çalışmak için kullanılır.
- Görüntü Tanıma:** Görüntülerin sınıflandırılması ve analiz edilmesi için PyTorch kullanılarak derin öğrenme modelleri geliştirilebilir.

Örnek 1:

Bir konuşma tanıma projesi üzerinde çalışıyorsunuz. PyTorch ile dil modelleri eğiterek, sesli komutları tanıyan bir sistem geliştirebilirsiniz. Bu sistem, sesli asistanlar ve diğer konuşma tabanlı uygulamalar için kullanılabilir.

PyTorch ile Konuşma Tanıma Modeli

```

1 # Modeli ve Diğer Bileşenleri Tanımlama
2 input_size = 13 # MFCC Özellik sayısı
3 hidden_size = 128
4 num_layers = 2
5 num_classes = 10 # Örneğin 10 farklı komut
6
7 model = SpeechRecognitionModel(input_size, hidden_size, num_layers, num_classes)
8 criterion = nn.CrossEntropyLoss()
9 optimizer = optim.Adam(model.parameters(), lr=0.001)
10
11 # Veri Setini Yükleyin
12 train_files = ['audio1.wav', 'audio2.wav', 'audio3.wav']
13 train_labels = [0, 1, 2]
14 train_dataset = SpeechDataset(train_files, train_labels)
15 train_loader = DataLoader(train_dataset, batch_size=2, shuffle=True)
16
17 # Modeli Eğitme
18 for epoch in range(10):
19     for waveforms, labels in train_loader:
20         # MFCC Özelliklerini Çıkarın
21         mfcc = torchaudio.transforms.MFCC()(waveforms)
22         mfcc = mfcc.squeeze(1).transpose(1, 2) # (batch, time, features)
23
24         # İleri Geçiş
25         outputs = model(mfcc)
26         loss = criterion(outputs, labels)
27
28         # Geri Yayılım ve Optimizasyon
29         optimizer.zero_grad()
30         loss.backward()
31         optimizer.step()
32
33         print(f'Epoch [{epoch+1}/10], Loss: {loss.item():.4f}')
34
35 # Modeli Kaydetme
36 torch.save(model.state_dict(), 'speech_recognition_model.pth')
37

```

```

1 import torch
2 import torchnn as nn
3 import torch.optim as optim
4 from torch.utils.data import DataLoader, Dataset
5 import torchaudio
6
7 # Basit Bir Konuşma Tanıma Veri Seti Sınıfı
8 class SpeechDataset(Dataset):
9     def __init__(self, file_list, labels):
10         self.file_list = file_list
11         self.labels = labels
12
13     def __len__(self):
14         return len(self.file_list)
15
16     def __getitem__(self, idx):
17         waveform, sample_rate = torchaudio.load(self.file_list[idx])
18         label = self.labels[idx]
19         return waveform, label
20
21 # Basit Bir RNN Modeli
22 class SpeechRecognitionModel(nn.Module):
23     def __init__(self, input_size, hidden_size, num_layers, num_classes):
24         super(SpeechRecognitionModel, self).__init__()
25         self.rnn = nn.LSTM(input_size, hidden_size, num_layers, batch_first=True)
26         self.fc = nn.Linear(hidden_size, num_classes)
27
28     def forward(self, x):
29         # x: (batch, time, features)
30         out, _ = self.rnn(x)
31         out = out[:, -1, :] # Son zamanızındaki çıktıyi al
32         out = self.fc(out)
33
34     return out

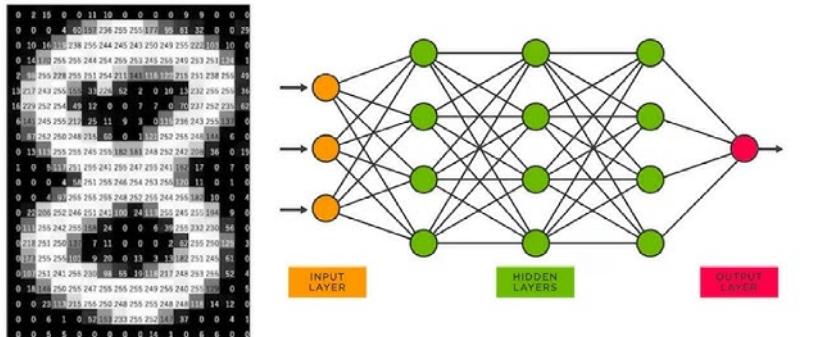
```

Şekil 1

Şekil 1

Bu örnekte, PyTorch kullanılarak basit bir konuşma tanıma modeli geliştirilmiştir. Model, ses dosyalarını alır, Mel Frekans Kepstral Katsayıları (MFCC) özelliklerini çıkarır ve bu özellikler üzerinden konuşulan komutu sınıflandırır. Model, LSTM tabanlı bir RNN (Recurrent Neural Network) kullanır ve son zaman adımındaki çıktıyı kullanarak komut sınıfını tahmin eder. Eğitim süreci boyunca, modelin kaybı (loss) minimize edilerek doğruluk artırılır.

Neural Networks



PyTorch

Örnek 2:

Bir medikal görüntü analizi projesi yapıyorsunuz. PyTorch ile MRI taramalarını analiz eden bir model geliştirerek, tümörlerin sınıflandırılmasını sağlayabilirsiniz.

PyTorch ile Medikal Görüntü Analizi

Şekil 1

Sekil 1

Bu örnekte, PyTorch kullanılarak medikal görüntü analizi için basit bir konvolüsyonel sinir ağısı (CNN) modeli geliştirilmiştir. Model, gri tonlamalı tıbbi görüntüler alır ve bu görüntüler üzerinden tümör varlığını sınıflandırır. Model iki konvolüsyon katmanı, iki max pooling katmanı ve iki yoğun katman içerir. Eğitim süreci boyunca, modelin kaybı minimize edilerek sınıflandırma doğruluğu artırılır. Bu tür modeller, medikal görüntülerin otomatik olarak analiz edilmesi ve hastalıkların erken tescisi için kullanılır.

Terms and Conditions Ekranı için Görev Uygulaması (Trello Task)

Bu görevde, mobil uygulama içinde kullanıcıya "Terms and Conditions" (Şartlar ve Koşullar) sayfasını göstermek için bir ekran geliştirilmesi amaçlandı. Yapılan uygulamanın hem kullanıcı deneyimi açısından uygun hem de teknik açıdan etkili bir çözüm sunması gerekiyordu. Bu bağlamda, uzun ve genellikle statik olan metinlerin kullanıcıya sunulmasında HTML formatı tercih edildi. JSON yerine HTML kullanılması, birçok teknik ve pratik avantaj sağlamlamaktadır. Bu tercihi, hem teknik hem de kullanım kolaylığı açısından ele alarak uzun ve akademik bir değerlendirme yapmak mümkündür.

Görev Kapsamı ve Teknik Detaylar

Flutter uygulamasında, kullanıcıya bir "Terms and Conditions" sayfası göstermek amacıyla bir yapı oluşturduk. Aşağıdaki işlev, kullanıcı bir sözleşme ekranına geçmek istediginde devreye girmektedir:

Bu fonksiyon, öncelikle cihazın bulunduğu dil ayarlarına göre uygun HTML dosyasını assets/terms/ klasöründen yükliyor. Dosya yüklemesi tamamlandıktan sonra, Flutter'in Navigator yapısı kullanılarak yeni bir sayfaya yönlendirme yapılıyor. Bu sayfa, yüklenen HTML içeriğini gösteren TermsScreen adlı widget ile temsil edilmektedir:

```

● ● ●

1 Future<void> _showTermsScreen(BuildContext context) async {
2   String htmlContent = await rootBundle.loadString(
3     'assets/terms/terms_conditions_${Get.locale?.languageCode.toString()}.html');
4
5   Navigator.push(
6     context,
7     MaterialPageRoute(
8       builder: (context) => TermsScreen(htmlContent),
9     ),
10    );
11 }

```

Şekil 1

Bu yapı, Flutter'da yaygın kullanılan bir tasarımdır. Sayfa, bir Scaffold widget ile düzenlenmiş ve uygulamanın üst bölümünde bir AppBar ile başlık gösterilmiştir. Sayfanın ana gövdesi, bir SingleChildScrollView içine yerleştirilmiş Html widget'ı kullanılarak HTML içeriğini göstermektedir.

```

● ● ●

1 class TermsScreen extends StatelessWidget {
2   final String htmlContent;
3
4   TermsScreen(this.htmlContent);
5
6   @override
7   Widget build(BuildContext context) {
8     return Scaffold(
9       appBar: AppBar(
10         title: Text("terms_conditions".tr),
11       ),
12       body: SingleChildScrollView(
13         child: Html(data: htmlContent),
14       ),
15     );
16   }
17 }

```

Şekil 1

Bu HTML dosyası, Flutter uygulamasında sabit metinleri göstermek için kullanılıyor ve kullanıcıların uygulamayı kullanırken kabul etmeleri gereken şartları ve koşulları içeriyor. Metin HTML formatında olduğu için, dil değişikliklerinde yalnızca ilgili HTML dosyasının yüklenmesi yeterli oluyor.

Neden HTML Tercih Edildi?

Bu görevde, "Terms and Conditions" metni uzun ve statik bir içerik olduğundan, JSON yerine HTML formatında sunulması tercih edildi. Bunun temel nedenleri şunlardır:

- Statik İçerikler için Uygunluk:** Şartlar ve koşullar gibi uzun ve nadiren değişen içerikler için HTML formatı oldukça uygundur. Bu metinler genellikle sabittir ve çok sık güncellenmez. HTML dosyaları, uzun metinleri ve yapısal düzenlemeleri (başlıklar, paragraflar, listeler vb.) kolayca yönetmek için mükemmelidir.

- Daha Az Yükleme Süresi ve Hafiza**

Kullanımı: JSON dosyaları, özellikle uzun metinlerde boyut açısından dezavantajlı hale gelebilir. Büyük JSON dosyalarının

yüklenmesi ve işlenmesi, mobil cihazlarda daha fazla bellek ve işlem gücü gerektirebilir. Ayrıca, JSON formatında bu kadar uzun metinlerin tutulması veri yapılarının karmaşıklamasına ve dosya boyutlarının büyümesine neden olabilir. HTML ise doğrudan tarayıcılar ve uygulama içi tarayıcılar (WebView gibi) tarafından desteklenen bir format olduğundan, daha hızlı ve hafif bir çözüm sunar.

- HTML'nin Formatlama Avantajı:** HTML, metinleri biçimlendirme ve yapılandırma açısından çok daha esnektir. Başlıklar, paragraflar, listeler ve hatta stil dosyaları (CSS) ile zenginleştirilmiş metin sunma imkanı sunar. JSON, bu tür yapısal düzenlemeler için uygun değildir. JSON formatında bu kadar detaylı biçimlendirme yapmak için özel işleyiciler geliştirmek ya da verileri daha karmaşık hale getirmek gerekecektir.
- Dil Desteği ve Yerelleştirme Kolaylığı:** Bu projede yerelleştirme (localization) önemli bir rol oynamaktadır. Uygulamanın farklı dillerde hizmet vermesi gereği için, `terms_conditions_{languageCode}.html` gibi bir dosya yapılandırması ile dil desteği sağlanmıştır. JSON formatında bu yapılandırmayı yönetmek daha zor olabilir. HTML'de ise her dil için ayrı bir dosya hazırlayarak dil desteğini sağlamak hem daha kolay hem de daha düzenlidir.
- Bakım Kolaylığı:** HTML formatı, uygulamanın güncellenmesi sırasında bakım açısından da daha kolaydır. Yeni bir şartlar ve koşullar güncellemesi olduğunda, yalnızca ilgili HTML dosyasını değiştirip uygulamaya entegre etmek yeterlidir. Bu dosyalar uygulama kaynakları (`assets/`) içerisinde tutulduğundan, dış bağımlılıklar olmadan uygulama güncellemeleri yapılabilir.

GetX ve Localization ile Entegrasyon

Görev kapsamında kullanılan GetX mimarisi, Flutter uygulamalarında durumu (state) yönetme ve yerelleştirme (localization) gibi işlemleri oldukça basitleştirir.

`terms_conditions_{Get.locale?.languageCode.toString()}.html` ifadesi, uygulamanın dil ayarlarına göre dinamik olarak doğru HTML dosyasını yüklemektedir. GetX ile yerelleştirme işlemi, dil dosyalarının otomatik olarak yüklenmesi ve uygulamaya entegre edilmesi açısından büyük bir avantaj sağlamaktadır.

Uygulama, kullanıcının cihazında hangi dil ayarları mevcutsa, o dile göre ilgili HTML dosyasını seçerek, kullanıcıya diline uygun bir şartlar ve koşullar metni sunmaktadır. Bu da kullanıcı deneyimini iyileştiren önemli bir faktördür.

Projenin bu aşamasında, şartlar ve koşullar metninin HTML formatında sunulması, teknik açıdan birçok avantaj sağlamaktadır. Uzun ve sabit metinlerin JSON yerine HTML ile yönetilmesi, uygulamanın performansını artırdığı gibi bakım kolaylığı ve dil desteği açısından da önemli avantajlar sunmaktadır. GetX ile yapılan yerelleştirme işlemi sayesinde kullanıcılar, kendi dillerinde hizmet alabilirken, uygulama içerik yönetimi basitleştirilmiş ve daha az kaynak kullanımı sağlanmıştır.

```

1  <!DOCTYPE html>
2  <html>
3  <head>
4      <title>Terms and Conditions</title>
5  </head>
6  <body>
7      <h1>Şartlar ve Koşullar</h1>
8      <p>Bu uygulamayı kullanarak aşağıdaki hüküm ve koşulları kabul etmiş olursunuz:</p>
9      <h2>1. Giriş</h2>
10     <p>Bu, hükmü ve koşullarınızın giriş bölümündür...</p>
11     <h2>2. Kullanıcı Sorumlulukları</h2>
12     <p>Bir kullanıcı olarak bu kurallara uymanız beklenir...</p>
13     ...
14  </body>
15  </html>

```

Customer ve Ticket Widget Yapılarının Tasarımı ve Uygulanması

1. Customer için Yeni Klasör Yapısının Oluşturulması:

Bu günü förevde, projede "Customer" ile ilgili widget ve bileşenlerin düzenli ve erişilebilir bir yapıya sahip olması için "View/Widgets" klasörü altında "customer" adında yeni bir klasör oluşturduk. Bu yapılanma, projede sürdürülebilirlik ve bakım açısından önemlidir, çünkü proje büyükçe her bileşen veya modülün ayrı bir klasörde saklanması, kodun anlaşılabilirliğini ve yönetilebilirliğini artırır. "Customer" ile ilgili tüm widget'lerin bu klasör altında toplanması, projenin modülerliğini sağlar ve bir takım çalışmasında geliştiriciler arasında uyumu kolaylaştırır. İyi organize edilmiş bir dosya yapısı, karmaşık projelerde hata tespitini hızlandırır ve yeni özelliklerin eklenmesini kolaylaştırır.



2. Ticket Görünümleri: Liste, Tablo ve Takvim Widget'lerinin Geliştirilmesi:

Projede "ticket" yapılarının çeşitli görünümlerle sunulması gerekiyordu. Ticket'lar hakkında bilgilerin farklı görünümlerde kullanıcıya sunulması, kullanıcının ihtiyaçlarına göre esnek bir arayüz sağlamaktadır. Bu doğrultuda üç temel widget tasarladım: List View (Liste Görünümü), Table View (Tablo Görünümü) ve Calendar View (Takvim Görünümü). Her bir widget, ticket'ların farklı düzenlerde görüntülenmesi ve kullanıcı etkileşimlerini desteklemek amacıyla özel olarak geliştirildi.

2.1 Ticket List View:

Dosya Adı: ticket_list_view.dart

Ticket'ların kart yapılarıyla listelendiği bu görünüm, kullanıcılarla ticket'lar hakkında hızlı bilgi edinme fırsatı sunar. Her bir kart, ticket'in adı, durumu ve atandığı kullanıcı gibi yüzeysel bilgileri içermektedir. Bu sayede kullanıcılar ticket'lar arasında hızlıca gezinebilir. Kartlara tıklandığında ise detay sayfasına yönlendirme yapılır. Liste görünümü, özellikle çok sayıda ticket'i aynı anda görmek isteyen kullanıcılar

```
1  class TicketCalendarViewState extends State<TicketCalendarView> {
2      DateTime _focusedDay = DateTime.now();
3      DateTime? _selectedDay;
4
5      @override
6      Widget build(BuildContext context) {
7          return Column(
8              children: [
9                  TableCalendar(
10                      focusedDay: _focusedDay,
11                      selectedDayPredicate: (day) => isSameDay(_selectedDay, day),
12                      onDaySelected: (selectedDay, focusedDay) {
13                          setState(() {
14                              _selectedDay = selectedDay;
15                              _focusedDay = focusedDay;
16                          });
17                      },
18                      eventLoader: _getEventsForDay,
19                  ),
20                  ..._getEventsForDay(_selectedDay!).map((ticket) => ListTile(
21                      title: Text(ticket.name),
22                      subtitle: Text(ticket.project),
23                      onTap: () {
24                          Get.to(TicketDetailScreen(ticket: ticket));
25                      },
26                  )));
27              );
28          List<Ticket> _getEventsForDay(DateTime day) {
29              return widget.tickets.where((ticket) => isSameDay(ticket.date, day)).toList();
30          }
31          Icon getStatusIcon(TicketStatus status) {
32              // Renk duruma göre belirlenir
33              return Icon(Icons.circle, color: getStatusColor(status));
34          }
35      }
36  }
```

Şekil 1

İçin uygundur. Kod yapısında, `TicketInfoScreen` sınıfı içerisinde bu liste widget'ını çağırarak, ticket'ların kart şeklinde listelenmesini sağladım. Bu yapı, liste şeklinde bilgileri sunmanın etkili ve kullanıcı dostu bir yoludur. Kart yapısının esnekliği, daha fazla bilgi eklenmesini veya düzenlemeler yapılmasını da kolaylaştırır.

2.2 Ticket Table View:

Dosya Adı: ticket_table_view.dart

Tablo görünümü, ticket'ların daha detaylı bir şekilde, sütunlar halinde gösterilmesini sağlar. Kullanıcılar, tablo yapısında ticket'ların durumunu, ID'sini, bilet adını, projeyi, oluşturma ve atanın kişiyi görebilirler. Tablo yapısının en önemli avantajı, çok sayıda veriyi düzenli bir şekilde sunabilmesidir. Kullanıcılar bu veriler arasında kolayca sıralama yapabilir ve sütuna tıklayarak ticket'ları farklı kriterlere göre filtreleyebilirler.

Bu yapının uygulanmasında `DataTable` widget'ını kullandım. `DataTable`, sütunlar ve satırlar halinde veriyi gösterebilen bir yapısı sunar. Burada `onSort` metodu ile kullanıcıların ticket'ları istedikleri kriterlere göre sıralamalarını sağladım. Örneğin, `ticket.name` sütunu üzerine tıklandığında, ticket'lar isimlerine göre sıralanır. `TicketDetail` sayfasına geçiş, satırlara tıklanarak sağlanır. Bu sayede kullanıcılar, tablo üzerinde gezindikten sonra ticket detaylarına ulaşabilirler.

```

1 class TicketTableViewState extends State<TicketTableView> {
2   int? _sortColumnIndex;
3   bool _sortAscending = true;
4   @override
5   Widget build(BuildContext context) {
6     return DataTable(
7       sortColumnIndex: _sortColumnIndex,
8       sortAscending: _sortAscending,
9       columns: [
10         DataColumn(label: Text('Durum'), onSort: _onSort),
11         DataColumn(label: Text('ID'), onSort: _onSort),
12         DataColumn(label: Text('Bilet Adı'), onSort: _onSort),
13         DataColumn(label: Text('Proje'), onSort: _onSort),
14         DataColumn(label: Text('Oluşturma Tarihi'), onSort: _onSort),
15         DataColumn(label: Text('Atanan Kişi'), onSort: _onSort),
16       ],
17       rows: widget.tickets.map((ticket) {
18         return DataRow(
19           cells: [
20             DataCell(Icon(getStatusIcon(ticket.status))),
21             DataCell(Text(ticket.id.toString())),
22             DataCell(Text(ticket.name), DataCellText(ticket.project)),
23             DataCell(Text(ticket.creationDate)),
24             DataCell(Text(ticket.assignedUser)),
25             onSelectChanged: (bool? selected) {
26               if (selected == true) {
27                 Get.to(TicketDetailScreen(ticket: ticket));
28               }
29             },
30           ).toList(),
31         );
32       }).toList(),
33       void onSort(int columnIndex, bool ascending) {
34         setState(() {
35           _sortColumnIndex = columnIndex;
36           _sortAscending = ascending;
37         });
38       });
39     }
40   }
41 }
```

tarihini
ilgili
tablo

Şekil 1

Tablo yapısının bir diğer avantajı, veri analizi ve karşılaştırmalarının hızlı yapılabilmesidir. Kullanıcılar, özellikle büyük veri setlerinde tablo görünümünü tercih eder, çünkü tablo, verilerin daha kompakt bir biçimde sunulmasına olanak tanır.

2.3 Ticket Calendar View:

Dosya Adı: ticket_calendar_view.dart

Takvim görünümü, ticket'ların tarih bazlı gösterilmesi gereken senaryolarda etkili bir çözümdür. Bu görünüm, ticket'ların belirli bir gün veya tarihler aralığında gösterilmesini sağlar. `TableCalendar` kütüphanesini kullanarak, tarih seçimi ve o tarihe ait ticket'ların gösterilmesi işlemlerini gerçekleştirdim.

Bu görünümde kullanıcılar, belirli bir güne tıkladıklarında o güne ait ticket'ları alt kısmda görebilecekleri bir listeye yönlendiriliyor. `TicketDetailScreen` sayfasına geçiş bu listeden bir ticket'e tıklanarak sağlanır. Aynı zamanda takvim üzerinde ticket'ların sayısına göre küçük durum daireleri gösterilir. Bu dairelerin renkleri, ticket'ların durumuna göre `getStatusColor` fonksiyonu ile belirlenir. Kullanıcılar, takvim üzerinde yoğunluğu bu renklerle takip edebilir.

Takvim görünümü, zaman bazlı projelerde ticket'ların takip edilmesi açısından önemlidir. Kullanıcılar belirli bir tarihte hangi ticket'ların oluşturulduğunu veya tamamlandığını bu takvim yapısıyla kolayca görebilirler. Ayrıca bu yapı, proje yönetimi süreçlerinde zaman çizelgelerinin yönetilmesi ve görevlerin takip edilmesi açısından da oldukça faydalıdır.

Bu görev, projeye modüler ve esnek bir kullanıcı arayüzü eklemek için önemli bir adımdı. "Customer" bileşenlerinin yeni bir klasörde düzenlenmesi, kodun yapısal düzenini iyileştirirken; ticket'ların farklı görünümle sunulması, kullanıcı deneyimini geliştirdi. Liste, tablo ve takvim gibi çeşitli görünüm seçenekleri ile kullanıcıların ticket'ları farklı şekillerde görmesi sağlandı. Bu çeşitlilik, projenin kullanıcı odaklı bir yapıya kavumasına katkı sağladı.

Gün 7 :

Ticket Yönetiminde Arama ve Filtreleme Fonksiyonlarının Eklenmesi ve Ticket Detay Ekranının Geliştirilmesi

Stajımın 7. gününde, mevcut Flutter projemde müşteri biletlerini (ticket) yönetmek için geliştirdiğimiz arayüzlerde kullanıcı deneyimini iyileştirmek amacıyla arama vefiltreleme fonksiyonlarını entegre ettim. Ayrıca, her bir biletin detaylı bilgilerini görüntülemek için yeni bir ekran olan `TicketDetailScreen`'i tasarladım ve kodladım. Bu süreçte, projenin mevcut mimarisi ve kullanılan teknolojiler göz önünde bulundurularak, kullanıcı dostu ve işlevsel bir yapı oluşturmayı hedefledim.

1. Arama Fonksiyonunun Eklenmesi

Görev Tanımı: Denetleyicim tarafından yapılan geri bildirimlere dayanarak, ticket'lar üzerinde etkili bir arama yapabilmek için `TicketInfoScreen`'e bir arama butonu eklemem gerekti. Bu özellik, kullanıcıların belirli kriterlere göre ticket'ları hızlıca bulmalarını sağlamak amacıyla geliştirilmiştir.

Uygulama Adımları:

1. Arama Butonunun Eklenmesi:

- `TicketInfoScreen` içinde bir arama butonu (`IconButton`) ekledim.
- Bu butona tıklandığında, arama çubuğu açıp kapatmak için bir `TextField` gösterilecek şekilde yapılandırdım.

2. Arama Metni Kontrolü:

- `_searchController` adlı bir `TextEditingController` kullanarak, kullanıcının girdiği metni kontrol ettim.
- `_isSearching` adında bir boolean değişken ile arama modunun aktif olup olmadığını takip ettim.

3. Arama İşlevinin Oluşturulması:

- `searchFilter` fonksiyonu ile kullanıcının girdiği metni ticket isimleriyle karşılaştırarak, arama sonuçlarını filtreledim.

Bu fonksiyon, liste, tablo ve takvim görüntülerine parametre olarak gönderilerek, her bir görünümde arama sonuçlarının doğru şekilde gösterilmesini sağladı.

```
1 //Arama ve Filtreleme Seçenekleri
2 Row(children: [
3   IconButton(icon: Icon(Icons.search),
4   onPressed: () => setState(() => _isSearching = !_isSearching)),
5   Expanded(child: _isSearching ? TextField(controller: _searchController) : Text('my_tickets'.tr)),
6   PopupMenuButton(items: _statuses.map((status) => CheckedPopupMenuItem(child: Text(status.tr),
7   value: status)).toList(),
8 ])
```

Şekil 1

- **IconButton:** Kullanıcının arama modunu açıp kapatmasını sağlar. Butona tıklandığında `_isSearching` değişkeni güncellenir ve arama çubuğu açılır veya kapanır.
- **TextField:** Arama modunda, kullanıcının arama metnini girmesi için bir metin alanı sağlar. `onChanged` metodu, her metin değişikliğinde `searchFilter` fonksiyonunu çağırarak dinamik arama yapar.
- **PopupMenuButton:** Kullanıcıların ticket'ları belirli kriterlere (örneğin, durum veya proje türüne göre) göre filtrelemelerini sağlar.

2. Filtreleme Fonksiyonunun Geliştirilmesi

Görev Tanımı: Ticket'ları proje türüne ve durumuna göre filtreleyebilmek için mevcut filtreleme mekanizmasını geliştirmem gerekti. Bu sayede kullanıcılar, ilgilendikleri spesifik ticket'ları daha kolay bulabilirler.

Uygulama Adımları:

1. Filtreleme Parametrelerinin Belirlenmesi:

- activeFilter değişkeni ile ticket durumlarını kontrol ettim ve filtreleme işlemini bu değişken üzerinden gerçekleştirdim.
- Filtreleme seçeneklerini kullanıcıya sunmak için PopupMenuItem içerisinde durumlara göre seçenekler oluştururdum.

2. Filtreleme İşlevinin Oluşturulması:

- filterTickets fonksiyonu, seçilen filtre kriterlerine göre ticket listesini günceller.
- Bu fonksiyon, hem liste hem de tablo ve takvim görüntülerinde ticket'ların doğru şekilde filtrelenmesini sağlar.

```

1  Widget _buildInfoRow(String label, String value, BuildContext context) {
2      return Padding(
3          padding: const EdgeInsets.symmetric(vertical: 8.0),
4          child: Row(
5              crossAxisAlignment: CrossAxisAlignment.start,
6              children: [
7                  SizedBox(
8                      width: 120,
9                      child: Text('$label',
10                         style: Theme.of(context)
11                             .textTheme
12                             .titleMedium
13                             ?.copyWith(fontWeight: FontWeight.bold)),
14                     ),
15                     Expanded(
16                         child: Text(value.toString()),
17                         style: Theme.of(context).textTheme.bodyLarge),
18                     ],
19                 );
20             );
21         }
22     }
23
24     Widget _buildStatusRow(String label, String value, BuildContext context) {
25         return Padding(
26             padding: const EdgeInsets.symmetric(vertical: 8.0),
27             child: Row(
28                 crossAxisAlignment: CrossAxisAlignment.start,
29                 children: [
30                     SizedBox(
31                         width: 120,
32                         child: Text('$label',
33                             style: Theme.of(context)
34                             .textTheme
35                             .titleMedium
36                             ?.copyWith(fontWeight: FontWeight.bold)),
37                     ),
38                     Expanded(
39                         child: Row(
40                             children: [
41                                 Container(
42                                     width: 12,
43                                     height: 12,
44                                     decoration: BoxDecoration(
45                                         shape: BoxShape.circle,
46                                         color: _getStatusColor(value),
47                                     ),
48                                 ),
49                                 const SizedBox(width: 8),
50                                 Text(value.tr, style: Theme.of(context).textTheme.bodyLarge),
51                             ],
52                         ),
53                     ],
54                 );
55             );
56         }
57     }
58 }
```

Şekil 1

- **filterTickets:** Kullanıcının seçtiği durum kriterine göre ticket listesini günceller. originalTickets, tüm ticket'ların saklandığı orijinal listeyi temsil ederken, widget.tickets ise görüntülenen ticket'ların listesini günceller.

3. TicketDetailScreen Sınıfının Tasarımı ve Kodlanması

Görev Tanımı: Her bir ticket'in detaylı bilgilerini göstermek için TicketDetailScreen adlı yeni bir ekran oluşturmak. Bu ekran, kullanıcının ticket'i tıklamasıyla açılacak ve ticket'in tüm önemli bilgilerini gösterecek.

Uygulama Adımları:

1. TicketDetailScreen Yapısının Oluşturulması:

- Bu sınıf, her bir ticket'in detaylarını göstermek üzere tasarlandı.
- Ticket bilgilerini key-value yapısı şeklinde düzenleyerek, kullanıcının kolayca anlayabileceği bir formatta sundum.

2. Bilgi Satırlarının Oluşturulması:

- `_buildInfoRow` ve `_buildStatusRow` yöntemlerini kullanarak, ticket bilgilerini ve durumunu görsel olarak daha anlaşılır hale getirdim.
- Durum bilgisi için, `getStatusBarColor` fonksiyonu ile renkli çemberler oluşturarak, kullanıcının ticket durumunu hızlıca fark etmesini sağladım.

```

1  class TicketDetailScreen extends StatelessWidget {
2      final Map<String, String> ticket;
3      const TicketDetailScreen({super.key, required this.ticket});
4      @override
5      Widget build(BuildContext context) {
6          final screenSize = MediaQuery.of(context).size;
7          final isTablet = screenSize.width > 600;
8          return Scaffold(
9              appBar: AppBar(
10                  title: Text('${ticket['tr']} ${ticket['id']}'),
11              ),
12              body: Center(
13                  child: Card(
14                      elevation: 4,
15                      margin: EdgeInsets.all(isTablet ? 32.0 : 16.0),
16                      child: Padding(
17                          padding: EdgeInsets.all(isTablet ? 32.0 : 16.0),
18                          child: SizedBox(
19                              width: isTablet ? screenSize.width * 0.7 : screenSize.width,
20                              child: Column(
21                                  crossAxisAlignment: CrossAxisAlignment.start,
22                                  children: [
23                                      Text(ticket['name']!.tr,
24                                          style: Theme.of(context).textTheme.headlineSmall),
25                                      SizedBox(height: isTablet ? 24 : 16),
26                                      _buildInfoRow('type', ticket['type']!.tr, context),
27                                      _buildInfoRow('project', ticket['project']!.tr, context),
28                                      _buildInfoRow(
29                                          'creation_date', ticket['createdAt']!, context),
30                                      _buildStatusRow('status', ticket['status']!, context),
31                                      _buildInfoRow(
32                                          'created_by', ticket['createdBy']!.tr, context),
33                                      ],
34                                  ),
35                              ),
36                          ),
37                      ),
38                  );
39              }
40 }
```

Müşteri Çekmecesi Tasarımı (Customer Drawer)

İlk olarak, müşteri tarafı için bir çekmece (drawer) tasarladım. Bu çekmecede kullanıcıya hızlı erişim sağlayacak üç ana etiket ekledim:

- **Home Page** (Ana Sayfa)
- **My Tickets** (Biletlerim)
- **Settings** (Ayarlar)

Her etiket için uygun ikonlar tasarıma yerleştirildi. Çekmecenin rengi primary olarak ayarlandı ve yazılar beyaz renkte seçildi, böylece kontrast artırılarak daha okunabilir hale getirildi. Bu çekmece sayesinde kullanıcılar istedikleri sayfalara hızlıca erişebilecek.

Bilet Oluşturma Sayfası (Create Ticket Screen) Bir sonraki aşamada, **Create Ticket** (Bilet Oluştur) sayfasını tasarladım. Bu sayfada, kullanıcıların bilet oluşturmaya olanak sağlayacak bir form oluştururdum. Formda üç ana alan bulunuyor:

- **Bilet Adı**
- **Proje**
- **Açıklama**

Her bir alan için kullanıcıların veri girebileceği text field'lar (metin kutuları) tasarladım. Özellikle açıklama kısmını için metin düzenleme seçenekleri ekledim:

- **Kalm (Bold)**
- **İtalik (Italic)**
- **Altı Çizili (Underline)**

```
● ● ●  
1 void _applyTextStyle(String style) {  
2     setState(() {  
3         switch (style) {  
4             case 'bold':  
5                 _fontWeight = _fontWeight == FontWeight.bold ? FontWeight.normal : FontWeight.bold;  
6                 break;  
7             case 'italic':  
8                 _fontStyle = _fontStyle == FontStyle.italic ? FontStyle.normal : FontStyle.italic;  
9                 break;  
10            case 'underline':  
11                _textDecoration = _textDecoration == TextDecoration.underline ? TextDecoration.none : TextDecoration.underline;  
12                break;  
13        }  
14    });  
15 }
```

Bu düzenleme seçenekleri, açıklama kısmının zenginleştirilmesini sağlıyor. Ek olarak, kullanıcıların dosya eklemesi için bir **dashed button** (kesik çizgili buton) yerleştirdim. Son olarak, formun altına **Bileti Oluştur** (Create Ticket) butonunu ekledim ve bu buton tasarımda **primary** renkte görünecek şekilde ayarlandı.

Kodun Önemli Kısımları:

- **_applyTextStyle()** fonksiyonu: Bu fonksiyon, kullanıcının açıklama kısmında metin formatını değiştirmesine olanak tanır. Örneğin, kullanıcı kalın yazmak için 'bold' ikonuna tıkladığında, yazı stili **FontWeight.bold** olarak ayarlanır. Aynı işlem italik ve altı çizili metinler için de geçerlidir.

- **_buildAttachmentButton()** fonksiyonu: Kullanıcıların biletlerine dosya eklemelerine izin veren buton tasarımını sağlar. Dosya ekleme işlemi için **FilePicker** kütüphanesini kullanıyorum. Butonun stili, tema rengine göre otomatik olarak değişir (kararlı mod veya açık mod).

```

1 Widget _buildAttachmentButton(...) {
2   return ElevatedButton(
3     onPressed: () async {
4       FilePickerResult? result = await FilePicker.platform.pickFiles();
5       if (result != null) {
6         // Seçilen dosya işleme kodu
7       }
8     },
9     style: ElevatedButton.styleFrom(
10       padding: EdgeInsets.zero,
11       shape: RoundedRectangleBorder(
12         borderRadius: BorderRadius.circular(8),
13         side: BorderSide(
14           color: isDarkMode ? Colors.white24 : Colors.white,
15           width: 2,
16           style: BorderStyle.none,
17         ),
18       ),
19       backgroundColor: isDarkMode ? Colors.grey[800] : Colors.white,
20     ),
21     child: ... // Tasarım detayları
22   );
23 }
...

```

Şekil 1

- **_createTicket()** fonksiyonu: Kullanıcı formu doldurduktan sonra 'Bileti Oluştur' butonuna basıldığında çalışan fonksiyondur. Bu fonksiyon önce bir **Snackbar** ile başarılı bir mesaj gösterir ve ardından kullanıcının bilet listesini görebileceği **TicketsInfoScreen** sayfasına yönlendirir.

```

1 void _createTicket() {
2   Get.snackbar(
3     'Success',
4     'ticket_created_successfully'.tr,
5     snackPosition: SnackPosition.BOTTOM,
6     backgroundColor: Colors.green,
7     colorText: Colors.white,
8     duration: const Duration(seconds: 2),
9   );
10 Future.delayed(const Duration(seconds: 2), () {
11   Get.offAll(() => const TicketsInfoScreen());
12 });
13 }

```

Şekil 1

Bilet Bilgileri Sayfası (Tickets Info Screen) Ardından, biletlerin kullanıcıya farklı görünümle sunulacağı **Tickets Info** (Bilet Bilgileri) sayfasını tasarladım. Bu sayfada biletler üç farklı şekilde görüntülenebilecek:

1. **Liste Görünümü**: Kartlar içinde biletlerin kısa özeti bilgilerini gösterecek.
2. **Tablo Görünümü**: Bilet bilgilerini satırlar ve sütunlar şeklinde düzenledim, yana kaydırma özelliği ile daha fazla bilgi görüntülenebilir hale getirildi.
3. **Takvim Görünümü**: Biletlerin durumlarını, renkli göstergelerle birlikte takvim günlerine yerleştirilen bir tasarımla sundum.

Bu sayede kullanıcılar biletlerine farklı açılardan bakabilecek ve ihtiyaçlarına uygun görünümü seçebilecek.

Bilet Detay Sayfası (Ticket Detail Screen) Son olarak, **Ticket Detail** (Bilet Detay) sayfasını tasarladım. Bu sayfada bir biletin tüm detayları kullanıcıya sunuluyor. Biletin durumu (açık, kapalı, ilerlemeye) gibi bilgileri renkli göstergelerle öne çıkardım. Ayrıca, biletin açıklaması, oluşturulma tarihi, ilgili proje gibi detaylar da bu sayfada yer almaktır.

Önemli Kod Kısımları:

1. **getStatusColor()** fonksiyonu: Biletin durumuna göre renk seçimi yapar. Örneğin, 'open' (açık) olan biletler yeşil, 'closed' (kapalı) olanlar gri renkte gösterilir.

```

1 Color _getStatusColor(String status) {
2   switch (status.toLowerCase()) {
3     case 'open':
4       return const Color.fromARGB(255, 0, 255, 0);
5     case 'closed':
6       return const Color.fromARGB(255, 68, 68, 68);
7     case 'in_progress':
8       return const Color.fromARGB(255, 0, 135, 245);
9     default:
10       return Colors.grey;
11   }
12 }

```

Şekil 1

Bu tasarımlar ile kullanıcı deneyimini artırarak, bilet oluşturma ve takip işlemlerinin daha kolay yapılmasını sağladım. Tüm bu sayfalar ve fonksiyonlar projeyi geliştiren ekip arkadaşımıza sunuldu ve onay aldım.

Gün 9 :

Firebase Entegrasyonu ve Giriş Sayfası Oluşturulması

Bugün stajimin onuncu günüydü ve üzerinde çalıştığım projede önemli bir aşamaya geldim. Projemin giriş (login) sayfasını oluştururdum ve Firebase ile entegre ederek kullanıcı kimlik doğrulama işlemlerini başlattım. Bu süreç oldukça detaylıydı ve her adımı dikkatle gerçekleştirmem gerekti. Aşağıda izlediğim adımları detaylı bir şekilde açıklayacağım.

Firebase Console'da Yeni Proje Oluşturma

İlk adım olarak, şirketin Flutter ile ilişkili e-posta adresini kullanarak Firebase Console üzerinden "SUREM" adında yeni bir proje oluştururdum. Firebase Authentication hizmetini kullanarak uygulamaya e-posta tabanlı kimlik doğrulama eklemem gereği için, e-posta sağlayıcısını (email provider) etkinleştirdim. Bu sayede kullanıcıların e-posta ve şifre ile giriş yapmasını sağlayacak yapıyı kurmuş oldum.

Firebase CLI Kurulumu ve Projeyi Firebase'e Bağlama

Firebase'i projeme bağlayabilmek için öncelikle Firebase CLI'yi (Command Line Interface) kurdum. Komut satırında şu adımları izledim:

1. Firebase CLI Kurulumu:

Bu komutla Firebase araçlarını bilgisayara kurdum.



2. Firebase Giriş:

Firebase'de oturum açarak projemi Firebase'e bağlamaya hazır hale getirdim.

```
1 npm install -g firebase-tools
2 firebase login
3 firebase init
```

3. Projeyi Firebase'e Bağlama:

Şekil 1

Bu adımda, Firebase ile Flutter projemi birbirine bağlayarak Firebase hizmetlerinin projede kullanılmasını sağladım.

Flutter Projesine Firebase Entegrasyonu



Firebase'i Flutter projemde kullanabilmek için bazı kütüphaneleri yüklemem gerekiyordu. Bunun için `pubspec.yaml` dosyasına şu bağımlılıkları ekledim:

```
1 dependencies:
2   firebase_core: latest_version
3   firebase_auth: latest_version
4   cloud_firestore: latest_version
```

Şekil 1

Bu kütüphaneler, Firebase'in temel fonksiyonları ile kimlik doğrulama ve veri tabanı gibi hizmetlerini kullanmamı sağladı.

Firebase Projesiyle Bağlantı Kurulması

Firebase ile Flutter projemin entegrasyonunu tamamlamak için proje kök dizinimde şu komutu çalıştırıldım:

[flutterfire configure](#)

Tarih ve İşyeri Amirinin İmzası	13/09/2024
---------------------------------	------------

Bu komut, Firebase projemi Flutter'a entegre etti ve gerekli ayar dosyalarını oluşturdu. Firebase Console üzerinden Android ve iOS platformları için uygulamayı tanımladım ve gerekli konfigürasyon dosyalarını (google-services.json ve GoogleService-Info.plist) indirerek projeme ekledim.

Giriş (Login) Sayfasının Oluşturulması

Firebase Authentication ile kullanıcı girişlerini yönetebilmek için Flutter'da bir login sayfası oluşturdum. Flutter projemde GetX kullandığım için bir GetX controller sınıfı ile kimlik doğrulama işlemlerini kontrol ettim. FirebaseAuth kütüphanesini kullanarak, kullanıcıların e-posta ve şifre bilgilerini doğruladım.

Kullanıcıdan email ve password bilgilerini almak için TextEditingController sınıflarını kullandım ve kimlik doğrulama işlemini aşağıdaki kod ile gerçekleştirdim:



```
1 UserCredential userCredential = await FirebaseAuth.instance.signInWithEmailAndPassword(  
2   email: emailController.text,  
3   password: passwordController.text,  
4 );
```

Şekil 1

Bu işlemin başarılı olup olmadığını kontrol etmek için FirebaseAuthException ile hataları yönettim ve kullanıcının başarılı bir şekilde giriş yapması durumunda anasayfaya yönlendirme işlemini gerçekleştirdim. Ayrıca, kullanıcıların Google ile giriş yapabilmesi için GoogleSignIn ile entegre bir giriş fonksiyonu da ekledim.

Gün 10 :

Giriş Ekranı Detayları ve Hata Yönetimi

Stajimin on birinci gününde, giriş (login) sayfasını daha da geliştirdim ve hatalara karşı daha sağlam bir yapı kurmak için gerekli iyileştirmeleri yaptım. Bugünkü çalışmalarımda, kullanıcı deneyimini artırmak ve giriş işlemini daha kullanışlı hale getirmek için ek özellikler ekledim.

Google ile Giriş

Öncelikle, Google ile giriş özelliğini entegre ettim. Google hesabı ile giriş yapabilmek için `GoogleSignIn` paketini kullanarak Firebase Authentication'a bağladım. Kullanıcı Google hesabıyla giriş yaptığından sonra Firebase'e gerekli kimlik bilgilerini ilettim ve kullanıcıyı oturum açtıktan sonra ana sayfaya yönlendirdim:



```

1 final GoogleSignInAccount? googleUser = await GoogleSignIn().signIn();
2 final GoogleSignInAuthentication? googleAuth = await googleUser?.authentication;
3 final credential = GoogleAuthProvider.credential(
4   accessToken: googleAuth?.accessToken,
5   idToken: googleAuth?.idToken,
6 );
7 await FirebaseAuth.instance.signInWithCredential(credential);

```

Şekil 1

Şifre Sıfırlama Fonksiyonu

Kullanıcıların şifrelerini unutmaları durumunda, şifre sıfırlama işlemi gerçekleştirebilmeleri için bir şifre sıfırlama fonksiyonu ekledim. Bu özellik, kullanıcıların e-posta adreslerini girerek şifre sıfırlama bağlantısını almasını sağladı:



```
1 await FirebaseAuth.instance.sendPasswordResetEmail(email: emailController.text);
```

Şekil 1

Bu işlem sonrasında, kullanıcının e-posta adresine şifre sıfırlama talimatlarını içeren bir e-posta gönderiliyor.

Hata Yönetimi ve Bildirimler

Giriş işlemi sırasında ortaya çıkabilecek hatalar için kapsamlı bir hata yönetimi sistemi kurdum. Firebase Authentication işlemlerinde `try-catch` blokları ile oluşabilecek hataları yakalayıp, kullanıcıya anlamlı hata mesajları gösterdim. `Get.snackbar` kullanarak, hata mesajlarını kullanıcıya bildirdim:



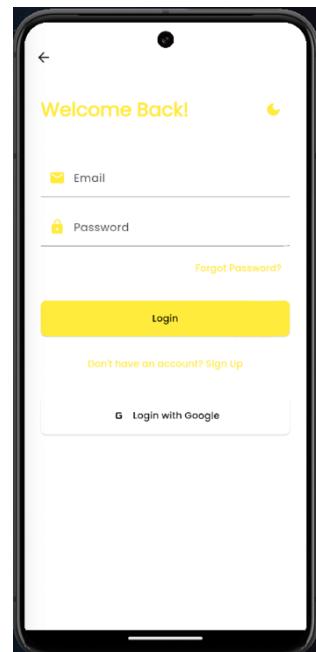
```
1 Get.snackbar('Login Error', e.message ?? 'Error occurred during login',
2   backgroundColor: Colors.red,
3   colorText: Colors.white,
4   snackPosition: SnackBarPosition.BOTTOM);
```

Şekil 1

Kullanıcı Deneyimi İyileştirmeleri

- Yüklenme Göstergesi:** Kullanıcı giriş butonuna tıkladığında, işlemin devam ettiğini göstermek için bir yüklenme (loading) simgesi ekledim. Bu, kullanıcının işlemi beklediğini anlaması açısından önemli bir adım oldu.
- Tema Desteği:** Projeye koyu ve açık tema desteği ekleyerek, kullanıcıların uygulamayı kişisel tercihleri doğrultusunda özelleştirebilmesini sağladım.

Bu iki gün içerisinde, hem teknik açıdan önemli geliştirmeler yaptım hem de kullanıcı deneyimini artıran yeni özellikler ekledim. Firebase ile başarılı bir entegrasyon kurmuş olmak, projeye hız kazandırdı ve bundan sonraki aşamalarda daha karmaşık özellikler ekleyemem sahip olacak sağlam bir temel oluşturdu.



Şekil 1

Gün 11 :

WelcomePage

Bugün stajimin on ikinci günüydü ve proje kapsamında oldukça önemli ilerlemeler kaydettim. Öncelikle, login (giriş) ve signup (kayıt ol) sayfalarını başarıyla tamamladım. Bu aşamadan sonra projede kullanıcıların giriş yaptıktan sonra karşılayacağı WelcomePage’ı (karşılama sayfası) oluşturmaya başladım.

WelcomePage’de tasarım açısından kullanıcıya hoş bir deneyim sunmak için animasyonlu bir logo ekledim. Bu animasyon, kullanıcıya uygulamanın dinamik bir yapıya sahip olduğunu hissettirmek amacıyla tasarlandı. Ayrıca sayfaya dil değiştirme ve tema değiştirme butonları ekledim. Bu butonlar kullanıcı deneyimini kişiselleştirmek adına oldukça önemlidir. Dil ve tema butonlarını ekledikten sonra bu butonları daha önce oluşturduğum Locale (yerel dil desteği) ve ThemeController (tema kontrolörü) ile entegre ettim. Bu entegrasyon sayesinde kullanıcılar, uygulamanın temasını ve dilini değiştirebiliyor.

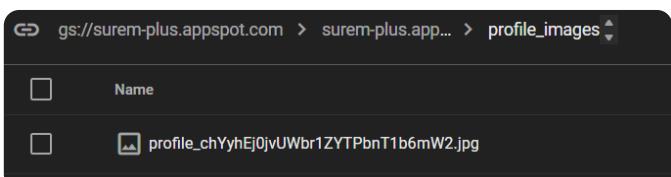
Bu aşamadan sonra projenin ana sayfası olan HomePage’ın yapımına geçtim. HomePage’de kullanıcılara daha interaktif bir deneyim sunmak adına "choose file" (dosya seç) butonunu ekledim. Bu buton sayesinde kullanıcı, cihazında bulunan galeriden ya da dosyalarından bir resim seçebiliyor ve seçtiği resmi geçici olarak ekranda görüntüleyebiliyor. Bunun altında, "process" (işle) butonu yer alıyor. Bu buton, ilerde API işlevi görecektir ve seçilen resim üzerinde yapılacak işlemleri API üzerinden gerçekleştirip sonuçlarını ekranda gösterecek. Bu aşamada API entegrasyonu henüz tamamlanmadı, ancak temel altyapıyı oluştururdum. Ayrıca, "save" (kaydet) butonunu ekleyerek kullanıcının seçtiği resmi kaydedebilmesi için bir fonksiyon yazdım.

Projenin ilerleyen aşamalarında, kullanıcı bilgilerini yönetmek için Profile Screen (profil ekranı) üzerinde çalıştım. Bu ekranda Firebase’den gelen verileri kullanarak kullanıcının profil bilgilerini doldurdum. Kullanıcıların profil resimlerini güncelleyebilmeleri için Firebase Storage kütüphanesini projeye ekledim. YAML dosyasına Firebase Storage ile ilgili gerekli bağımlılıkları ekledikten sonra, profil resmini yüklemek için bir dosya yükleme sistemi oluşturup şu komut ile kontrol ettim:

```

1 // Profil resmini gösterme ve değiştirme seçenekleri
2 Obx(() => Stack(
3   alignment: Alignment.bottomRight,
4   children: [
5     CircleAvatar(
6       radius: 60,
7       backgroundImage: controller.profileImageUrl.value != null
8         ? NetworkImage(controller.profileImageUrl.value!)
9         : const AssetImage('assets/default_profile.png') as ImageProvider,
10    ),
11    GestureDetector(
12      onTap: controller.changeProfileImage,
13      child: Container(
14        padding: const EdgeInsets.all(8),
15        decoration: BoxDecoration(
16          color: Theme.of(context).primaryColor,
17          shape: BoxShape.circle,
18        ),
19        child: const Icon(Icons.edit, size: 20, color: Colors.black),
20      ),
21    ),
22  ],
23 )),
24 // Kullanıcı adı alanı
25 Obx(() => TextField(
26   controller: TextEditingController(text: controller.username.value),
27   onChanged: controller.changeUsername,
28   style: Theme.of(context).textTheme.titleLarge,
29   textAlign: TextAlign.center,
30   decoration: const InputDecoration(
31     border: InputBorder.none,
32   ),
33 )),
34 // Çıkış butonu
35 OutlinedButton(
36   onPressed: controller.logout,
37   style: OutlinedButton.styleFrom(
38     side: const BorderSide(color: Colors.red),
39   ),
40   child: const Text('Logout'),
41 ),

```



```

1 // "choose file"
2 ElevatedButton(
3   onPressed: controller.pickImage,
4   child: const Text('Choose File'),
5 ),
6 // Seçilen resmi geçici olarak gösterme
7 controller.image != null
8 ? Image.file(controller.image!)
9 : const Text('No image selected'),
10 // "process" butonu, API fonksiyonunu simüle eder
11 ElevatedButton(
12   onPressed: controller.processImage,
13   child: const Text('Process'),
14 ),
15 // "save" butonu ile işlenen resmi kaydetme
16 ElevatedButton(
17   onPressed: controller.saveImage,
18   child: const Text('Save'),
19 ),

```

```

rules_version = '2';
service firebase.storage {
  match /b/{bucket}/o {
    match /{allPaths=**} {
      allow read, write: if request.auth != null;
    }
  }
}

```

Sign-in providers		
Provider		Status
Email/Password		Enabled
Google		Enabled

Bununla birlikte, ImagePicker kütüphanesini kullanarak kullanıcının cihazından bir resim seçmesine olanak sağladım. Ancak, bu aşamada karşılaştığım bir sorun vardı: Seçilen resim Firebase Storage'a yüklenmiyordu. Hatanın kaynağını bulmak için mentorumla görüştüm ve Firebase Storage'daki "rules" (kurallar) kısmını kontrol etmem gerektiğini öğrendim. Bu alanı incelediğimde, yalnızca "read" (okuma) izni olduğunu fark ettim. Bu nedenle Firebase Storage'da hem "read" hem de "write" (yazma) izinlerini ekledim ve böylece sorun çözüldü.

Firebase ile ilgili bir diğer sorun, seçilen resmin boyutunun büyük olmasıydı. Bu nedenle VScode konsolunda hata alıyorum. Bu hatayı çözmek için resmi sıkıştırma işlemi yaptım ve resim boyutunu 1024x1024 piksel (yaklaşık 1MB) olacak şekilde optimize ettim. Resmi sıkıştırdıktan sonra sorunsuz bir şekilde Firebase'e yüklemeyi başardım.

Profil ekranında kullanıcı deneyimini artırmak amacıyla, kullanıcıların kullanıcı adlarını değiştirebilmesi için bir "changeUsername" (kullanıcı adı değiştir) fonksiyonu ekledim. Bu sayede kullanıcılar profil ayarları üzerinden kullanıcı adlarını kolaylıkla güncelleyebilecekler. Ayrıca, AppBar (uygulama çubuğu) üzerinde geri butonu, dil değiştirme ve tema değiştirme butonlarını ekledim. Bu butonlar, kullanıcıların uygulama içinde daha hızlı ve kullanıcı dostu bir şekilde gezinmesine olanak sağlıyor.

Bunlara ek olarak, Firebase Authentication yardımıyla şifre değiştirme işlemi için bir fonksiyon yazdım. Bu fonksiyon, kullanıcılar e-posta göndererek şifrelerini sıfırlamalarını sağlıyor. Kullanıcı, gelen e-posta üzerinden şifresini değiştirebiliyor. Bu işlem güvenlik açısından önemli bir adım olduğu için Firebase ile sorunsuz bir şekilde çalışmasını sağladım.

Son olarak, uygulamaya bir "logout" (çıkış) butonu ekledim. Bu buton sayesinde kullanıcılar uygulamadan güvenli bir şekilde çıkış yapabiliyor ve çıkış işlemi tamamlandığında WelcomePage'e yönlendiriliyorlar.

Tüm bu işlemler sırasında sayfanın responsive (duyarlı) olmasını sağlamak için MediaQuery yapısını kullandım. Farklı cihaz boyutlarına göre düzenlemeler yaptım ve kullanıcı deneyimini çeşitli cihazlarda optimize ettim. Ayrıca, uygulamanın çok dilli desteğini genisletmek için JSON dosyaları ile dil değiştirme özelliğini yaygınlaştırdım. Tüm sayfalarda bu dil değişikliklerinin sorunsuz bir şekilde çalışmasını sağladım.

```
1 class LoginController extends GetxController {
2   final emailController = TextEditingController();
3   final passwordController = TextEditingController();
4   final isLoading = false.obs;
5
6   Future<void> login() async {
7     if (emailController.text.isNotEmpty && passwordController.text.isNotEmpty) {
8       isLoading.value = true;
9
10    try {
11      UserCredential userCredential =
12        await FirebaseAuth.instance.signInWithEmailAndPassword(
13          email: emailController.text,
14          password: passwordController.text,
15        );
16    // User successfully Logged in
17    Get.snackbar('Login', 'Login successful',
18      backgroundColor: Colors.green,
19      colorText: Colors.white,
20      snackPosition: SnackPosition.BOTTOM);
21    Get.offAll(() => const HomeScreen());
22  } on FirebaseAuthException catch (e) {
23    Get.snackbar('Login Error', e.message ?? 'Error occurred during login');
24    backgroundColor: Colors.red,
25    colorText: Colors.white,
26    snackPosition: SnackPosition.BOTTOM);
27  } finally {
28    isLoading.value = false;
29  }
30 } else {
```

```
1 // E-posta ve şifre alanlarıyla giriş forma
2 TextField(
3   controller: controller.emailController,
4   decoration: const InputDecoration(
5     labelText: 'Email',
6     prefixIcon: Icon(Icons.email, color: Colors.yellow),
7   ),
8 ),
9 TextField(
10   controller: controller.passwordController,
11   decoration: const InputDecoration(
12     labelText: 'Password',
13     prefixIcon: Icon(Icons.lock, color: Colors.yellow),
14   ),
15   obscureText: true,
16 ),
17 // Yüklenme durumıyla giriş butonu
18 Obx(() => ElevatedButton(
19   onPressed: controller.isLoading.value ? null : controller.isLoading.value ? const CircularProgressIndicator() : const Text('Login'),
20   child: controller.isLoading.value ? const CircularProgressIndicator() : const Text('Login'),
21 )),
22 // Google ile giriş butonu
23 Obx(() => ElevatedButton.icon(
24   onPressed: controller.isLoading.value ? null : controller.isLoading.value ? const CircularProgressIndicator() : const Text('Login with Google'),
25   icon: const Icon(Icons.g_mobiledata, size: 24),
26   label: controller.isLoading.value ? const CircularProgressIndicator() : const Text('Login with Google'),
27   style: ElevatedButton.styleFrom(
28     primary: Colors.indigo,
29     shape: RoundedRectangleBorder(
30       borderRadius: BorderRadius.circular(10),
31     ),
32   ),
33 )),
34 
```

Users					
Identifier	Providers	Created	Signed In	User UID	Actions
as@gmail.c...	Email	Oct 2, 2...	Oct 2, 2...	h1MAdolfhM5...	
its.3rsa@g...	Email	Sep 28, 2...	Oct 7, 2...	chYyhEj0JvUWbr...	
dsadas@gd...	Email	Sep 28, 2...	Sep 28, 2...	TAoykzFPMGPk...	

Gün 12:

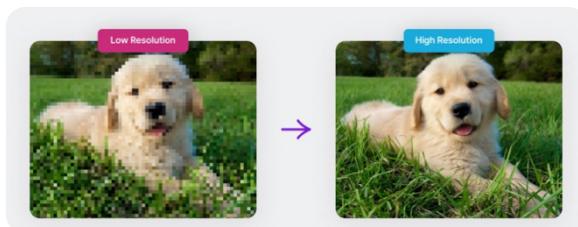
Yapay Zeka Modeli Geliştirme ve Super-Resolution (Süper Çözünürlük) Uygulaması

Bugün stajimin on üçüncü gününde, denetleyicim tarafından daha önce tamamladığım görevler inceledi ve bazı küçük düzeltmeler yapmam önerildi. Bu düzeltmeler hızlıca gerçekleştirdikten sonra, yapılan bir toplantıda Turgut Bey, projede bir yapay zeka modeline geçmem gerektiğini belirtti. Bu model, görüntülerin kalitesini artırmaya yönelik bir **süper çözünürlük (super-resolution)** algoritması olacak. Daha önce bu konuda deneyimim olmamakla birlikte, birlikte çalıştığım diğer stajyer arkadaşımın bu alanda bilgisi vardı ve bu benim için öğrenme sürecimi hızlandıracak bir fırsat sundu.

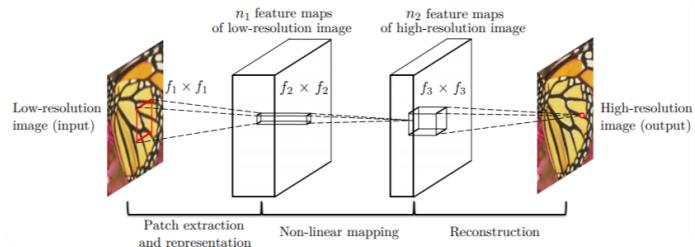
Super-Resolution ve SRCNN Modeli

Super-resolution, düşük çözünürlüklü görüntülerin kalitesini artırma işlemidir. Bu yöntem, bulanık veya piksel yoğunluğu düşük olan görüntülerdeki eksik detayların tahmin edilerek yeniden oluşturulmasını sağlar. Super-resolution işlemi genellikle bir sinir ağısı, özellikle **konvolüsyonel sinir ağısı (Convolutional Neural Network - CNN)** kullanılarak gerçekleştirilir. Model, düşük çözünürlüklü bir görüntüyü analiz ederek yüksek çözünürlüklü bir versiyon üretir. Projemizde, yaygın olarak kullanılan **SRCCN (Super-Resolution Convolutional Neural Network)** modelini tercih edeceğiz. SRCCN, görüntüdeki düşük çözünürlüklü detayları CNN katmanları yardımıyla işleyip, kaybolan detayları geri getirerek yüksek çözünürlüklü bir görüntü üretir.

SRCCN modelinin başarısı, geniş çaplı bir eğitim verisi gerektirmektedir. Bu nedenle, modeli eğitmek için düşük ve yüksek çözünürlüklü görüntü çiftlerinden oluşan büyük bir veri seti kullanmayı planlıyoruz. Eğitim sürecinde, düşük çözünürlüklü görüntüler modele verilecek ve model, bu görüntülerini yüksek çözünürlüklü versiyonlarına dönüştirmeye çalışacak. Eğitim sırasında kullanılan verilerin çeşitliliği, modelin genel performansını artıracak ve çeşitli görüntü formatlarıyla daha başarılı sonuçlar elde etmemizi sağlayacaktır.



Şekil 1



Şekil 1

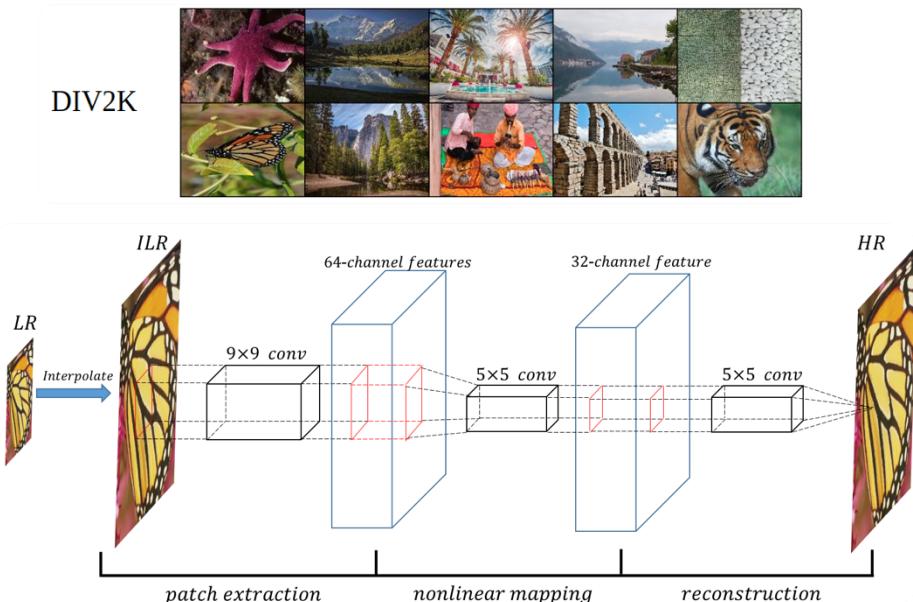
Model Geliştirme Süreci ve Kullanılacak Araçlar

Proje kapsamında geliştirilecek model, Flutter uygulaması ile entegre edilecektir. Kullanıcılar, uygulama üzerinden bir görüntü yüklediklerinde bu görüntü yapay zeka modeline gönderilecek, işlenmiş yüksek çözünürlüklü versiyon geri dönecek ve kullanıcıya sunulacak. Bu süreçte modeli eğitmek için **Python** programlama dilini ve yapay zeka kütüphanelerinden biri olan **TensorFlow** veya **PyTorch** kullanmayı planlıyoruz.

Proje İlerleyışı ve Gelecek Adımlar

Projenin bu aşamasında, yapay zeka modelinin geliştirilmesi ve eğitilmesi üzerine odaklanacağım. Bunun için, görüntüleri işleyebilecek altyapıyı kurarak, modelin verimli çalışabilmesi adına gerekli eğitim ve değerlendirme süreçlerini yürüteceğim. Eğitim aşamasında farklı veri setleri kullanılarak modelin öğrenme kapasitesini artırmayı ve olası hataların minimize edilmesini hedefliyoruz. Bununla birlikte, modelin performansını ölçmek için çeşitli metrikler geliştireceğiz ve bu metrikler doğrultusunda sürekli geri bildirim sağlayarak modelin iyileştirilmesine katkı sağlayacağız.

Sonuç olarak, proje kapsamında geliştirilecek bu super-resolution modeli, kullanıcılarla düşük çözünürlüklü görüntülerini daha yüksek kalitede sunma imkanı verecek. Bu süreçteki çalışmam, hem yapay zeka modelinin teorik temelini öğrenmemi sağlayacak hem de Flutter uygulaması ile entegrasyonunu gerçekleştirecek teknik yetkinliklerimi geliştirmeme katkıda bulunacaktır.



Şekil 1

Gün 13 :

Derin Öğrenme ve Süper Çözünürlük: Google Colab ile Yapay Zeka Modeli Eğitimi

Bugün stajimin on dördüncü günü. Proje üzerinde ilerlemek ve süper çözünürlük modeli (SRCNN) ile ilgili çalışmalarla başlamak için araştırmalar yaptıktan sonra, projeyi Google Colab platformu üzerinde gerçekleştirmeye kararım aldım. Google Colab, özellikle derin öğrenme projeleri için çok sayıda avantaj sunan, Jupyter Notebook'un bulut tabanlı bir versiyonudur. Colab'ı tercih etmemdeki temel sebeplerden biri, sağladığı ücretsiz GPU ve TPU kaynaklarının model eğitim süreçlerinde hesaplamaları hızlandırmasıdır. Derin öğrenme modelleri genellikle büyük veri setleriyle çalışmayı ve yüksek hesaplama gücü gerektirir. Colab'ın sunduğu güçlü donanım kaynakları, projeyi daha verimli bir şekilde yürütmemi sağlayacaktır. Ayrıca, Colab'ın bulut tabanlı yapısı sayesinde internet bağlantısı olan her yerden projeye erişim sağlanabilir ve projeler kolayca paylaşılabilir.



Jupyter Notebook, yerel makinelerde çalışan bir araç olarak tanımlanabilir. Kullanıcılara esneklik sunar ancak donanım yeterliliği olmayan makinelerde bazı sınırlamalar yaratabilir. Bu açıdan Google Colab, bulut altyapısının avantajlarını sunarak kaynak yetersizliği olan bilgisayarlarda dahi büyük ölçekli projeleri yürütebilme imkânı tanır. Bunun yanında, model eğitimi süresince oluşabilecek teknik kısıtlamalarla karşılaşma riskini azaltır.

Projemizin temelinde, derin öğrenme modellerinin geliştirilmesi ve eğitimi için güçlü bir araç olan **TensorFlow** yer alıyor. TensorFlow, Google tarafından geliştirilen açık kaynaklı bir kütüphane olup, özellikle yapay zeka ve makine öğrenimi uygulamaları için geniş kapsamlı bir platform sağlar. **Keras**, TensorFlow üzerinde çalışan ve derin öğrenme modellerini hızlandıran yüksek seviyeli bir API olarak kullanılır. Keras'ın modüler yapısı ve kullanımı kolaydır, bu da model geliştirme sürecini oldukça basit hale getirir. Veriler üzerinde sayısal işlemler yapmak ve veri yapıları ile çalışmak için ise **NumPy** kullanılmaktadır. NumPy, Python'un sayısal işlem kütüphanesi olup, özellikle çok boyutlu diziler ve matrisler üzerinde çalışmayı kolaylaştırır. Model eğitimi sırasında verilerin optimize edilmesinde büyük rol oynar.



İlk adım olarak, Google Colab üzerinde bu kütüphaneleri yükleyeceğim. Colab platformunda, gerekli kütüphaneleri yüklemek oldukça basittir ve yalnızca birkaç satır kod yazarak proje ortamına entegre edilebilmektedir. Bu kütüphaneleri yükledikten sonra, proje için veri hazırlama aşamasına geçeceğim. **SRCNN** (Super-Resolution Convolutional Neural Network) modelini eğitmek için düşük ve yüksek çözünürlüklü resim çiftlerine ihtiyaç duyacağım. Bu veri çiftleri, düşük çözünürlüklü bir görüntüyü giriş olarak alacak ve modelin yüksek çözünürlüklü bir çıktıyı üretmesi için eğitim verisi olarak kullanılacaktır.

Google Colab üzerinde çalışırken, kaynakların verimli kullanımı büyük önem taşiyor. Bu nedenle, GPU veya TPU seçenekleri kullanılarak model eğitimi hızlandırılabilir. SRCNN modelinin eğitimi sırasında farklı hiperparametrelerle oynayarak modelin performansını optimize etmeye çalışacağım. Ayrıca, modelin sonuçlarını değerlendirmek için çeşitli metrikler (örneğin PSNR - Peak Signal-to-Noise Ratio ve SSIM - Structural Similarity Index) kullanacağım. Bu metrikler, modelin çıktı kalitesini ölçümede ve iyileştirme süreçlerinde kritik bir rol oynayacaktır. Eğitim sürecinin sonunda, modelin test verileri üzerindeki performansını değerlendirerek gerekli iyileştirmeleri yapmayı planlıyorum.

Bu süreçte kullanılan teknolojiler ve kütüphaneler, derin öğrenme projelerinin temel taşılarını oluşturuyor. Google Colab, TensorFlow, Keras ve NumPy gibi araçların sağladığı avantajları en etkili şekilde kullanarak, yapay zeka modelimizi Flutter uygulamasına entegre edecek ve kullanıcıların düşük çözünürlüklü görüntülerini daha kaliteli hale getiren bir çözüm sunacağım.

Gün 14 :

Süper Çözünürlük Modelinin Kurulumu ve İlk Aşamaları

Bugün stajimin on beşinci gününde, süper çözünürlük modeli oluşturma sürecine odaklandım. Bu, yüksek çözünürlülü görünümlerin üretimini hedefleyen derin öğrenme modellerini içeriyor ve bu amaca yönelik olarak Google Colab ortamını hazırladım. İlk olarak, Google Colab üzerinde çalışacak ortamı oluşturmak için gerekli kütüphaneleri indirdim. `pip install` komutlarını kullanarak PyTorch, Torchvision, Matplotlib gibi temel kütüphaneleri yükledim. Bu kütüphaneler, görüntü işleme ve derin öğrenme modellerinin oluşturulması ve eğitimi için gerekli temel araçları sağlıyor.

Ardından, bu kütüphaneleri doğru şekilde içe aktardığımın emin olmak için `import` işlemlerini gerçekleştirdim. Tüm kütüphanelerin başarılı bir şekilde içe aktarılmasının ardından, süper çözünürlük modelinin temel yapısını oluşturmaya başladım. Modelin yapısı, katmanlardan oluşuyor ve her bir katman görüntü üzerinde farklı bir işlevi yerine getiriyor.

- **İlk katman:** Düşük çözünürlüklü verileri işleyerek temel özelliklerini çıkarıyor.
- **Orta katmanlar:** Görüntüde eksik olan detayları tamamlayarak görüntünün genel kalitesini artırıyor.
- **Son katman:** Görüntünün yeniden yapılandırıldığı ve yüksek çözünürlüklü bir çıktıının elde edildiği final katmanıdır.

Bunu başarmak için önce süper çözünürlük modeli için kullanılacak veri setine ihtiyaç duyдум. Bu bağlamda **DIV2K** veri setini Google Colab ortamına indirdim ve veri setiyle çalışmak için gerekli dosyaları açtım. DIV2K veri seti, görüntü süper çözünürlük algoritmalarını eğitmek için yaygın olarak kullanılan bir veri setidir ve yüksek çözünürlüklü görüntüler sunar. Veriler indirildikten sonra, PyTorch ve diğer veri işleme araçlarını model için hazırladım. Modelin eğitilmesi sürecinde kullanılan cihazların seçimi de önemli bir adımdı; bu nedenle GPU kullanımını kontrol ettim ve en uygun cihazı seçtim.

Bu işlemlerin ardından, modelin yapı taşlarını oluştururdum. **Generator** ve **Discriminator** olmak üzere iki ana sınıf oluşturuldu:

1. **Generator (Üretici) Sınıfı:** Düşük çözünürlüklü görüntülerini alıp bunları yüksek çözünürlüklü hale getirmek için katmanları içerir. Bu sınıf, modelin temelini oluşturur ve görüntüdeki kayıp detayları yeniden oluşturma görevini üstlenir.
2. **Discriminator (Ayırıcı) Sınıfı:** Gerçek görüntüler ile model tarafından üretilen sahte (sentetik) görüntüler arasındaki farkı ayırt etmek için tasarlanmıştır. Bu sınıf, modelin başarısını değerlendirmeye ve iyileştirmeye yardımcı olur.

Veri işleme kısmı için de ayrı bir sınıf oluştururdum: **DIV2KDataset**. Bu sınıf, yüksek çözünürlüklü görüntülerleri alır ve düşük çözünürlüklü versiyonlarını üretir. Bu dönüşümler, süper çözünürlük modelinin eğitim sürecinde önemli bir rol oynar.

```

1 !wget http://data.vision.ee.ethz.ch/cvl/DIV2K/DIV2K_train_HR.zip
2 !unzip DIV2K_train_HR.zip
Gizli çıkış göster

1 from torch.utils.data import random_split
2 import matplotlib.pyplot as plt

1 from torchvision.utils import save_image

1 import torch
2 import torch.nn as nn
3 import torch.optim as optim
4 from torch.utils.data import Dataset, DataLoader, random_split
5 from torchvision import transforms
6 from torchvision.utils import save_image
7 import os
8 from PIL import Image
9 import matplotlib.pyplot as plt
10 import numpy as np
11

1 GPU'nun etkin olduğunu kontrol eder
2 device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
3 print(f"Using device: {device}")
4

king device: cuda

```

Şekil 1

```

1 class Generator(nn.Module):
2     def __init__(self, scale_factor=4):
3         super(Generator, self).__init__()
4         self.conv1 = nn.Conv2d(3, 64, kernel_size=9, padding=4)
5         self.conv2 = nn.Conv2d(64, 64, kernel_size=3, padding=1)
6         self.conv3 = nn.Conv2d(64, 64, kernel_size=3, padding=1)
7         self.conv4 = nn.Conv2d(64, 64, kernel_size=3, padding=1)
8         self.conv5 = nn.Conv2d(64, 64, kernel_size=3, padding=1)
9         self.conv6 = nn.Conv2d(64, 64, kernel_size=3, padding=1)
10        self.conv7 = nn.Conv2d(64, 3, kernel_size=9, padding=4)
11        self.relu = nn.ReLU(inplace=True)
12        self.upsample = nn.Upsample([scale_factor*scale_factor, mode='nearest'])

13
14    def forward(self, x):
15        x = self.relu(self.conv1(x))
16        residual = x
17        x = self.relu(self.conv2(x))
18        x = self.relu(self.conv3(x))
19        x = self.relu(self.conv4(x))
20        x = self.relu(self.conv5(x))
21        x = self.relu(self.conv6(x))
22        x += residual
23        x = self.upsample(x)
24        x = self.conv7(x)
25        return torch.tanh(x)

```

Şekil 1

Gün 15:

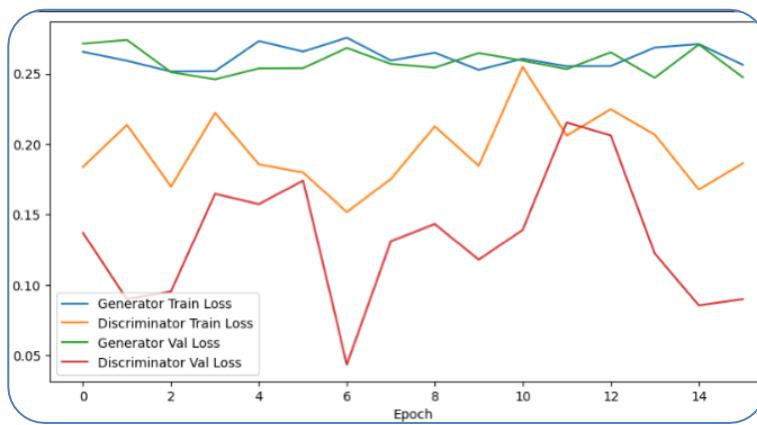
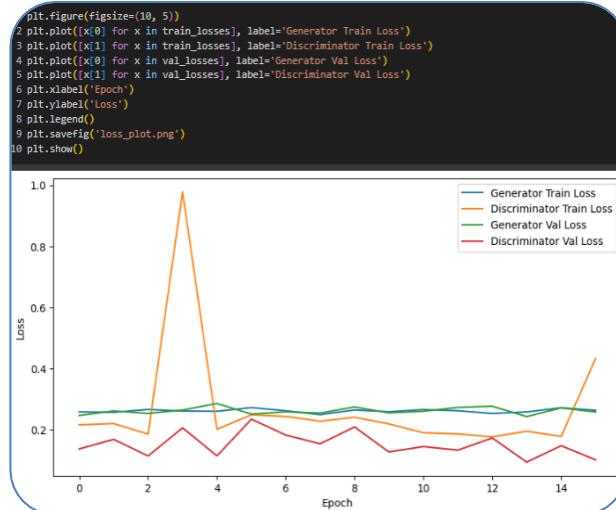
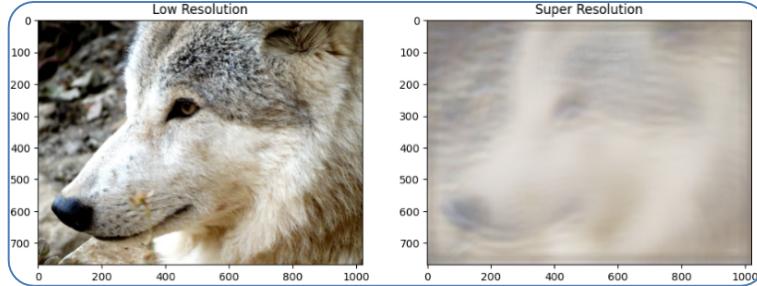
Eğitim Süreci ve Modelin Performans Analizi

Bugün, modelin eğitim sürecine odaklandım. İlk olarak, eğitim ve doğrulama verilerini hazırladım. **DIV2K** veri setini %80 eğitim ve %20 doğrulama olarak ayırdım. Bu ayrılmış, modelin eğitimi ve doğrulanması aşamalarında kullanılacak veri kümelerini belirledi. Eğitim sürecine başlamadan önce, modelin performansını optimize edebilmek için gerekli **optimizer** ve **kayıp fonksiyonlarını** tanımladım.

- **Optimizer:** Modelin öğrenme sürecini yöneten algoritmadır. Genellikle Stokastik Gradient Descent (SGD) veya Adam Optimizer gibi teknikler kullanılır.
- **Kayıp Fonksiyonu:** Modelin performansını değerlendirmek için kullanılır. Bu fonksiyon, modelin ürettiği sonuç ile gerçekle beklenen sonuç arasındaki farkı ölçer. Eğitim sırasında bu farkı minimize etmeye çalışılır.

Bu adımların ardından, eğitim ve doğrulama döngülerini başlattım. Her epoch'ta, hem **generator** hem de **discriminator** modelleri güncellenir. Her epoch'ta kayıplar kaydedilir ve modelin performansı takip edilir. Eğitim süreci boyunca düzenli olarak kayıp fonksiyonları ve doğrulama sonuçları görselleştirilir. Bu görselleştirmeler, modelin ilerleyişini ve başarısını anlamamda önemli bir rol oynadı. Aşağıda örnek bir kod parçası bulunuyor:

```
Epoch [1/16] | G Loss: 0.2564, D Loss: 0.2276 | Val G Loss: 0.2562, Val D Loss: 0.2382
Epoch [2/16] | G Loss: 0.2504, D Loss: 0.2281 | Val G Loss: 0.2765, Val D Loss: 0.1756
Epoch [3/16] | G Loss: 0.2605, D Loss: 0.2247 | Val G Loss: 0.2888, Val D Loss: 0.1140
Epoch [4/16] | G Loss: 0.2664, D Loss: 0.2053 | Val G Loss: 0.2920, Val D Loss: 0.1085
Epoch [5/16] | G Loss: 0.2664, D Loss: 0.1947 | Val G Loss: 0.2665, Val D Loss: 0.1810
Epoch [6/16] | G Loss: 0.2736, D Loss: 0.1774 | Val G Loss: 0.2577, Val D Loss: 0.1058
Epoch [7/16] | G Loss: 0.2723, D Loss: 0.1872 | Val G Loss: 0.2627, Val D Loss: 0.1615
Epoch [8/16] | G Loss: 0.2683, D Loss: 0.1516 | Val G Loss: 0.2685, Val D Loss: 0.1515
Epoch [9/16] | G Loss: 0.2626, D Loss: 0.1389 | Val G Loss: 0.2751, Val D Loss: 0.1058
Epoch [10/16] | G Loss: 0.2671, D Loss: 0.1873 | Val G Loss: 0.2761, Val D Loss: 0.1330
Epoch [11/16] | G Loss: 0.2599, D Loss: 0.1751 | Val G Loss: 0.2622, Val D Loss: 0.0700
Epoch [12/16] | G Loss: 0.2613, D Loss: 0.1584 | Val G Loss: 0.2715, Val D Loss: 0.0695
Epoch [13/16] | G Loss: 0.2619, D Loss: 0.1814 | Val G Loss: 0.2643, Val D Loss: 0.1345
Epoch [14/16] | G Loss: 0.2637, D Loss: 0.1302 | Val G Loss: 0.2646, Val D Loss: 0.1449
Epoch [15/16] | G Loss: 0.2749, D Loss: 0.1838 | Val G Loss: 0.2803, Val D Loss: 0.1448
Epoch [16/16] | G Loss: 0.2657, D Loss: 0.2254 | Val G Loss: 0.2871, Val D Loss: 0.1433
```



Tarih ve İşyeri Amirinin İmzası

23/09/2024

Gün 16 :

Yeni Modelin Geliştirilmesi ve SRCNN Yapısı

Bugün, stajimin on altinci günü ve dünkü başarısız model deneyiminden sonra süreci yeniden gözden geçirerek daha sağlam bir model geliştirmeye karar verdim. Dünkü testlerde düşük çözünürlüklü görüntülerde bulanıklık sorununu çözemediğim için, bu problemi kökünden halletmek adına sıfırdan bir model oluşturmaya başladım. İlk adım olarak, gerekli veri setlerinin Google Colab ortamına indirilmesi ve açılması işlemlerini gerçekleştirdim. **DIV2K** veri setini kullanarak, düşük çözünürlüklü (LR) ve yüksek çözünürlüklü (HR) görüntülerini içeren iki ayrı veri setini işleme aldım.

Bu veri setlerini işlerken, düşük çözünürlüklü görüntülerin boyutlarının model tarafından daha iyi analiz edilmesi adına her birini **256x256** hedef boyutlarına yeniden boyutlandırdım. Bu işlem, modelin giriş verilerini daha standart hale getirmeye yönelik bir önlem olarak önemli bir adımdır. Görüntü boyutlandırma süreci, modelin performansını artırmak için en iyi şekilde optimize edilmelidir.

Veri hazırlama adımlarının tamamlanmasının ardından, model geliştirme sürecine geçtim. Bu kez, **SRCNN (Super-Resolution Convolutional Neural Network)** mimarisine dayanan bir model geliştirdim. SRCNN, görüntüleri süper çözünürlüklü hale getirmek için yaygın olarak kullanılan bir CNN modelidir ve üç ana katmandan oluşur:

- İlk katman**, düşük çözünürlüklü görüntülerini filtreleyerek temel bilgileri çıkarır. Bu katman, bir tür önişleme görevini üstlenir ve giriş verilerini modelin geri kalanına hazır hale getirir.
- İkinci katman**, görüntüdeki eksik detayları tamamlar ve genel çözünürlüğü artırır. Bu katmanda, daha derin özniteliklerin işlenmesi sağlanır.
- Son katman**, yüksek çözünürlüklü görüntü üretir ve sürecin tamamlanmasını sağlar. Bu katman, modelin çıktısının gerçek dünyadaki yüksek çözünürlüklü bir görüntüye yakın olmasını amaçlar.

Aşağıda, SRCNN modelinin tanımlandığı kod yer alıyor:

```

1 wget http://data.vision.ee.ethz.ch/cvl/DIV2K/DIV2K_train_LR_bicubic_X2.zip
2 unzip DIV2K_train_LR_bicubic_X2.zip
3 wget http://data.vision.ee.ethz.ch/cvl/DIV2K/DIV2K_train_HR.zip
4 unzip DIV2K_train_HR.zip

Gizli çıkış göster

1 import os
2 import cv2
3 import numpy as np
4 from glob import glob
5 low_res_images = sorted(glob('DIV2K_train_LR_bicubic/X2/*.png'))
6 high_res_images = sorted(glob('DIV2K_train_HR/*.png'))
7 target_height, target_width = 256, 256
8 def load_and_resize_images(image_paths):
9     images = []
10    for img_path in image_paths:
11        img = cv2.imread(img_path)
12        img = cv2.resize(img, (target_width, target_height))
13        images.append(img)
14    return np.array(images)
15 low_res_data = load_and_resize_images(low_res_images)
16 high_res_data = load_and_resize_images(high_res_images)

1 import tensorflow as tf
2 from tensorflow.keras.layers import Conv2D, Input
3 from tensorflow.keras.models import Model
4 def build_srcnn():
5     input_img = Input(shape=(None, None, 3))
6     x = Conv2D(64, (9, 9), activation='relu', padding='same')(input_img)
7     x = Conv2D(32, (1, 1), activation='relu', padding='same')(x)
8     output_img = Conv2D(3, (5, 5), activation='linear', padding='same')(x)
9     model = Model(inputs=input_img, outputs=output_img)
10    return model
11 model = build_srcnn()
12 model.summary()

Model: "functional_1"
Layer (type)          Output Shape         Params #
Input Layer (InputLayer)   (None, None, None, 3)      0
conv2d_3 (Conv2D)        (None, None, None, 64)    15,360
conv2d_4 (Conv2D)        (None, None, None, 32)    2,048
conv2d_5 (Conv2D)        (None, None, None, 3)     2,403

Total params: 20,009 (78.51 KB)
Trainable params: 19,000 (78.51 KB)
Non-trainable params: 0 (0.00 B)

import numpy as np
2 import cv2
3 import os
4 def load_lr_hr_images(lr_dir, hr_dir, target_size=(128, 128)):
5     lr_images = []
6     hr_images = []
7     for lr_image_name, hr_image_name in zip(sorted(os.listdir(lr_dir)), sorted(os.listdir(hr_dir))):
8         lr_image_path = os.path.join(lr_dir, lr_image_name)
9         hr_image_path = os.path.join(hr_dir, hr_image_name)
10        lr_img = cv2.imread(lr_image_path, cv2.IMREAD_GRAYSCALE)
11        hr_img = cv2.imread(hr_image_path, cv2.IMREAD_GRAYSCALE)
12        if lr_img is None or hr_img is None:
13            continue
14        lr_img_resized = cv2.resize(lr_img, target_size, interpolation=cv2.INTER_CUBIC)
15        hr_img_resized = cv2.resize(hr_img, target_size, interpolation=cv2.INTER_CUBIC)
16        lr_images.append(lr_img_resized)
17        hr_images.append(hr_img_resized)
18    lr_images = np.array(lr_images).astype('float32') / 255.0
19    hr_images = np.array(hr_images).astype('float32') / 255.0
20    lr_images = np.expand_dims(lr_images, axis=-1)
21    hr_images = np.expand_dims(hr_images, axis=-1)
22    return lr_images, hr_images

1 def preprocess_data(low_res_img, high_res_img):
2     low_res_img = tf.image.convert_image_dtype(low_res_img, tf.float32)
3     high_res_img = tf.image.convert_image_dtype(high_res_img, tf.float32)
4     return low_res_img, high_res_img
5 train_dataset = tf.data.Dataset.from_tensor_slices((low_res_data, high_res_data))
6 train_dataset = train_dataset.map(preprocess_data)
7 train_dataset = train_dataset.batch(16) # (batch size)

1 model.compile(optimizer='adam', loss='mean_squared_error', metrics=['accuracy'])
2 model.fit(train_dataset, epochs=10, verbose=1)

Epoch 1/10
50/50 [=====] 372s 7s/step - accuracy: 0.5736 - loss: 0.0302
Epoch 2/10
50/50 [=====] 374s 7s/step - accuracy: 0.7870 - loss: 0.0030
Epoch 3/10
50/50 [=====] 363s 7s/step - accuracy: 0.8242 - loss: 0.0019
Epoch 4/10
50/50 [=====] 381s 7s/step - accuracy: 0.8521 - loss: 0.0015
Epoch 5/10
50/50 [=====] 385s 7s/step - accuracy: 0.8610 - loss: 0.0015
Epoch 6/10
50/50 [=====] 379s 7s/step - accuracy: 0.8733 - loss: 0.0012
Epoch 7/10
50/50 [=====] 383s 7s/step - accuracy: 0.8802 - loss: 0.0012
Epoch 8/10
50/50 [=====] 382s 7s/step - accuracy: 0.8843 - loss: 0.0011
Epoch 9/10
50/50 [=====] 381s 7s/step - accuracy: 0.8845 - loss: 0.0011
Epoch 10/10
50/50 [=====] 364s 7s/step - accuracy: 0.8577 - loss: 0.0018

```

Bu model, oldukça temel bir mimariye sahip olsa da süper çözünürlük görevlerinde etkili sonuçlar verebilir. Modelin mimarisini ve katmanları tamamlandıktan sonra, eğitim sürecine geçtim.

Tarih ve İşyeri Amirinin İmzası

24/09/2024

Gün 17 :

Modelin Eğitimi ve Performans Analizi

Bugün, dün geliştirdiğim SRCNN modelini eğitmeye başladım. Eğitim süreci, veri setlerinin uygun şekilde hazırlanması ve modelin doğru parametrelerle eğitilmesi üzerine yoğunlaştı. Eğitim sürecini başlatmadan önce, modelin eğitiminde kullanacağım **optimizer** ve **kayıp fonksiyonlarını** seçtim. **Adam Optimizer** ve **Mean Squared Error (MSE)** kayıp fonksiyonunu kullanmaya karar verdim, çünkü bu kombinasyon genellikle görüntü işleme görevlerinde etkili sonuçlar sunar.

Eğitim sırasında, modelin her epoch'ta kaydettiği **kayıpları** düzenli olarak takip ettim ve grafikler yardımıyla bu kayipları görselleştirdim. Eğimin ilk aşamalarında, modelin kayipları hızla azaldı ve bu, modelin veri setinden öğrenmeye başladığının bir göstergesi oldu. Eğitim esnasında kaydettiğim bazı kodlar aşağıda yer alıyor:

```

1 import cv2
2 import numpy as np
3 import matplotlib.pyplot as plt
4 def test_image(model, low_res_image_path):
5     low_res_img = cv2.imread(low_res_image_path)
6     if low_res_img is None:
7         print("Error: Image not found or path is incorrect")
8         return
9     original_width, original_height = low_res_img.shape[:2]
10    low_res_img_resized = cv2.resize(low_res_img, (128, 128))
11    low_res_img_input = np.expand_dims(low_res_img_resized, axis=0)
12    predicted_img = model.predict(low_res_img_input)
13    predicted_img = np.squeeze(predicted_img)
14    predicted_img = np.clip(predicted_img / np.max(predicted_img), 0, 1)
15    predicted_img_resized = cv2.resize(predicted_img, (original_width, original_height))
16    plt.figure(figsize=(15, 15))
17    plt.subplot(1, 2, 1)
18    plt.imshow(cv2.cvtColor(low_res_img, cv2.COLOR_BGR2RGB))
19    plt.title('Low Resolution Image')
20    plt.axis('off')
21    plt.subplot(1, 2, 2)
22    plt.imshow(predicted_img_resized)
23    plt.title('High Resolution Image')
24    plt.axis('off')
25    plt.show()
26 low_res_image_path = '/content/DIV2K_train_LR_bicubic/X2/0005x2.png'
27 test_image(model, low_res_image_path)

```

Şekil 1



Şekil 1

Ancak, eğimin ilerleyen aşamalarında test için seçtiğim düşük çözünürlüklü görüntülerden biri üzerinde modelin hala bulanık çıktılar verdiği fark ettim. Bu bulanıklığın birçok nedeni olabilir. Denetleyicim bana bu sorun üzerinde çeşitli faktörlerin etkili olabileceğini belirtti:

- Modelin yeterince eğitilmemiş olması:** Model henüz yeterli sayıda epoch boyunca eğitilmediği için sonuçlar beklenenden daha düşük performans sergileyebilir.
- Kayıp fonksiyonu ve optimizasyonun uygun olmaması:** Yanlış seçilmiş kayıp fonksiyonları ve optimizasyon algoritmaları, modelin öğrenme hızını ve doğruluğunu doğrudan etkileyebilir.
- Eğitim verilerinin çeşitliliği:** Eğitim verilerinin çeşitliliği de modelin genel performansını artırmada önemli bir faktördür.

Buna rağmen, modelin %40 - %50 oranında eğitildiğinde bile bazı testlerde daha net ve yüksek çözünürlüklü görüntüler elde etmeye başladığımı fark ettim. Eğimin ilerleyen süreçlerinde modelin başarısını daha fazla artırmayı hedefliyorum ve optimizasyon seçeneklerini gözden geçireceğim.

Elde ettiğim sonuçlar ve analizler doğrultusunda, modelin eğitim sürecini daha da iyileştirmek için:

- Eğitim sürecine daha fazla epoch ekleyerek modelin daha derin öğrenme gerçekleştirmesini sağlamayı,
- Alternatif kayıp fonksiyonlarını ve optimizasyon stratejilerini denemeyi,
- Veri çeşitliliğini artırarak modeli daha geniş bir yelpazede test etmeyi planlıyorum.

Modelin Performansını Artırma ve Kayıp Fonksiyonlarının Değerlendirilmesi

Bugün stajimin on sekizinci gününde, önceki günlerde geliştirilen **SRCNN** modelinin performansını artırmak ve optimizasyon süreçlerini daha etkili hale getirmek için çalışmalara devam ettim. Modelin halen bazı test verilerinde bulanık sonuçlar üretmesi, eğitimin daha fazla derinleştirilmesi ve kayıp fonksiyonlarının yeniden değerlendirilmesi gerektiğini ortaya koydu.

İlk olarak, modelin eğitilme sürecini gözden geçirdim. Daha önce seçtiğim **Adam Optimizer'ı** kullanmaya devam ettim, ancak modelin daha hızlı ve doğru öğrenebilmesi için farklı optimizasyon stratejilerini test etmeye karar verdim. Bu bağlamda, **SGD (Stochastic Gradient Descent)** optimizasyonunu denemeye başladım. SGD, modelin daha hızlı öğrenmesini sağlarken, küçük adımlarla global minimuma daha stabil ulaşmasını destekleyebilir.

Ayrıca, modelin kayıplarını daha doğru bir şekilde ölçmek için **Mean Absolute Error (MAE)** kayıp fonksiyonunu ek olarak test etmeye karar verdim. MAE, MSE'ye kıyasla daha doğrudan bir kayıp fonksiyonu olarak bilinir ve özellikle süper çözünürlük gibi görsel kalitenin kritik olduğu alanlarda daha etkili olabilir. **MAE** ile **MSE** arasındaki farkları test ederek, hangisinin daha iyi sonuç verdiği analiz ettim.

Eğitim sürecinde MAE kayıp fonksiyonunun sonuçlarını incelediğimde, eğitim kayıplarının daha hızlı düşüğünü ve modelin doğrulama verileri üzerinde daha iyi performans sergilediğini gözlemledim. Aşağıda, eğitim sürecine dair bazı önemli kod parçaları ve grafikler yer almaktadır:

```
# Modelin MAE ile eğitimi
model.compile(optimizer='adam', loss='mae')

history = model.fit(train_images, train_labels, epochs=50, validation_split=0.2)

# Eğitim ve doğrulama kayıplarını görselleştirme
import matplotlib.pyplot as plt
plt.plot(history.history['loss'], label='Eğitim Kaybı (MAE)')
plt.plot(history.history['val_loss'], label='Doğrulama Kaybı (MAE)')
plt.legend()
plt.title("Eğitim ve Doğrulama Kayıpları (MAE)")
plt.show()
```

Bu noktada, **MAE** ve **MSE** kayıplarını karşılaştırarak modelin hangi kayıp fonksiyonuyla daha iyi sonuçlar verdienenini belirledim. Yaptığım analizler sonucunda, MAE'nin daha iyi sonuçlar ürettiğini ve eğitim sürecinde daha tutarlı bir performans sağladığını gözlemledim. Bununla birlikte, modelin doğrulama kayıplarını izleyerek ilerleyen epochlarda aşırı öğrenmeye (overfitting) maruz kalmamasını sağlamak için erken durdurma (early stopping) kriterlerini ekledim.

Bu değişikliklerin ardından, modelin doğrulama kayıplarında önemli bir düşüş elde ettim ve test verilerinde daha net, yüksek çözünürlüklü görüntüler elde etmeye başladım. Ancak bu iyileştirmeler sonucunda dahi bazı görüntülerde halen küçük detayların eksik olduğunu fark ettim.

Gün 19 :

Veri Çeşitlendirme ve Augmentasyon Stratejileri

Stajimin on dokuzuncu gününde, modelin performansını artırmak için veri çeşitliliğini artırma ve augmentasyon stratejilerini denemeye karar verdim. Eğitim verilerinin çeşitlendirilmesi, modelin daha geniş bir yelpazede öğrenme gerçekleştirmesini ve daha sağlam bir genel performans sergilemesini sağlar. **Veri augmentasyonu**, modelin eğitim verilerini yapay olarak genişletmenin en yaygın ve etkili yöntemlerinden biridir.

İlk olarak, veri augmentasyonu için **çevirmeler (flip)**, **rotasyonlar (rotate)**, **parlaklık ayarları (brightness adjustments)** ve **gürültü ekleme (noise addition)** gibi yöntemler kullandım. Bu işlemler, modelin her bir görüntüyü farklı varyasyonlarla öğrenmesini sağlayarak daha genelleştirilebilir sonuçlar elde etmesine yardımcı olabilir.

Aşağıda, veri augmentasyonu için kullanılan bazı kodlar yer almaktadır:

```

1  from tensorflow.keras.preprocessing.image import ImageDataGenerator
2  # Veri augmentasyonu
3  datagen = ImageDataGenerator(
4      rotation_range=20,
5      width_shift_range=0.2,
6      height_shift_range=0.2,
7      horizontal_flip=True,
8      brightness_range=[0.8, 1.2],
9      zoom_range=0.2
10 )
11
12 # Eğitim verilerini augmentasyon ile besleme
13 train_generator = datagen.flow(train_images, train_labels, batch_size=32)

```

Bu augmentasyon teknikleri, modelin eğitim sürecini daha sağlam bir hale getirdi. **Rotasyon** ve **çevrim** gibi basit işlemler bile modelin farklı açılardan görüntülerini anlamasını kolaylaştırdı. Ayrıca, **gürültü ekleme** işlemi, modelin daha gerçek dünyaya uygun hale gelmesine yardımcı oldu. Doğal görüntülerde çeşitli bozulmalar olabileceğinden, modelin bu tür durumlara karşı daha dirençli hale getirilmesi çok önemliydi.

Eğitim sürecinde augmentasyonlu verileri kullanarak modeli yeniden eğittim ve sonuçların, augmentasyonsuz eğitimlere kıyasla daha başarılı olduğunu gördüm. Özellikle, modelin doğrulama kayıplarının düşmesi ve daha net sonuçlar üretmesi augmentasyonun faydalarını kanıtlar nitelikteydi.

Son olarak, augmentasyon işlemi sonrası, modelin daha önce bulanık olan çıktılarında belirgin bir iyileşme gözlemledim. Daha net ve keskin detaylar üretmeye başlayan model, süper çözünürlük görevinde hedeflenen kaliteye yakın sonuçlar vermeye başladı.

Eğitim Sürecinin Tamamlanması ve Model Testleri

Bugün stajimin yirminci günü ve modelin eğitim sürecini tamamladım. Dünkü veri augmentasyonu stratejileri ve farklı kayıp fonksiyonları denemelerinden sonra modelin performansı önemli ölçüde arttı. Bugünkü çalışmalarla, modelin nihai testlerini gerçekleştirdim ve çeşitli düşük çözünürlüklü görüntüler üzerinde sonuçlarını değerlendirdim.

İlk olarak, modelin eğitim sürecini **100 epoch** boyunca tamamladım ve eğitim kayıplarının çok düşük seviyelere indiğini gözlemledim. Aşırı öğrenmeyi önlemek için kullandığım erken durdurma kriterleri sayesinde modelin gereğinden fazla eğitim almasının önüne geçtim ve böylece modelin doğrulama performansı da yüksek seviyede kalmaya devam etti.

Test sürecinde, düşük çözünürlüklü birkaç görüntü seçerek bunları modelde denedim. **SRCNN modeli**, ilk günlere kıyasla daha net ve keskin sonuçlar vermeye başladı. Test sonuçlarını görselleştirmek analizlerimi derinleştirdim:

```
1 # Test görüntüsünü modele verip çıktıyu elde etme
2 test_image = low_res_test_image.reshape((1, 256, 256, 1))
3 predicted_image = model.predict(test_image)
4
5 # Görselleştirme
6 plt.figure(figsize=(12, 6))
7 plt.subplot(1, 2, 1)
8 plt.title("Düşük Çözünürlüklü Görüntü")
9 plt.imshow(low_res_test_image, cmap='gray')
10
11 plt.subplot(1, 2, 2)
12 plt.title("Yüksek Çözünürlüklü Tahmin")
13 plt.imshow(predicted_image.reshape((256, 256)), cmap='gray')
14 plt.show()
```

Elde edilen sonuçlar, eğitim sürecinde kaydedilen iyileştirmelerin model performansına doğrudan yansındığını gösterdi. Modelin test sonuçları, başlangıçta yaşadığım bulanıklık sorununu büyük oranda çözmüş görünüyor. Ancak, daha karmaşık görüntülerde halen küçük hatalar olduğunu ve bu hataların giderilmesi için modelin daha fazla veri ile eğitilmesi gerektiğini düşündüm.

Stajimin bu son gününde, genel olarak modelin geliştirme, eğitim ve test süreçlerinde öğrendiklerimi toparladım. SRCNN modeli, süper çözünürlük görevlerinde etkili sonuçlar verebilecek bir model olsa da, daha karmaşık yapılar (örneğin, GAN tabanlı süper çözünürlük modelleri) daha yüksek performans sunabilir. Gelecekte bu tür daha gelişmiş modelleri araştırmayı planlıyorum.

STAJ DEFTERİNİN DOLDURULMASINDA VE DEĞERLENDİRİLMESİNE DİKKAT EDİLECEK GENEL İLKELER

1. Staj defteri Bölüm tarafından aksi belirtilmediği sürece mürekkepli/tükenmez kalemle ya da antet yapısı korunarak bilgisayar çıktısı alınarak doldurulabilir.
2. Defterdeki bilgiler okunaklı yazılmalı, kullanılacak şekil, tablo veya fotoğraflardan defter sayfalarını aşmayanlar ilgili bölüme konulmalı; sayfa yazım alanını aşanlar uygun boyutta katlanıp ek olarak verilmelidir. Bu ekler metin içindeki değişim sırasına göre "EK.1, EK.2 ..." biçiminde numaralandırılmış olmalıdır.
3. Çalışma yapılan laboratuvar veya fabrikalara ilişkin yerleşim planları verilmelidir. Çalışılan makine, cihaz ve ölçüm aletlerinin özellikleri ile temel çalışma prensipleri belirtilmelidir.
4. Sadece kitap, broşür gibi basılı kaynaklardan aktarılan bilgi ve şekilleri içeren defterler değerlendirilmeyecektir. Basılı kaynaklardan alınmış bilgi ve belgelere (şekil ve fotoğraf gibi) mutlaka referans gösterilmelidir.
5. Aynı işyerinde staj yapan öğrenciler aynı bilgi ve kaynaklardan yararlanmış olabilir. Ancak bu durum, defterlerin birbirinin aynısı veya çok benzeri olmasını gerektirmez. Defterler biçim ve içerik bakımından özgün olmalıdır.
6. Belirtilen yerlerinde işyeri sorumlusunun onayı bulunmayan defterler değerlendirilemez.
7. Staj Yönergesinde belirtilen zorunlu nedenlerle iki farklı alandaki staj aynı dönemde yapılmışsa, her alan için ayrı defter doldurulmalıdır.
8. Uygulama bitiminde staj yapılmış olan kurum tarafından gönderilmesi gereken değerlendirme formunun bölüme ulaştırılmasından öğrenci sorumludur.
9. Öğrenci, yaptığı staj juri önünde sunacak; yapılan sunum ve staj belgeleri birlikte değerlendirilerek staj kabul edilecek veya edilmeyecektir.
10. Her Bölümün stajlarda dikkat edilmesi gereken konularla ilgili istedikleri diğer hususlar aşağıda maddeler halinde verilmiştir.

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

1. Öğrencilerin mezun olmadan önce her biri en az 20 iş günü olan iki stajı tamamlamaları gerekmektedir. 20 günlük staj bir defada tamamlanır.
2. Öğrenci staj yapmak istediği yeri Staj komisyonuna önerir ve komisyon onayını alır. Komisyon tarafından onaylanmamış veya geçerli mazereti olmaksızın stajını eksik bırakınca öğrencinin stajı geçersiz sayılır.
3. Staj yapılan Kurum/Kuruluşa en az bir Bilgisayar Mühendisi olmalıdır.
4. Staj defteri, staj programına ve staj kurallarına uygun olarak, elle yazılacaktır. Her bir staj günü için en az bir sayfa olmak üzere toplamda en az 20 sayfa yazılmalıdır.
5. Staj süresince yapılan uygulama ve pratikler stajın yapıldığı kurumun çalışma alanından olacaktır.
6. Stajı süresince öğrencilerin staja devam edip etmedikleri staj komisyonunda ilgili kurumlar aranarak kontrol edilecektir. Staja devam etmeyen veya staj raporu uygun görülmeyen öğrencilerin stajları geçersiz sayılacaktır.
7. Öğrenciler aynı kurumda ve aynı zaman dilimleri arasında staj yapmış olsalar ve aynı projede çalışmış olsalar bile defterleri farklı olmak zorundadır. Aksi durumda ilgili öğrencilerin stajları geçersiz sayılır.
8. Staj yapılabilecek alanlar ve staj ile ilgili diğer detaylar bölüm staj komisyonuna bölümün web sayfasından ilan edilmektedir.

BİYOMÜHENDİSLİK BÖLÜMÜ

1. 20'şer iş gündünden oluşan stajlardan ilki laboratuvar ikincisi ise işletme stajı olacaktır.
2. Laboratuvar stajında Biyomühendislik/Biyoloji bilimiyle ilgili alanlarda laboratuvar çalışma düzeni, güvenlik önlemleri, kullanılan cihazlar ve analiz yöntemleri incelenip genel kurallarlığında rapor halinde sunulacaktır.
3. İşlette stajında ise prosesin akım şeması, yönetim organizasyonu, kütle ve enerji denklikleri oluşturulacak, işgücü analiz edilecektir. Proseste önerilecek iyileştirmeler öneri olarak sunulacaktır.

ÇEVRE MÜHENDİSLİĞİ BÖLÜMÜ

1. Stajlar laboratuvar/büro ve şantiye/isletme olmak üzere iki aşamalı olup ilk aşamada Laboratuvar/Büro stajı ikinci aşamada ise şantiye/isletme stajı yapılmalıdır.
2. Stajların yapılabileceği temel alanlar: Çevre kimyası, çevre mikrobiyolojisi, çevre biyoteknolojisi ile ilgili laboratuvar çalışmaları, su kirlenmesi ve kontrolü, su temini ve atıksuların uzaklaştırılması, su ve atıksu arıtma teknolojileri, hava kirlenmesi ve kontrolü, katı ve tehlikeli atıkların yönetimi ve bertarafı, gürültü kirliliği kontrolü, endüstriyel atıkların yönetimi, çevresel etki değerlendirmesi, çevre yönetimi ve planlaması.
3. İki dönemlik zorunlu stajın her biri 20'şer iş gündünden oluşmalı ve stajlar aynı temel alanları içeren kurumlarda yapılmamalıdır.
4. Staj raporu, yazım kurallarına uygun bir şekilde mürekkepli kalemlle hazırlanmalıdır.
5. Staj ile ilgili işletmenin akım şeması çizilmeli ve atık oluşturan birimler, atık miktarları ve atık yükleri ayrıntılı bir şekilde verilmelidir.
6. Yapılan deneylerin ve kullanılan cihazların özellikleri, prensipleri, deneyleri amaçları ve analiz sonuçlarının yorumlanı mutlaka verilerek standartlarla karşılaştırılmalıdır.
7. Staj süresince yapılan işler ve/veya elde edilen veriler çevre mevzuatı ile ilişkilendirilip yorumlanmalıdır.

ELEKTRİK-ELEKTRONİK MÜHENDİSLİĞİ BÖLÜMÜ

1. İlk staja 2.sınıftan en az 40 AKTS, ikinci staja da 3. sınıfından en az 30 AKTS ders alan öğrenciler başvurabilir.
2. İki staj ayrı iş yerlerinde yapılmalıdır. Özel durumlar Staj Komisyonunun iznine tabiidir. Her stajda, işyerinin farklı birimlerinde bulunulmaya çalışılmalıdır.
3. Staj yapılacak işyerinde, en az 1 (bir) Elektrik/Elektronik/Elektrik-Elektronik Mühendisi olmalıdır.
4. a. Staj defteri mürekkepli ya da tükenmez kalemle günlük iş planına göre düzgün bir şekilde elle yazılmalı; gerekiyorsa fotoğraflar, çizimler ve projeler defterin ekinde verilmelidir.
b. Staj defteri günlük yazılmalı, yazılınlar işyeri ile uyumlu olmalıdır. Defter resimlerle ya da şekillerle doldurulmamalıdır.
c. Defterde ilgili yerlerde sorumlu mühendis bilgileri açık olmalı, kurum kaşe ve mührü mutlaka vurulmalıdır. Her sayfada, sorumlu kişinin imzası olmalıdır.
d. Staj defterlerinin başında staj yerinin organizasyonu ve faaliyetleri hakkında bilgi verilmelidir (en fazla 3 sayfa olabilir).
e. Staj defterleri birbirinin aynı olan öğrencilerin stajları kabul edilmeyecektir.
5. Staj sonunda, staj değerlendirme formu kurum tarafından doldurulduktan sonra posta ile bölüme gönderilebilir ya da gizli kaşesi taşıyan kapalı ve mühürlü zarf içinde öğrenciye teslim edilebilir. Değerlendirme formu kapalı zarf içinde öğrenciye teslim edilmişse, öğrenci bölüme teslim eder.
6. Defter ve (öğrenciye teslim edilmişse) staj değerlendirme formu öğretim yılının ilk 15 günü içerisinde (staj dersleri 1 ve 2 alınması dahi) bölüme imza karşılığında teslim edilmelidir. Teslim etmeyen öğrencilerin stajları kabul edilmeyecektir. Defterler öğrenciye geri verilmeyeceğinden dolayı, öğrenci teslim edilmeden önce sunum hazırlıkları için gerekli dokümanları almalıdır.
7. Öğrenciler, öğretim yılının ilk 15 günü içerisinde staj değerlendirme formunun bölüme ulaşıp ulaşmadığını öğrenmelidir. Eğer ulaşmamış ise staj yaptıkları kurumla iletişime geçerek, formların bölüme ulaşmasını sağlamalıdır.
8. Staj hariç tüm derslerinden başarılı olanlar ve Sınav haftalarını kapsamayacak tarihlerde olmak şartıyla, bulunduğu dönemdeki derslerin hiç birisinden devam mecburiyeti olmayan öğrenciler Öğretim dönemi içerisinde staj yapabilir.
9. İlk defa staj yapacak olan 2. sınıf öğrencileri önceden en az bir staj sunumu dinlemelidir. Aksi halde, staj evrakları onaylanmayacaktır.

İNŞAAT MÜHENDİSLİĞİ BÖLÜMÜ

Staj şantiye ve büro stajı olmak üzere iki alanda yapılacaktır.

Şantiye Stajı:

Yapı, yol, su yapıları vb. şantiyelerden birisinde proje uygulaması, çeşitli imalatların yapılması veya denetleme konularını kapsar.

Büro Stajı:

Fiyat Analizleri keşif özetleri, İhale, Çeşitli projelerin düzenlenmesi (mimari statik, yol, su getirme ve kanalizasyon vb. Dinamik Hidrolik hesaplamaların yapılması, İnşaat Mühendisliği ile ilgili paket programları kapsar.

JEOLOJİ MÜHENDİSLİĞİ BÖLÜMÜ

Staj, jeoloji mühendisliğiyle ilgili arazi veya laboratuvar çalışması gerçekleştirilen kurum veya kuruluşlarda yapılmalıdır.

KİMYA MÜHENDİSLİĞİ BÖLÜMÜ

1. İşletmenin her bölümünü; hammadde, ürün, reaksiyon şartları ve kapasite yönünden dikkatle incelenmelidir.
2. Her bölüm için ayrı ayrı ve tüm proses için enerji ve madde balansı yapılmalıdır.
3. İşletmenin her bölümünün akış şemaları ve bu böltümeler arasındaki ilişkiye gösteren bir total akış şeması çizilmelidir.
4. Tesiste bulunan önemli cihazların özelliklerini, yapıları ve çalışma prensipleri incelenmelidir.
5. İmalat ve kalite kontrol için yapılan analizler hakkında bilgiler derlenmelidir.
6. İşletmede kullanılan yardımcı tesisler ve bunların işletme ekonomisi, çevresel duyarlılık ve teknolojik gelişme yönlerinden katkılarını da belirten bilgiler derlenmelidir.
7. İşletmenin organizasyon şeması oluşturulmalıdır.
8. İşletmede elde edilen ürünler için maliyet analizlerine ilişkin bilgiler derlenmelidir.
9. İşletmede kullanılan üretim teknolojilerinin modern teknolojilerle karşılaştırılması yapılarak verimliliğin ve kapasitenin artırılması ve maliyetlerin düşürülmesi için neler yapılabileceğini belirten bir değerlendirme yapılmalıdır.

MAKİNE MÜHENDİSLİĞİ BÖLÜMÜ

Stajlar aşağıda belirtilen konuları kapsayan iki farklı alanda yapılmalıdır.

I. Alan: Atölye

Atölye alanındaki çalışmaların 10 iş gününü talaşlı imalat, 5 iş gününü döküm, 5 iş gününü de kaynak ve şekillendirme işlemleri oluşturur. Atölye çalışması; imalat yöntemleri, imalattaki iş sırası ve imalat makinalarının tanımı, belirgin özellikleri ve çalışma sistemlerini kapsar. Teknolojik bilgi, gözlem, imalat resimlerinin çizimleri ve uygulamaya dayanır. Bu çalışmalar mümkün olduğunda, seri üretim yapılan ve tam teşekküllü atölyeleri bulunan kurumlarda yapılır.

Talaşlı İmalat Yöntemleri: Bu bölümde torna, taşlama, freze, matkap, planya gibi tezgâhlarda talaş kaldırma işlemleri bilfil takip edilmelidir. Bu işlemler sırasında kullanılan her türlü alet ve tezgâhların özellikleri araştırılıp tezgâh üzerinde uygulama yapılacak, tezgâhlarda imal edilen parçaların teknik resimleri norm ve standartlara uygun şekilde kurşun kaleme deftere çizilerek, parçaların tezgâha bağlama ve işleme yöntemleri kısaca açıklanacaktır. Ayrıca varsa bilgisayar destekli tezgâhlarla ilgili program hazırlanması, ofset işlemleri ve tezgâhlarda parçaların işlenmesi takip edilerek gerekli açıklamalar yazılacaktır.

Döküm: Dökümcülük, Döküm Kalıcılığı ve Maden Ergitme Tekniği olarak iki grup altında toplanabilir. Staj sırasında, genel döküm bilgileri ışığında dökümcülükte kullanılan ocaklar, kapasite ve verimleri, dökümcülük alet ve gereçleri, kalıplama yöntemleri, kalıba ergiyik metalin dökülmesi, döküm sonrası işlemlerin değerlendirilmesi ve imalat resimlerinin çizilmesi, döküm çeşitleri ve döküm işleminde dikkat edilecek hususlar incelenip deftere yazılmalıdır.

Kaynak ve Plastik Şekil Verme: Kaynak yöntemleri hakkında bilgi verilip yapılan uygulamalara ait Teknik Resimler çizilip gerekli açıklamalar yapılmalıdır. Plastik şekil verme, dövme, haddeleme ve saç işleme gibi işlemlerin özellikleri incelenmeli ve gerekli açıklamalar yapılmalıdır.

II. Alan: Fabrika Organizasyonu ve Yönetimi

Fabrika Organizasyonu ve Yönetimi alanındaki stajlar, ürün ve/veya hizmet üreten işletmelerde yapılabileceği gibi ısitma, soğutma ve havalandırma projelerini yapan işletmelerde yapılabilir. 20 iş gününü kapsayan uygulama aşağıdaki konularda olmalıdır.

Fabrika Organizasyonu ve Yönetimi (süresi 2 hafta): Fabrikanın örgütsel yapısı, fabrikada yer alan iş etüdü çalışmalarının araştırılması, üretim planlama ve kontrol teknikleri, iş güvenliği, işçi-işveren ilişkileri, satın alma işlemlerinin uygulanış şekli, hammadde temini, depolama ve stoklanmanın işletme içindeki önemi, stok bulundurma nedenleri, stok kontrolde maliyet unsurları, bakım üniteleri ve hedefleri, üretimi artırma çabaları, kalite kontrol düzenleri, toplam kalite yönetimine ilişkin çalışmaların tespiti, güç ve enerji ünitelerinin analizi (elektrik dağıtım şebekesi bağlantı ve güçleri ile), AR-GE faaliyetlerinin araştırılması.

Üretim ve Montaj İşlemleri (süresi 2 hafta): Üretimi yapılan malzeme ve teçhizatın projelendirme aşamalarının etüdü; üretimde kullanılan tezgâh ve makinelerde iş akışı ve imalat zamanının incelenmesi; montajda uygulanan yöntem ve teknikler belirlenerek varsa önerilerle birlikte deftere yazılır.

MEKATRONİK MÜHENDİSLİĞİ BÖLÜMÜ

Her biri 20 iş gününden ibaret, staj-1 ve staj-2 olarak belirtilen stajlar farklı kurumlarda yapılmalıdır. Ancak büyük ölçekli kurumların farklı yerlesimlerde yer alan, farklı birimlerinde(AR-GE, Üretim, Tasarım, Kalite vb. birimlerinde) öğrenciler her iki stajını da yapabilirler.

1. Öğrenciler stajlarını, Mekatronik Mühendisinin bulunduğu birimlerde yapmalıdır. Şayet Mekatronik Mühendisi bulunmuyorsa, Makine Mühendisi veya Elektrik Elektronik Mühendisinin bulunma şartı aranır.

2. Staj defterleri, defter sayfalarındaki formatın korunması şartı ile bilgisayar çıktısı şeklinde yazılarak hazırlanabilir. Zorunlu kalınması halinde el yazısı ile staj defteri yazılabilir.

3. Staj defterinde stajın yapıldığı her bir günün tarihi açıkça belirtilmelidir, her bir sayfası staj yapılan ilgili birimin mühendisi tarafından imzalanıp kaşelennelidir. Kaşede unvan ve diploma numarası belirtilmiş olmalıdır.

4. Staj yapacak öğrenciler, staj defterlerinde, staj yaptığı kurumu tanıtan bilgilerini, ilgili birimin faaliyet alanlarını, ürün ile ilgili detaylı teknik bilgileri, öğrencinin bizzat kendisi tarafından gerçekleştirmiş olduğu en az 5 farklı uygulamaları kapsayacak biçimde, açıklayıcı bir dilde yazarak aktarmalıdır. Staj defterine aktarılan bilgiler gerek görüldüğünde teknik çizimlerle, resimlerle ve açıklayıcı şemalarla desteklenmelidirler.

5. Staj yapacak öğrenciler uygulamalarını mekanik ve elektrik-elektronik sistemlerini barındıran, yazılım içerikli ürünler üzerine gerçekleştireceklerdir.

METALURJİ VE MALZEME MÜHENDİSLİĞİ BÖLÜMÜ

Staj raporu, staj defterindeki biçim korunmak şartıyla ve 20 sayfadan az olmamak kaydıyla okunaklı ve tertipli bir şekilde el yazısıyla veya bilgisayar çıktısı olarak hazırlanabilir.

Stajlar için demir-çelik fabrikaları, döküm fabrikaları, otomotiv ve makine imalat sanayi, alüminyum ve bakır gibi demir dışı metal üretim fabrikaları, şşe -cam sanayi ve ıslı işlem fabrikaları gibi metaltürji ve malzeme konusunda faaliyet gösteren fabrikalar seçilebilir.

Stajlar aşağıda belirtilen konuları kapsayan iki farklı alanda yapılmalıdır.

I. Alan: Üretim

- Öğrencilerin, Metalurji ve Malzeme Mühendisliği kapsamı içerisinde giren üretim stajı yapmaları gerekmektedir.
- Öğrenciler yukarıda anılan sektörlerde 20 iş günü staj yapmakla sorumlu olup, bu sektörlerde yaptıkları staj süresince üretime bafil katılmak şartı aranır.

II. Alan: Fabrika Organizasyonu ve Yönetimi

- Fabrikanın organizasyonu, fabrikada yer alan iş etüdü ve çalışmaları araştırılması, üretim planlama ve kontrol teknikleri, iş güvenliği, işçi-işveren ilişkileri, satın alma işlemlerinin uygulanışı, ham madde temini, depolama ve stoklamanın işletme içindeki önemi, stok bulundurma nedenleri, stok kontrolde maliyet unsurları, bakım üniteleri ve hedefleri, üretimi artırma çabaları, kalite kontrol düzenleri, toplam kalite yönetimine ilişkin çalışmaların tespiti, güç ve enerji ünitelerinin analizi (elektrik dağıtım şebekesi bağlantı ve güçleri ile), araştırma-geliştirme (AR-GE) faaliyetlerinin araştırılması incelenecaktır.
- Fabrika Organizasyonu ve Yönetim Stajı 20 iş günüdür.

YAZILIM MÜHENDİSLİĞİ BÖLÜMÜ

- Staj raporunda uygulamalar ağırlıklı olarak yer alacak ve gerekli yerlerde teorik bilgi verilecektir.
- Eğer günlük iş yerine proje üzerinde çalışma yapılmışsa, staj raporunda günlük anlatım yerine proje ve nasıl gerçekleştirildiği açıklanmalıdır.
- Staj raporu bilgisayar çıktısı olarak hazırlanacaktır.
- Staj yapılan yerde mutlaka yazılım mühendisi bulunacaktır. Bu mümkün değilse, bilgisayar mühendisi bulunacaktır.
- Staj raporunun yazılmasında F.Ü. Fen Bilimleri Enstitüsü yüksek lisans tez yazım kuralları esas alınacaktır.
- Aksi ilan edilmedikçe staj sunumları, Güz döneminin 6. haftasının Çarşamba günü ilgili jüri huzurunda yapılacaktır.
- Staj belgelerinin benzerlik oranı test edilecek ve sonuç rapor sunulacaktır.