

BİLİM TÜRKİYE
YAPAY ZEKA YAZ OKULU
DERS PLANLARI KİTABI

Cilt – 2
(3. ve 4. Haftalar)

Hazırlayan
Doç. Dr. Utku KÖSE
Süleyman Demirel Üniversitesi, Türkiye
Kuzey Dakota Üniversitesi, ABD.

Haziran 2024

Kitap Kullanım Kılavuzu

Bilim Türkiye Yapay Zeka Yaz Okulu ders planlarını içeren bu kitap, öğretmenler ve öğrencilerin dersler bazında işlenecek konuları ve gerçekleştirilecek uygulamalar karşısındaki rollerini belirlemek adına yönlendirici açıklamalarla donatılmış durumdadır. Konuların haftalar ve dersler içerisindeki sırası ve düzeni, Yapay Zeka uygulamalarının öğrenciler tarafından bilgi ve beceri boyutunda kolay özümsenmesi adına tasarlanmıştır. Dersler içerisinde sunulan açıklamalar öğretmenlere ve öğrencilere yol gösterici olmakla beraber aşamalı bir öğretim-öğrenim sürecini de desteklemektedir. Bununla birlikte öğretmenler dersler esnasında öğrencilerden gelen dönütler ya da hakim oldukları konularda öğrencilere aktarmak istedikleri teorik ya da uygulamalı olgular konusunda inisiyatif kullanabileceklerdir.

Sunulan her bir ders planı öncelikli olarak ders ya da derslerin açılış planı ile başlamaktadır. Bu planlar genel olarak **Ders Etiketleri** altında işlenecek *Konu*, hedef öğrenci kitlesinin *Sınıf* düzeyi, toplam *Süre*, öğrencilerden beklenen *Hazır Bulunuşluk* durumları ve işlenecek konuları temsil eden *Anahtar Kelimeler* ile desteklenmiştir. Ders Etiketleri ardından **Beceriler** başlığı altında öğrencilerden ders(ler) esnasında beklenen beceri durumları listelenmiştir. Öğretmenler ilgili beceri durumlarını dikkate alarak öğrencileri yönlendirebilecek, gözlemleyebileceklerdir. Yine ders(ler) sonunda öğrencilerin kazanması beklenen bilgi ve beceri durumları **Kazanımlar** başlığı altında, sınıf ortamında kullanılması beklenen materyaller ise **Ders Materyalleri** tablosu içerisinde listelenmiştir. **Öğretmen Hazırlık Çalışmaları** başlığı altında öğretmenlerin ders öncesi okuyabilecekleri ve izleyebilecekleri kaynaklar listelenmekte, **Atölye Hazırlık Çalışmaları** başlığı altında ise yine derslere hazırlık adına tavsiye edilen ön-hazırlık yaklaşımları ifade edilmektedir. Sunulan ders planı içerisinde kullanılan kaynaklar ise **Kaynakça** başlığı altında listelenmektedir.

Etkin bir öğretim-öğrenim süreci için ders planlarının beş aşamalı; artırımlı ve ilerleyici bir düzende temsil edilen adımlar altında sunumu gerçekleştirilmiştir. Ders açılış planlarında **Genel Bakış** başlığı altında kısaca ifade edilen ve gerekli noktalarda ders planı içeriklerinde detaylandırılan adımlar ve işlevleri genel hatlarıyla şu şekildedir:



1. ADIM: HAREKETE GEÇ: İşlenecek olan konulara yönelik öğrenci ilgisini çekecek aktivitelerin gerçekleştirilmesi, gerekli teorik konuların aktarılması.



2. ADIM: KEŞFET: Bir önceki adımda gerçekleşen ilgi çekme ve teorik girizgahlar ardından ilerleyen konuların ve uygulama adımlarının işlenmeye başlanması, gerekli noktalarda yeni bilgi ve becerilerin elde edilmesi yönünde aksiyonların alınması.



3. ADIM: ÜRET: Bir önceki adımdaki aktivitelerin üzerine yeni aksiyonların alınarak işlenen konu kapsamının derinleştirilmesi ve gerekirse nihayete erdirilmesi.



4. ADIM: İLERLET: Öğrencilerin işlenen konular ve gerçekleştirilen uygulamalar üzerine kendi inisiyatifleri ile düzenlemeler gerçekleştirilmesi, değerlendirmelerde bulunması.



5. ADIM: DEĞERLENDİRME: İşlenen konuların genel değerlendirmesinin yapılması ve önerilen sorular kapsamında ve alternatif soru-cevap süreçleriyle pekiştirmelerin sağlanması.

3. HAFTA – 1. GÜN – 1. DERS: OPENCV KÜTÜPHANESİ TANITIMI

DERS PLANI

DERS ETİKETLERİ



KONU: OpenCV Kütüphanesi Tanıtımı



SINIF: 12-14 Yaş



SÜRE: 60 dk.



HAZIR BULUNUŞLUK: Temel İngilizce eğitimi, Yapay Zeka farkındalığı, Programlamaya yönelik temel bilgi



ANAHTAR KELİMELER: Yapay Zeka, Kodlama, OpenCV



BECERİLER:



Tahmin



Sorgulama



Günlük hayat bağlantıları kurma



KAZANIMLAR:

Bu derste öğrenciler;

- Görüntü işleme kavramı hakkında genel bilgi sahibi olur.
- OpenCV kütüphanesi hakkında genel bilgi sahibi olur.
- Yapay Zeka uygulamaları geliştirmede OpenCV kütüphanesinden nasıl yararlanabileceği konusunda farkındalık elde eder.



DERS MATERİYALLERİ:

Malzeme	Miktar
Yazı Tahtası	Öğrenci sayısı ve sınıf büyüklüğüne göre en az bir adet
Projeksiyon	Öğrenci sayısı ve sınıf büyüklüğüne göre en az bir adet
Defter, Kalem	Öğrenci sayısı kadar
Bilgisayar	Öğrenci sayısının yarısı kadar
Tablet	Öğrenci sayısının yarısı kadar



ÖĞRETMEN HAZIRLIK ÇALIŞMALARI:

Okunabilecek Kaynaklar:

Bradski, G., & Kaehler, A. (2008). *Learning OpenCV: Computer vision with the OpenCV library*. O'Reilly Media, Inc.

Samtaş, G., & Gülesin, M. (2011). Sayısal görüntü işleme ve farklı alanlardaki uygulamaları. *Ejovoc (Electronic Journal of Vocational Colleges)*, 2(1), 85-97.

İzlenebilecek Kaynaklar:

Karmaşık Sistemler ve Veri Bilimi. (2020). Görüntü İşleme Nedir?. Çevrimiçi: <https://www.youtube.com/watch?v=OviHYvtgRFA> (Erişim 17 Haziran 2024).

Kocaaslan, B. (2022). Görüntü İşleme Nedir?. Çevrimiçi:

https://www.youtube.com/watch?v=zEIMz_qQKyk&list=PLP6TjrWzAOA1wkUS_0GDWjk0dBWC6z_Ru (Erişim 17 Haziran 2024).

Programming Knowledge. (2019). Welcome to the Hugging Face course. Çevrimiçi: <https://www.youtube.com/watch?v=kdLM6AOd2vc&list=PLS1QulWo1RIa7D1O6skqDQ-JZ1GGHK-K> (Erişim 17 Haziran 2024).

ATÖLYE HAZIRLIK ÇALIŞMALARI:

Öğretmen;

- Dersten önce ders materyallerini hazırlamalıdır.
- Görüntü işleme hakkında ön bilgi edinmelidir.
- OpenCV kütüphanesi hakkında ön bilgi ve beceri sahibi olmalıdır.



KAYNAKÇA:

Acharya, T., & Ray, A. K. (2005). *Image processing: principles and applications*. John Wiley & Sons.

Garcia, G. B., Suarez, O. D., Aranda, J. L. E., Tercero, J. S., Gracia, I. S., & Enano, N. V. (2015). *Learning image processing with OpenCV*. Birmingham (UK): Packt Publishing.

Howse, J. (2013). *OpenCV computer vision with python* (Vol. 27). Birmingham, UK: Packt Publishing.

Mu, Q., Wang, X., Wei, Y., & Li, Z. (2021). Low and non-uniform illumination color image enhancement using weighted guided image filtering. *Computational Visual Media*, 7, 529-546.

GENEL BAKIŞ:

- 1) Harekete Geç:** Öğrencilere Yapay Zeka uygulamalarının görsel verileri daha kolay kullanabilmesi için neler yapılabileceği sorular ve görüşler tartışılar.
- 2) Keşfet:** Öğrencilere görüntü işleme kavramı hakkında genel bilgi verilir.
- 3) Üret:** Öğrencilere OpenCV kütüphanesi genel hatlarıyla açıklanır.
- 4) İlerlet:** Öğrencilere bireysel olarak görüntü işleme ve OpenCV kütüphanesinin faydalı olacağı Yapay Zeka uygulama fikirleri üzerinden düşünmeleri ve bu fikirler hakkında genel açıklamaları yazıya dökmeleri istenir.
- 5) Değerlendir:** Yazılan açıklamalar ve derste öğrenilenler ışığında soru-cevap etkinliği gerçekleştirilerek görüntü işleme kavramı ve OpenCV kütüphanesine dair bilgiler pekiştirilir.

UYGULAMA



1. ADIM: HAREKETE GEÇ

Öğrencilere Yapay Zeka uygulamalarının görsel verileri daha kolay kullanabilmesi için neler yapılabileceği sorular ve görüşler tartışılar. İlgili süreç için 10 dakikalık bir zaman ayırılır.



2. ADIM: KEŞFET

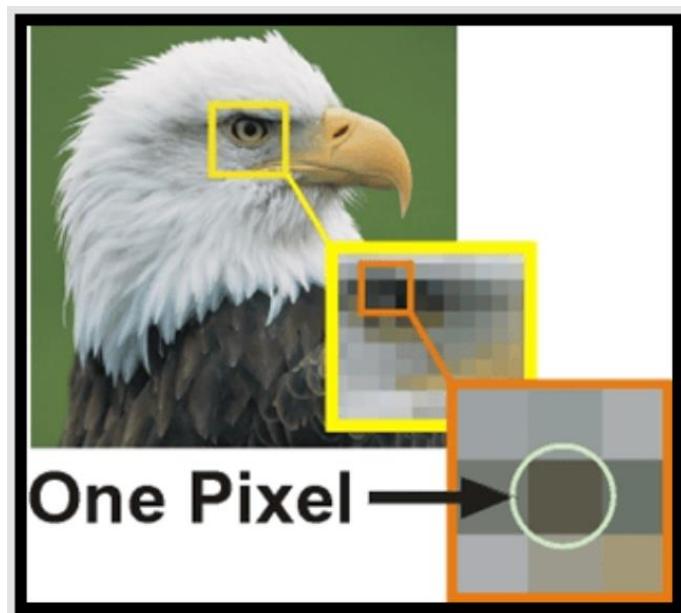
Öğrencilere görüntü işleme kavramı hakkında genel bilgi verilir.

Görüntü İşleme

Bilgisayar tabanlı sistemlerin gelişimiyle beraber farklı türden verilerin işlenmesi ve anlaşılmaları da her geçen zaman daha kolay hale gelmiştir. Bu durum özellikle farklı

türden verilerin bilgisayar ortamına aktarımı ve kullanımı aşamalarında karşılaşılan zorlukların ve çabaların karşısında daha pratik çözümlere olan ihtiyaçlar neticesinde ortaya çıkmıştır. En basitinden sayısal ve metin tabanlı verilerin bilgisayarlarda işlenmesi nispeten hızlı ve pratiktir. Ancak dış ortamdan alınan görüntü ya da ses verilerinin kullanımı daha karmaşık adımları da gerektirir. Özellikle görüntü verileri, gelişen görüntüleme teknolojilerinin de bir getirisidir, teknolojik araçların olmazsa olmaz unsurları haline gelmiş, bu nedenle gerçek dünyadan alınan görüntülerin bilgisayarlarda işlenmesi ya da bilgisayarlarda tamamen dijital görüntülerin oluşturulması için özel çözüm araçları ortaya çıkmıştır.

Dijital ortamda temsil edilen görüntüler, **piksəl** adı verilen küçük görüntü birimlerinin bir araya gelmesiyle oluşmakta ve pikseller renk, parlaklık, doygunluk gibi farklı parametrelerin etkisi altında farklı sayısal değerlerin görsel versiyonları halinde temsil edilmektedir. Temsil edilen piksellerin çeşitli matematiksel fonksiyonlar yardımıyla manipüle edilmesi ve böylelikle genel görüntünün farklı versiyonlara dönüştürülmesi ve mümkün hale gelmektedir (Şekil 1). Bu tür dönüşüm ve manipülasyon araçlarının geliştirildiği teknoloji alanı kısaca “**görüntü işleme (image processing)**” olarak adlandırılmaktadır (Acharya, & Ray, 2005).



Şekil 1. Dijital görüntülerde pikseller.

Görüntü işleme aslında Photoshop ya da Corel Draw gibi yazılımlarda ve hatta MS Paint gibi temel görüntü düzenleme ortamlarında sıkılıkla kullanılmaktadır. Fotoğraflardan belli alanların çıkarılması, renkli bir fotoğrafın siyah beyaz hale getirilmesi (ya da tam tersi işlemler), bir görüntünün parlaklığının ayarlanması ya da “**filtre**” adı verilen araçlarla görüntüde ileri düzey manipülasyonların yapılması görüntü işleme algoritmaları sayesinde gerçekleşmektedir. Genel olarak tipik bir görüntü işleme süreci şu işlemleri içerebilmektedir (Garcia vd., 2015):

- Görüntü parlaklık, doygunluk ve renk durumlarının oynanması.
- Görüntü genişlik ve yükseklik gibi parametrelerin değiştirilmesi.
- Görüntüden belli parçaların alınması ya da görüntüde belli bölümlerin yok edilmesi.
- Görüntünün manipülasyonlarla orijinalinden farklı şekilde bir versiyona dönüştürülmesi.

- Farklı görüntülerin bir araya getirilerek yeni bir görüntü elde edilmesi.

İlgili işlemlerin yapılabilmesi için görüntü işleme teknolojisi genel olarak bazı dönüşüm fonksiyonlarını kullanmaktadır. Görüntü işleme algoritmaları Yapay Zeka algoritmalarında olduğu gibi tahmin ve tanımlama odaklı, “probleme uyum sağlayan” çözümler ortaya koymamakta, daha çok pikseller üzerinde dönüşümler yaparak genel görüntünün (görselin) değişmesini sağlamaktadır. Ancak bu değişimler görüntünün farklı versiyonlara dönüşmesini ya da ileri düzey veri işleme aşamalarında daha pratik bir şekilde temsil edilmesini sağladığı için görüntü işleme teknolojisi popüleritesini artırarak sürdürmüştür.



Şekil 2. Görüntü işleme ile örnek bir görsel üzerinde farklı sonuçlar (Mu vd., 2021).

Görüntü işleme ileri düzey veri işleme aşamalarında daha fazla kolaylık ve daha etkili sonuçlara neden olduğu için Yapay Zeka uygulamalarında bir tür “veri önişleme” aracı olarak kullanılabilir, hatta Derin Öğrenme modellerinin içerisinde entegre bir biçimde yer almaktadır. Bu nedenle görsel veri setleri içeren Yapay Zeka uygulamalarında görüntü işleme aşamaları büyük önem taşımaktadır.



3. ADIM: ÜRET

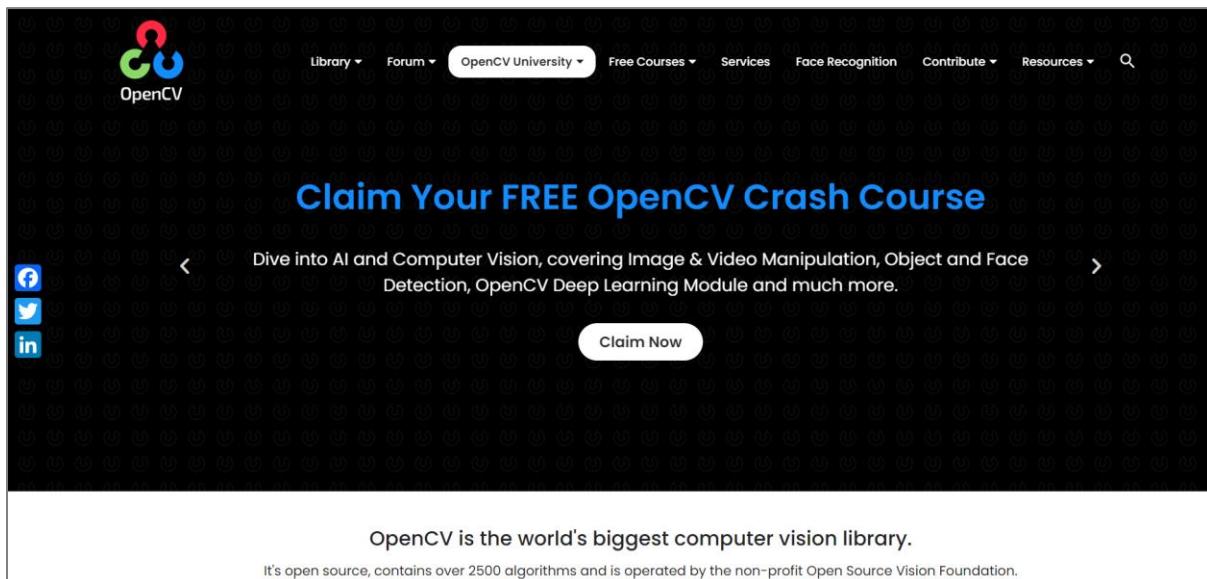
Üret adımında öğrencilere OpenCV kütüphanesi genel hatlarıyla açıklanır.

OpenCV Kütüphanesi

OpenCV (Open Source Computer Vision Library) görüntü işleme uygulamaları için yaygın bir şekilde kullanılan, açık kaynak kodlu bir kodlama kütüphanesidir. İlk olarak Intel firması tarafından geliştirilmeye başlanmıştır, ardından Willow Garage ve yine Intel tarafından satın alınan Itseez firması tarafından güncellenmeye devam etmiştir. Görüntü işleme çözümlerinin Yapay Zeka gibi ileri teknolojilerle bir arada kullanımı neticesinde, algılanan dış dünyaya göre kararlar oluşturan sistemlerin geliştirildiği bilgisayarlı görü (computer vision) çözümleri de OpenCV kütüphanesinin etkin bir biçimde faydalandığı faaliyet alanları içerisinde yer almaktadır.

almaktadır. Esasında C ve C++ programlama dilleri ile yazılmış olan OpenCV, Python gibi güncel programlama dilleriyle kullanılabilen şekilde genişleme bağlantıları da içermektedir (Howse, 2013).

Öğrencilerden <https://opencv.org/> adresi üzerinden OpenCV resmî Web platformunu görüntülemeleri istenir. OpenCV Web platformu, ilgili kütüphaneyle alakalı sürüm bilgilerinin, örnek uygulamaların, ücretsiz eğitimlerin ve kütüphane üzerinden geliştirmeler yapan kullanıcıların sorularını / görüşlerini paylaştıkları forum ortamlarının sunulduğu kapsamlı bir Web sitesidir (Şekil 3).



Şekil 3. OpenCV kütüphanesi Web platformu.

Öğrencilerin OpenCV Web platformu üzerinde sunulan linklere tıklama suretiyle farklı sayfaları incelemeleri istenir ve platform ortamında sunulan farklı sayfalar gözlemlenerek daha önce Python kütüphaneleri Web sayfaları (Bkz. 1. Hafta – 3. Gün) ve Hugging Face, Kaggle gibi platformlarla (Bkz. 1. Hafta – 4. Gün) olan benzerlikler ya da farklılıklar irdelenir.

OpenCV kütüphanesi açık kaynak paylaşım ilkeleri bağlamında GitHub ortamında çeşitli uygulama örneklerini de kullanıcılar sunmaktadır (Şekil 4). Öğrencilerden <https://github.com/opencv/opencv> linkini ziyaret ederek OpenCV GitHub kod paylaşım ortamını incelemeleri istenir.

The screenshot shows the GitHub repository page for 'opencv / opencv'. At the top, there are navigation links for Product, Solutions, Resources, Open Source, Enterprise, and Pricing. On the right, there are search, sign-in, and sign-up buttons. Below the header, the repository name 'opencv / opencv' is shown with a 'Public' status. A navigation bar includes links for Code, Issues (2.4k), Pull requests (156), Discussions, Actions, Projects (2), Wiki, Security, and Insights. A dropdown menu for 'Code' is open, showing options like '4.x' (selected), '6 Branches', and '125 Tags'. The main content area displays a list of recent pull requests from 'kaingwade'. Each pull request includes a summary, date, and number of commits. To the right, there's an 'About' section with repository details: 'Open Source Computer Vision Library', 'opencv.org', and various tags: 'opencv', 'c-plus-plus', 'computer-vision', 'deep-learning', and 'image-processing'. Other sections include Readme, Apache-2.0 license, Security policy, Activity, Custom properties, 76.8k stars, 2.7k watching, and 55.7k forks.

Şekil 4. OpenCV GitHub sayfası.

OpenCV kütüphanesi görsel verilere dayanan ileri düzey uygulamalarda ortaya koyduğu başarılar neticesinde ek bileşenlerle gelişmeye devam etmiş ve nihayetinde alternatif görüntü işleme filtrelerinin uygulanmasını, görüntülerde nesne tespiti ya da segmentasyonunun sağlanması, statik ya da gerçek zamanlı görüntülerle pratik bir şekilde çalışmalar yapılmasını ve hatta içerisinde taşıdığı Makine Öğrenmesi bileşenleri ile Yapay Zeka uygulamalarının da doğrudan OpenCV üzerinden kodlanmaya başlanması içerecek düzeyde gelişmiştir (Garcia vd., 2015; Howse, 2013). Bu yönyle OpenCV, günümüz ileri düzey robotik sistemlerde, görüntü üzerinden otomatikleştirilmiş karar destek araçlarında öncelikli olarak tercih edilen bir bileşen konumundadır.



4. ADIM: İLERLET

Öğrencilere bireysel olarak görüntü işleme ve OpenCV kütüphanesinin faydalı olacağı Yapay Zeka uygulama fikirleri üzerinden düşünmeleri ve bu fikirler hakkında genel açıklamaları yaziya dökmeleri istenir.



5. ADIM: DEĞERLENDİRME

İlerlet adımında yazılan açıklamalar ve derste öğrenilenler ışığında soru-cevap etkinliği gerçekleştirilerek görüntü işleme kavramı ve OpenCV kütüphanesine dair bilgiler pekiştirilir.

Öğrencilere şu sorular sorulabilir:

- Görüntü işleme yaklaşımı hakkında neler düşünüyorsunuz?

- Görüntü işlemenin gelecek teknolojiler üzerindeki rolü hakkında fikirleriniz nelerdir?
- Görüntü işleme ve bilgisayarlı görü gibi çözümler gerçek hayatı bizlere ne gibi avantajlar sağlamamaktadır?
- OpenCV kütüphanesi hakkında neler düşünüyorsunuz?
- Python kodlama ve Yapay Zeka uygulamaları konusunda şu ana kadar öğrendiklerinize ek olarak OpenCV'nin sağlayabileceği ek avantajlar neler olabilir?

3. HAFTA – 1. GÜN – 2., 3. DERSLER: OPENCV İLE TEMEL GÖRÜNTÜ İŞLEME

DERS PLANI

DERS ETİKETLERİ



KONU: OpenCV ile Temel Görüntü İşleme



SINIF: 12-14 Yaş



SÜRE: 120 dk.



HAZIR BULUNUŞLUK: İngilizce eğitimi, Yapay Zeka kavramı ve uygulamaları hakkında temel bilgi, OpenCV kütüphanesi hakkında temel bilgi, Python programlama dili hakkında temel bilgi



ANAHTAR KELİMELER: Yapay Zeka, Algoritma, Kodlama, Python, Görüntü İşleme, OpenCV



BECERİLER:



Tahmin



Sorgulama



Grup çalışması



Günlük hayat bağlantısı kurma



KAZANIMLAR:

Bu derste öğrenciler;

- OpenCV kütüphanesi kapsamında kullanılabilecek temel görüntü düzenleme ve işleme bileşenleri hakkında bilgi ve beceri sahibi olurlar.
- Python ile OpenCV destekli görüntü işleme uygulamaları gerçekleştirirler.
- Python tabanlı OpenCV bileşenlerini kullanmak suretiyle geliştirilebilecek Yapay Zeka uygulamaları konusunda fikir ve farkındalık sahibi olurlar.



DERS MATERİYALLERİ:

Malzeme	Miktar
Yazı Tahtası	Öğrenci sayısı ve sınıf büyüklüğüne göre en az bir adet
Projeksiyon	Öğrenci sayısı ve sınıf büyüklüğüne göre en az bir adet
Defter, Kalem	Öğrenci sayısı kadar
Bilgisayar	Öğrenci sayısının yarısı kadar
Tablet	Öğrenci sayısının yarısı kadar
Python Anaconda Framework	Her bilgisayarda kurulu olacak şekilde



ÖĞRETMEN HAZIRLIK ÇALIŞMALARI:

Okunabilecek Kaynaklar:

Bradski, G., & Kaehler, A. (2008). *Learning OpenCV: Computer vision with the OpenCV library*. O'Reilly Media, Inc.

Joshi, P., Escrivá, D. M., & Godoy, V. (2016). *OpenCV by example*. Packt Publishing Ltd.

İzlenebilecek Kaynaklar:

Kendi Çapında Mühendis. (2021). OpenCV ile Görüntü İşleme. Çevrimiçi:

<https://www.youtube.com/watch?v=LeuA7jIx8E&list=PLpmnLzMgTTpAucGwBMaLhcUOy60FN5-T8>

(Erişim 17 Haziran 2024).

Programming Knowledge. (2019). OpenCV Python Tutorial For Beginners. Çevrimiçi:
<https://www.youtube.com/watch?v=kdLM6AOd2vc&list=PLS1QulWo1RIa7D1O6skqDQ-JZ1GGHKK-K> (Erişim 17 Haziran 2024).

ATÖLYE HAZIRLIK ÇALIŞMALARI:

Öğretmen;

- Dersten önce ders materyallerini hazırlamalıdır.
- OpenCV kütüphanesi hakkında bilgi ve beceri sahibi olmalıdır.

- Python ortamında OpenCV kütüphanesi kullanımı konusunda bilgi ve beceri sahibi olmalıdır.



KAYNAKÇA:

Howse, J. (2013). *OpenCV computer vision with python* (Vol. 27). Birmingham, UK: Packt Publishing.

Minichino, J., & Howse, J. (2015). *Learning OpenCV 3 computer vision with python*. Packt Publishing Ltd.

GENEL BAKIŞ:

- 1) Harekete Geç:** Öğrencilere OpenCV kütüphanesinin Python ortamına dahil edilmesine ilişkin temel bilgiler verilir.
- 2) Keşfet:** Öğrencilere OpenCV kütüphanesi yardımıyla görseller üzerinde gerçekleştirilebilecek temel düzenlemelere ilişkin Python kodlamaları açıklanır.
- 3) Üret:** Öğrencilere OpenCV kütüphanesi ve Python kodlama ile uygulanabilen görüntü işleme çözümleri açıklanır.
- 4) İlerlet:** Öğrencilerin gruplar halinde çalışarak OpenCV kütüphanesi yardımıyla farklı uygulamalar gerçekleştirmeleri sağlanır.
- 5) Değerlendir:** Öğrencilerin yaptıkları uygulamalar ve derste öğrenilenler ışığında soru-cevap etkinliği gerçekleştirilerek OpenCV kütüphanesiyle Python uygulamaları kodlamaya dair bilgiler pekiştirilir.

UYGULAMA



1. ADIM: HAREKETE GEÇ

Öğrencilere OpenCV kütüphanesinin Python ortamına dahil edilmesine ilişkin temel bilgiler verilir.

Python Ortamında OpenCV Kütüphanesinin Kullanımı

OpenCV kütüphanesi Python uygulamalarına kolaylıkla entegre edilebilmektedir. Bu amaçla öncelikli olarak bilgisayarda yüklü olan Python ekosistemine OpenCV'nin kurulması gereklidir. Öğrencilerin öncelikli olarak Spyder editörü konsol ortamında **pip install** komut dizilimi yardımıyla OpenCV kütüphanesini kurmaları sağlanır:

pip install opencv-python

OpenCV kütüphanesinin kurulumu sonrasında herhangi bir Python uygulaması kodlarken OpenCV bileşenleri “import” komutu yardımıyla uygulama ortamına dahil edilir:

import cv2

Öğrencilerden Python kodlama ortamında OpenCV kütüphanesini çağrımları ve ardından Şekil 5’de gösterilen versiyon kontrol komutlarını yazmak suretiyle OpenCV’nin kurulumunu teyit etmeleri istenir.

```
7 import cv2
8 print("OpenCV versiyon:",cv2.__version__)
```

(a)

OpenCV versiyon: 4.7.0

(b)

Şekil 5. (a) OpenCV kütüphane çağrıları ve versiyon kontrolüne ilişkin kodlar
(b) kodların çalıştırılması sonucu konsol görüntüsü.



2. ADIM: KEŞFET

Öğrencilere OpenCV kütüphanesi yardımıyla görseller üzerinde gerçekleştirilebilecek temel düzenlemelere ilişkin Python kodlamaları açıklanır.

OpenCV ile Görsel Okuma ve Görüntüleme

OpenCV kütüphanesinde yer alan fonksiyonlar, görsellerin Python ortamında kolaylıklar düzenlemesine olanak sağlamaktadır. Görsellerin boyut özellikleri, hizalanma durumları ayarlanabilmekte, hatta görseller üzerinde kesme işlemleri ya da görsellere ait farklı piksel özelliklerine erişimler OpenCV’nin temel fonksiyonları ile gerçekleştirilmektedir.

OpenCV yardımıyla düzenleme yapılacak görsellerin öncelikli olarak çağrılması gerekmektedir. Bu amaçla **imread** fonksiyonu kullanılmaktadır. Yine üzerinde işlem yapılan bir görselin görüntülenmesi için de **imshow** fonksiyonu kullanılır.

Öğrenciler Şekil 6’de gösterilen komutları yazmak suretiyle örnek bir görseli okur, Python ortamında görselin tekrar görüntülenmesini sağlar ve ilgili görselin temel genişlik, yükseklik ve derinlik (renk kanal sayısı) özelliklerini ekrana yazdırır.

12. satırda yazılan “waitKey” fonksiyonu görselin ekranda görüntülenerek sonra ilerleyen komutların çalıştırılması için kullanıcıdan pencereyi kapamasını beklemektedir.

14. satırda yazılan **shape** kodu okunan görsel özelliklerinin değişkenlere çekilmesini sağlamaktadır.

```

10 gorsel = cv2.imread("image.jpg")
11 cv2.imshow("Image", gorsel)
12 cv2.waitKey(0)
13
14 (h, w, d) = gorsel.shape
15 print("Genişlik={}, Yükseklik={}, Derinlik={}".format(w, h, d))

```

(a)



(b)

Genişlik=599, Yükseklik=450, Derinlik=3

(c)

Şekil 6. (a) GörSEL dosyasının okunması, görüntülenmesi ve çeşitli özelliklerinin elde edilmesine yönelik kodlar
 (b) kodların çalıştırılması sonucu pencere görüntüsü
 (c) kodların çalıştırılması sonucu konsol görüntüsü

OpenCV ile GörSEL Boyutlandırma ve Piksellere Erişim

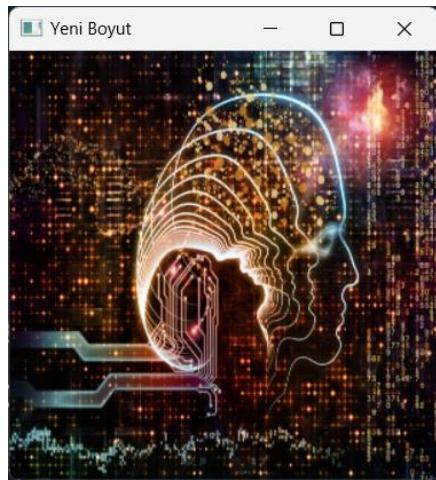
OpenCV'nin “**resize**” fonksiyonu kullanılmak suretiyle görsellerin boyutu değiştirilebilmekte, yine görsellerin tutulduğu değişkenlerde piksellere satır ve sütun indeks değerleri yardımıyla ulaşarak üç kanallı renk değerleri (R: Red -Kırmızı-, G: Green -Yeşil-, B: Blue -Mavi-) elde edilebilmektedir. Öğrencilerden Şekil 7'deki kodları yazarak görseli 300 * 300 boyutuna değiştirmeleri ve ardından 150. Y ve 150. X koordinat noktasındaki piksel renklerini okumaları istenir.

```

17 yeni_gorsel = cv2.resize(gorsel, (300, 300))
18 cv2.imshow("Yeni Boyut", yeni_gorsel)
19 cv2.waitKey(0)
20
21 (B, G, R) = yeni_gorsel[150, 150]
22 print("R={}, G={}, B={}".format(R, G, B))

```

(a)



(b)

R=46, G=12, B=3

(c)

Şekil 6. (a) GörSEL dosyasının okunması, görüntülenmesi ve çeşitli özelliklerinin elde edilmesine yönelik kodlar
 (b) kodların çalıştırılması sonucu pencere görüntüsü
 (c) kodların çalıştırılması sonucu konsol görüntüsü

OpenCV ile GörSEL Döndürme ve Kesme

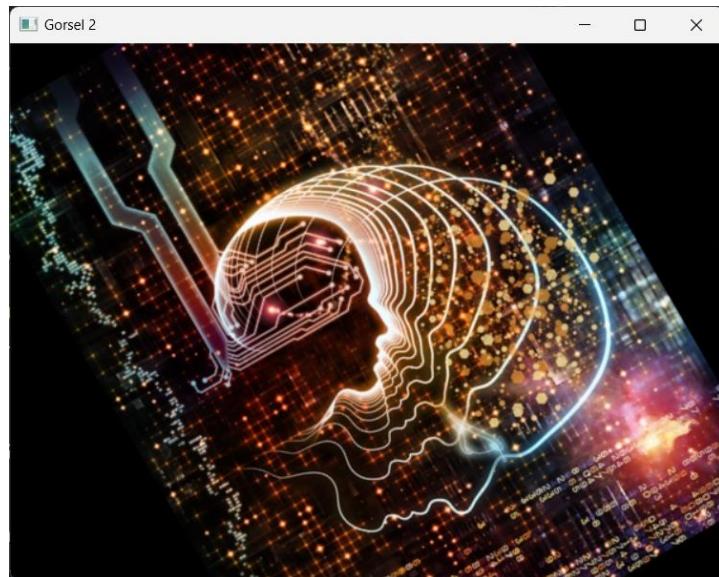
OpenCV okunan görsellerin belli açı değerleri yönünde döndürülmesini ve hatta belli görsel bölümlerinin kesilerek ayrı bir görsel olarak kullanımını sağlayan fonksiyonlar da içermektedir. Döndürme işlemi için görsel merkez noktası belirlenmekte, akabinde **warpAffine** fonksiyonu ile döndürme işlemi tamamlanmaktadır. Öğrencilerin okunan orijinal görseli Şekil 7'de olduğu gibi 60 derecelik açıyla saat yönünde döndürmesi yönünde ilgili kodları yazarak sonuçları gözlemllemeleri istenir.

```

24  merkez_noktasi = (w // 2, h // 2)
25  X = cv2.getRotationMatrix2D(merkez_noktasi, -60, 1.0)
26  dondurulmus_gorsel = cv2.warpAffine(gorsel, X, (w, h))
27  cv2.imshow("Gorsel 2", dondurulmus_gorsel)
28  cv2.waitKey(0)

```

(a)



(b)

Şekil 7. (a) OpenCV ile döndürme işlemlerine dair kodlar
 (b) kodların çalıştırılması sonucu elde edilen pencere / görsel görüntüsü

Görseller üzerinde sıkılıkla başvurulan temel düzenleme işlemlerinden biri de “kesme” işlemidir. Bu amaç için tipki piksel erişiminde olduğu gibi görsellerin X ve Y koordinat noktalarına ilişkin başlangıç ve bitiş indeks değerleri kullanılır. Buna göre, bir görselden alınacak parça için köşeli parantezler arasında **Y başlangıç:Y bitiş, X başlangıç:X bitiş** değerleri belirtilmektedir.

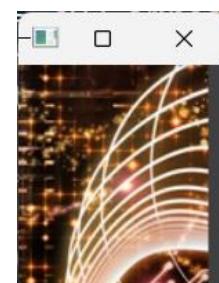
Öğrencilerle Şekil 8’de olduğu gibi orijinal görselin belli bir parçasının alınarak yeni görsel değişkenine aktarılması ve bu parçanın ekranda görüntülenmesi yönünde kodlar yazılır.

```

32  parça = gorsel[100:230, 150:260]
33  cv2.imshow("Parça", parça)
34  cv2.waitKey(0)

```

(a)



(b)

Şekil 8. (a) Görsel parça kesmeye ilişkin kodlar
 (b) kodların çalıştırılması sonucu elde edilen pencere görüntüsü



3. ADIM: ÜRET

Bu adımda öğrencilere OpenCV kütüphanesi ve Python kodlama ile uygulanabilen görüntü işleme çözümleri açıklanır.

OpenCV ile Görseller Üzerinde Görüntü İşleme Çözümleri

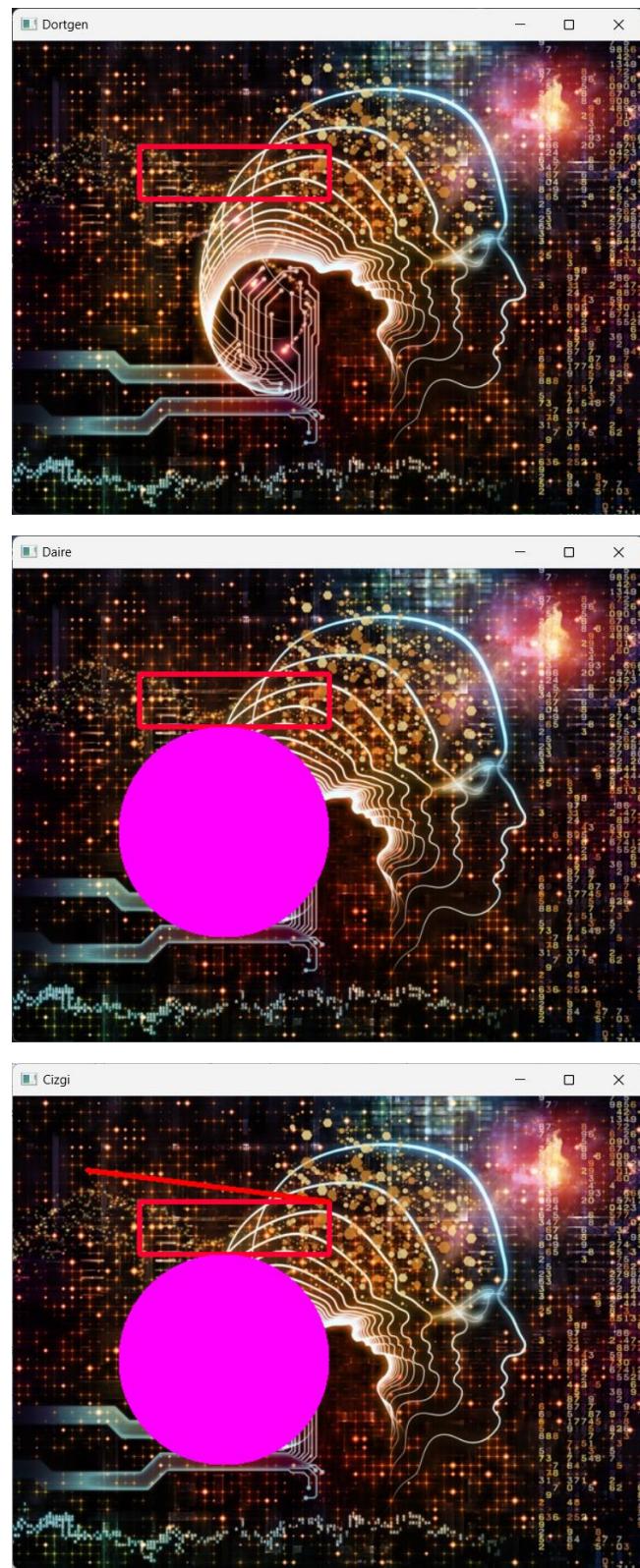
OpenCV görseller üzerindeki temel düzenleme işlemleri dışında olmak üzere, daha ileri seviye manipülasyonların yani görüntü işleme çözümlerinin işletilmesi için çeşitli fonksiyonlar içermektedir. Bu fonksiyonlar genellikle orijinal görselin farklı stillerde ve yapılarda versiyonlara dönüştürülmesi ve böylelikle alternatif görüntü analizi adımlarına ve Yapay Zeka gibi ileri düzey teknolojik çözümlerin gerçekleştireceği tespit süreçlerine etkin girdilerin oluşturulması mümkün olmaktadır.

Görsellere Çizimler ve Metinler Eklenmesi

OpenCV okunan bir görsel üzerine belirtilen noktalar kapsamında çeşitli geometrik çizimlerin ve hatta metinlerin eklenmesi için pratik fonksiyonlar içermektedir. Buna göre **rectangle**, **line**, **circle** gibi fonksiyonlar geometrik çizimler, **putText** fonksiyonu ise metinler eklemek amacıyla kullanılır. Öğrencilerle birlikte Şekil 9'da gösterilen komutların yazılarak sonuçların incelenmesi ve yorumlanması sağlanır. Özellikle farklı fonksiyonlar içerisinde yer alan çizim alanlarına dair değerler, renk kodları (yuvarlak parantez içerisindeki R-G-B renk değerleri) ve çizgi kalınlığı (genellikle son parametre değeri) gibi parametreler özellikle vurgulanır.

```
36 cv2.rectangle(gorsel, (300, 100), (120, 150), (50, 0, 255), 3)
37 cv2.imshow("Dortgen", gorsel)
38 cv2.waitKey(0)
39
40 cv2.circle(gorsel, (200, 250), 100, (255, 0, 255), -1)
41 cv2.imshow("Daire", gorsel)
42 cv2.waitKey(0)
43
44 cv2.line(gorsel, (70, 70), (300, 100), (0, 0, 255), 3)
45 cv2.imshow("Cizgi", gorsel)
46 cv2.waitKey(0)
47
48 cv2.putText(gorsel, "Yapay Zeka", (20, 50),
49             cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 255, 0), 3)
50 cv2.imshow("Metin", gorsel)
51 cv2.waitKey(0)
```

(a)



(b)



(c)

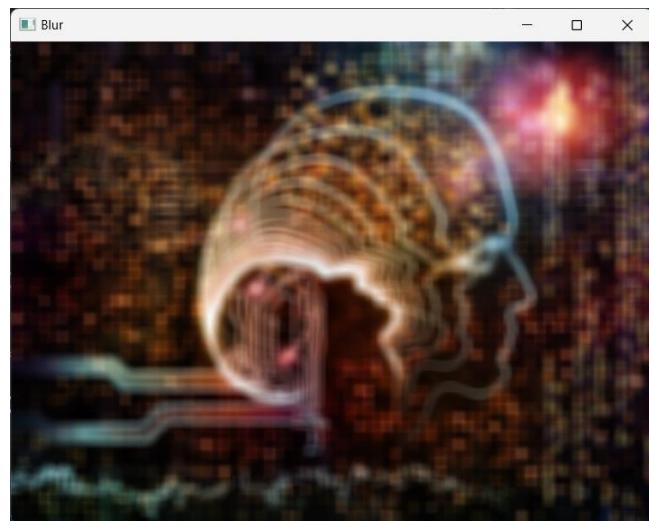
Şekil 9. (a) Görsellere çizimler ve metinler eklenmesine yönelik kodlar
 (b) kodların çalıştırılması sonucunda eklenen çizimler / şekillere dair pencere görüntüsü
 (c) kodların çalıştırılması sonucu eklenen metine (“Yapay Zeka”) dair pencere görüntüsü

Görselin Bulanıklaştırılması

Okunan görseller OpenCV sayesinde bulanıklaştmaya (blur efektine) tabi tutulabilmektedir. Bu amaçla “**GaussianBlur**” adlı fonksiyon kullanılır (Şekil 10).

```
55 bulanik = cv2.GaussianBlur(gorsel, (11, 11), 50)
56 cv2.imshow("Blur", bulanik)
57 cv2.waitKey(0)
```

(a)



(b)

Şekil 10. (a) OpenCV ile görüntü bulanıklaştmaya (blur efektine) dair kodlar
 (b) kodların çalıştırılması sonucu elde edilen pencere görüntüsü

Görselin Gri Tonlara Dönüşürülmesi

Özellikle renklerden ziyade siyah ve beyaz renk tonlarındaki değişimlerin önemli olduğu uygulamalarda görsellerin gri tonlara dönüştürülmesi gerekebilir. Bu amaçla OpenCV içerisindeki “**cvtColor**” fonksiyonu çağırılabilmektedir. Gri tonlara dönüşüm için ilgili fonksiyonun ikinci parametresi **cv2.COLOR_BGR2GRAY** olarak ifade edilir (Şekil 11).

```
61  gri = cv2.cvtColor(gorsel, cv2.COLOR_BGR2GRAY)
62  cv2.imshow("Gri", gri)
63  cv2.waitKey(0)
```

(a)



(b)

Şekil 11. (a) OpenCV ile gri tonlara dönüşümü dair kodlar
(b) kodların çalıştırılması sonucu elde edilen pencere görüntüsü

Görselde Kenar Tespiti

Genellikle medikal görüntüler üzerinde çalışan ya da nesneler üzerinden tespitler yapan Yapay Zeka uygulamalarında gri tonlara dönüştürülen görsellerde kenar tespitleri yapılır. Kenar tespiti bir bakıma farklı renk tonlarına geçiş noktalarının ön plana çıkarılması yönünde gerçekleştirilen bir filtreleme işlemidir. Birçok farklı kenar tespiti algoritması var olsa da **Canny** algoritması sıkılıkla tercih edilen temel bir kenar tespit aracıdır (Howse, 2013; Minichino, & Howse, 2015).

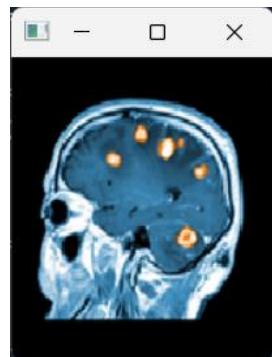
Öğrencilerden bu defa örnek bir renkli medikal beyin görüntüsünü kullanmak suretiyle önce griye dönüşüm ardından kenar tespiti adımlarını kodlamalarını ve elde edilen çıktıları gözlemlemeleri istenir. Burada uygulanan örnektan yola çıkarak, görüntü işlemenin medikal görüntülerde yer alan anormalliliklerin tespiti noktasında Yapay Zeka algoritmalarının görevlerini kolaylaştırmaları yönündeki avantajları vurgulanır. Şekil 12 söz konusu kodları ve ilgili çıktıları göstermektedir.

```

67 gorsel2 = cv2.imread("medical_image.png")
68 cv2.imshow("Image", gorsel2)
69 cv2.waitKey(0)
70
71 gri2 = cv2.cvtColor(gorsel2, cv2.COLOR_BGR2GRAY)
72 cv2.imshow("Gri", gri2)
73 cv2.waitKey(0)
74
75 kenar_edged = cv2.Canny(gri2, 30, 150)
76 cv2.imshow("Edged", kenar_edged)
77 cv2.waitKey(0)

```

(a)



(b)



(c)



(d)

Şekil 12. (a) OpenCV ile medikal görüntüde gri ton dönüşümü ve kenar tespitine dair kodlar
 (b) orijinal medikal görüntü verisi
 (c) kodların çalıştırılması sonucu elde edilen griye dönüşüm görüntüsü
 (d) kodların çalıştırılması sonucu elde edilen kenar tespiti görüntüsü

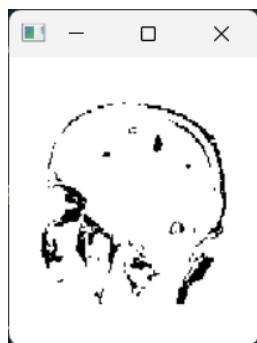
Görselde Thresholding İşlemleri

Görüntü verileri üzerinde gerçekleştirilen uygulamalarda dikkate alınan görsellerde kimi zaman bazı bölümlerin daha net gözlemlenmesi gerekebilmiştir. Bu durumda kullanılan farklı görüntü işleme çözümleri arasında yer alan thresholding, görseldeki koyu ya da parlak bölgelerin ya da çözümde dezavantaj yaratan bazı bölgelerin ortadan kaldırılmasında etkili olmaktadır (Howse, 2013; Minichino, & Howse, 2015).

Öğrencilerin bir önceki aşamada elde edilen gri tonlardaki medikal görüntü üzerinde Şekil 13'te gösterilen kodlar yardımıyla thresholding işlemi uygulamaları sağlanır. Elde edilen sonuçlar gözlemlenir ve gri tonlara dönüşüm, kenar bulma, thresholding gibi aşamalar arasında ortaya çıkan farklılıklar irdelenir.

```
79 t = cv2.threshold(gri2, 225, 255, cv2.THRESH_BINARY_INV)[1]
80 cv2.imshow("Thresh", t)
81 cv2.waitKey(0)
```

(a)



(b)

Şekil 13. (a) OpenCV ile thresholding sürecine yönelik kodlar
(b) kodların çalıştırılması sonucu elde edilen pencere görüntüsü

UYGULAMANIN PYTHON KODLARI

```
import cv2
#print("OpenCV versiyon:",cv2.__version__)
gorsel = cv2.imread("image.jpg")
cv2.imshow("Image", gorsel)
cv2.waitKey(0)
#
=====
=====
# (h, w, d) = gorsel.shape
# print("Genişlik={}, Yükseklik={}, Derinlik={}".format(w, h, d))
```

```

#
# yeni_gorsel = cv2.resize(gorsel, (300, 300))
# cv2.imshow("Yeni Boyut", yeni_gorsel)
# cv2.waitKey(0)
#
# (B, G, R) = yeni_gorsel[150, 150]
# print("R={ }, G={ }, B={ }".format(R, G, B))
#
# merkez_noktasi = (w // 2, h // 2)
# X = cv2.getRotationMatrix2D(merkez_noktasi, -60, 1.0)
# dondurulmus_gorsel = cv2.warpAffine(gorsel, X, (w, h))
# cv2.imshow("Gorsel 2", dondurulmus_gorsel)
# cv2.waitKey(0)
#
=====

=====

#
=====

=====

# parca = gorsel[100:230, 150:260]
# cv2.imshow("Parca", parca)
# cv2.waitKey(0)
#
# cv2.rectangle(gorsel, (300, 100), (120, 150), (50, 0, 255), 3)
# cv2.imshow("Dortgen", gorsel)
# cv2.waitKey(0)
#
# cv2.circle(gorsel, (200, 250), 100, (255, 0, 255), -1)
# cv2.imshow("Daire", gorsel)
# cv2.waitKey(0)
#
# cv2.line(gorsel, (70, 70), (300, 100), (0, 0, 255), 3)

```

```

# cv2.imshow("Cizgi", gorsel)
# cv2.waitKey(0)
#
# cv2.putText(gorsel, "Yapay Zeka", (20, 50),
#             cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 255, 0), 3)
# cv2.imshow("Metin", gorsel)
# cv2.waitKey(0)
#
=====

=====

# =====

=====

# bulanik = cv2.GaussianBlur(gorsel, (11, 11), 50)
# cv2.imshow("Blur", bulanik)
# cv2.waitKey(0)
#
=====

=====

# =====

=====

# gri = cv2.cvtColor(gorsel, cv2.COLOR_BGR2GRAY)
# cv2.imshow("Gri", gri)
# cv2.waitKey(0)
#
=====

=====

gorsel2 = cv2.imread("medical_image.png")
cv2.imshow("Image", gorsel2)

```

```
cv2.waitKey(0)
```

```
gri2 = cv2.cvtColor(gorsel2, cv2.COLOR_BGR2GRAY)
```

```
cv2.imshow("Gri", gri2)
```

```
cv2.waitKey(0)
```

```
kenar_edged = cv2.Canny(gri2, 30, 150)
```

```
cv2.imshow("Edged", kenar_edged)
```

```
cv2.waitKey(0)
```

```
t = cv2.threshold(gri2, 225, 255, cv2.THRESH_BINARY_INV)[1]
```

```
cv2.imshow("Thresh", t)
```

```
cv2.waitKey(0)
```



4. ADIM: İLERLET

İlerlet adımı içerisinde öğrencilerin gruplar halinde çalışarak OpenCV kütüphanesi yardımıyla farklı uygulamalar gerçekleştirmeleri sağlanır. Söz konusu süreç için toplam 15 dakikalık bir süre ayrılr.



5. ADIM: DEĞERLENDİRME

Öğrencilerin yaptıkları uygulamalar ve derste öğrenilenler ışığında soru-cevap etkinliği gerçekleştirilerek OpenCV kütüphanesiyle Python uygulamaları kodlamaya dair bilgiler pekiştirilir.

Öğrencilere şu sorular sorulabilir:

- Gerçekleştirilen Python tabanlı OpenCV uygulamalarından elde ettiğiniz bilgi ve beceriler neler olmuştur?
- OpenCV ile görüntü işleme kodlama süreçleri hakkında görüşleriniz nelerdir?
- OpenCV'nin görüntüler üzerinde gerçekleştirilebilecek Yapay Zeka uygulamaları üzerindeki etkileri hakkında neler düşünüyorsunuz?
- Gerçekleştirilen görüntü işlemlerini hangi tür görsel veriler üzerinde uygulamak isterdiniz?

- Kenar tespit çözümleri konusunda fikirleriniz nelerdir?
- Thresholding işlemi konusunda neler düşünüyorsunuz?
- Edindiğiniz bilgi ve becerilerden hareketle ne gibi Yapay Zeka uygulamaları geliştirmek isterdiniz?

3. HAFTA – 2. GÜN – 1. DERS: POSE DETECTION UYGULAMASI

DERS PLANI

DERS ETİKETLERİ



KONU: Pose Detection Uygulaması



SINIF: 12-14 Yaş



SÜRE: 60 dk.



HAZIR BULUNUŞLUK: İngilizce eğitimi, Yapay Zeka kavramı ve uygulamaları hakkında temel bilgi, OpenCV kütüphanesi hakkında temel bilgi, Python programlama dili hakkında temel bilgi



ANAHTAR KELİMELER: Yapay Zeka, Algoritma, Kodlama, Python, Görüntü İşleme, OpenCV, Pose Detection



BECERİLER:



Tahmin



Sorgulama



Grup çalışması



Günlük hayat bağlantısı kurma



KAZANIMLAR:

Bu derste öğrenciler;

- Görsel veriler üzerinden pose detection (duruş tespitinin) dayandığı temeller konusunda bilgi edinirler.
- OpenCV kütüphanesi ve ön eğitimli bir Yapay Zeka modeli yardımıyla pose detection (duruş tespiti) uygulaması geliştirme konusunda bilgi ve beceri sahibi olurlar.
- Sabit ve hareketli görseller üzerinde pose detection (duruş tespiti) konusunda bilgi ve beceri sahibi olurlar.



DERS MATERİYALLERİ:

Malzeme	Miktar
Yazı Tahtası	Öğrenci sayısı ve sınıf büyüklüğüne göre en az bir adet
Projeksiyon	Öğrenci sayısı ve sınıf büyüklüğüne göre en az bir adet
Defter, Kalem	Öğrenci sayısı kadar
Bilgisayar	Öğrenci sayısının yarısı kadar
Tablet	Öğrenci sayısının yarısı kadar
Python Anaconda Framework	Her bilgisayarda kurulu olacak şekilde



ÖĞRETMEN HAZIRLIK ÇALIŞMALARI:

Okunabilecek Kaynaklar:

Halıcı, A. S., & Demirhan, A. (2022). Kalmanfiltresi ve küresel en yakın komşu yöntemi ile çok kişili gerçek zamanlı poz takibi. *Politeknik Dergisi*, 26(2), 889-899.

Singh, A. K., Kumbhare, V. A., & Arthi, K. (2021, June). Real-time human pose detection and recognition using mediapipe. In *International conference on soft computing and signal processing* (pp. 145-154). Singapore: Springer Nature Singapore.

İzlenebilecek Kaynaklar:

Augmented AI. (2019). OpenPose Tutorials. Çevirmiçi:

https://www.youtube.com/watch?v=4FZrE3cmTPA&list=PL_Nji0JOuXg24bHB60SB2TwF0PpwhJkCF (Erişim 17 Haziran 2024).

Wood, K. (2023). OpenCV Python Pose Estimation. Çevirmiçi:

<https://www.youtube.com/watch?v=bs81DNsMrnM> (Erişim 17 Haziran 2024).

ATÖLYE HAZIRLIK ÇALIŞMALARI:

Öğretmen;

- Dersten önce ders materyallerini hazırlamalıdır.
- Python ortamında OpenCV kütüphanesi kullanımı konusunda bilgi ve beceri sahibi olmalıdır.

- Pose detection (duruş tespiti) uygulamaları konusunda ön bilgi ve farkındalık sahibi olmalıdır.



KAYNAKÇA:

Boesch, G. (2023). The Complete Guide to OpenPose in 2024. Online: <https://viso.ai/deep-learning/openpose/> (Erişim 17 Haziran 2024).

Viswakumar, A., Rajagopalan, V., Ray, T., & Parimi, C. (2019). Human gait analysis using OpenPose. In *2019 fifth international conference on image information processing (ICIIP)* (pp. 310-314). IEEE.

GENEL BAKIŞ:

1) Harekete Geç: Öğrencilere OpenCV ve Python yardımıyla insan poz (duruş) tespitinin nasıl yapılabileceği konusunda fikirleri sorulur ve dönütler tartışılırak poz (duruş) tespitinin temel mantığı açıklanır.

2) Keşfet: Öğrencilerin OpenCV ve Python kodlama yardımıyla ön eğitimli bir Yapay Zeka modeli üzerinden sabit görselde pose detection (duruş tespiti) gerçekleştirmeleri sağlanır.

3) Üret: Öğrencilerin önceki uygulamada tespit süreçlerinin hareketli görüntüler yönünde revize etmeleri sağlanır.

4) İlerlet: Öğrencilerin gruplar halinde çalışarak gerçekleştirildikleri uygulamalar üzerinde değişikliklerle denemeler yapmaları istenir.

5) Değerlendir: Öğrencilerin yaptıkları uygulamalardan elde ettikleri tecrübeler ve derste öğrenilenler ışığında soru-cevap etkinliği gerçekleştirilerek pose detection (duruş tespiti) uygulamalarına dair bilgiler pekiştirilir.

UYGULAMA



1. ADIM: HAREKETE GEÇ

Öğrencilere OpenCV ve Python yardımıyla insan poz (duruş) tespitinin nasıl yapılabileceği konusunda fikirleri sorulur ve dönütler tartışılırak poz (duruş) tespitinin temel mantığı açıklanır.

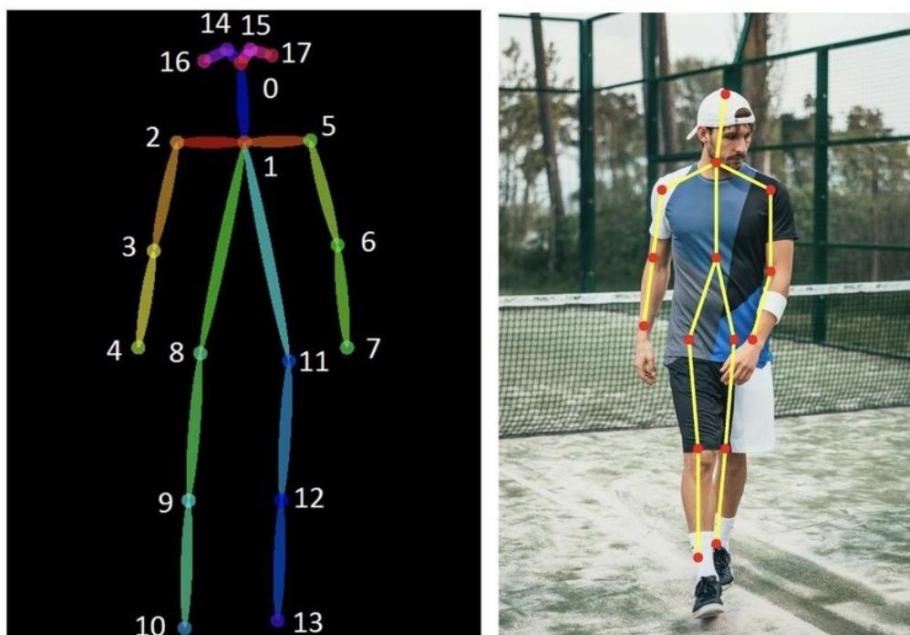
Görüntü İşleme ve Yapay Zeka Yardımıyla Poz (Duruş) Tespiti

Yapay Zeka uygulamalarında görüntü verileri üzerinden gerçekleştirilen uygulamalar özellikle hareketli nesneler üzerinden tespitler konusunda aktif bir şekilde ilerlemektedir. Görüntü işleme gibi aşamaların desteğiyle Yapay Zeka modelleri tarafından başarılı bir şekilde işlenecek hale getirilen görüntüler, çözümlemek istenen hedef problemlere göre çeşitli tahmin

ve karar verme aşamalarına bağlanabilmektedir. Özellikle insanlar üzerinden yapılan tespit uygulamaları, yürüyüş tespiti, poz (duruş) tespiti gibi yaklaşımların otomatikleştirilmesi yönünde kurgulanmaktadır (Viswakumar vd., 2019). Gerçekleştirilen tespitler, kişi ya da olay tahmini, fizyoterapi / tedavi takibi ya da sportif aktivitelerin değerlendirilmesi gibi çok çeşitli uygulamalarla ilişkilendirilmektedir.

Yapay Zeka uygulamaları yardımıyla insanlar üzerinde gerçekleştirilen tespitlerin en yaygın olanı, vücut bileşenlerinin temsili noktalarla tespitine dayanmaktadır. Vücut bileşenlerinin temsili noktalarla tespiti, **pose detection (duruş tespiti)**, **face detection (yüz tanıma)** gibi farklı uygulamaları desteklemekle beraber, amaca yönelik uygulamalarda daha spesifik tespitler de (Örneğin, kozmetik ürünlerinin el, yüz...vs. üzerinde sanal olarak denenmesi) yapılmaktadır. Bütün bu tespitlerin odak noktası Yapay Zeka uygulamalarının tahminlerini destekleyecek noktaların belirlenerek, sayısal girdilerle (x, y, z koordinatları) amaca yönelik tahmin çıktılarının elde edilmesi yönündedir.

Şekil 14 tipik bir pose detection uygulamasının vücut bileşenlerini temsil eden noktaları belirleme yönünde izlediği yaklaşımı göstermektedir (Boesch, 2023). Öğrencilere şekildeki her bir noktanın farklı vücut bileşenini temsil ettiği ve bu temsillere göre Yapay Zeka modelleri üzerinden girdi-çıktı mantığı üzerinden tahmin çözümlerinin geliştirilebildiği açıklanır.



Şekil 14. Pose detection yaklaşımı (Boesch, 2023).

Pose Detection İçin OpenPose Kütüphanesi

Pose detection çözümleri için OpenCV desteğini arkasına alarak problemlere özgü çözümler üretebilen birçok farklı kütüphane vardır. Bu kütüphaneler arasında yer alan OpenPose, özellikle insan görüntüleri üzerinde tespitlerle ön plana çıkan bir kütüphane olarak, pose detection uygulamalarında pratik bir biçimde kullanılır. Genel olarak 3D ya da 2D görüntüler üzerinde tespitler yapabilen OpenPose, bu amaç doğrultusunda ön eğitilmiş Yapay Zeka

modellerini (özellikler yapay sinir ağlarına dayalı Derin Öğrenme tekniklerini) kullanabilmekte; vücut odaklı tespitlerde 15, 18, 27 temsil noktasını, yüz odaklı tespitlerde 70 temsil noktasını, el odaklı tespitlerde ise 21 temsil noktasını belirleyebilmektedir (Boesch, 2023).



2. ADIM: KEŞFET

Öğrencilerin OpenCV ve Python kodlama yardımıyla ön eğitimli bir Yapay Zeka modeli üzerinden sabit görselde pose detection (duruş tespiti) gerçekleştirmeleri sağlanır.

Kütüphane Çağrıları ve Ön Tanımlamalar

Öğrencilerle birlikte uygulama kapsamında kullanılacak OpenCV ve Numpy kütüphane çağrıları kodlanır. Kütüphane çağrıları sonrasında OpenPose modellerine uygun bir biçimde vücut bileşenlerini temsil eden bölümler ve bu bölümleri temsil eden sınıfların tanımlanması gerekmektedir. Bu amaçla görsel arkaplani da dahil olmak üzere 19 farklı bileşeni temsil eden noktalar ve vücut bileşenleri arası bağlantılarına yönelik ikili bağlantı tanımları kodlanır (Şekil 15).

```
7 import cv2
8 import numpy as np
9
10 BODY_PARTS = { "Nose": 0, "Neck": 1, "RShoulder": 2, "RElbow": 3, "RWrist": 4,
11     "LShoulder": 5, "LElbow": 6, "LWrist": 7, "RHip": 8, "RKnee": 9,
12     "RAngle": 10, "LHip": 11, "LKnee": 12, "LAngle": 13, "REye": 14,
13     "LEye": 15, "REar": 16, "LEar": 17, "Background": 18 }
14
15 POSE_PAIRS = [ ["Neck", "RShoulder"], [ "Neck", "LShoulder"], [ "RShoulder", "RElbow"],
16     [ "RElbow", "RWrist"], [ "LShoulder", "LElbow"], [ "LElbow", "LWrist"],
17     [ "Neck", "RHip"], [ "RHip", "RKnee"], [ "RKnee", "RAngle"], [ "Neck", "LHip"],
18     [ "LHip", "LKnee"], [ "LKnee", "LAngle"], [ "Neck", "Nose"], [ "Nose", "REye"],
19     [ "REye", "REar"], [ "Nose", "LEye"], [ "LEye", "LEar"] ]
```

Şekil 15. Pose detection için kütüphane çağrıları ve ön tanımlamalar.

Yapay Zeka Modeli ile Görselin Analizi

Tanımlamalar ardından pose detection bölgesini temsil eden genişlik ve yükseklik değişkenleri tanımlanır. Yine ön eğitimli OpenPose MobileNet adlı Derin Sinir Ağları teknigine dayalı model dosyası (**graph_opt.pb**) indirilerek Python uygulamasının yer aldığı dizine kaydedilir OpenCV kütüphanesi “**dnn**” bileşeni üzerinden uygulama ortamına çağrılmaması yönünde kodlar yazılır (Şekil 16).

MODEL İÇİN İNDİRME LİNKİ

https://github.com/quanhua92/human-pose-estimation-opencv/blob/master/graph_opt.pb

Yapay Zeka modelinin tanımlanması ardından “pose.png” olarak isimlendirilen örnek görselin okunması, okunan görselin genişlik-yükseklik değerlerinin alınması ve görselin Yapay Zeka modeline girdi olarak tanımlanmasını takiben (Şekil 16 31. satır), “**forward**” fonksiyonu ile pose detection işleminin yapılması (Şekil 16 32. satır) ve elde edilen çıktıların 19 temsili nokta üzerinden “detect” adlı değişken altında tutulması (Şekil 16 33. Satır) yönünde kodlar yazılır.

```

24 model = cv2.dnn.readNetFromTensorflow("graph_opt.pb")
25
26 img = cv2.imread("pose.png")
27
28 f_w = img.shape[1]
29 f_h = img.shape[0]
30
31 model.setInput(cv2.dnn.blobFromImage(img, 1.0, (w, h), (127.5, 127.5, 127.5), swapRB=True, crop=False))
32 detect = model.forward()
33 detect = detect[:, :, :19, :, :]

```

Şekil 16. Pose detection kapsamında örnek görselin Yapay Zeka modeli ile analiz edilmesi.

Pose Detection Çıktısının Elde Edilmesi

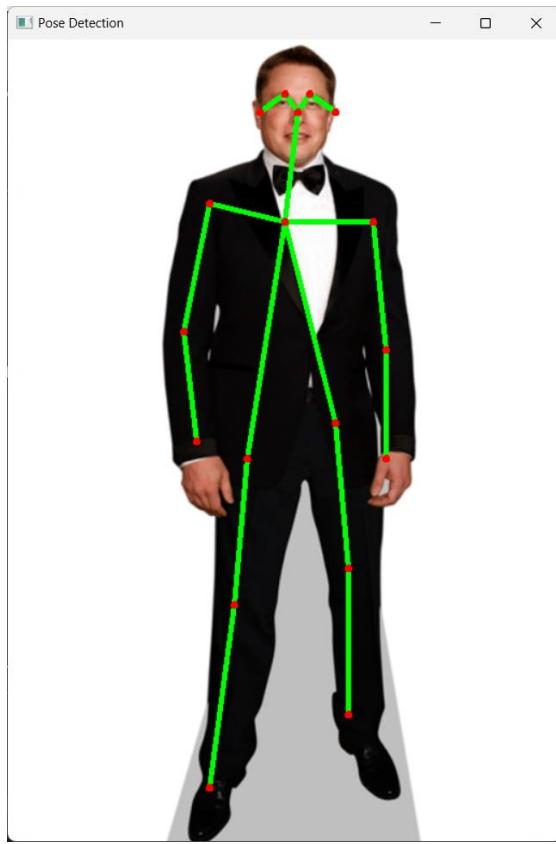
Yapay Zeka modelinin elde ettiği temsil çıktılarının orijinal görsel üzerinde ekrana yansıtılması için Şekil 17'de görüldüğü gibi; elde edilen çıktılardan destek alınarak vücut bileşenleri arası bağlantıların çizgiler ve daireler yardımıyla gösterilmesi yönünde kodlar yazılır. Öğrencilere kullanılan iki adet “for” döngüsü işaret edilerek döngüler içerisindeki adımlar genel hatlarıyla vurgulanır. Son olarak elde edilen pose detection çıktısı “imshow” fonksiyonu yardımıyla ekrana yansıtılır.

```

35 assert(len(BODY_PARTS) == detect.shape[1])
36
37 points = []
38 for i in range(len(BODY_PARTS)):
39     heatMap = detect[0, i, :, :]
40     _, conf, _, point = cv2.minMaxLoc(heatMap)
41     x = (f_w * point[0]) / detect.shape[3]
42     y = (f_h * point[1]) / detect.shape[2]
43     points.append((int(x), int(y)) if conf > 0.2 else None)
44
45 for pair in POSE_PAIRS:
46     partFrom = pair[0]
47     partTo = pair[1]
48     assert(partFrom in BODY_PARTS)
49     assert(partTo in BODY_PARTS)
50
51     idFrom = BODY_PARTS[partFrom]
52     idTo = BODY_PARTS[partTo]
53
54     if points[idFrom] and points[idTo]:
55         cv2.line(img, points[idFrom], points[idTo], (0, 255, 0), 3)
56         cv2.ellipse(img, points[idFrom], (3, 3), 0, 0, 360, (0, 0, 255), cv2.FILLED)
57         cv2.ellipse(img, points[idTo], (3, 3), 0, 0, 360, (0, 0, 255), cv2.FILLED)
58
59 cv2.imshow('Pose Detection', img)
60 cv2.waitKey(0)

```

(a)



(b)

Şekil 17. (a) Pose detection çıktısının elde edilmesine yönelik kodlar
 (b) kodların çalıştırılması sonucu elde edilen pencere görüntüsü

UYGULAMANIN PYTHON KODLARI

```

import cv2
import numpy as np

BODY_PARTS = { "Nose": 0, "Neck": 1, "RShoulder": 2, "RElbow": 3, "RWrist": 4,
    "LShoulder": 5, "LElbow": 6, "LWrist": 7, "RHip": 8, "RKnee": 9,
    "RAngle": 10, "LHip": 11, "LKnee": 12, "LAngle": 13, "REye": 14,
    "LEye": 15, "REar": 16, "LEar": 17, "Background": 18 }

POSE_PAIRS = [ ["Neck", "RShoulder"], ["Neck", "LShoulder"], ["RShoulder",
    "RElbow"], ["RElbow", "RWrist"], ["LShoulder", "LElbow"], ["LElbow", "LWrist"],
    ["Neck", "RHip"], ["RHip", "RKnee"], ["RKnee", "RAngle"], ["Neck", "LHip"],
    ["LHip", "LKnee"], ["LKnee", "LAngle"], ["Neck", "Nose"], ["Nose", "REye"],
    ["Nose", "LEye"] ]

```

```
["REye", "REar"], ["Nose", "LEye"], ["LEye", "LEar"] ]
```

```
w = 350
```

```
h = 350
```

```
model = cv2.dnn.readNetFromTensorflow("graph_opt.pb")
```

```
img = cv2.imread("pose.png")
```

```
f_w = img.shape[1]
```

```
f_h = img.shape[0]
```

```
model.setInput(cv2.dnn.blobFromImage(img, 1.0, (w, h), (127.5, 127.5, 127.5),  
swapRB=True, crop=False))
```

```
detect = model.forward()
```

```
detect = detect[:, :19, :, :]
```

```
assert(len(BODY_PARTS) == detect.shape[1])
```

```
points = []
```

```
for i in range(len(BODY_PARTS)):
```

```
    heatMap = detect[0, i, :, :]
```

```
    _, conf, _, point = cv2.minMaxLoc(heatMap)
```

```
    x = (f_w * point[0]) / detect.shape[3]
```

```
    y = (f_h * point[1]) / detect.shape[2]
```

```
    points.append((int(x), int(y))) if conf > 0.2 else None)
```

```
for pair in POSE_PAIRS:
```

```
    partFrom = pair[0]
```

```
    partTo = pair[1]
```

```
    assert(partFrom in BODY_PARTS)
```

```
    assert(partTo in BODY_PARTS)
```

```

idFrom = BODY_PARTS[partFrom]
idTo = BODY_PARTS[partTo]

if points[idFrom] and points[idTo]:
    cv2.line(img, points[idFrom], points[idTo], (0, 255, 0), 3)
    cv2.ellipse(img, points[idFrom], (3, 3), 0, 0, 360, (0, 0, 255), cv2.FILLED)
    cv2.ellipse(img, points[idTo], (3, 3), 0, 0, 360, (0, 0, 255), cv2.FILLED)

cv2.imshow('Pose Detection', img)
cv2.waitKey(0)

```



3. ADIM: ÜRET

Bu adımda öğrencilerin önceki uygulamada tespit süreçlerinin hareketli görüntüler yönünde revize etmeleri sağlanır.

OpenCV ile Hareketli Görüntünün Okunması

OpenCV kütüphanesi yardımıyla sabit görüntüler dışında hareketli görüntülerin de okunması ve **frame (çerçeve / görsel parçası)** halinde işlenmesi de mümkün olmaktadır. Hareketli görüntülerin okunmasında kameralardan akan görüntüler kullanıldığı gibi, video dosyaları da kullanılabilir. Her iki durum için de “**VideoCapture**” fonksiyonu kullanılmaktadır.

Öğrencilerin örnek video dosyası (<https://github.com/intel-iot-devkit/sample-videos/raw/master/worker-zone-detection.mp4>) üzerinden pose detection yapabilmeleri için önceki uygulama üzerindeki kodlarda, Yapay Zeka modeli çağrısı sonrasında yer alan görsel okuma bölümlerinin video capture moduna alınması ve for döngülerinin genel bir “while” döngüsü altına alınarak videodan okuma sürecinin döngüsel bir pose detection sürecine dönüştürmeleri yönünde kod revizyonlarını yapmaları sağlanır (Şekil 18).

Not: Sürekli görüntü okuma ve tespit süreçlerinin başarılı bir şekilde çalışması için önceki kodlama sürecinde en satırda yazılan cv2.waitKey(0) komutu silinmelidir.

```

24 model = cv2.dnn.readNetFromTensorflow("graph_opt.pb")
25
26 img = cv2.imread("pose.png")
27
28 f_w = img.shape[1]
29 f_h = img.shape[0]
30
31 model.setInput(cv2.dnn.blobFromImage(img, 1.0, (w, h), (127.5, 127.5, 127.5), swapRB=True, crop=False))
32 detect = model.forward()
33 detect = detect[:, :19, :, :]

```

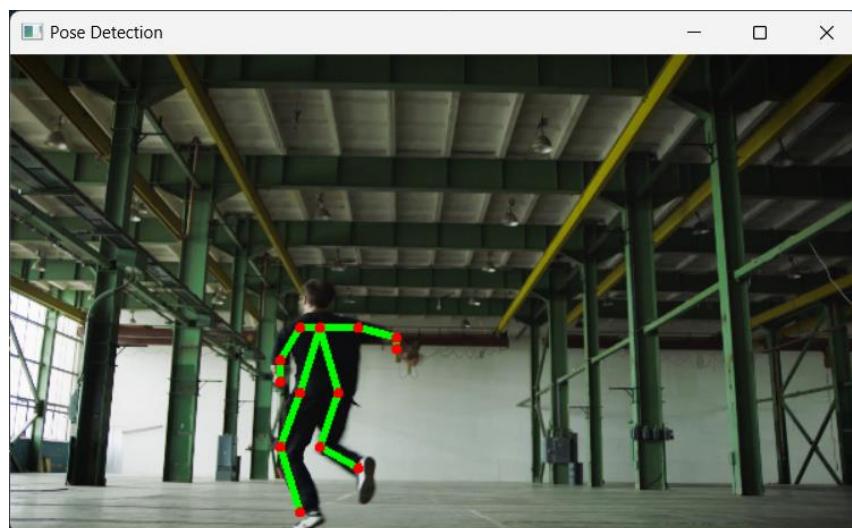
```

26 capture = cv2.VideoCapture("pose_video.mp4")
27
28 while cv2.waitKey(1) < 0:
29     hasFrame, img = capture.read()
30     img = cv2.resize(img, (600, 338))
31     if not hasFrame:
32         cv2.waitKey()
33         break
34
35 f_w = img.shape[1]
36 f_h = img.shape[0]
37
38 model.setInput(cv2.dnn.blobFromImage(img, 1.0, (w, h), (127.5, 127.5, 127.5), swapRB=True, crop=False))
39 detect = model.forward()
40 detect = detect[:, :19, :, :]
41
42 assert(len(BODY_PARTS) == detect.shape[1])
43
44 points = []
45 for i in range(len(BODY_PARTS)):
46     heatMap = detect[0, i, :, :]
47     _, conf, _, point = cv2.minMaxLoc(heatMap)
48     x = (f_w * point[0]) / detect.shape[3]
49     y = (f_h * point[1]) / detect.shape[2]
50     points.append((int(x), int(y)) if conf > 0.2 else None)
51
52 for pair in POSE_PAIRS:
53     partFrom = pair[0]
54     partTo = pair[1]
55     assert(partFrom in BODY_PARTS)
56     assert(partTo in BODY_PARTS)
57
58     idFrom = BODY_PARTS[partFrom]
59     idTo = BODY_PARTS[partTo]
60
61     if points[idFrom] and points[idTo]:
62         cv2.line(img, points[idFrom], points[idTo], (0, 255, 0), 3)
63         cv2.ellipse(img, points[idFrom], (3, 3), 0, 0, 360, (0, 0, 255), cv2.FILLED)
64         cv2.ellipse(img, points[idTo], (3, 3), 0, 0, 360, (0, 0, 255), cv2.FILLED)
65
66 cv2.imshow('Pose Detection', img)

```

Şekil 18. Hareketli görüntüler için Python kodları üzerindeki revizyonlar.

Şekil 19'da görüldüğü üzere, revizyon sonrası çalıştırılan kodlar, video üzerinde pose detection süreçlerini aktif bir biçimde göstermektedir.



Şekil 19. Hareketli görüntü üzerinde pose detection.

UYGULAMANIN PYTHON KODLARI

```
import cv2
import numpy as np

BODY_PARTS = { "Nose": 0, "Neck": 1, "RShoulder": 2, "RElbow": 3, "RWrist": 4,
    "LShoulder": 5, "LElbow": 6, "LWrist": 7, "RHip": 8, "RKnee": 9,
    "RAngle": 10, "LHip": 11, "LKnee": 12, "LAngle": 13, "REye": 14,
    "LEye": 15, "REar": 16, "LEAR": 17, "Background": 18 }

POSE_PAIRS = [ ["Neck", "RShoulder"], ["Neck", "LShoulder"], ["RShoulder",
    "RElbow"], ["RElbow", "RWrist"], ["LShoulder", "LElbow"], ["LElbow", "LWrist"],
    ["Neck", "RHip"], ["RHip", "RKnee"], ["RKnee", "RAngle"], ["Neck", "LHip"],
    ["LHip", "LKnee"], ["LKnee", "LAngle"], ["Neck", "Nose"], ["Nose", "REye"],
    ["REye", "REar"], ["Nose", "LEye"], ["LEye", "LEAR"] ]

w = 350
h = 350

model = cv2.dnn.readNetFromTensorflow("graph_opt.pb")

capture = cv2.VideoCapture("pose_video.mp4")

while cv2.waitKey(1) < 0:
    hasFrame, img = capture.read()
    img = cv2.resize(img, (600, 338))
    if not hasFrame:
        cv2.waitKey()
        break

    f_w = img.shape[1]
    f_h = img.shape[0]
```

```

model.setInput(cv2.dnn.blobFromImage(img, 1.0, (w, h), (127.5, 127.5, 127.5),
swapRB=True, crop=False))
detect = model.forward()
detect = detect[:, :19, :, :]

assert(len(BODY_PARTS) == detect.shape[1])

points = []
for i in range(len(BODY_PARTS)):
    heatMap = detect[0, i, :, :]
    _, conf, _, point = cv2.minMaxLoc(heatMap)
    x = (f_w * point[0]) / detect.shape[3]
    y = (f_h * point[1]) / detect.shape[2]
    points.append((int(x), int(y)) if conf > 0.2 else None)

for pair in POSE_PAIRS:
    partFrom = pair[0]
    partTo = pair[1]
    assert(partFrom in BODY_PARTS)
    assert(partTo in BODY_PARTS)

    idFrom = BODY_PARTS[partFrom]
    idTo = BODY_PARTS[partTo]

    if points[idFrom] and points[idTo]:
        cv2.line(img, points[idFrom], points[idTo], (0, 255, 0), 3)
        cv2.ellipse(img, points[idFrom], (3, 3), 0, 0, 360, (0, 0, 255), cv2.FILLED)
        cv2.ellipse(img, points[idTo], (3, 3), 0, 0, 360, (0, 0, 255), cv2.FILLED)

cv2.imshow('Pose Detection', img)

```

Hareketli Görüntülerin Kameradan Alınması

Öğrencilerden hareketli görüntü üzerinden pose detection sürecini kameraları üzerinden deneyimlemeleri istenir. Bunun için kodlar içerisinde **VideoCapture** fonksiyonunda yer alan video dosyası parametresi silinerek yerine **0** yazılır (İlgili kod satırında fonksiyon kullanımı **VideoCapture(0)** şeklinde olacaktır).



4. ADIM: İLERLET

Öğrencilerin gruplar halinde çalışarak gerçekleştirdikleri uygulamalar üzerinde değişikliklerle denemeler yapmaları istenir. Bu aşama için toplam 10 dakikalık bir süre ayrılr.



5. ADIM: DEĞERLENDİRME

Öğrencilerin yaptıkları uygulamalardan elde ettikleri tecrübeler ve derste öğrenilenler ışığında soru-cevap etkinliği gerçekleştirilerek pose detection (duruş tespiti) uygulamalarına dair bilgiler pekiştirilir.

Öğrencilere şu sorular sorulabilir:

- Gerçekleştirilen pose detection uygulaması sayesinde elde ettiğiniz bilgi ve beceriler neler olmuştur?
- Gerçekleştirdiğiniz pose detection uygulamalarının sonuçları hakkında ne düşünüyorsunuz?
- Edindiğiniz pose detection odaklı bilgi ve becerilerden hareketle farklı ne gibi Yapay Zeka uygulamaları geliştirmek isterdiniz?

3. HAFTA – 2. GÜN – 2., 3. DERSLER: FACE DETECTION UYGULAMASI

DERS PLANI

DERS ETİKETLERİ



KONU: Face Detection Uygulaması



SINIF: 12-14 Yaş



SÜRE: 120 dk.



HAZIR BULUNUŞLUK: İngilizce eğitimi, Yapay Zeka kavramı ve uygulamaları hakkında temel bilgi, OpenCV kütüphanesi hakkında temel bilgi, Python programlama dili hakkında temel bilgi



ANAHTAR KELİMELER: Yapay Zeka, Algoritma, Kodlama, Python, Görüntü İşleme, OpenCV, Face Detection



BECERİLER:



Tahmin



Sorgulama



Grup çalışması



Günlük hayat bağlantısı kurma



KAZANIMLAR:

Bu derste öğrenciler;

- Face detection (yüz tespiti) yaklaşımının dayandığı temeller konusunda bilgi edinirler.
- OpenCV kütüphanesi ve ön eğitimli bir Yapay Zeka modeli üzerinden face detection (yüz tespiti) uygulaması geliştirme konusunda bilgi ve beceri sahibi olurlar.
- Sabit ve hareketli görseller kapsamında face detection (yüz tespiti) konusunda bilgi ve beceri sahibi olurlar.



DERS MATERİYALLERİ:

Malzeme	Miktar
Yazı Tahtası	Öğrenci sayısı ve sınıf büyüklüğüne göre en az bir adet
Projeksiyon	Öğrenci sayısı ve sınıf büyüklüğüne göre en az bir adet
Defter, Kalem	Öğrenci sayısı kadar
Bilgisayar	Öğrenci sayısının yarısı kadar
Tablet	Öğrenci sayısının yarısı kadar
Python Anaconda Framework	Her bilgisayarda kurulu olacak şekilde



ÖĞRETMEN HAZIRLIK ÇALIŞMALARI:

Okunabilecek Kaynaklar:

Hjelmas, E., & Low, B. K. (2001). Face detection: A survey. *Computer vision and image understanding*, 83(3), 236-274.

Tan, M., & Emeksiz, C. (2023). Yüz Tanıma Sistemleri İçin Geliştirilmiş Veri Artırma Temelli Adaptif Yüz Tanıma Modeli. *Düzce Üniversitesi Bilim ve Teknoloji Dergisi*, 11(2), 588-606.

Turan, H., & Doğan, H. (2024). Yüz Tanıma Tabanlı Öğrenci Takip Sistemi. *Türk Bilim ve Mühendislik Dergisi*, 6(1), 1-7.

İzlenebilecek Kaynaklar:

Çelebi, V. (2021). Python-OpenCV ile Yüz Tanıma. Çevirmişi:
<https://www.youtube.com/watch?v=D2bWa4-FAss> (Erişim 18 Haziran 2024).

Learn OpenCV. (2023). What is Face Detection? – The Ultimate Guide for 2022. Çevirmişi:
<https://www.youtube.com/watch?v=-Di50FIoq8w> (Erişim 18 Haziran 2024).

ATÖLYE HAZIRLIK ÇALIŞMALARI:

Öğretmen;

- Dersten önce ders materyallerini hazırlamalıdır.
- Python ortamında OpenCV kütüphanesi kullanımı konusunda bilgi ve beceri sahibi olmalıdır.

- Face detection (yüz tespiti) uygulamaları konusunda ön bilgi ve farkındalık sahibi olmalıdır.



KAYNAKÇA:

Guo, G., Wang, H., Yan, Y., Zheng, J., & Li, B. (2020). A fast face detection method via convolutional neural network. *Neurocomputing*, 395, 128-137.

Oloyede, M. O., Hancke, G. P., & Myburgh, H. C. (2020). A review on face recognition systems: recent approaches and challenges. *Multimedia Tools and Applications*, 79(37), 27891-27922.

Soo, S. (2014). Object detection using Haar-cascade Classifier. *Institute of Computer Science, University of Tartu*, 2(3), 1-12.

Taşkıran, M., Kahraman, N., & Erdem, C. E. (2020). Face recognition: Past, present and future (a review). *Digital Signal Processing*, 106, 102809.

GENEL BAKIŞ:

1) Harekete Geç: Öğrencilere OpenCV ve Python yardımıyla insanlarda yüz tanımanın nasıl yapılabileceği konusunda fikirleri sorulur ve dönütler tartışılırak yüz tanımanın temel mantığı açıklanır.

2) Keşfet: Öğrencilerin OpenCV ve Python kodlama yardımıyla ön eğitimli bir Yapay Zeka modeli üzerinden sabit görselde face detection (yüz tespiti) gerçekleştirmeleri sağlanır.

3) Üret: Öğrencilerin önceki uygulamada yüz tespiti süreçlerini hareketli görüntüler yönünde revize etmeleri sağlanır.

4) İlerlet: Öğrencilerin gruplar halinde çalışarak gerçekleştirdikleri uygulamalar üzerinde değişikliklerle denemeler yapmaları istenir.

5) Değerlendir: Öğrencilerin yaptıkları uygulamalarдан elde ettikleri tecrübeler ve derste öğrenilenler ışığında soru-cevap etkinliği gerçekleştirilerek face detection (yüz tespiti) uygulamalarına dair bilgiler pekiştirilir.

UYGULAMA



1. ADIM: HAREKETE GEÇ

Öğrencilere OpenCV ve Python yardımıyla insanlarda yüz tanımanın nasıl yapılabileceği konusunda fikirleri sorulur ve dönütler tartışılırak yüz tanımanın temel mantığı açıklanır.

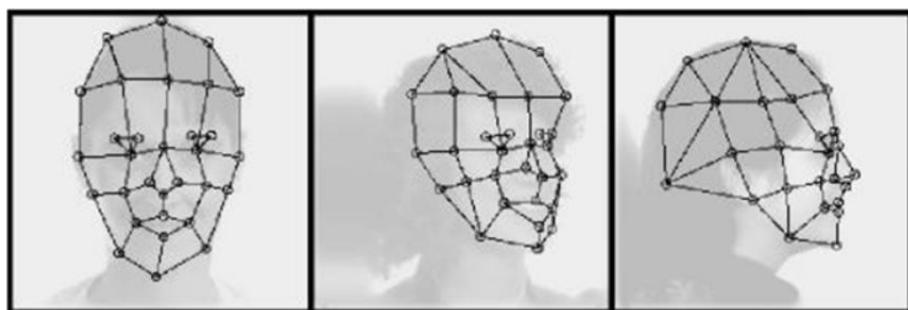
Görüntü İşleme ve Yapay Zeka Yardımıyla Yüz Tanıma

Görüntü işleme ve Yapay Zeka birleşimi, tipki pose detection uygulamalarında olduğu gibi face detection (yüz tespiti) uygulamalarında da ilgi görmektedir. Face detection çözümleri

gerçekleşen tespitler üzerinden insanların tanımlanması, duygusal durumlarının tespiti gibi farklı tahminler amacıyla tercih edilmektedir (Guo vd., 2020; Taşkiran vd., 2020).

Face detection yaklaşımı pose detection için izlenen yöntemlere (Bkz. 3. Hafta – 2. Gün – 1. Ders) benzer şekilde temsili noktaların bulunmasına dayanmaktadır. Bu noktaların bulunması sadece insan yüzlerinin tespitinde değil, nokta pozisyonlarına göre ifadelerin (dolayısıyla duyguların, yüz mimiklerine tekabül eden anımların...vs.) sınıflandırılması yönünde çözümlerin tasarlanması kolaylaştırmaktadır.

Öğrencilere aynı gün içerisinde değinilen OpenPose kütüphanesinde yüz tespiti için kullanılan 70 adet temsil noktasının söz konusu yaklaşımı desteklediği, farklı yüz tespiti çözümlerinde nokta sayısının değişkenlik gösterebileceği açıklanır. İlgili noktaların genel hatlarıyla insan yüzünde yer alan yüz bölgesi çevresi, gözler, burun, dudak, çene gibi bölgelerin belirlenmesinde kullanıldığı irdelenir. Şekil 20 bu konuda örnek bir temsili göstermektedir ()�.



Şekil 20. Face detection yaklaşımında yüzde yer alan temsil noktalarına yönelik bir örnek
(Oloyede vd., 2020).

Face Detection İçin Haar Cascade Modeli

Face detection uygulamalarında performans ve başarı odaklı çalışmalar birçok farklı teknığın ve Yapay Zeka modellerinin geliştirilmesini sağlamıştır. Bu noktada “Haar Cascade” modeli özellikle yüksek başarılı güncel modeller karşısında hızlı tespit avantajıyla popüleritesini korumaktadır. Haar Cascade temel olarak görsel üzerinde taramalar yaparak bir nesneyi temsil eden özelliklerin yer aldığı alanların dikkate alınması suretiyle nesne tespitlerinin yapılabilmesini sağlar (Soo, 2014). Anlaşılacağı üzere, söz konusu özellikler konusunda insan yüzünü temsil eden özelliklerin kullanımını face detection uygulamalarını desteklemektedir.



2. ADIM: KEŞFET

Öğrencilerin OpenCV ve Python kodlama yardımıyla ön eğitimli bir Yapay Zeka modeli üzerinden sabit görselde face detection (yüz tespiti) gerçekleştirmeleri sağlanır.

Kütüphane Çağrıları ve Ön Tanımlamalar

Öğrencilerin Haar Cascade Yapay Zeka modelinin ön eğitimli olarak yer aldığı model dosyasını indirmeleri sağlanır.

MODEL İÇİN İNDİRME LİNKİ

https://raw.githubusercontent.com/opencv/opencv/master/data/haarcascades/haarcascade_frontalface_default.xml

Model dosyası Python uygulamasının kodlanacağı klasör altına konumlandırılır ve Şekil 21'de gösterildiği gibi OpenCV kütüphanesi çağrıları yapılarak OpenCV'nin sağladığı “**CascadeClassifier**” fonksiyonu yardımıyla model tanımının Python değişkenine okunması yönünde kod yazılır.

```
7 import cv2
8
9 face_model = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
```

Şekil 21. OpenCV ile Haar Cascade modelinin oluşturulması.

Model çağrısını takip eden kod satırlarında içerisinde insan yüzlerinin yer aldığı örnek görüntü dosyası okunur ve OpenCV “**cvtColor**” fonksiyonu yardımıyla görsel gri tonlamalı versiyona dönüştürülür (Şekil 22).

```
11 img = cv2.imread('faces.jpg')
12
13 gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
```

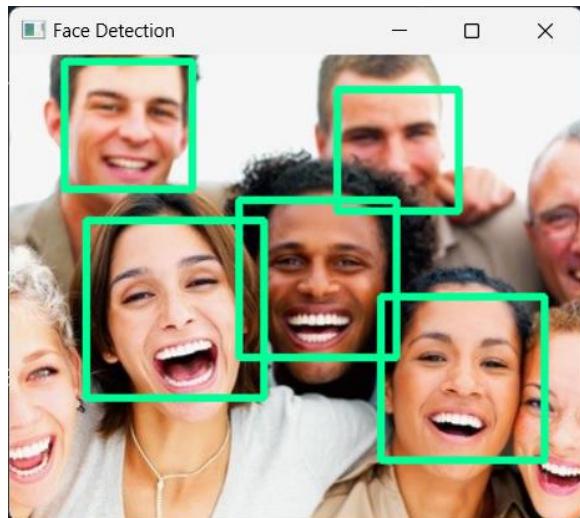
Şekil 22. Örnek görüntü dosyasının okunması ve gri tonlamaya dönüştürülmesi.

Haar Cascade ile Face Detection Aşamaları

Oluşturulan model “**detectMultiScale**” fonksiyonunu kullanmak suretiyle gri tonlamalı görsel üzerinde face detection sürecini otomatik olarak gerçekleştirmektedir. Bu amaçla Şekil 23'de gösterilen 15. satırdaki kod yazılır ve ardından for döngüsü kullanılmak suretiyle tespit edilen yüzlerin orijinal görsel üzerinde dikdörtgenler çizilmek suretiyle (Şekil 23 17. ve 18. satırlar) tespiti sağlanmış olur. Yapılan tespitler imshow fonksiyonu yardımıyla ekrana yansıtılır.

```
15 detected_faces = face_model.detectMultiScale(gray, 1.1, 4)
16
17 for (x, y, w, h) in detected_faces:
18     cv2.rectangle(img, (x, y), (x+w, y+h), (150, 255, 0), 3)
19
20 cv2.imshow('Face Detection', img)
21 cv2.waitKey()
```

(a)



(b)

Şekil 23. (a) Face detection çıkışısının elde edilmesine dair kodlar
(b) kodların çalıştırılması sonucu elde edilen pencere görüntüsü

UYGULAMANIN PYTHON KODLARI

```
import cv2
face_model = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
img = cv2.imread('faces.jpg')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
detected_faces = face_model.detectMultiScale(gray, 1.1, 4)

for (x, y, w, h) in detected_faces:
    cv2.rectangle(img, (x, y), (x+w, y+h), (150, 255, 0), 3)

cv2.imshow('Face Detection', img)
cv2.waitKey()
```



3. ADIM: ÜRET

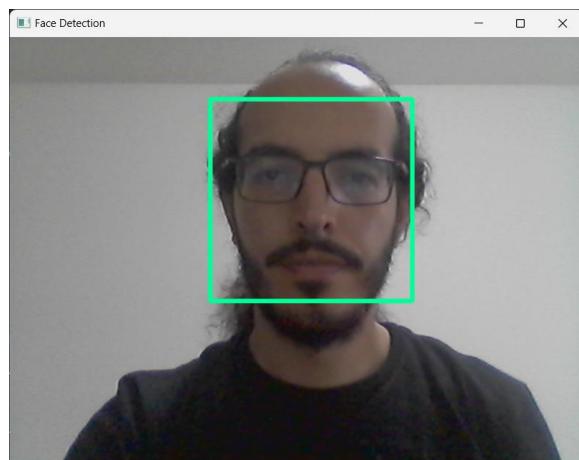
Öğrencilerin önceki uygulamada yüz tespiti süreçlerini hareketli görüntüler yönünde revize etmeleri sağlanır.

Kamera Görüntüsü Üzerinden Face Detection

Öğrencilere pose detection uygulamasında gerçekleştirildiği gibi; model çağrısı sonrasında yer alan kod bölümlerinde “**VideoCapture**” fonksiyonu kullanılmak suretiyle kamera görüntüsü üzerinden face detection gerçekleştirilebileceği belirtilir. Bu noktada mevcut Python kodları Şekil 24’te olduğu gibi revize edilerek kamera görüntüsünden face detection yapılması sağlanır (Şekil 24b’de örnek bir kamera görüntüsü tespit süreci gösterilmiştir). Öğrencilere pose detection uygulamasında yapılan revizyon ile bu uygulamada yapılan revizyon arası benzerlikler vurgulanarak öğrencilere kodlama aşamalarında izlenebilecek pratik yolların önemi açıklanır.

```
11 capture = cv2.VideoCapture(0)
12
13 while cv2.waitKey(1) < 0:
14     hasFrame, img = capture.read()
15     if not hasFrame:
16         cv2.waitKey()
17         break
18
19     gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
20
21     detected_faces = face_model.detectMultiScale(gray, 1.1, 4)
22
23     for (x, y, w, h) in detected_faces:
24         cv2.rectangle(img, (x, y), (x+w, y+h), (150, 255, 0), 3)
25
26 cv2.imshow('Face Detection', img)
```

(a)



(b)

Şekil 24. (a) Kamera görüntüsü için face detection çıkışının elde edilmesine dair kodlar
(b) kodların çalıştırılması sonucu elde edilen pencere görüntüsü

UYGULAMANIN PYTHON KODLARI

```
import cv2

face_model = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
capture = cv2.VideoCapture(0)

while cv2.waitKey(1) < 0:
    hasFrame, img = capture.read()
    if not hasFrame:
        cv2.waitKey()
        break

    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

    detected_faces = face_model.detectMultiScale(gray, 1.1, 4)

    for (x, y, w, h) in detected_faces:
        cv2.rectangle(img, (x, y), (x+w, y+h), (150, 255, 0), 3)

    cv2.imshow('Face Detection', img)
```



4. ADIM: İLERLET

Öğrencilerin gruplar halinde çalışarak gerçekleştirdikleri uygulamalar üzerinde değişikliklerle denemeler yapmaları istenir. Söz konusu etkinlik için toplamda 10 dakikalık bir süre kullanılır.



5. ADIM: DEĞERLENDİRME

Öğrencilerin yaptıkları uygulamalardan elde ettikleri tecrübeler ve derste öğrenilenler ışığında soru-cevap etkinliği gerçekleştirilerek face detection (yüz tespiti) uygulamalarına dair bilgiler pekiştirilir.

Öğrencilere şu sorular sorulabilir:

- Yapmış olduğunuz face detection uygulaması ile elde ettiğiniz bilgi ve beceriler neler olmuştur?
- Face detection uygulamalarının sonuçları hakkında fikirleriniz nelerdir?
- Elde ettiğiniz face detection odaklı bilgi ve becerilerden hareketle farklı ne gibi Yapay Zeka uygulamaları geliştirmek isterdiniz?
- Şu ana kadar öğrendiğiniz pose detection ve face detection uygulamalarının birleşimi ile ne gibi Yapay Zeka çözümleri elde edilebilir?

3. HAFTA – 3. GÜN – 1., 2. DERSLER: SELENIUM / PYTORCH KÜTÜPHANE TANITIMI

DERS PLANI

DERS ETİKETLERİ



KONU: Selenium / Pytorch Kütüphane Tanıtımı



SINIF: 12-14 Yaş



SÜRE: 120 dk.



HAZIR BULUNUŞLUK: İngilizce eğitimi, Yapay Zeka kavramı ve uygulamaları hakkında temel bilgi, Python programlama dili hakkında temel bilgi



ANAHTAR KELİMELER: Yapay Zeka, Algoritma, Kodlama, Python, Selenium, Pytorch



BECERİLER:



Tahmin



Sorgulama



Grup çalışması



Günlük hayat bağlantısı kurma



KAZANIMLAR:

Bu derste öğrenciler;

- Python Selenium kütüphanesi konusunda bilgi edinirler.
- Python Pytorch kütüphanesi konusunda bilgi edinirler.
- Söz konusu Selenium ve Pytorch kütüphanelerinin kullanılması suretiyle gerçekleştirilebilecek Yapay Zeka tabanlı uygulamalar konusunda farkındalık sahibi olurlar.



DERS MATERİYALLERİ:

Malzeme	Miktar
Yazı Tahtası	Öğrenci sayısı ve sınıf büyüklüğüne göre en az bir adet
Projeksiyon	Öğrenci sayısı ve sınıf büyüklüğüne göre en az bir adet
Defter, Kalem	Öğrenci sayısı kadar
Bilgisayar	Öğrenci sayısının yarısı kadar
Tablet	Öğrenci sayısının yarısı kadar
Python Anaconda Framework	Her bilgisayarda kurulu olacak şekilde



ÖĞRETMEN HAZIRLIK ÇALIŞMALARI:

Okunabilecek Kaynaklar:

Nyamathulla, S., Ratnababu, P., & Shaik, N. S. (2021). A review on selenium web driver with python. *Annals of the Romanian Society for Cell Biology*, 16760-16768.

Stevens, E., Antiga, L., & Viehmann, T. (2020). *Deep learning with PyTorch*. Manning Publications.

İzlenebilecek Kaynaklar:

Loeber, P. (2020). PyTorch Tutorials - Complete Beginner Course. Çevirmiçi: <https://www.youtube.com/watch?v=EMXfZB8FVUA&list=PLqnslRFeH2UrcDBWF5mfPGpqQDSta6VK4> (Erişim 18 Haziran 2024).

Makine Öğrenmesi. (2019). PyTorch Dersleri. Çevirmiçi: https://www.youtube.com/watch?v=Y1a_uzYHVps&list=PLRRY18KNZTgWqNVa5SeqUi7r925NjXTJn (Erişim 18 Haziran 2024).

Tech with Tim. (2020). Python Selenium Tutorials. Çevirmiçi: <https://www.youtube.com/watch?v=Xjv1sY630Uc&list=PLzMcBGfZo4-n40rB1XaJ0ak1bemvlqumQ> (Erişim 19 Haziran 2024).

Test Otomasyon Mühendisi. (2022). Python ve Selenium ile web test otomasyonu. Çevirmiçi: <https://www.youtube.com/watch?v=MM10SYKfqwA&list=PLbFzTYWXNIJ4BrHcQ5dVhy9xhLtLg2Kcr> (Erişim 19 Haziran 2024).

ATÖLYE HAZIRLIK ÇALIŞMALARI:

Öğretmen;

- Dersten önce ders materyallerini hazırlamalıdır.
- Python ortamında Selenium kütüphanesi kullanımı konusunda bilgi ve beceri sahibi olmalıdır.
- Python ortamında Pytorch kütüphanesi kullanımı konusunda bilgi ve beceri sahibi olmalıdır.



KAYNAKÇA:

Gundecha, U. (2015). *Selenium Testing Tools Cookbook*. Packt Publishing Ltd.

Raschka, S., Liu, Y. H., & Mirjalili, V. (2022). *Machine Learning with PyTorch and Scikit-Learn: Develop machine learning and deep learning models with Python*. Packt Publishing Ltd.

GENEL BAKIŞ:

- 1) Harekete Geç:** Öğrencilere Selenium ve Pytorch kütüphaneleri hakkında daha önce bilgi sahibi olup olmadıkları sorulur ve ilgili kütüphanelerin Python ve Yapay Zeka uygulamalarına getirdikleri avantajları vurgulayacak örnekler verilerek öğrencilerin ilgisini çekilir.
- 2) Keşfet:** Öğrencilere Selenium ve Pytorch kütüphanelerine ilişkin genel bir tanıtım yapılır.
- 3) Üret:** Öğrencilerin Selenium ve Pytorch kütüphaneleri ile bağlantılı temel kodlama süreçlerini örnek uygulamalar üzerinden gözlemlenmeleri sağlanır.
- 4) İlerlet:** Öğrencilerin gruplar halinde çalışarak gerçekleştirdikleri uygulamalar üzerinde değişikliklerle denemeler yapmaları istenir.
- 5) Değerlendir:** Öğrencilerin yaptıkları uygulamalardan elde ettikleri tecrübeler ve derste öğrenilenler ışığında soru-cevap etkinliği gerçekleştirilerek Selenium ve Pytorch kütüphanelerine dair bilgiler pekiştirilir.

UYGULAMA



1. ADIM: HAREKETE GEÇ

Harekete geç adımda öğrencilere Selenium ve Pytorch kütüphaneleri hakkında daha önce bilgi sahibi olup olmadıkları sorulur ve ilgili kütüphanelerin Python ve Yapay Zeka uygulamalarına getirdikleri avantajları vurgulayacak örnekler verilerek öğrencilerin ilgisini çekilir.



2. ADIM: KEŞFET

Öğrencilere Selenium ve Pytorch kütüphanelerine ilişkin genel bir tanıtım yapılır.

Python Selenium Kütüphanesi

Selenium sunduğu driver bileşeni ve çeşitli fonksiyonlar yardımıyla Web browser ortamıyla kullanıcı etkileşiminin kodlanmış adımlar içerisinde gerçekleştirilebilmesini mümkün kılan bir Python kütüphanesidir. Selenium'un sunduğu olanaklar sayesinde Web browser yardımıyla gerçekleştirilebilen birçok işlemin bir Python uygulaması tarafından gerçekleştirilebilmesi sağlanabilir; hatta bu sayede Web platformları üzerindeki işlemler otomatikleştirip, test, kontrol, uygulama süreçlerine yönelik araçlar geliştirilebilir (Gundecha, 2015).

Selenium kütüphanesinin Python ortamlarına dahil edilmesi için **pip install selenium** komutuyla kurulum yapılır. Öğrencilerden Spyder ortamında Selenium kurulumunu gerçekleştirmeleri istenir.

Öğrencilerle birlikte <https://www.selenium.dev/> Web adresi ziyaret edilerek farklı linkler altında sunulan sayfalar ziyaret edilir. Selenium Web platformu giriş sayfası Şekil 25'teki gibidir.

The screenshot shows the official Selenium website at https://www.selenium.dev/. The header features the Selenium logo and navigation links for About, Downloads, Documentation, Projects, Support, Blog, and English. The main banner has a green gradient background with the text "Selenium automates browsers. That's it!" and "What you do with that power is entirely up to you.". Below the banner, there are three main sections: "Getting Started" with links to Selenium WebDriver, IDE, and Grid. Each section includes a brief description and a "READ MORE" button.

Section	Description	Link
Selenium WebDriver	If you want to create robust, browser-based regression automation suites and tests, scale and distribute scripts across many environments, then you want to use Selenium WebDriver, a collection of language specific bindings to drive a browser - the way it is meant to be driven.	READ MORE
Selenium IDE	If you want to create quick bug reproduction scripts, create scripts to aid in automation-aided exploratory testing, then you want to use Selenium IDE; a Chrome, Firefox and Edge add-on that will do simple record-and-playback of interactions with the browser.	READ MORE
Selenium Grid	If you want to scale by distributing and running tests on several machines and manage multiple environments from a central point, making it easy to run the tests against a vast combination of browsers/OS, then you want to use Selenium Grid.	READ MORE

Şekil 25. Selenium Web platformu giriş sayfası.

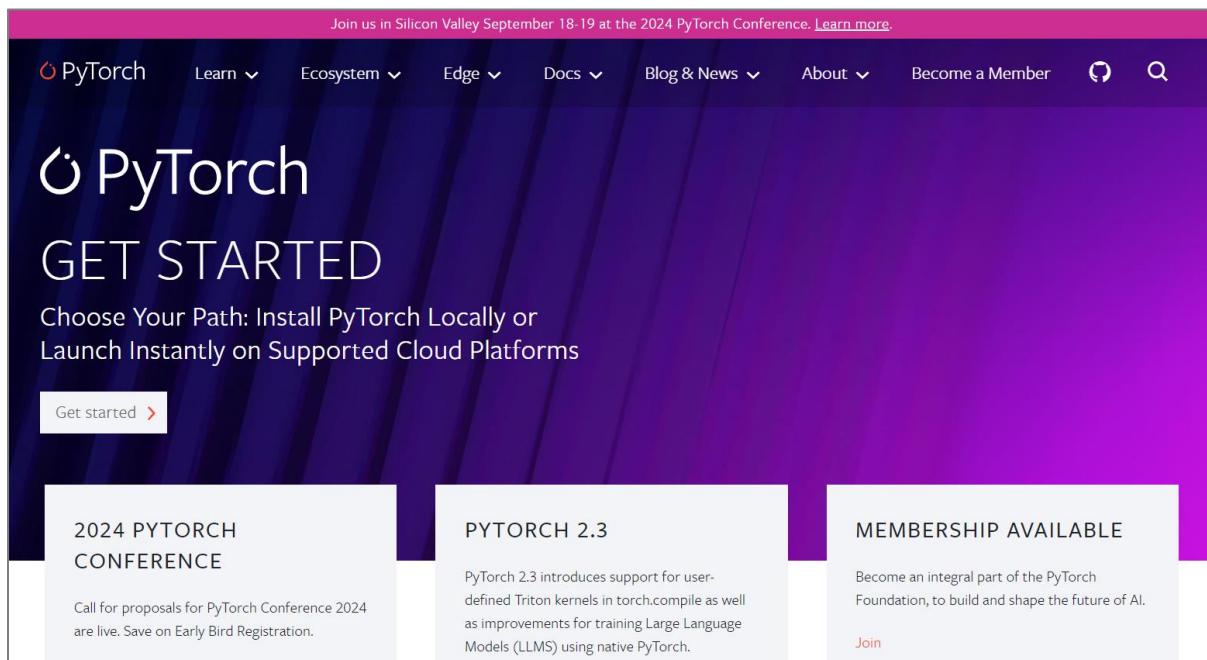
Selenium kütüphanesi Python uygulamaları ortamında özellikle **Selenium WebDriver** bileşeni yardımıyla dahil olmaktadır. Bunun haricinde **Selenium IDE** bileşeni otomatik test süreçlerine yönelik kodlamaların yapılması için Web browser odaklı eklentiler sunmakta, **Selenium Grid** bileşeni de yine test süreçlerine yönelik geliştirilen araçların aynı anda farklı sistemlerde çalışabilmesini mümkün kılmaktadır (Gundecha, 2015).

Python Pytorch Kütüphanesi

Pytorch Facebook AI Araştırma Laboratuvarı (FAIR) tarafından ilk olarak 2016 yılında yayınlanan ve günümüzde yaygınlığını sürdürden, Makine Öğrenmesi ve Derin Öğrenme çözümlerinde tercih edilen bir Python kütüphanesidir. Pytorch spesifik olarak bilgisayarlı görü ve doğal dil işleme uygulamalarında ön plana çıkmaktadır (Raschka vd., 2022).

Öğrencilerden Pytorch kütüphanesini de Python ortamlarına kurmaları istenir. Bu amaçla konsolda **pip install torch** komutu çalıştırılır.

Öğrencilerle beraber <https://pytorch.org/> Web adresi ziyaret edilir ve farklı linkler altında sunulan Pytorch sayfaları ziyaret edilir. Şekil 26 Pytorch kütüphanesine ait Web platformunu göstermektedir.



Şekil 26. Pytorch Web platformu giriş sayfası.



3. ADIM: ÜRET

Bu adımda öğrencilerin Selenium ve Pytorch kütüphaneleri ile bağlantılı temel kodlama süreçlerini örnek uygulamalar üzerinden gözlemlemeleri sağlanır.

Selenium Kütüphanesi ile Temel Web Browser İşlemleri

Öğrencilerin Şekil 27'de olduğu gibi Python kod sayfasında **from selenium import webdriver** komutu yardımıyla Selenium kütüphanesini çağırmaları ve ardından webdriver bileşeni altından Firebox yazılımını kontrol edecek driver bileşenini çağırarak brw değişkenini tanımlamaları sağlanır. Son olarak "**get**" fonksiyonunu kullanmak suretiyle Selenium anasayfasının açılması yönünde kod yazılarak uygulama çalıştırılır. Şekil 27b'de görüldüğü

gibi Firefox penceresi sayfa açılış süreciyle birlikte browserdaki kontrolün uzaktan (bot tabanlı) yapıldığını tespit etmektedir.

```

7   from selenium import webdriver
8
9   brw = webdriver.Firefox()
10  brw.get('http://selenium.dev/')

```

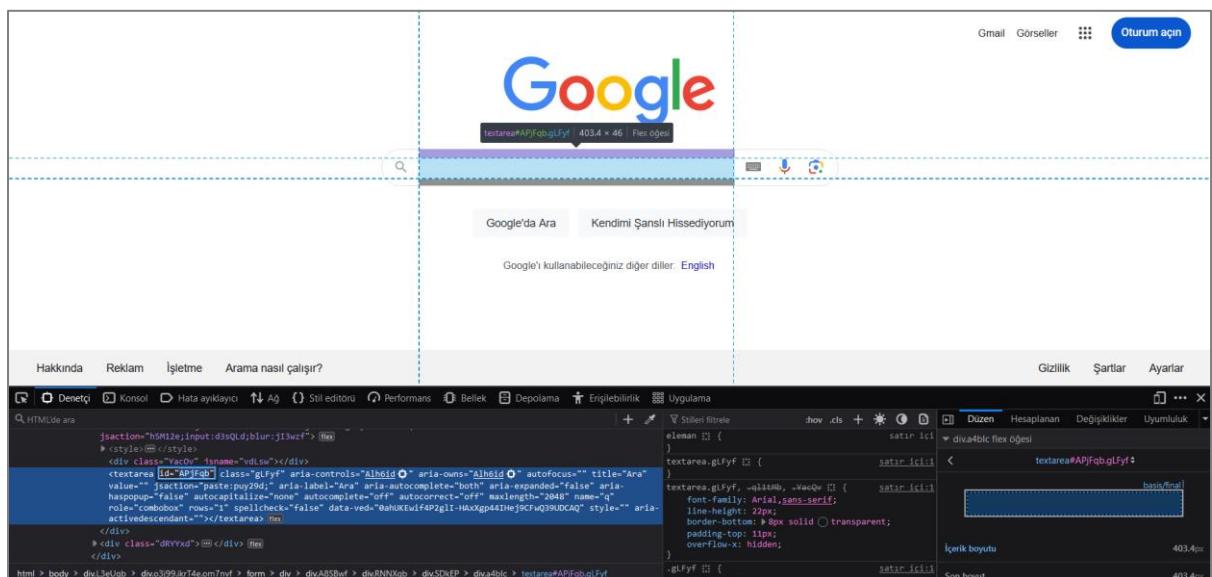
(a)



(b)

Şekil 27. (a) Selenium ile Firefox üzerinden Web sayfası ziyaretine yönelik kodlar
(b) Firefox penceresinde gösterilen uzaktan kontrol (bot) kullanımına dair ibare

Öğrencilere Selenium kütüphanesinin browser ile etkileşim konusunda kendi içerisinde birçok farklı mekanizmayı barındırdığı ve bu nedenle gerekli olması durumunda Web kaynakları ve Selenium dokümantasyonu üzerinde ayrıca çalışılması gerektiği vurgulanır. Bu aşamada Web browser ortamında etkileşim kurulacak sayfalarla ilgili olarak otomatik kontrol edilecek bileşenler hakkında öncelikli olarak bilgi edinilmesi gerektiği ifade edilir. Örneğin, Google arama motorunda arama yapmak için öncelikli olarak arama kutucuğunun Firefox ortamında sağ tuş tıklanarak **Denetle** düğmesi ile incelenmesi ve bileşenle ilgili herhangi bir tanımlayıcı bilginin Selenium ortamında **find_element** fonksiyonu ile tespit edilmesi gerektiği açıklanır. Şekil 28, Google sayfası arama kutucuğunun ID bilgisinin temin ediliş şeklini göstermektedir.



Şekil 28. Google sayfasında arama kutucuğu ID bilgisinin alınması

Öğrenciler, bu aşamadan sonraki komutları daha iyi gözlemlerek adına kodlama ortamına **import time** kodunu yazarak komutlar arası bekleme sağlayacak **sleep** fonksiyonunu kullanmayı sağlayacak kütüphaneyi çağırırlar. Ardından Web bileşenine ID ile erişimi destekleyecek Selenium bileşeni **By** ve browsera klavye tuş komutu gönderimini sağlayacak **Keys** bileşeni kodlama ortamına dahil edilir (Şekil 29).

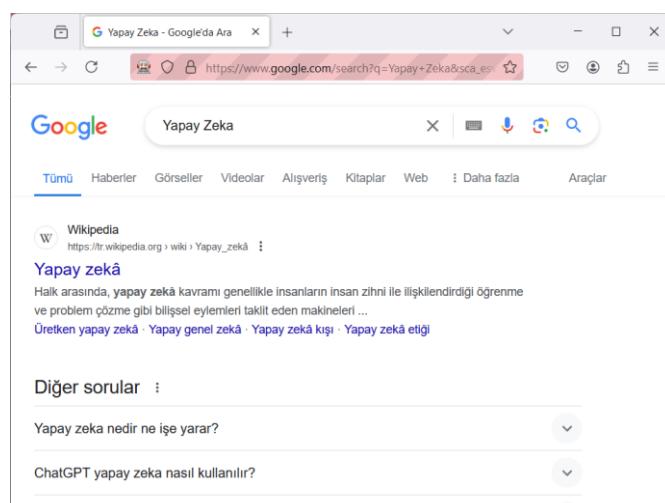
```
14 import time
15 from selenium.webdriver.common.by import By
16 from selenium.webdriver.common.keys import Keys
```

Şekil 29. Kod ortamına yeni kütüphanelerin dahil edilmesi

Öğrenciler Şekil 30'daki kodları yazarak Selenium ile bazı temel browser komutlarını çalıştırır ve hatta Google ortamında “Yapay Zeka” kelimesini aratırlar. 18. satırda **get** fonksiyonu ile Google arama motoru adresine yönlenme gerçekleştirildikten sonra, **time.sleep(1)** komutu ile 1 saniyelik bir bekleme gerçekleşmekte ardından 20. satırda pencere boyutu 800*600 boyutuna düşürülmektedir. Yine 22. satırdaki kod ile sayfa yenilemesi yapılmakta, 24. satırda ID bilgisi **APjFqb** olarak bulunan arama kutucuğu “search” adında bir değişken ile temsil edilmektedir. Son olarak 25. satırda **send_keys** fonksiyonu ile arama kutucığına “Yapay Zeka” kelimesi ve ardından Keys bileşeni altından **Enter** tuşu komutu gönderilerek arama işlemi gerçekleştirilmektedir.

```
18 brw.get('https://www.google.com')
19 time.sleep(1)
20 brw.set_window_size(800,600)
21 time.sleep(1)
22 brw.refresh()
23 time.sleep(1)
24 search = brw.find_element(By.ID, 'APjFqb')
25 search.send_keys('Yapay Zeka' + Keys.RETURN)
```

(a)



(b)

Şekil 30. (a) Selenium ile çeşitli işlemler ve Google ortamında aramaya dair kodlar
(b) Arama sonucu ulaşılan nihai pencere görüntüsü

Öğrencilere ihtiyaç olması halinde alternatif fonksiyonlar da gösterilebileceği gibi kendilerinin bir sonraki adım olan “İlerlet” adımı kapsamında Web kaynaklarında aramalar yaparak buldukları örnek fonksiyonları denemeleri yönünde teşvik edilmeleri sağlanabilecektir.

UYGULAMANIN PYTHON KODLARI

```
from selenium import webdriver
brw = webdriver.Firefox()
#
=====
=====
# brw.get('http://selenium.dev/')
#
=====
=====

import time
from selenium.webdriver.common.by import By
from selenium.webdriver.common.keys import Keys

brw.get('https://www.google.com')
time.sleep(1)
brw.set_window_size(800,600)
time.sleep(1)
brw.refresh()
time.sleep(1)
search = brw.find_element(By.ID, 'APjFqb')
search.send_keys('Yapay Zeka' + Keys.RETURN)
```

Pytorch Kütüphanesi ile Yapay Sinir Ağları Uygulaması

Pytorch kütüphanesi Makine Öğrenmesi kapsamında birçok farklı uygulamanın geliştirilmesine olanak sağlayan bir yapıdadır. Öğrencilere tipki Scikit-learn (sklearn) kütüphanesi gibi Pytorch kütüphanesinin de kapsamı geniş bir kütüphane olduğu vurgulanır. Ders kapsamında örnek olarak daha önce kullanılan şeker hastalığı (diyabet) veri seti (Bkz. 2. Hafta – 5. Gün – 1., 2., 3. Dersler) üzerinden Pytorch destekli Yapay Sinir Ağları modeli ile

uygulama yapılacakı açıklanır. Bu amaçla ilgili veri seti dosyasının yeni Python uygulama ortamı bağlamında hazır hale getirilmesi sağlanır.

Kütüphanelerin Dahil Edilmesi ve Veri Seti Hazırlığı

Pytorch kütüphanesi yapay sinir ağları ve eğitim algoritmaları kapsamında kendi içerisinde bileşenler taşımaktadır. Bu nedenle öğrencilerin Şekil 31'de olduğu gibi Pytorch kütüphanesi başta olmak üzere, **nn** ve **optim** bileşenlerini ayrı ayrı kod ortamına dahil etmesi sağlanır. Ayrıca, veri setinin işlenmesi noktasında kullanılacak olan Numpy kütüphanesi de kod ortamına dahil edilir. Yine Şekil 31 kapsamında 12. ve 17. satırlar arasında gösterilen kodlar yazılmak suretiyle şeker hastalığı veri setinin okunması ve girdi (X) ile çıktı (Y) değişkenleri altında organize edilmeleri yönünde adımlar atılır.

```
7 import torch
8 import torch.nn as nn
9 import torch.optim as optim
10 import numpy as np
11
12 data_set = np.loadtxt('diabetes_data.csv', skiprows=1, delimiter=',')
13 X = data_set[:,0:8]
14 Y = data_set[:,8]
15
16 X = torch.tensor(X, dtype=torch.float32)
17 Y = torch.tensor(Y, dtype=torch.float32).reshape(-1, 1)
```

Şekil 31. Pytorch tabanlı yapay sinir ağları uygulaması için kütüphanelerin kod ortamına dahil edilmesi ve veri seti hazırlığının tamamlanması

Numpy kütüphanesi ile birlikte kullanılan “**loadtxt**” fonksiyonu tipki CSV dosyalarını okumada olduğu gibi Numpy’ın veri seti dosyalarını okumak için kullanılan bir fonksiyondur (Bu fonksiyonda **skiprows=1** parametresi ile dosya içerisinde birinci satırda yer alan sütun isimlerinin okunmaması sağlanmaktadır). Bunun haricinde, X ve Y değişkenlerinin oluşturulması aşamasında Pytorch kütüphanesi aracılığıyla **tensor** adı verilen veri modeli yapılanması kullanılmaktadır. Tensor özellikle Yapay Zeka ve Veri Bilimi uygulamalarında bilinen ve verilerin dizi / matris düzleminde tanımlanmasını sağlayan bir temsil türü olarak tanımlanabilir.

Yapay Sinir Ağları ve Eğitim Algoritmasının Tanımlanması

Pytorch’ın “nn” bileşeni sayesinde katmanlı yapay sinir ağları pratik bir biçimde tanımlanabilmektedir. Bu amaçla öğrencilerin Şekil 32’de görüldüğü gibi katmanlı bir sinir ağı yapısı oluşturmaları sağlanır. Kodlarda “nn.Linear” tanımlaması birinci parametrede belirtilen sayı kadar girdiyi, ikinci parametrede belirtilen sayıda çıktıya dönüştürme noktasında girdi verilerine ağırlıklı dönüşüm uygulayan bir katmanın oluşturulmasını sağlamaktadır. Bu noktada “nn.ReLU” ve “nn.Sigmoid” tanımlamaları da ReLU ve Sigmoid dönüşüm fonksiyonları içeren katman tanımlarına tekabül eder.

Şekil 32’de yapay Sinir Ağları modeli oluşturulduktan sonra model yapısı ekrana yazdırılmakta, ardından modeli eğitecek algoritma da Pytorch’ın “optim” bileşeni üzerinden

çağırılmaktadır. Eğitim algoritması olarak sinir ağlarının etkin eğitimini sağlayan **Adam Optimizer** algoritması 0,002 öğrenme oranı (learning rate) parametresi ile tanımlanmıştır. Yine 30. satırda modelin eğitim performansını ölçen loss fonksiyonu tanımlanmış, 32. satırdaki “batch_size” değişkeni ile sinir ağlarının her eğitim döngü adımda modele sunulacak olan toplam örnek sayısı, 33. satırdaki “total_epochs” değişkeni ile toplam eğitim döngü sayısı belirlenmiştir.

```

19 ANN_model = nn.Sequential(
20     nn.Linear(8, 16),
21     nn.ReLU(),
22     nn.Linear(16, 10),
23     nn.ReLU(),
24     nn.Linear(10, 1),
25     nn.Sigmoid()
26 )
27 print("Oluşturulan Model:\n",ANN_model)
28
29 optimizer_alg = optim.Adam(ANN_model.parameters(), lr=0.002)
30 loss_func = nn.BCELoss()
31
32 batch_size = 10
33 total_epochs = 200

```

Şekil 32. Yapay sinir ağları modeli, eğitim algoritması ve eğitim sürecine ilişkin tanımlamalara dair kodlar

Öğrencilerin kodların devamında iç içe iki for döngüsü yardımıyla veri setinden örneklerin sinir ağı modelinden geçirilerek çıktı değerlerinin elde edildiği ve her döngü aşamasında loss değerlendirme değerinin elde edilerek eğitim algoritması yardımıyla ağı ağırlıklarının yenilendiği süreç kodlanır (Şekil 33). İlgili kodlar 37.-43. satırlar arasında yazılmış olup, her döngü (epoch) adımı sonunda ise (birinci for döngüsünün son adımı olarak) döngü değeri ve loss değerlendirme değeri ekrana yansıtılmaktadır.

Eğitimi tamamlanan yapay sinir ağları modeli için şeker hastalığı veri setindeki örnekler kapsamındaki doğruluk değeri uygulamanın son bölümlerinde (46.-49. satırlar arasında) kodlanarak, başarım değeri ekrana yansıtılmaktadır (Şekil 33).

```

35 for e in range(total_epochs):
36     for i in range(0, len(X), batch_size):
37         X_batch = X[i:i+batch_size]
38         Y_pred = ANN_model(X_batch)
39         Y_batch = Y[i:i+batch_size]
40         loss_s = loss_func(Y_pred, Y_batch)
41         optimizer_alg.zero_grad()
42         loss_s.backward()
43         optimizer_alg.step()
44     print(f'Geçekleşen epoch {e}, En son Loss değeri {loss_s}')
45
46 with torch.no_grad():
47     Y_pred = ANN_model(X)
48 acc_rate = (Y_pred.round() == Y).float().mean()
49 print(f"Doğruluk {acc_rate}")

```

Şekil 33. Yapay sinir ağları modelinin eğitimi ve doğruluk değerinin hesaplanması dair kodlar

Uygulamanın çalıştırılması neticesinde 200. epoch üzerinden eğitilen model örnek olarak 0,18 loss (hata) değeri ile birlikte şeker hastalığı veri seti için %83'lük bir başarım göstermiştir (Şekil 34). Öğrencilere söz konusu performansın her zaman aynı olmayabileceği açıklanarak ilgili uygulamanın birkaç kez daha çalıştırılması ve elde edilen sonuçların gözlemlenmesi istenir.

```
Oluşturulan Model:  
Sequential(  
    (0): Linear(in_features=8, out_features=16, bias=True)  
    (1): ReLU()  
    (2): Linear(in_features=16, out_features=10, bias=True)  
    (3): ReLU()  
    (4): Linear(in_features=10, out_features=1, bias=True)  
    (5): Sigmoid()  
)  
Gerçekleşen epoch 0, En son loss değeri 0.5949808359146118  
Gerçekleşen epoch 1, En son loss değeri 0.5537654161453247  
Gerçekleşen epoch 2, En son loss değeri 0.5387256145477295  
Gerçekleşen epoch 3, En son loss değeri 0.5319718718528748  
Gerçekleşen epoch 4, En son loss değeri 0.5276117920875549  
Gerçekleşen epoch 5, En son loss değeri 0.5219323039054871  
Gerçekleşen epoch 6, En son loss değeri 0.5206228494644165  
Gerçekleşen epoch 7, En son loss değeri 0.5179624557495117  
Gerçekleşen epoch 8, En son loss değeri 0.516030490398407  
...  
Gerçekleşen epoch 190, En son loss değeri 0.18929660320281982  
Gerçekleşen epoch 191, En son loss değeri 0.18090450763702393  
Gerçekleşen epoch 192, En son loss değeri 0.17980191111564636  
Gerçekleşen epoch 193, En son loss değeri 0.18871460855007172  
Gerçekleşen epoch 194, En son loss değeri 0.18754923343658447  
Gerçekleşen epoch 195, En son loss değeri 0.19716745615005493  
Gerçekleşen epoch 196, En son loss değeri 0.17398729920387268  
Gerçekleşen epoch 197, En son loss değeri 0.1858605593442917  
Gerçekleşen epoch 198, En son loss değeri 0.16416582465171814  
Gerçekleşen epoch 199, En son loss değeri 0.17924195528030396  
Doğruluk 0.8255208134651184
```

Şekil 34. Pytorch ile kodlanan yapay sinir ağları modeli uygulaması ile konsol ortamında elde edilen örnek ekran görüntüleri

UYGULAMANIN PYTHON KODLARI

```
import torch  
import torch.nn as nn  
import torch.optim as optim  
import numpy as np  
  
data_set = np.loadtxt('diabetes_data.csv', skiprows=1, delimiter=',')  
X = data_set[:,0:8]  
Y = data_set[:,8]
```

```

X = torch.tensor(X, dtype=torch.float32)
Y = torch.tensor(Y, dtype=torch.float32).reshape(-1, 1)

ANN_model = nn.Sequential(
    nn.Linear(8, 16),
    nn.ReLU(),
    nn.Linear(16, 10),
    nn.ReLU(),
    nn.Linear(10, 1),
    nn.Sigmoid()
)
print("Oluşturulan Model:\n",ANN_model)

optimizer_alg = optim.Adam(ANN_model.parameters(), lr=0.002)
loss_func = nn.BCELoss()
batch_size = 10
total_epochs = 200

for e in range(total_epochs):
    for i in range(0, len(X), batch_size):
        X_batch = X[i:i+batch_size]
        Y_pred = ANN_model(X_batch)
        Y_batch = Y[i:i+batch_size]
        loss_s = loss_func(Y_pred, Y_batch)
        optimizer_alg.zero_grad()
        loss_s.backward()
        optimizer_alg.step()
    print(f'Gerçekleşen epoch {e}, En son loss değeri {loss_s}'')

with torch.no_grad():
    Y_pred = ANN_model(X)
    acc_rate = (Y_pred.round() == Y).float().mean()
    print(f'Dogruluk {acc_rate}')

```



4. ADIM: İLERLET

Öğrencilerin gruplar halinde çalışarak gerçekleştirdikleri uygulamalar üzerinde değişikliklerle denemeler yapmaları istenir. Bu noktada öğrencilere mevcut uygulamalara Selenium ve Pytorch Web kaynaklarını araştırarak tespit edebilecekleri farklı fonksiyonları da kullanabilecekleri ifade edilir ve süreç içerisinde takıldıkları noktalarda yardım edilir. İlgili süreç için 20 dakikalık bir süre ayrılr.



5. ADIM: DEĞERLENDİRME

Öğrencilerin yaptıkları uygulamalardan elde ettikleri tecrübeler ve derste öğrenilenler ışığında soru-cevap etkinliği gerçekleştirilerek Selenium ve Pytorch kütüphanelerine dair bilgiler pekiştirilir.

Öğrencilere şu sorular sorulabilir:

- Python Selenium kütüphanesi hakkında fikirleriniz nelerdir?
- Python Pytorch kütüphanesi hakkında fikirleriniz nelerdir?
- Selenium ve Pytorch kütüphaneleri ile gerçekleştirdiğiniz uygulamalar hakkında neler düşünüyorsunuz?
- Selenium ve Pytorch kütüphaneleri yardımıyla geliştirilebilecek Python ve Yapay Zeka uygulamaları hakkında neler düşünürsünüz?

3. HAFTA – 3. GÜN – 3. DERS: SESLİ ASİSTAN

DERS PLANI

DERS ETİKETLERİ



KONU: Sesli Asistan



SINIF: 12-14 Yaş



SÜRE: 60 dk.



HAZIR BULUNUŞLUK: İngilizce eğitimi, Yapay Zeka kavramı ve uygulamaları hakkında temel bilgi, Python programlama dili hakkında temel bilgi, Python Pytorch kütüphanesi hakkında temel bilgi



ANAHTAR KELİMELER: Yapay Zeka, Algoritma, Kodlama, Python, Pytorch, Sesli Asistan



BECERİLER:



Tahmin



Sorgulama



Grup çalışması



Günlük hayat bağlantısı kurma



KAZANIMLAR:

Bu derste öğrenciler;

- Yapay Zeka tabanlı sesli asistan uygulamalarının izlediği çözüm süreçleri konusunda farkındalık sahibi olurlar.
- Python ortamında Yapay Zeka tabanlı bir sesli asistan geliştirme konusunda bilgi ve beceri sahibi olurlar.
- Pytorch ve bağlı kütüphaneler yardımıyla sesli asistan kodlama konusunda bilgi ve beceri sahibi olurlar.



DERS MATERİYALLERİ:

Malzeme	Miktar
Yazı Tahtası	Öğrenci sayısı ve sınıf büyüklüğüne göre en az bir adet
Projeksiyon	Öğrenci sayısı ve sınıf büyüklüğüne göre en az bir adet
Defter, Kalem	Öğrenci sayısı kadar
Bilgisayar	Öğrenci sayısının yarısı kadar
Tablet	Öğrenci sayısının yarısı kadar
Python Anaconda Framework	Her bilgisayarda kurulu olacak şekilde



ÖĞRETMEN HAZIRLIK ÇALIŞMALARI:

Okunabilecek Kaynaklar:

Maedche, A., Legner, C., Benlian, A., Berger, B., Gimpel, H., Hess, T., ... & Söllner, M. (2019). AI-based digital assistants: Opportunities, threats, and research perspectives. *Business & Information Systems Engineering*, 61, 535-544.

Manojkumar, P. K., Patil, A., Shinde, S., Patra, S., & Patil, S. (2023). AI-based virtual assistant using python: a systematic review. *International Journal for Research in Applied Science & Engineering Technology (IJRASET)*, 11.

İzlenebilecek Kaynaklar:

Assembly AI. (2024). Coding an AI Voice Bot from Scratch: Real-Time Conversation with Python. Çevirmiçi: https://www.youtube.com/watch?v=Nyo5m_glZXs (Erişim 19 Haziran 2024).

SkillCurb. (2024). Coding My Own Virtual Voice Assistant | OpenAI Whisper & StreamLit. Çevirmiçi: <https://www.youtube.com/watch?v=0-kvvEChjQo> (Erişim 19 Haziran 2024).

ATÖLYE HAZIRLIK ÇALIŞMALARI:

Öğretmen;

- Dersten önce ders materyallerini hazırlamalıdır.

- Python ortamında Yapay Zeka tabanlı sesli asistan geliştirme adımları konusunda bilgi ve beceri sahibi olmalıdır.
- Ders kapsamında kullanılan Python kütüphaneleri hakkında genel farkındalık sahibi olmalıdır.



KAYNAKÇA:

Akash, S., Jayaram, N., & Jesudoss, A. (2022). Desktop based smart voice assistant using python language integrated with arduino. In *2022 6th International Conference on Intelligent Computing and Control Systems (ICICCS)* (pp. 374-379). IEEE.

Pandey, D., Ali, A., Dubey, S., Srivastava, M., Dwivedi, S., & Raza, M. S. (2022). Voice Assistant Using Python and AI. *International Research Journal of Engineering and Technology*, 9(05), 832-838.

Pavitra, A. R. R., Ganeshan, M., Pavithran, P., & Rajamurugan, G. (2023). A Review on Intelligent Voice Assistant with Multilingual Support using Artificial Intelligence. *JETIR*. 10(4), b286-b290.

GENEL BAKIŞ:

- 1) Harekete Geç:** Öğrencilere Yapay Zeka tabanlı sesli asistan kullanıp kullanmadıkları sorularak, söz konusu sistemlerin nasıl geliştirilmiş olabileceği hakkındaki görüşler tartışılar.
- 2) Keşfet:** Öğrenciler sesli asistan için ilgili kütüphane ve model çağrılarını kodlar.
- 3) Üret:** Öğrenciler mikrofon üzerinden sesli komut dinlenmesi ve komutların model üzerinden tahmin edilmesi yönünde uygulama kodlarını yazar.
- 4) İlerlet:** Öğrencilerin gruplar halinde çalışarak gerçekleştirdikleri uygulamalar üzerinde değişikliklerle denemeler yapmaları istenir.
- 5) Değerlendir:** Öğrencilerin yaptıkları uygulamalardan elde ettikleri tecrübeler ve derste öğrenilenler ışığında soru-cevap etkinliği gerçekleştirilerek sesli asistan geliştirmeye dair bilgiler pekiştirilir.

UYGULAMA



1. ADIM: HAREKETE GEÇ

Harekete geç adımda öğrencilere Yapay Zeka tabanlı sesli asistan kullanıp kullanmadıkları sorularak, söz konusu sistemlerin nasıl geliştirilmiş olabileceği hakkındaki görüşler tartışılar.

Yapay Zeka Tabanlı Sesli Asistan Geliştirme

Günümüz Yapay Zeka geliştirmeleri, kullanıcıların sesli komutlarını dinleyerek çözümler üreten zeki asistanların geliştirilmesini oldukça kolaylaştırmaktadır. Özellikle mobil

teknolojilerin gelişimi ve gömülü sistemlerin popülerite kazanması, kompakt sesli asistan tasarımlarının ilerleyişinde rol oynamaktadır (Akash vd., 2022). Yine farklı mobil işletim sistemleri ve sosyal medya platformları, OpenAI, Meta ve Google gibi önde gelen şirketlerin teknoloji dünyasına kazandırdığı Yapay Zeka modellerini sesli asistan çözümleri yönünde etkin bir biçimde kullanmaktadır. Bu noktada sesli asistan modellerinin çalışma mantığı kısaca kullanıcı taraflı ses komutlarıyla eğitilmiş Yapay Zeka modellerinin anlık dinlenen ses verileri üzerinden tahminlerde bulunmasına ve tahminlere göre yazılımlardaki ilerleyen işlevlerin tetiklenmesi doğrultusunda şekillenmektedir (Pandey vd., 2022; Pavitra vd., 2023). Python programlama dili düşünüldüğünde söz konusu adımları ortaya koyabilmek için çeşitli kütüphanelerin etkileşimi şarttır.



2. ADIM: KEŞFET

Öğrenciler sesli asistan için ilgili kütüphane ve model çağrılarını kodlar.

Kütüphane ve Model Çağrıları

Öğrencilere sesli asistan geliştirme aşamasında birçok farklı Python kütüphanesi ve model kullanılabileceği açıklanır ve ders kapsamında gerçekleştirilecek uygulama için Pytorch kütüphanesinin ses verileri için tasarlanan “**Torchaudio**” versiyonu ile birlikte Hugging Face platformundan (Bkz. 1. Hafta – 4. Gün – 1. Ders) ön eğitilmiş bir model kullanılacağı ifade edilir.

Öğrencilerin uygulamadaki kütüphane bileşenleri ve fonksiyonları sahaklı bir şekilde kullanılmeleri için konsol ortamında sırasıyla aşağıdaki komutları yazarak yükleme işlemlerini tamamlamaları istenir (İlk iki yükleme Pytorch taraflı ses dosyaları işlemek için kullanılırken, son yükleme olan pvrecorder ise mikrofondan ses kaydı / kullanımı için gereklidir):

```
pip install torchaudio  
pip install soundfile  
pip install pvrecorder
```

Öğrenciler Şekil 35'te gösterilen komutları yazmak suretiyle Pytorch, torchaudio ve Hugging Face platformundan ön eğitimli Yapay Zeka modelini çekmek için kullanılacak **transformers** bileşenlerini çağırır. Yine okunan ses dosyalarını görselleştirmek için Matplot Pyplot bileşeni uygulama ortamına dahil edilmiştir. 13. ve 14. satırlarda ön eğitimli Türkçe Yapay Zeka modeli AutoProcessor ve AutoModelForCTC bileşenleri yardımıcılık Hugging Face ortamından Python ekosistemine dahil edilmiştir. Model ilk kez kullanıldığında (kodlar çalıştırıldığında) sistem model dosyalarını otomatik olarak indirmektedir.

Öğrencilerin uygulamada kullanılan ön eğitimli modeli incelemek adına <https://huggingface.co/m3hrdadfi/wav2vec2-large-xlsr-turkish> Web adresini ziyaret etmeleri sağlanır ve Web sayfasındaki açıklamalar birlikte irdelenir.

```

7 import torch
8 import torchaudio
9
10 from transformers import AutoProcessor, AutoModelForCTC
11 import matplotlib.pyplot as plt
12
13 processor = AutoProcessor.from_pretrained("m3hrdadfi/wav2vec2-large-xlsr-turkish")
14 model = AutoModelForCTC.from_pretrained("m3hrdadfi/wav2vec2-large-xlsr-turkish")

```

Şekil 35. Kütüphane ve model çağrılarına ilişkin kodlar



3. ADIM: ÜRET

Öğrenciler mikrofon üzerinden sesli komut dinlenmesi ve komutların model üzerinden tahmin edilmesi yönünde uygulama kodlarını yazar.

Mikrofon Üzerinden Sesli Komut Dinlenmesi

Öğrenciler **pvrecorder** kütüphanesi yardımıyla uygulamanın bilgisayar ortamında tanımlı mikrofon(lar) üzerinden sesli komut dinlemesini ve dinlenen komutun **wav** formatında kaydedilmesi yönündeki kodları yazar (Şekil 36). Şekil 36'da 16.-18. satırlar arasında pvrecorder ve destekleyici bileşenler olan “wave” ile “struct” bileşenleri uygulama ortamında dahil edilmektedir. 24. satırdaki **recorder.start()** kodu ile mikrofondan ses kaydı başlamakta, “while” döngüsü altındaki komutlar arasında yer alan **recorder.read()** komutu ile sesler aktif bir biçimde “audio” değişkeni kapsamında toplanmaktadır. Uygulama klavyeden **Ctrl + C** tuşları basılıncaya kadar kayda devam etmekte ve kayıt durdurulduğunda dinlenen ses verileri **“except KeyboardInterrupt”** komut bölümü altında tanımlanan kodlar yardımıyla uygulama dizinine **ses.wav** ismiyle kaydedilmektedir. Aktif ses dinleme ve kaydın durdurulması aşamaları **print** fonksiyonu üzerinden “Dinliyorum...” ve “Komut alındı.” ifadeleriyle ekran'a belirtilmektedir. Ses dinleme ve kayıt süreci genel olarak durduktan sonra 39. ve 40. satırlarda yer alan kodlar yardımıyla “ses.wav” dosyasının model için hazır hale getirilmesi sağlanmaktadır.

```

16 from pvrecorder import PvRecorder
17 import wave
18 import struct
19
20 recorder = PvRecorder(device_index=0, frame_length=512)
21 audio = []
22
23 try:
24     recorder.start()
25
26     while True:
27         print("Dinliyorum...")
28         frame = recorder.read()
29         audio.extend(frame)
30 except KeyboardInterrupt:
31     print("Komut alındı.")
32     recorder.stop()
33     with wave.open("ses.wav", 'w') as f:
34         f.setparams((1, 2, 16000, 512, "NONE", "NONE"))
35         f.writeframes(struct.pack("h" * len(audio), *audio))
36 finally:
37     recorder.delete()
38
39 waveform, sample_rate = torchaudio.load("ses.wav")
40 waveform_resampled = torchaudio.transforms.Resample(orig_freq=sample_rate, new_freq=16000)(waveform)

```

Şekil 36. Mikrofon üzerinden sesli komut verilerinin alınmasına dair kodlar

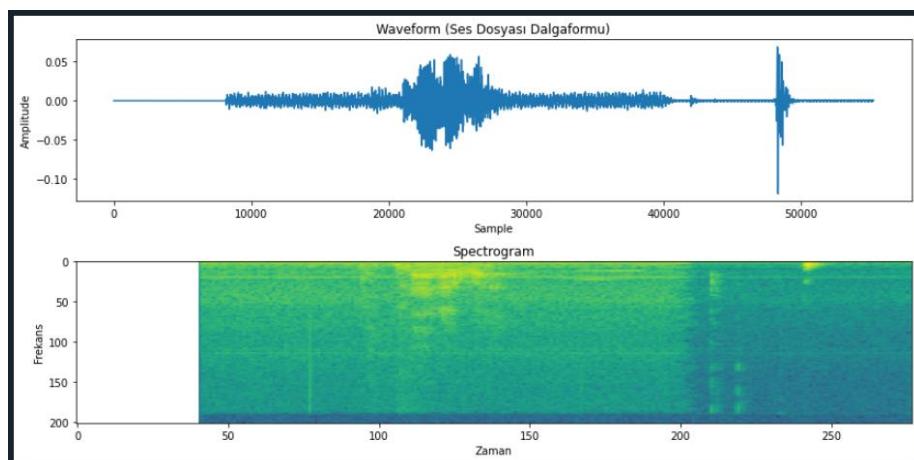
Öğrencilerin mikrofondan elde edilen ses verilerini Pyplot üzerinden görselleştirebilecekleri belirtilerek Şekil 37'de gösterilen kodları yazmaları sağlanır. İlgili kodlar kaydedilen sesin dalga formu ve spektrogram adı verilen iki farklı grafik ile görüntülenmesini sağlamakta; dalga formu için “waveform” değişkenine torchaudio ile yüklenen verilerin **plot** fonksiyonu ile görüntüleme gerçekleştirilirken, spektrogram için torchaudio ile Pyplot bir arada kullanılmaktadır. Şekil 38 kaydedilen “Merhaba” ifadesi için oluşan grafikleri göstermektedir.

```

42 plt.figure(figsize=(12, 6))
43 plt.subplot(2, 1, 1)
44 plt.plot(waveform.t().numpy())
45 plt.title('Waveform (Ses Dosyası Dalgaformu)')
46 plt.xlabel('Sample')
47 plt.ylabel('Amplitude')
48
49 plt.subplot(2, 1, 2)
50 spe = torchaudio.transforms.Spectrogram()(waveform_resampled)
51 spe_ch1 = spe[0, :, :]
52 plt.imshow(spe_ch1.log2().numpy(), aspect='auto', cmap='viridis')
53 plt.title('Spectrogram')
54 plt.xlabel('Zaman')
55 plt.ylabel('Frekans')
56
57 plt.tight_layout()
58 plt.show()

```

Şekil 37. Ses verisinin görselleştirilmesine yönelik kodlar



Şekil 38. “Merhaba” ifadesi için oluşan grafikler

Sesli Komutun Tahmini

Öğrencilerin dosya olarak kaydedilen ses verisinin ön eğitimli Hugging Face modelinden tahmin edilmesi için Şekil 39'da gösterilen kodları yazmaları istenir. Kodlar önceki kod satırlarında düzenlenen ses verisinin model tarafından değerlendirilmesini sağlayarak komutun tahmin edilmesini sağlamaktadır. Öğrencilere uygulamanın tahmin edilen komuttan yola çıkararak kullanıcıyı asiste edecek işlemler konusunda tercihe göre düzenlenmeye devam edilebileceği belirtilir.

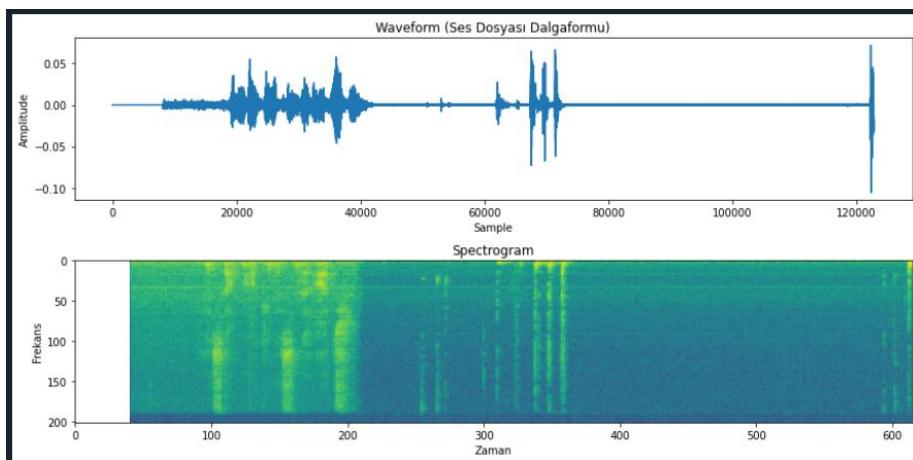
```

60     with torch.no_grad():
61         logits = model(waveform_resampled).logits
62
63     output_ids = torch.argmax(logits, dim=-1)
64     command = processor.batch_decode(output_ids)
65
66     print("Komutunuz:", command)

```

Şekil 39. Sesli komutun tahminine yönelik kodlar

Kodlanan uygulamanın nihai hali çalıştırıldığında sesli komutun dinlenmesi ve tahmin edilmesi yönünde Şekil 40'da gösterildiği gibi bir çıktı elde edilebilmektedir.



(a)

```

Dinliyorum...
Dinliyorum...
Dinliyorum...
Komut alındı.
Komutunuz: ['hesap makinessini aç']

```

(b)

Şekil 40. (a) “Hesap makinesini aç” komutu için ses verisi grafikleri
(b) İlgili komuta dair konsol çıktıları

UYGULAMANIN PYTHON KODLARI

```

import torch
import torchaudio

from transformers import AutoProcessor, AutoModelForCTC
import matplotlib.pyplot as plt

```

```

processor = AutoProcessor.from_pretrained("m3hrdadfi/wav2vec2-large-xlsr-turkish")
model = AutoModelForCTC.from_pretrained("m3hrdadfi/wav2vec2-large-xlsr-turkish")

from pvrecorder import PvRecorder
import wave
import struct

recorder = PvRecorder(device_index=0, frame_length=512)
audio = []

try:
    recorder.start()

    while True:
        print("Dinliyorum...")
        frame = recorder.read()
        audio.extend(frame)

    except KeyboardInterrupt:
        print("Komut alındı.")
        recorder.stop()

        with wave.open("ses.wav", 'w') as f:
            f.setparams((1, 2, 16000, 512, "NONE", "NONE"))
            f.writeframes(struct.pack("h" * len(audio), *audio))

    finally:
        recorder.delete()

waveform, sample_rate = torchaudio.load("ses.wav")
waveform_resampled = torchaudio.transforms.Resample(orig_freq=sample_rate,
new_freq=16000)(waveform)

plt.figure(figsize=(12, 6))
plt.subplot(2, 1, 1)
plt.plot(waveform.t().numpy())

```

```

plt.title('Waveform (Ses Dosyası Dalgaformu)')
plt.xlabel('Sample')
plt.ylabel('Amplitude')

plt.subplot(2, 1, 2)
spe = torchaudio.transforms.Spectrogram()(waveform_resampled)
spe_ch1 = spe[0, :, :]
plt.imshow(spe_ch1.log2().numpy(), aspect='auto', cmap='viridis')
plt.title('Spectrogram')
plt.xlabel('Zaman')
plt.ylabel('Frekans')

plt.tight_layout()
plt.show()

with torch.no_grad():
    logits = model(waveform_resampled).logits

    output_ids = torch.argmax(logits, dim=-1)
    command = processor.batch_decode(output_ids)

    print("Komutunuz:", command)

```



4. ADIM: İLERLET

İlerlet adımda öğrencilerin gruplar halinde çalışarak gerçekleştirdikleri uygulamalar üzerinde değişikliklerle denemeler yapmaları istenir. Etkinlik için 20 dakikalık bir süre ayrılr.

Öğrencilere kod ortamında kullandıkları Türkçe Hugging Face modeli dışında farklı modelleri de kullanabilecekleri açıklanır. Örneğin, aşağıdaki kod dizilimi İngilizce bir model için tercih edilebilmektedir:

```
from transformers import Wav2Vec2ForCTC, Wav2Vec2Processor
```

```
model = Wav2Vec2ForCTC.from_pretrained("facebook/wav2vec2-xls-r-300m")
processor = Wav2Vec2Processor.from_pretrained("facebook/wav2vec2-xls-r-300m")
```

Öğrencilerin alternatif modelleri “Hugging Face Wav2Vec2forCTC” ya da “Hugging Face AutoModelForCTC” gibi aramalarla tespit ederek deneyebilecekleri ifade edilir.



5. ADIM: DEĞERLENDİRME

Öğrencilerin yaptıkları uygulamalardan elde ettikleri tecrübeler ve derste öğrenilenler ışığında soru-cevap etkinliği gerçekleştirilerek sesli asistan geliştirmeye dair bilgiler pekiştirilir.

Öğrencilere şu sorular sorulabilir:

- Sesli asistan geliştirme sürecinde kullandığınız farklı kütüphaneler hakkında fikirleriniz nelerdir?
- Geliştirdığınız sesli asistan uygulaması ve kullandığınız Yapay Zeka modeli hakkındaki görüşleriniz nelerdir?
- Sesli asistan uygulamalarının hayatımızdaki rolü ve geleceği hakkında neler düşünüyorsunuz?
- Sesli asistan uygulamalarının günlük hayatı kişisel aktivitelerin organize edilmesi ve böyle bir durumda kişisel verilerin gizli kalmama durumu hakkında fikirleriniz nelerdir?

3. HAFTA – 4. GÜN – 1., 2., 3. DERSLER: FACE DETECTION VE SESLİ ASİSTAN ENTEGRASYONU

DERS PLANI

DERS ETİKETLERİ



KONU: Face Detection ve Sesli Asistan Entegrasyonu



SINIF: 12-14 Yaş



SÜRE: 180 dk.



HAZIR BULUNUŞLUK: İngilizce eğitimi, Yapay Zeka kavramı ve uygulamaları hakkında temel bilgi, Python programlama dili hakkında temel bilgi, Python Pytorch kütüphanesi hakkında temel bilgi



ANAHTAR KELİMELER: Yapay Zeka, Algoritma, Kodlama, Python, Pytorch, Face Detection, Sesli Asistan



BECERİLER:



Tahmin



Sorgulama



Günlük hayat bağlantısı kurma



KAZANIMLAR:

Bu derste öğrenciler;

- Face detection ve sesli asistan uygulamalarının bir arada kullanımı konusunda bilgi ve beceri sahibi olurlar.
- OpenCV, Pytorch ve diğer bağlı kütüphanelerin bir arada kullanımı neticesinde kamera ve ses verilerinin bir arada nasıl kullanılabileceği ilişkin farkındalık elde ederler.
- Bağımsız uygulama kodlarının yeni bir uygulama ortamında bir arada çalışması için yapılabilecek ileri düzey revizyonlar konusunda bilgi ve beceri sahibi olurlar.



DERS MATERİYALLERİ:

Malzeme	Miktar
Yazı Tahtası	Öğrenci sayısı ve sınıf büyüklüğüne göre en az bir adet
Projeksiyon	Öğrenci sayısı ve sınıf büyüklüğüne göre en az bir adet
Defter, Kalem	Öğrenci sayısı kadar
Bilgisayar	Öğrenci sayısının yarısı kadar
Tablet	Öğrenci sayısının yarısı kadar
Python Anaconda Framework	Her bilgisayarda kurulu olacak şekilde



ÖĞRETMEN HAZIRLIK ÇALIŞMALARI:

Okunabilecek Kaynaklar:

Gupta, D., Hossain, E., Hossain, M. S., Andersson, K., & Hossain, S. (2019). A digital personal assistant using bangla voice command recognition and face detection. In *2019 IEEE International Conference on Robotics, Automation, Artificial-intelligence and Internet-of-Things (RAAICON)* (pp. 116-121). IEEE.

Iannizzotto, G., Bello, L. L., Nucita, A., & Grasso, G. M. (2018). A vision and speech enabled, customizable, virtual assistant for smart environments. In *2018 11th International Conference on Human System Interaction (HSI)* (pp. 50-56). IEEE.

İzlenebilecek Kaynaklar:

Blitz, G. (2020). Building a Virtual Assistant (using Python) V3 (Face Recognition). Çevrimiçi: <https://www.youtube.com/watch?v=9Gh81eI1cGY> (Erişim 20 Haziran 2024).

PyCoding Tech. (2023). Major Project: Voice Assistant Project in Python Programming | Step by step Tutorial. Çevrimiçi: <https://www.youtube.com/watch?v=c1EeSl2i5mg> (Erişim 20 Haziran 2024).

ATÖLYE HAZIRLIK ÇALIŞMALARI:

Öğretmen;

- Dersten önce ders materyallerini hazırlamalıdır.

- Python ortamında Yapay Zeka tabanlı face detection ve sesli asistan geliştirme adımları konusunda bilgi ve beceri sahibi olmalıdır.
- Ders kapsamında kullanılan Python kütüphaneleri hakkında genel farkındalık sahibi olmalıdır.



KAYNAKÇA:

Aravindan, D., & Supriya, P. (2021). Face recognition authenticated voice assistant system for the disabled. In *2021 Second International Conference on Electronics and Sustainable Communication Systems (ICESC)* (pp. 1220-1226). IEEE.

Pandey, D., Ali, A., Dubey, S., Srivastava, M., Dwivedi, S., & Raza, M. S. (2022). Voice Assistant Using Python and AI. *International Research Journal of Engineering and Technology*, 9(05), 832-838.

GENEL BAKIŞ:

- 1) Harekete Geç:** Öğrencilere şu ana kadar edindikleri bilgi ve beceriler ile daha önce geliştirdikleri face detection ve sesli asistan uygulamalarını bir araya toplayan bir uygulama gerçekleştirip gerçekleştiremeyecekleri sorulur ve gelen dönütler tartışılır.
- 2) Keşfet:** Öğrencilerden face detection ve sesli asistan süreçlerine yönelik kütüphaneleri ve ön tanımlamalarını yapmaları istenir.
- 3) Üret:** Öğrenciler face detection durumuna göre sesli asistanı tetikleyen uygulama kapsamına dair kodları yazar.
- 4) İlerlet:** Öğrencilerin bireysel bir şekilde çalışarak gerçekleştirilen uygulama üzerinde düzenlemeler yapmaları ve elde ettikleri sonuçları not almaları istenir.
- 5) Değerlendir:** Öğrencilerin aldıkları notlar ve derste öğrenilenler ışığında soru-cevap etkinliği gerçekleştirilerek face detection ve sesli asistan entegrasyonuna dair bilgiler pekiştirilir.

UYGULAMA



1. ADIM: HAREKETE GEC

Öğrencilere şu ana kadar edindikleri bilgi ve beceriler ile daha önce geliştirdikleri face detection ve sesli asistan uygulamalarını bir araya toplayan bir uygulama gerçekleştirip gerçekleştiremeyecekleri sorulur ve gelen dönütler tartışılır.



2. ADIM: KEŞFET

Keşfet adımda öğrencilerden face detection ve sesli asistan süreçlerine yönelik kütüphaneleri ve ön tanımlamalarını yapmaları istenir.

Face Detection ve Sesli Asistan Uygulamaları

Face detection ve sesli asistan uygulamaları mobil teknolojilerin gerek yazılım gerekliliklerinin donanım kapasitelerinin ilerleyişi ile birlikte bir arada kullanılabilecek seviyelere ulaşmış durumdadır. Özellikle kullanıcı güvenlik ve onaylama ihtiyaçları ile birlikte erişilebilirlik çerçevesinde bireylerin ihtiyaçlarına cevap verebilecek yüz ve ses verilerinin entegre kullanıldığı uygulamalar araştırmala sıkılıkla konu olmaktadır (Aravindan, & Supriya, 2021; Pandey vd., 2022). Bu noktada yüz ve ses verilerinin besleyeceği bir uygulamayı geliştirmek farklı Python bileşenlerinin ve Yapay Zeka modellerinin işbirliği içerisinde kullanımını gerektirmektedir.

Kütüphane Çağrıları ve Ön Tanımlamalar

Öğrencilerden önceki derslerde gerçekleştirdikleri face detection (Bkz. 3. Hafta – 2. Gün – 2., 3. Dersler) ve sesli asistan (Bkz. 3. Hafta – 3. Gün – 3. Ders) uygulamalarında kullanılan çeşitli kütüphaneleri yeni Python uygulama ortamına import etmeleri istenir. Ayrıca yeni uygulamada sesli komutun saniye bazlı alınmasını sağlamak adına tarih ve saat bilgilerine ulaşmayı sağlayan **datetime** kütüphanesinin ve kod ortamına dahil edilmesi sağlanır (import datetime) (Şekil 41).

```
7 import cv2
8 import torch
9 import torchaudio
10 from transformers import AutoProcessor, AutoModelForCTC
11 from pvrecorder import PvRecorder
12 import wave
13 import struct
14 import datetime
15
16 face_model = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
17
18 processor = AutoProcessor.from_pretrained("m3hrdadfi/wav2vec2-Large-xlsr-turkish")
19 model = AutoModelForCTC.from_pretrained("m3hrdadfi/wav2vec2-Large-xlsr-turkish")
20
21 recorder = PvRecorder(device_index=0, frame_length=512)
22 audio = []
23 command_count = 0
```

Şekil 41. Kütüphane çağrıları ve ön tanımlamalara dair kodlar

Kütüphane çağrıları yanında, yine Şekil 41'de gösterildiği gibi face detection ön eğitimi model dosyasının ve Hugging Face ortamındaki Türkçe ses tanıma modelinin kod ortamına dahil edilmesi sağlanmaktadır (Bu noktada öğrenciler face detection için gereken model dosyasını önceki face detection uygulama klasöründen kopyalayabilirler. Hugging Face'den edinilen ses tanıma modeli ise daha önceki uygulamada bilgisayarlarla otomatik olarak indirildiği için bu model için tekrar işlem yapmaya ihtiyaç olmayacağından).

Model çağrılarını takiben, “pvrecorder” kütüphanesi destekli recorder değişkeni ve kaydedilen sesi tutacak audio değişkeni tanımlanır. Ayrıca, uygulama çalışırken kaydedilen her yeni sesi / komutu ayrı dosyada tutabilmek için komut girişinde +1 artış göstererek değeri dosya ismi sonuna eklenecek olan sayaç değişkeni de (command_count) 0 başlangıç değeri ile tanımlanır (Şekil 41).



3. ADIM: ÜRET

Öğrenciler face detection durumuna göre sesli asistanı tetikleyen uygulama kapsamına dair kodları yazar.

Gerçek Zamanlı Görüntü ve Sesli Komut Süreçlerinin Entegrasyonu

Öğrenciler ile birlikte kamera üzerinden aktif görüntü alımını sağlayacak OpenCV komutu yazılır ve ardından görüntü akışının sürekli olmasını sağlayacak OpenCV komutları için “sonsuz döngü” içerisinde yer alan bir while döngüsü oluşturulmaya başlanır. Bu döngü içerisinde OpenCV görüntü aktarımı sürerken her 1 milisaniyede kullanıcının olası tuş basma eylemlerini okumak için `k = cv2.waitKey(1)` kodu yazılır. Ardından daha önce face detection uygulamasına benzer yapıda kullanıcı yüzlerinin tanınmasına dair kodlar (gri tonlara dönüştürme ve tanınmış olası yüz verilerini “detected_faces” değişkenine aktarma adımları) yazılır (Şekil 42).

```
25 capture = cv2.VideoCapture(0)
26
27 while True:
28     k = cv2.waitKey(1)
29     hasFrame, img = capture.read()
30     if not hasFrame:
31         cv2.waitKey()
32         break
33     gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
34
35     detected_faces = face_model.detectMultiScale(gray, 1.1, 4)
```

Şekil 42. Görüntü okuma ve face detection adımlarına dair giriş kodları

Öğrencilere kod yazımına halen “tanımlanmış olan while döngüsü içerisinde devam ettikleri hatırlatılır”. Gerçekleştirilen uygulama kullanıcı yüzü tanımlandığında sesli komuta açık durumda olan, herhangi bir yüz tanıma olmadığından ise sesli komut kabul etmeyen bir uygulama yönünde kurgulanmıştır. Bu doğrultuda devam eden kodlarda toplam tanınan yüz sayısının 0'dan büyük olup olmadığı kontrol edilerek, kullanıcının “**a tuşuna basmak suretiyle komut gönderebileceğini**” belirten OpenCV text görüntüsü ve yine tanınan yüzlerin dörtgenler içerisine alınma işlemlerine dair kodlar yazılır (Şekil 43).

Şekil 43’de gösterilen ilgili kodların devamında öğrencilerin “a tuşuna basılması” halinde `sec_to_run` adlı bir değişkende belirtilen süre değeri (saniye) kadar dinleme sürecinin

başlayacağı kodları yazması sağlanır. Bu aşamada yazılan kodlar sesli asistan uygulamasındaki kodların revize edilerek face detection ile entegrasyonunu destekler şekilde yansıtılır. 43. kod satırında ses kaydı başlatılırken, 44. satırda her yeni ses dosyasını temsil edecek sayaç değişkeni +1 artırılmakta, 46. satırda ise tanımlanan sürenin takip edilmesini sağlayacak bitiş süresi anlık zaman ve istenilen saniye süresi üzerinde yapılan basit matematiksel işlemle tespit edilmektedir. 47.-52. satırlar arasında tanımlanan bir başka “sonsuz while döngüsü” istenilen süre kapsamında ses kaydını gerçekleştirmektedir. Döngü sonrasında tanımlanan kodlar, kaydedilen ses dosyasının işlenmesi ve Hugging Face modeli tarafından komutun tahmin edilerek konsol ortamına yansıtılması yönündedir. Face detection OpenCV ortamında görsel olarak kullanıcıya yansıtılırken, sesli komut sürecine yönelik bildirimler konsol ortamına yansımaktadır (Şekil 43).

```

37     if (len(detected_faces) > 0):
38         cv2.putText(img, "Asistan aktif. Komut göndermek için a'ya bas", (20, 50),
39                     cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 255, 0), 2)
40         for (x, y, w, h) in detected_faces:
41             cv2.rectangle(img, (x, y), (x+w, y+h), (150, 255, 0), 3)
42         if k & 0xFF == ord("a"):
43             recorder.start()
44             command_count = command_count + 1
45             sec_to_run = 3
46             exec_end_time = datetime.datetime.now() + datetime.timedelta(seconds=sec_to_run)
47             while True:
48                 if datetime.datetime.now() >= exec_end_time:
49                     break
50                 print("Dinliyorum...")
51                 frame = recorder.read()
52                 audio.extend(frame)
53                 print("Komut alındı.")
54                 recorder.stop()
55                 file_name = "ses"+str(command_count)+".wav"
56                 with wave.open(file_name, 'w') as f:
57                     f.setparams((1, 2, 16000, 512, "NONE", "NONE"))
58                     f.writeframes(struct.pack("h" * len(audio), *audio))
59                 audio = []
60                 waveform, sample_rate = torchaudio.load(file_name)
61                 waveform_resampled = torchaudio.transforms.Resample(orig_freq=sample_rate,
62                                         new_freq=16000)(waveform)
63
64                 with torch.no_grad():
65                     logits = model(waveform_resampled).logits
66                     output_ids = torch.argmax(logits, dim=-1)
67                     command = processor.batch_decode(output_ids)
68                     print("Komutunuz:", command)

```

Şekil 43. Face detection ve kullanıcı tuş basma eylemine göre ses verisi kaydı ve komut tahmini süreçlerine yönelik kodlar

Uygulamanın geriye kalan kod satırlarında, yüz tespitinin olmaması durumuna yönelik olarak, 37. kod satırındaki “if” açılımının tersi durumunu temsil eden **else** bloğu yazılır ve OpenCV ekranına yüz tanıma durumunun olmaması nedeniyle sesli asistanın aktif olmadığı bilgisi yansıtılır. Else bloğundan sonra uygulamanın 27. kod satırında açılmış olan while döngüsünün son komutunda “cv2.imshow” fonksiyonu ile OpenCV arayüzünün aktif yansıtmasına dair gerekli olan kod yazılmış olur (Şekil 44).

```

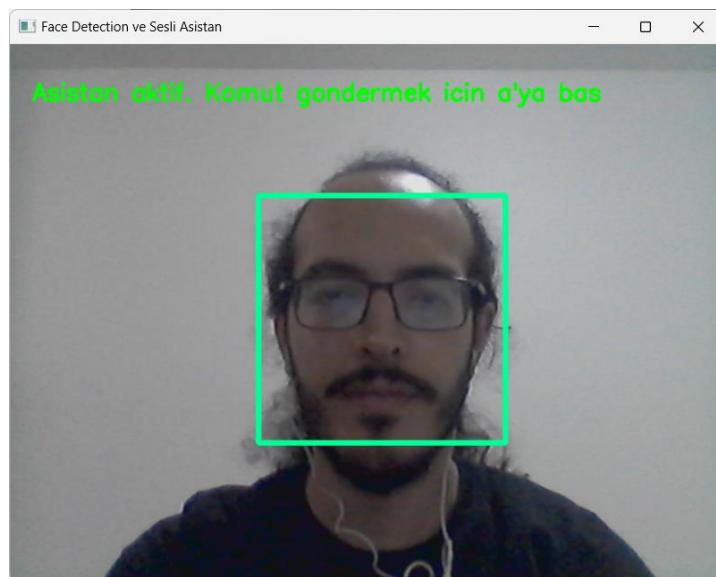
69     else:
70         cv2.putText(img, "Yuz tanimlanmadı. Sesli asistan pasif...", (20, 50),
71                     cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
72         cv2.imshow('Face Detection ve Sesli Asistan', img)

```

Şekil 44. Uygulamanın son bölümleri kapsamındaki OpenCV'ye yönelik kodları

Uygulamanın Çalışması

Öğrencilerle birlikte uygulama çalıştırılır ve açılan arayüzde kullanıcıda face detection olması ve olmaması durumunda gösterilen farklı bildirimler gözlemlenir (Şekil 45).



(a)

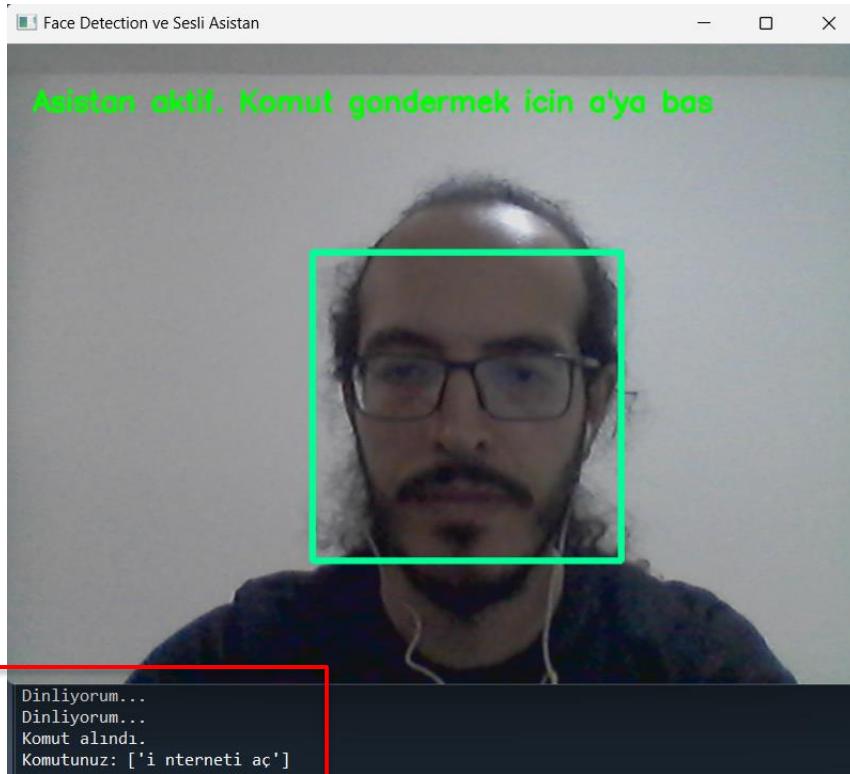


(b)

Şekil 45. (a) Face detection olmasi durumunda ekran görüntüsü

(b) Face detection olmaması durumunda ekran görüntüsü

Kullanıcı “a” tuşuna bastığında konsol bölümünde de görüleceği üzere, 3 saniyelik komut dinleme süreci başlamakta ve dinlenen komut Yapay Zeka modeli tarafından tespit edilerek yine konsolda bildirilmektedir (Şekil 46). Yapılan her yeni komut kaydı uygulama klasöründe yeni bir “wav” dosyası olarak kayıt altına alınmaktadır.



Şekil 46. Uygulamanın komut algılamasına dair konsol çıktıları

UYGULAMANIN PYTHON KODLARI

```
import cv2
import torch
import torchaudio
from transformers import AutoProcessor, AutoModelForCTC
from pvrecorder import PvRecorder
import wave
import struct
import datetime

face_model = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')

processor = AutoProcessor.from_pretrained("m3hrdadfi/wav2vec2-large-xlsr-turkish")
```

```

model = AutoModelForCTC.from_pretrained("m3hrdadfi/wav2vec2-large-xlsr-turkish")

recorder = PvRecorder(device_index=0, frame_length=512)
audio = []
command_count = 0

capture = cv2.VideoCapture(0)

while True:
    k = cv2.waitKey(1)
    hasFrame, img = capture.read()
    if not hasFrame:
        cv2.waitKey()
        break
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

    detected_faces = face_model.detectMultiScale(gray, 1.1, 4)

    if (len(detected_faces) > 0):
        cv2.putText(img, "Asistan aktif. Komut gondermek icin a'ya bas", (20, 50),
                   cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 255, 0), 2)
        for (x, y, w, h) in detected_faces:
            cv2.rectangle(img, (x, y), (x+w, y+h), (150, 255, 0), 3)
        if k & 0xFF == ord("a"):
            recorder.start()
            command_count = command_count + 1
            sec_to_run = 3
            exec_end_time = datetime.datetime.now() +
                datetime.timedelta(seconds=sec_to_run)
            while True:
                if datetime.datetime.now() >= exec_end_time:
                    break
                print("Dinliyorum...")

```

```

frame = recorder.read()
audio.extend(frame)
print("Komut alındı.")
recorder.stop()
file_name = "ses"+str(command_count)+".wav"
with wave.open(file_name, 'w') as f:
    f.setparams((1, 2, 16000, 512, "NONE", "NONE"))
    f.writeframes(struct.pack("h" * len(audio), *audio))
audio = []
waveform, sample_rate = torchaudio.load(file_name)
waveform_resampled = torchaudio.transforms.Resample(orig_freq=sample_rate,
                                                    new_freq=16000)(waveform)

with torch.no_grad():
    logits = model(waveform_resampled).logits
    output_ids = torch.argmax(logits, dim=-1)
    command = processor.batch_decode(output_ids)
    print("Komutunuz:", command)

else:
    cv2.putText(img, "Yuz tanimlanmadi. Sesli asistan pasif...", (20, 50),
               cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
    cv2.imshow('Face Detection ve Sesli Asistan', img)

```



4. ADIM: İLERLET

Öğrencilerin bireysel bir şekilde çalışarak gerçekleştirilen uygulama üzerinde düzenlemeler yapmaları ve elde ettikleri sonuçları not almaları istenir. Bu etkinlik için toplamda 30 dakikalık bir süre ayrılr.

İlerlet adımı kapsamında öğrencilerle birlikte farklı Hugging Face modelleri denenerek, Türkçe komut algılamaya yönelik farklı başarımlar gözlemlenebilecektir. Yine gerekirse öğrencilerin farklı modellerle İngilizce komutları da denemeleri sağlanabilir.



5. ADIM: DEĞERLENDİRME

Öğrencilerin aldıkları notlar ve derste öğrenilenler ışığında soru-cevap etkinliği gerçekleştirilerek face detection ve sesli asistan entegrasyonuna dair bilgiler pekiştirilir.

Öğrencilere şu sorular sorulabilir:

- Face detection ve sesli asistan entegrasyonu yönünde gerçekleştirdiğiniz kod revizyonları hakkında neler düşünüyorsunuz?
- Geliştirdiğiniz uygulamalarındaki görüşleriniz nelerdir?
- Face detection ve sesli asistanın bir arada kullanıldığı bu tür bir uygulama hangi tür problemlerin çözümünde başarılı bir şekilde kullanılabilir?

3. HAFTA – 5. GÜN – 1., 2. DERSLER: SESLİ ASİSTANA GOOGLE SUNUCUSU BAĞLAMA

DERS PLANI

DERS ETİKETLERİ



KONU: Sesli Asistana Google Sunucusu Bağlama



SINIF: 12-14 Yaş



SÜRE: 120 dk.



HAZIR BULUNUŞLUK: İngilizce eğitimi, Yapay Zeka kavramı ve uygulamaları hakkında temel bilgi, Python programlama dili hakkında temel bilgi



ANAHTAR KELİMELER: Yapay Zeka, Algoritma, Kodlama, Python, Sesli Asistan, Google Cloud, Google Speech-To-Text



BECERİLER:



Tahmin



Sorgulama



Günlük hayat bağlantısı kurma



KAZANIMLAR:

Bu derste öğrenciler;

- Google Cloud kapsamında Speech-To-Text çözümleri konusunda farkındalık ve bilgi sahibi olurlar.
- Sesli asistan uygulama süreçleri için Google Speech-To-Text sunucusunun kullanım konusunda bilgi ve beceri sahibi olurlar.
- Python ortamında daha önce kodlanan sesli asistan kütüphane yapıları ile Google Speech-To-Text kütüphanesi arasında etkileşimlerin nasıl kurulabileceği konusunda bilgi ve beceri sahibi olurlar.
- Python ortamında Google Speech-To-Text çözümü içeren bir uygulamanın nasıl geliştirileceği konusunda bilgi ve beceri sahibi olurlar.



DERS MATERİYALLERİ:

Malzeme	Miktar
Yazı Tahtası	Öğrenci sayısı ve sınıf büyüklüğüne göre en az bir adet
Projeksiyon	Öğrenci sayısı ve sınıf büyüklüğüne göre en az bir adet
Defter, Kalem	Öğrenci sayısı kadar
Bilgisayar	Öğrenci sayısının yarısı kadar
Tablet	Öğrenci sayısının yarısı kadar
Python Anaconda Framework	Her bilgisayarda kurulu olacak şekilde



ÖĞRETMEN HAZIRLIK ÇALIŞMALARI:

Okunabilecek Kaynaklar:

Harman, G., & Aydemir, E. (2022). Kolay Kişiselleştirilebilir Akıllı Sanal Asistan. *International Journal of Multidisciplinary Studies and Innovative Technologies*, 6(2), 143-151.

Odabas, M. S., Çakır, D., & Cengiz, T. E. P. E. (2022). Akıllı Ev Sistemlerinde Yapay Zekâ Kullanımı Üzerine Bir Değerlendirme. *OMÜ Mühendislik Bilimleri ve Teknolojisi Dergisi*, 2(2), 185-204.

Shakhovska, N., Basystiuk, O., & Shakhovska, K. (2019, May). Development of the Speech-to-Text Chatbot Interface Based on Google API. In *MoMLET* (pp. 212-221).

İzlenebilecek Kaynaklar:

Jenn, J. (2023). Google Cloud Speech-To-Text API With Python For Beginners. Çevrimiçi: https://www.youtube.com/watch?v=izdDHVLc_Z0 (Erişim 20 Haziran 2024).

ATÖLYE HAZIRLIK ÇALIŞMALARI:

Öğretmen;

- Dersten önce ders materyallerini hazırlamalıdır.
- Google Cloud çözümleri ve Google Speech-To-Text konusunda bilgi sahibi olmalıdır.

- Google Speech-To-Text API kullanımı için geçici olarak kullanılabilecek ve / veya öğrencilerle paylaşılabilecek Google Key temin etmiş olmalıdır.
- Python ortamında Google Speech-To-Text uygulamaları geliştirme konusunda bilgi ve beceri sahibi olmalıdır.
- Ders kapsamında kullanılan Python kütüphaneleri hakkında genel farkındalık sahibi olmalıdır.



KAYNAKÇA:

Choi, J., Gill, H., Ou, S., Song, Y., & Lee, J. (2018). Design of voice to text conversion and management program based on Google Cloud Speech API. In *2018 International Conference on Computational Science and Computational Intelligence (CSCI)* (pp. 1452-1453). IEEE.

Google. (2024). Speech-to-Text – Turn speech into text using Google AI. Çevrimiçi: <https://cloud.google.com/speech-to-text> (Erişim 20 Haziran 2024).

Isyanto, H., Arifin, A. S., & Suryanegara, M. (2020). Performance of smart personal assistant applications based on speech recognition technology using IoT-based voice commands. In *2020 International conference on information and communication technology convergence (ICTC)* (pp. 640-645). IEEE.

GENEL BAKIŞ:

1) Harekete Geç: Öğrencilere Google Speech-To-Text hakkında bilgi sahibi olup olmadıkları sorulur ve sesli asistanlar ile iletişime, çevresel kontrole ve Google Speech-To-Text çözümüne yönelik birtakım bilgiler verildikten sonra konu kapsamı karşılıklı olarak tartışılar.

2) Keşfet: Öğrencilerden Google Speech-To-Text tabanlı uygulamaya yönelik Google Cloud ortamında ilgili tanımlamaları yapmaları ve ardından Python ortamında ise uygulamaya dönük kütüphane çağrıları ve ön tanımlamaları kodlamaları istenir.

3) Üret: Öğrenciler sesli komut aktarımıları ve Google Speech-To-Text taraflı dönütlere yönelik kodlamaları gerçekleştirir.

4) İlerlet: Öğrenciler geliştirilen uygulama sonuçlarını gözlemler ve ardından bireysel olarak çalışarak uygulamanın etkileşim kapsamının artırılmasına dair geliştirmeler yapar.

5) Değerlendir: Öğrencilerin uygulamalarında yaptıkları geliştirmeler ve derste öğrenilenler ışığında soru-cevap etkinliği gerçekleştirilerek sesli asistan ile Google Speech-To-Text entegrasyonuna dair bilgiler pekiştirilir.

UYGULAMA



1. ADIM: HAREKETE GEÇ

Öğrencilere Google Speech-To-Text hakkında bilgi sahibi olup olmadıkları sorulur ve sesli asistanlar ile iletişime, çevresel kontrole ve Google Speech-To-Text çözümüne yönelik birtakım bilgiler verildikten sonra konu kapsamı karşılıklı olarak tartışılar.

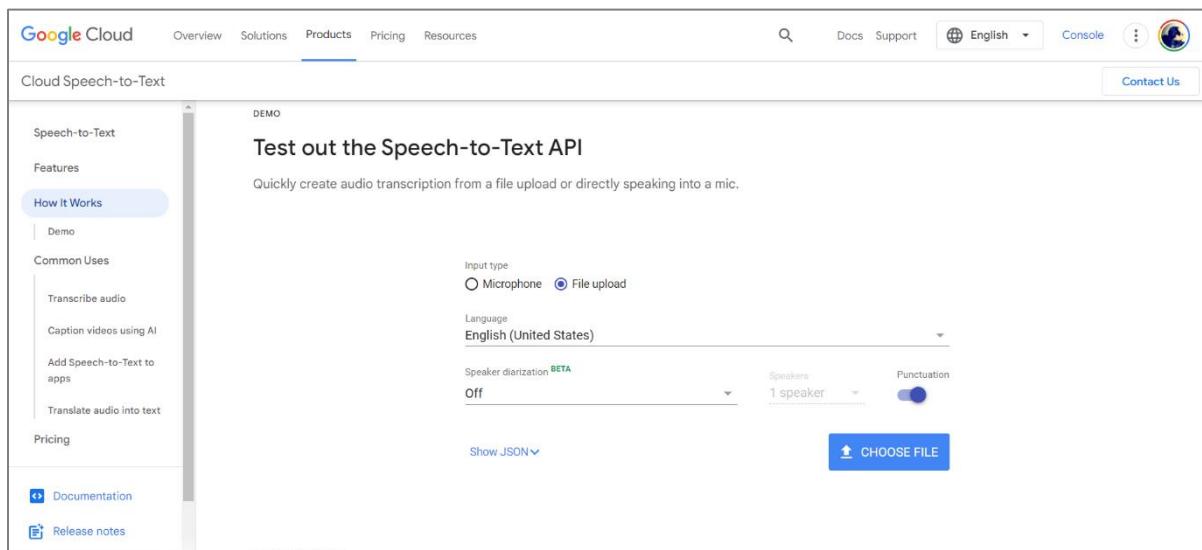
Sesli Asistanlar ile İletişim ve Çevresel Kontrol

Sesli asistan uygulamaları son yıllarda özellikle akıllı ortam kontrolleri açısından odak noktası haline gelmiştir. Google ve Apple gibi onde gelen teknoloji şirketlerinin konuşma tanıyalım ve hatta Nesnelerin İnterneti (IoT) teknolojisi bağlamında akıllı cihazlara hitap edebilme yönünde kullanıcılarla sundukları olanaklar, sesli asistanlar üzerinden kurgulanan yaklaşımı hızlandırmıştır. Bu noktada en basitinden Alexa ya da Siri gibi zeki asistanlar IoT cihazlarının ve Web ortamının kontrolünü sağlayan dinleme, dinlenen komutları analiz etme ve komutlara karşılık gelen eylemleri tetikleme yönünde önemli işlevlere sahip durumdadır (Isyanto vd., 2020; Natale, & Cooke, 2021). Bu noktada Google tarafından geliştirilen Speech-To-Text gibi çözümler konuşmadan metne dönüşüm konusunda etkin ve yüksek performanslı uygulamaların kullanımını mümkün hale getirmektedir (Choi vd., 2018).

Google Speech-To-Text

Google Speech-To-Text, Google şirketi tarafından konuşmadan metin analizi çözümleri sunan Yapay Zeka tabanlı bir hizmettir. Google'ın Chirp olarak adlandırdığı bir model üzerine kurulu olan sistem, bulut tabanlı servis hizmeti yaklaşımı ile beraber 125 farklı dili desteklemekte ve geliştiricilere ön eğitimli modeller yardımıyla pratik sesli asistan uygulamaları geliştirme yönünde fırsatlar sunmaktadır (Google, 2024).

Öğrencilerle birlikte <https://cloud.google.com/speech-to-text> Web adresi ziyaret edilerek Speech-To-Text hizmetine ilişkin sunulan açıklamalar incelenir ve demo uygulaması test edilir (Şekil 47).



Şekil 47. Google Cloud platformu ile birlikte Speech-To-Text Web platformu.

Speech-To-Text hizmeti, Google Cloud ortamına dahil olunarak, çeşitli ücretsiz hizmet olanakları ve daha ileri düzey çözümler için ücretli planlar kullanılmak suretiyle Python tabanlı uygulamalara da entegre edilebilmektedir. Bu amaçla Google Cloud ortamında Speech-To-Text hizmetine yönelik anahtarlar ve API fonksiyonları üzerinden Google sunucularıyla iletişim kurulabilmektedir.



2. ADIM: KEŞFET

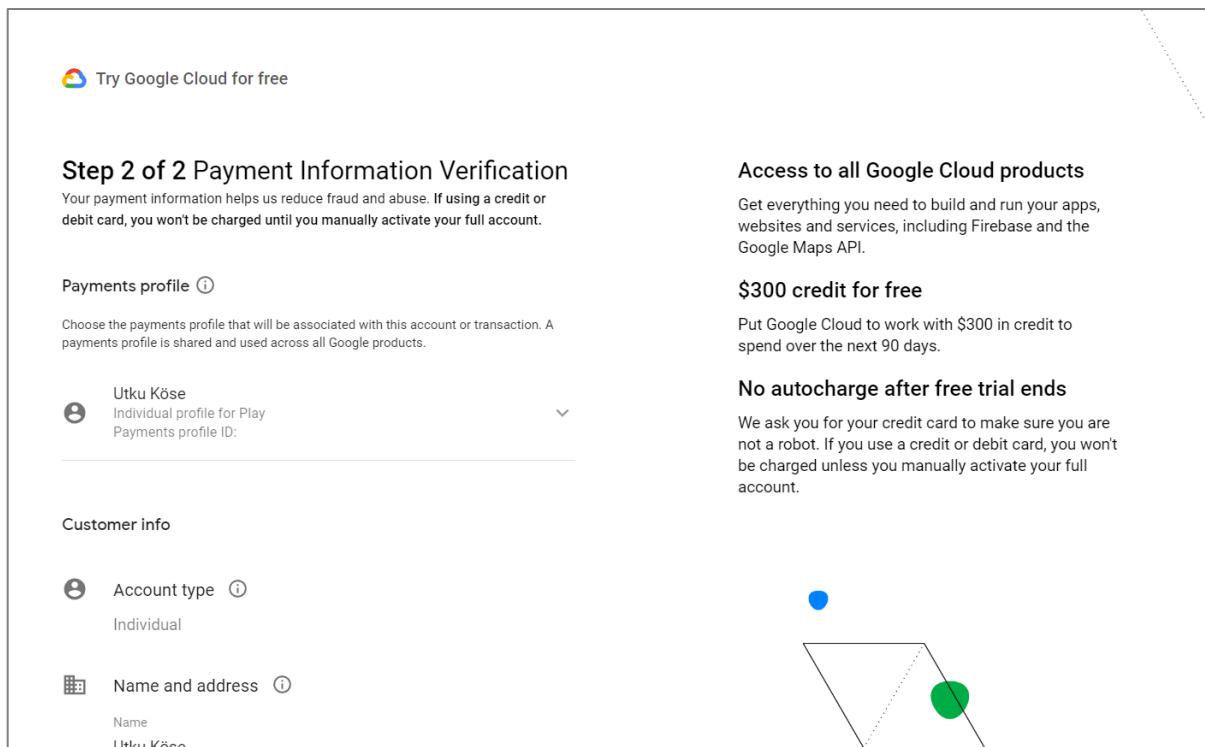
Öğrencilerden Google Speech-To-Text tabanlı uygulamaya yönelik Google Cloud ortamında ilgili tanımlamaları yapmaları ve ardından Python ortamında ise uygulamaya dönük kütüphane çağrıları ve ön tanımlamaları kodlamaları istenir.

Google Cloud ve Speech-To-Text Platformuna Üyelik

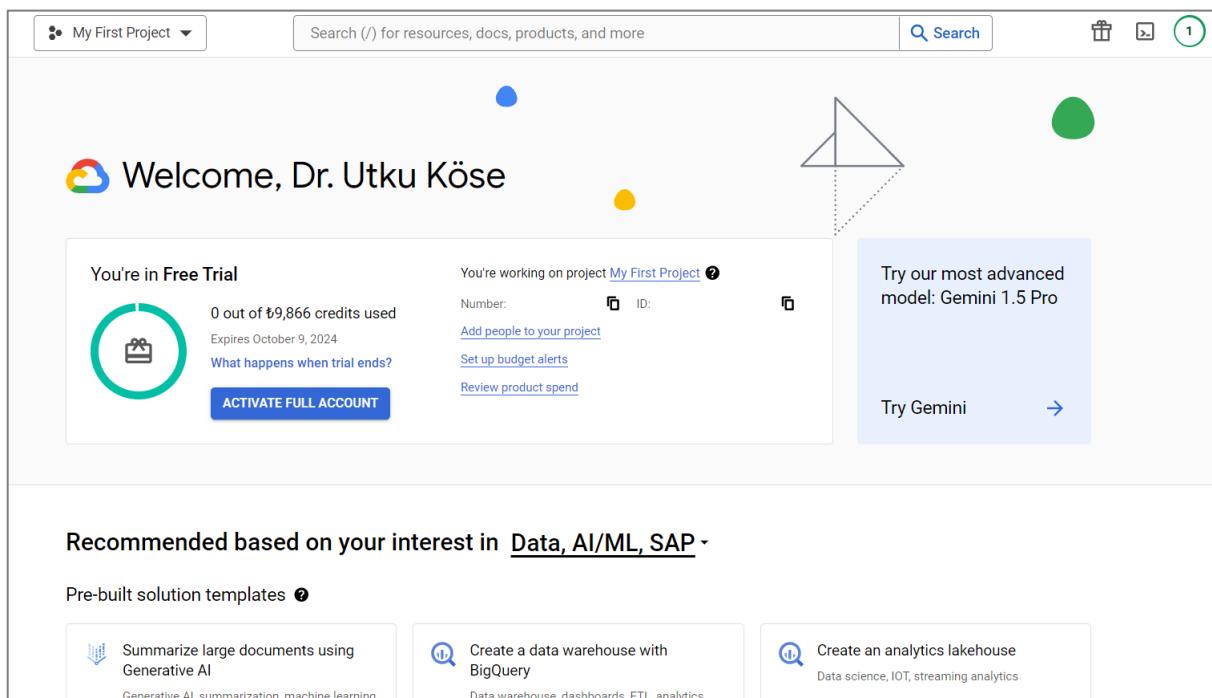
Öğrencilerin Python tabanlı uygulama geliştirmeden önce Google Cloud platformu üzerinden Speech-To-Text platformuna üye olmaları istenir. Speech-To-Text ana sayfasında **Start Free** düğmesine tıklamak suretiyle başlatılan üyelik (Şekil 48a), iki aşamalı arayüzler üzerinden kullanıcı üyelik bilgileri, ad soyad, adres ve ödemelerde kullanılabilecek kart bilgilerinin tanımlanması suretiyle tamamlanmaktadır (Şekil 48b). Bir sonraki adımda platform kullanım amaçlarına yönelik çeşitli soruların cevaplandırılmışından sonra Google Cloud platformunun giriş arayüzü görüntülenmektedir (Şekil 48c). Söz konusu arayüz Google bulut altyapısından destek almak suretiyle kullanılabilecek, özellikle Yapay Zeka desteğiyle ön plana çıkan hizmetlerin listelendiği, aynı zamanda kullanıcı projelerine de erişimin sağlandığı çeşitli seçenekleri görüntülemektedir.

The screenshot shows the Google Cloud Speech-to-Text product page. The top navigation bar includes links for Overview, Solutions, Products (which is highlighted in blue), Pricing, and Resources. On the right side of the header are links for Docs, Support, English (language selection), Console, Contact Us, and Start free. The main content area has a sidebar on the left with sections for Features, How It Works (with a Demo link), Common Uses (Transcribe audio, Caption videos using AI, Add Speech-to-Text to apps, Translate audio into text), Pricing, Documentation, and Release notes. The main content area features a heading 'Turn speech into text using Google AI' and a sub-section 'Speech-to-Text'. It describes how to convert audio into text transcriptions and integrate speech recognition into applications. It mentions free credits for new customers. Below this are two buttons: 'Start transcribing' and 'Contact sales'. A note at the bottom states: '*Applies to processing audio with the Speech-to-Text V1 API only.' To the right of the main content is a 'Product highlights' section with four bullet points: 'Easily add Speech-to-Text to apps', 'Transcribe audio files or real-time audio', 'Supports over 125 languages', and 'Use AI to caption videos'. Below this is a video thumbnail titled 'How to use Speech-to-Text' with a duration of 02:26 mins.

(a)



(b)

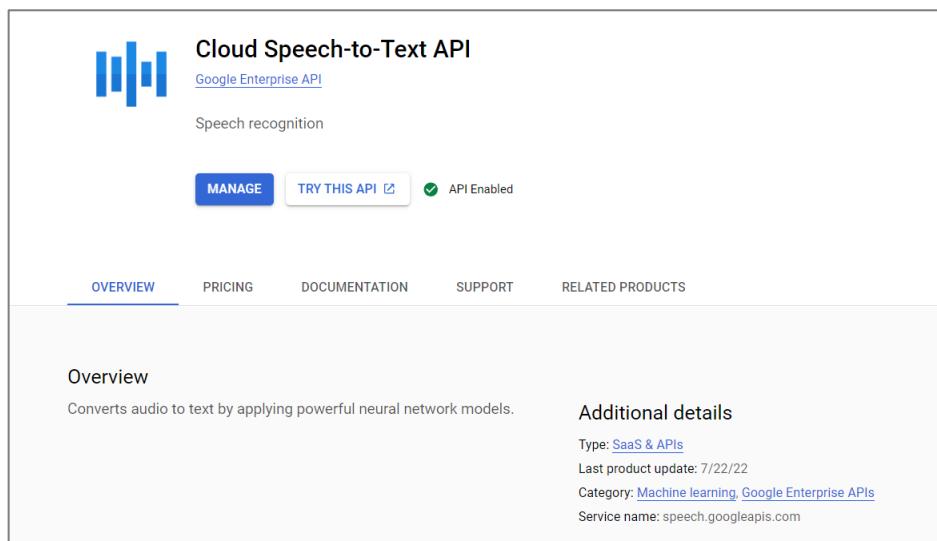
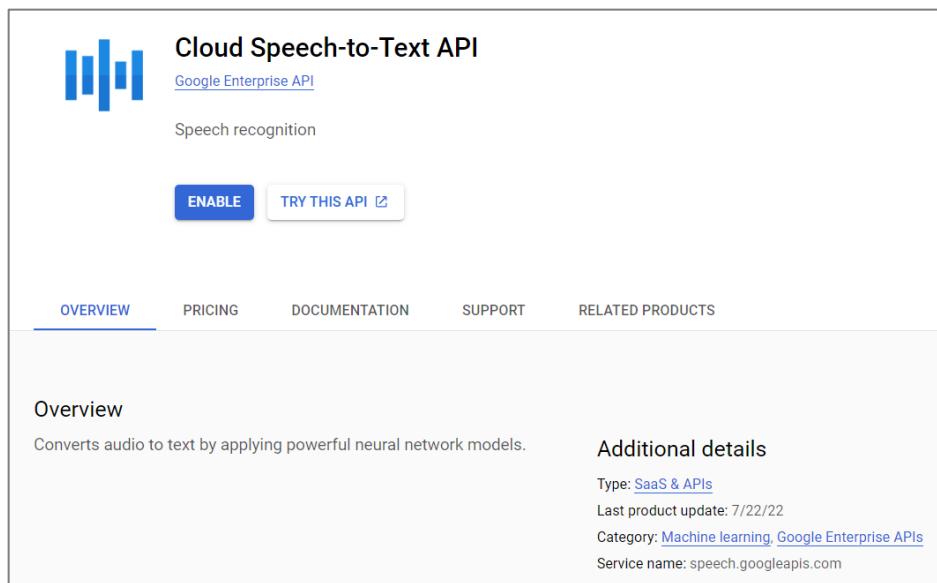


(c)

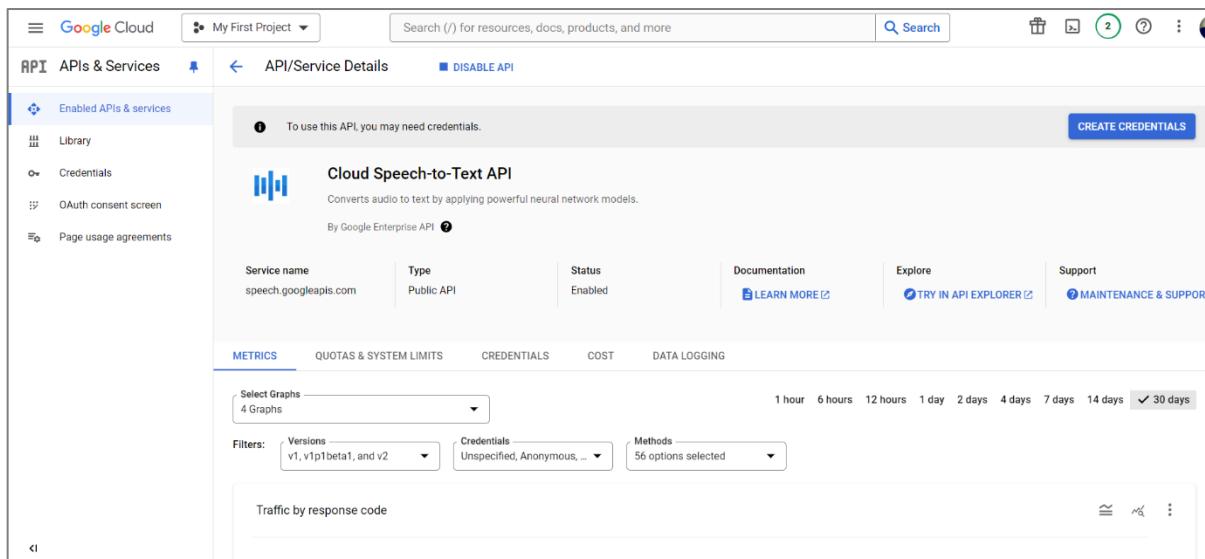
- Şekil 48. (a) Google Speech-To-Text ana sayfası
 (b) Google Cloud ve Speech-To-Text hizmetlerine yönelik üyelik aşamaları
 (c) Kullanıcıya yönelik Google Cloud giriş arayüzü

Python Uygulaması İçin Proje Anahtar Bilgilerinin Oluşturulması

Python uygulamalarından Google Speech-To-Text sunucuları ile iletişim kurabilmek adına öncelikli olarak proje tanımının yapılması ve uygulamaya has “anahtar” verilerinin alınması gerekmektedir. Bu amaçla öğrencilerle birlikte giriş arayüzünde “**Products**” bölümü altında **Convert speech to text** ifadesi taşıyan düğme tıklanmak suretiyle hizmet aktivasyonu sayfasına geçiş yapılır. İlgili sayfada **Enable** düğmesi tıklanarak hizmet aktif hale getirilir. Artık Enable düğmesi **Manage** düğmesine dönüşmüştür (Şekil 49a). Aktivasyon sonrası otomatik açılan ve yine Manage düğmesi üzerinden erişilen hizmet sayfasında hizmet seçenekleri ve kullanıma ilişkin çeşitli detay bilgiler ve istatistikler görüntülenebilmektedir (Şekil 49b).



(a)



(b)

Şekil 49. (a) Google Speech-To-Text hizmetinin aktif hale getirilmesi
 (b) Google Speech-To-Text hizmeti yönetim arayüzü

İlgili hizmetin Python uygulamasında kullanılması için **Create Credentials** düğmesine tıklanarak anahtar bilgilerinin oluşturulması sağlanır. Bu bilgiler ile Python ortamında Speech-To-Text API sistemi ile iletişim kurulabilecektir.

“Create Credentials” düğmesine tıklandıktan sonra geliştirilmesi düşünülen uygulama ile ilgili çeşitli bilgilerin girilmesi gerekmektedir. Bu aşamada geliştirilecek uygulama için erişim sağlanacak veri bilgileri sorusuna **application data** seçeneği işaretlenir ve böylelikle “service_account” kapsamında bilgilerin istediği sekmelere ulaşılır. Bu sekmelerde **service_account_name** bilgisi girildikten sonra **Done** düğmesine tıklayarak diğer opsiyonel seçeneklerin çevresinden geçilerek süreç tamamlanabilmektedir (Şekil 50a).

Service account oluşturulduktan sonra Credentials sekmesine geçilerek link olarak gösterilen service account ismi üzerine tıklanır. Yeni arayüzde **Key** sekmesine geçiş yapılır ve ardından **Add Key / Create new key** seçenekleri takip edilerek key veri dosyasının JSON formatında olacak şekilde indirilmesi sağlanır (Şekil 50b). Öğrenciler erişim kolaylığı adına JSON dosyasını Python uygulaması dizinine kaydeder.

[←](#) Create service account

1 Service account details

Service account name: python_sesli_asistan

Display name for this service account:

Service account ID *: python-sesli-asistan X C

Email address: python-sesli-asistan@rapid-pact-429002-r4.iam.gserviceaccount.com

Service account description:

Describe what this service account will do

[CREATE AND CONTINUE](#)

2 Grant this service account access to project (optional)

3 Grant users access to this service account (optional)

[DONE](#) [CANCEL](#)

(a)

[←](#) python_sesli_asistan

DETAILS PERMISSIONS **KEYS** METRICS LOGS

Keys

⚠ Service account keys can't be recovered if lost. Learn about the best way to automatically download your private key.

ℹ Google automatically downloads your private key for you. Learn more ↗

Add a new key pair or upload a public key.

Block service account key creation until further notice. Learn more about setting organization policy.

[ADD KEY ▾](#)

Type	Status	Key	Create time
No rows to display			

Create private key for "python_sesli_asistan"

Downloads a file that contains the private key. Store the file securely because this key can't be recovered if lost.

Key type

JSON
Recommended

P12
For backward compatibility with code using the P12 format

[CANCEL](#) [CREATE](#)

(b)

Şekil 50. (a) Google Speech-To-Text service account oluşturulması
(b) Key veri dosyasının JSON formatında indirilmesi

Python Uygulamasında Kütüphane Çağrıları ve Ön Tanımlamalar

Öğrenciler uygulamalarında kullanılacak key verilerini temin ettikten sonra Python kod ortamına geçerek kütüphane çağrıları ve ön tanımlamalarına yönelik kodları yazmaya başlar.

Bu noktaya kadar olan süreçlerde key verilerinin temin sorunları yaşanması durumunda öğrencilerin ortak kullanabilecekleri “hazır key verileri” ile hızlı çözüm üretilebilecektir (Söz konusu seçenek öğrencilerden gelecek dönütlere ve mevcut şartlara göre tercih edilebilir).

Öğrenciler öncelikli olarak Speech-To-Text Python kütüphanelerinin yüklenmesi için Spyder editörü konsol ortamında şu komutu çalıştırırlar:

pip install google-cloud-speech

Kütüphane yüklandıktan sonra öğrencilerin Şekil 51’de gösterilen kütüphane çağrı kodlarını yazmaları istenir. Uygulamanın sesli komut kaydı aşamasında, daha önce geliştirilen sesli asistan uygulamasındaki (Bkz. 3. Hafta – 3. Gün – 3. Ders) çeşitli kodların revize edilerek kullanılacağı açıklanır. Bu doğrultuda uygulamaya dahil olan **speech**, **os** ve **io** kütüphaneleri haricinde **wave**, **struct**, **datetime** ve **PvRecorder** yapıları da kod ortamına çağrılmaktadır.

Yapılan kütüphane çağrıları ardından öğrencilerin sırasıyla ses kaydedici değişkeni tanımlamaları, **os** kütüphanesinin **environ** fonksiyonu üzerinden daha önce indirilen JSON dosyasını key veri dosyası olarak göstergemeleri (18. kod satırı) ve son olarak Speech-To-Text hizmeti alacak istemci değişkenini tanımlamaları (19. kod satırı) sağlanır (Şekil 51). Öğrencilere kodlarda ‘key.json’ olarak gözüken kısımda kendi kaydettikleri JSON dosyası isminin kullanılması gerektiği belirtilir.

```
7  from google.cloud import speech
8  import os
9  import io
10
11 import wave
12 import struct
13 import datetime
14 from pvrecorder import PvRecorder
15
16 recorder = PvRecorder(device_index=0, frame_length=512)
17
18 os.environ['GOOGLE_APPLICATION_CREDENTIALS'] = 'key.json'
19 client = speech.SpeechClient()
```

Şekil 51. Kütüphane tanımları ve key verileri ile hizmet istemci değişkeninin tanımlanması



3. ADIM: ÜRET

Öğrenciler sesli komut aktarımıları ve Google Speech-To-Text taraflı dönütlere yönelik kodlamaları gerçekleştirir.

Uygulama Sesli Komut Kaydı ve Tahmin Aşamalarının Kodlanması

Uygulamanın ilerleyen aşamalarının organizasyonu genel olarak şu şekilde planlanmış durumdadır: Uygulama “sonsuz while döngüsü” üzerinden kullanıcıya sürekli tuş basması yönünde dönütlerde bulunmakta, kullanıcının “a” tuşuna basması durumunda sesli kayıt ve Speech-To-Text tahmin aşamalarına geçmektedir. Farklı bir tuşa basılması durumunda ise uygulama kapanmaktadır. Diğer yandan sesli kayıt ve tahmin aşaması ayrı bir kullanıcı tanımlı fonksiyon altında gerçekleştirilmektedir. Öğrenciler öncelikli olarak Şekil 52’de gösterildiği gibi **command_record_and_predict** olarak adlandırılan kullanıcı tanımlı fonksiyonu kodlar. Fonksiyonda genel akış 3 saniyelik bir zaman diliminde sesli komutun dinlenmesi ve **ses.wav** dosyası olarak kaydedilmesi ile başlamakta, ardından **Google-Text-To-Speech** sunucunun ses verisi üzerinden tahminde bulunarak sonucun ekrana yazdırılması ile sona ermektedir. 22.-37. kod satırları arası geleneksel sesli komut dinleme ve dosya yazdırma sürecine odaklanırken, 39. kod satırında başlayan “with” bloğu kaydedilen dosyayı okumaktadır. Yapay Zeka tabanlı tahmin süreci ise 43.-47. satırlar arasındaki ses verisi ve dil parametrelerinin belirlenmesinden sonra, 49. satırda kodlanan **recognize** fonksiyonu ile gerçekleştirilmektedir.

```
21 def command_record_and_predict():
22     audio = []
23     recorder.start()
24     sec_to_run = 3
25     exec_end_time = datetime.datetime.now() + datetime.timedelta(seconds=sec_to_run)
26     while True:
27         if datetime.datetime.now() >= exec_end_time:
28             break
29         print("Dinliyorum...")
30         frame = recorder.read()
31         audio.extend(frame)
32     print("Komut alındı.")
33     recorder.stop()
34     with wave.open("ses.wav", 'w') as f:
35         f.setparams((1, 2, 16000, 512, "NONE", "NONE"))
36         f.writeframes(struct.pack("h" * len(audio), *audio))
37     audio = []
38
39     with io.open("ses.wav", "rb") as audio_file:
40         content = audio_file.read()
41         audio = speech.RecognitionAudio(content=content)
42
43     config = speech.RecognitionConfig(
44         encoding=speech.RecognitionConfig.AudioEncoding.LINEAR16,
45         enable_automatic_punctuation=True,
46         audio_channel_count=1,
47         language_code="tr-TR")
48
49     response = client.recognize(request={"config": config, "audio": audio})
50
51     for result in response.results:
52         print("Komutunuz: {}".format(result.alternatives[0].transcript))
```

Şekil 51. Uygulama kapsamında sesli komut verisinin alınması ve Google-Text-To-Speech ile tahmin sürecinin gerçekleştirilmesine yönelik kodlar

Öğrencilerin son olarak Şekil 52'de gösterilen kodlarla uygulamanın ana kod bloğunu tamamlamaları istenir. Uygulama çalıştırıldığında kullanıcıdan dinlenen sesli komutlar Google Speech-To-Text sunucusunda tahmin edilmekte ve sonuçlar ekrana yansıtılmaktadır (Şekil 53).

```
54 while True:  
55     print("Sesli asistan arayüzü aktif...")  
56     opt = str(input("Komut gönderimi için a'ya çıkış için herhangi bir tuşa basınız:"))  
57     if (opt == "a"):  
58         command_record_and_predict()  
59     else:  
60         print("Sesli asistan iyi günler diler...")  
61         break
```

Şekil 53. Uygulama ana kodbloğu

```
Dinliyorum...  
Dinliyorum...  
Komut alındı.  
Komutunuz: Merhaba  
Sesli asistan arayüzü aktif...  
Komut gönderimi için a'ya çıkış için herhangi bir tuşa basınız:q  
Sesli asistan iyi günler diler...
```

Şekil 54. Uygulamanın çalıştırılması neticesinde elde edilen örnek bir çıktı

UYGULAMANIN PYTHON KODLARI

```
from google.cloud import speech  
import os  
import io  
  
import wave  
import struct  
import datetime  
from pvrecorder import PvRecorder  
  
recorder = PvRecorder(device_index=0, frame_length=512)  
  
os.environ['GOOGLE_APPLICATION_CREDENTIALS']= 'key.json'  
client = speech.SpeechClient()  
  
def command_record_and_predict():  
    audio = []
```

```

recorder.start()
sec_to_run = 3
exec_end_time = datetime.datetime.now() + datetime.timedelta(seconds=sec_to_run)
while True:
    if datetime.datetime.now() >= exec_end_time:
        break
    print("Dinliyorum...")
    frame = recorder.read()
    audio.extend(frame)
    print("Komut alındı.")
    recorder.stop()
    with wave.open("ses.wav", 'w') as f:
        f.setparams((1, 2, 16000, 512, "NONE", "NONE"))
        f.writeframes(struct.pack("h" * len(audio), *audio))
    audio = []

    with io.open("ses.wav", "rb") as audio_file:
        content = audio_file.read()
        audio = speech.RecognitionAudio(content=content)

    config = speech.RecognitionConfig(
        encoding=speech.RecognitionConfig.AudioEncoding.LINEAR16,
        enable_automatic_punctuation=True,
        audio_channel_count=1,
        language_code="tr-TR")

    response = client.recognize(request={"config": config, "audio": audio})

    for result in response.results:
        print("Komutunuz: {}".format(result.alternatives[0].transcript))

    while True:
        print("Sesli asistan arayüzü aktif...")

```

```
opt = str(input("Komut gönderimi için a'ya çıkış için herhangi bir tuşa basınız:"))

if (opt == "a"):

    command_record_and_predict()

else:

    print("Sesli asistan iyi günler diler...")

    break
```



4. ADIM: İLERLET

Öğrenciler geliştirilen uygulama sonuçlarını gözlemler ve ardından bireysel olarak çalışarak uygulamanın etkileşim kapsamının artırılmasına dair geliştirmeler yapar.



5. ADIM: DEĞERLENDİRME

Öğrencilerin uygulamalarında yaptıkları geliştirmeler ve derste öğrenilenler ışığında soru-cevap etkinliği gerçekleştirilerek sesli asistan ile Google Speech-To-Text entegrasyonuna dair bilgiler pekiştirilir.

Öğrencilere şu sorular sorulabilir:

- Google Speech-To-Text hakkında neler düşünüyorsunuz?
- Sesli asistan ve Google Speech-To-Text entegrasyonu yönünde gerçekleştirdiğiniz uygulama hakkında neler düşünüyorsunuz?
- Sesli asistan ve Google Speech-To-Text etkileşimine dayalı bu tür bir uygulama günlük hayatı hangi tür problemlerin çözümünde etkin bir şekilde kullanılabilir?
- Gerçekleştirdiğiniz uygulamayı alternatif olarak hangi tür problemler / uygulamalar kapsamında genişletmek isterdiniz?

3. HAFTA – 5. GÜN – 3. DERS: SESLİ ASİSTANA TELEGRAM BOTU BAĞLAMA

DERS PLANI

DERS ETİKETLERİ



KONU: Sesli Asistana Telegram Botu Bağlama



SINIF: 12-14 Yaş



SÜRE: 60 dk.



HAZIR BULUNUŞLUK: İngilizce eğitimi, Yapay Zeka kavramı ve uygulamaları hakkında temel bilgi, Makine Öğrenmesi hakkında temel bilgi, Telegram hakkında temel bilgi, Google Speech-To-Text hakkında temel bilgi



ANAHTAR KELİMELER: Yapay Zeka, Algoritma, Kodlama, Python, Sesli Asistan, Google Speech-To-Text, Telegram



BECERİLER:



Tahmin



Sorgulama



Grup çalışması



Günlük hayat bağlantısı kurma



KAZANIMLAR:

Bu derste öğrenciler;

- Telegram botu oluşturma konusunda bilgi ve beceri sahibi olurlar.
- Python ile Telegram bot mekanizmalarının geliştirilmesi konusunda bilgi ve beceri sahibi olurlar.
- Sesli asistan ile Telegram botu kontrol etme hakkında farkındalık ve bilgi sahibi olurlar.
- Python üzerinden sesli asistan süreci yardımcıyla Telegram bot kontrolü gerçekleştirilmesi kapsamında uygulama gerçekleştirirler.



DERS MATERİYALLERİ:

Malzeme	Miktar
Yazı Tahtası	Öğrenci sayısı ve sınıf büyüklüğüne göre en az bir adet
Projeksiyon	Öğrenci sayısı ve sınıf büyüklüğüne göre en az bir adet
Defter, Kalem	Öğrenci sayısı kadar
Bilgisayar	Öğrenci sayısının yarısı kadar
Tablet	Öğrenci sayısının yarısı kadar
Python Anaconda Framework	Her bilgisayarda kurulu olacak şekilde



ÖĞRETMEN HAZIRLIK ÇALIŞMALARI:

Okunabilecek Kaynaklar:

Aisyah, R. N., Istiqomah, D. M., & Muchlisin, M. (2021, March). Developing E-learning Module by Using Telegram Bot on ICT for ELT Course. In *5th International Conference on Arts Language and Culture (ICALC 2020)* (pp. 106-111). Atlantis Press.

İşeri, İ., Aydın, Ö., & Tutuk, K. (2021). Müşteri Hizmetleri Yönetiminde Yapay Zeka Temelli Chatbot Geliştirilmesi. *Avrupa Bilim ve Teknoloji Dergisi*, (29), 358-365.

Kadali, B., Prasad, N., Kudav, P., & Deshpande, M. (2020). Home automation using chatbot and voice assistant. In *ITM Web of Conferences* (Vol. 32, p. 01002). EDP Sciences.

İzlenebilecek Kaynaklar:

Bassel Tech. (2021). Telegram Bot - P11 - Voice Handling (Speech-To-Text). Çevrimiçi: <https://www.youtube.com/watch?v=1xQ7Xj6cEOw> (Erişim 22 Haziran 2024).

Idently. (2023). How To Create A Telegram Bot In Python For Beginners (2023 Tutorial). Çevrimiçi: <https://www.youtube.com/watch?v=vZtm1wuA2yc> (Erişim 22 Haziran 2024).

ATÖLYE HAZIRLIK ÇALIŞMALARI:

Öğretmen;

- Dersten önce ders materyallerini hazırlamalıdır.
- Telegram bot oluşturma ve Python ortamında kodlama konusunda bilgi ve beceri sahibi olmalıdır.

- Google Text-To-Speech hizmeti konusunda bilgi ve beceri sahibi olmalıdır.
- Ders kapsamında kullanılan Python kütüphaneleri hakkında genel farkındalık sahibi olmalıdır.



KAYNAKÇA:

Adamopoulou, E., & Moussiades, L. (2020). An overview of chatbot technology. In *IFIP International Conference on Artificial Intelligence Applications and Innovations* (pp. 373-383). Springer, Cham.

İlkbahar, F., Ünal, Ş., Karakaya, A. T., & Eren, B. (2021). Akıllı ev sistemleri üzerine bir model önerisi. *AJIT-e: Academic Journal of Information Technology*, 12(45), 90-105.

Lokman, A. S., & Ameedeen, M. A. (2019). Modern chatbot systems: A technical review. In *Proceedings of the Future Technologies Conference (FTC) 2018: Volume 2* (pp. 1012-1023). Springer International Publishing.

GENEL BAKIŞ:

- 1) Harekete Geç:** Öğrencilere Telegram ya da benzeri platformlar üzerinden bot hesaplarla etkileşim kurup kurmadıkları sorulur ve gelen dönütlere göre konu kapsamı tartışılar.
- 2) Keşfet:** Öğrencilere Telegram ortamında bot oluşturma adımlarına yönelik bilgiler verilir.
- 3) Üret:** Öğrencilerle birlikte Python ortamında Google Text-To-Speech tabanlı sesli asistan Telegram bot uygulamasına yönelik kütüphane çağrıları ve ön tanımlamalar gerçekleştirilir.
- 4) İlerlet:** Öğrencilerin Telegram bot fonksiyonları ve Google Text-To-Speech entegrasyonuna yönelik fonksiyonları kodlayarak uygulamayı çalışırmaları, sonuçları gözlemlemeleri ve gruplar halinde çalışarak genel denemeler / düzenlemeler yapmaları sağlanır.
- 5) Değerlendir:** Derste edinilen bilgilerden hareketle soru-cevap aşaması gerçekleştirilerek sesli asistan ve Telegram bot entegrasyonuna yönelik edinimler pekiştirilir.

UYGULAMA



1. ADIM: HAREKETE GEÇ

Harekete geç adımında öğrencilere Telegram ya da benzeri platformlar üzerinden bot hesaplarla etkileşim kurup kurmadıkları sorulur ve gelen dönütlere göre konu kapsamı tartışılar.

Anlık Mesajlaşma Sistemleri, Telegram ve Chat Bot Araçları

Web teknolojilerinin gelişimiyle beraber kullanıcılar arası iletişim ve etkileşimi artıran mesajlaşma sistemlerinde de hızlı ilerlemeler gerçekleşmiştir. 20. Yüzyıl'ın sonları ve 21. Yüzyıl başları chat odalarından anlık mesajlaşma sistemlerinin ilk örneklerinin yer aldığı gelişmelere sahne olmuşken, özellikle 2010'lu yıllarda sonra mobil cihazlarda etkinliğini gösteren daha özel uygulamalar geniş kitlelerce kullanılır hale gelmiştir. WhatsApp ve

Telegram yazılımları söz konusu uygulamalar arasında oldukça popüler olmakla beraber zamanla yeni işlevler de kazanmakta, kullanıcılar arası bireysel ve kitlesel etkileşim olanaklarını etkili bir biçimde genişletmektedir. Öyle ki, mobil cihazlarla gerçekleştirilen akıllı ortam, akıllı bina ya da akıllı ev kontrolleri, söz konusu özel mesajlaşma yazılımlarıyla da ilişkilendirilebilmektedir (İlkbahar vd., 2021).

Özel ve anlık mesajlaşma imkanları ve çok sayıda kullanıcının dahil olduğu grup / sayfa yapılanmaları, kullanıcıların bot (yazılımsal robot) tabanlı araçların geliştirilmesi yönünde atılımların da gerçekleşmesine sebep olmuştur. Anlık mesajlaşma yazılımlarının sağladığı geliştirme yolları, çok sayıda kullanıcının etkileşim içerisinde girdiği ortamlar için istege özel bot hesaplarının tasarlanması mümkün kılmakta, bu amaçla Python başta olmak üzere çeşitli programlama dilleri de kullanılabilir ve etkileşim içeriğine girdiği ortamlar için istege özel bot hesaplarının tasarlanması mümkün kılmakta, bu amaçla Python başta olmak üzere çeşitli programlama dilleri de kullanılabilir. Bu sayede kullanıcı ile otomatik etkileşim içeriğine giren, kullanıcı eylemlerine göre otomatik görevler yerine getiren ve **chat bot** olarak isimlendirilen araçlar kullanıcı etkileşimine önem veren birçok platformun önemli bileşenleri haline gelmiş durumdadır. Chat botlar günümüzde kullanıcının isteklerine uygun hizmet süreçlerinin yönetilmesi, belli bilgi ya da uzmanlık alanı dahilinde otomatik bilgi ve kaynak sunan araçların oluşturulması ve Yapay Zeka desteğiyle birlikte çok daha esnek kabiliyetlere sahip etkileşim unsurlarının elde edilmesinde ön plana çıkmaktadır (Lokman, & Ameedeen, 2019). Bu kapsamdaki gelişmeler, chat bot çözümlerinin de kendi içerisinde çeşitli sınıflara ayrılmasına ve Doğal Dil İşleme gibi etkileşim yönelik yöntemlerle her geçen zaman çok daha etkin seviyelere ulaşmasına olanak tanımıştır (Adamopoulou, & Moussiades, 2020).

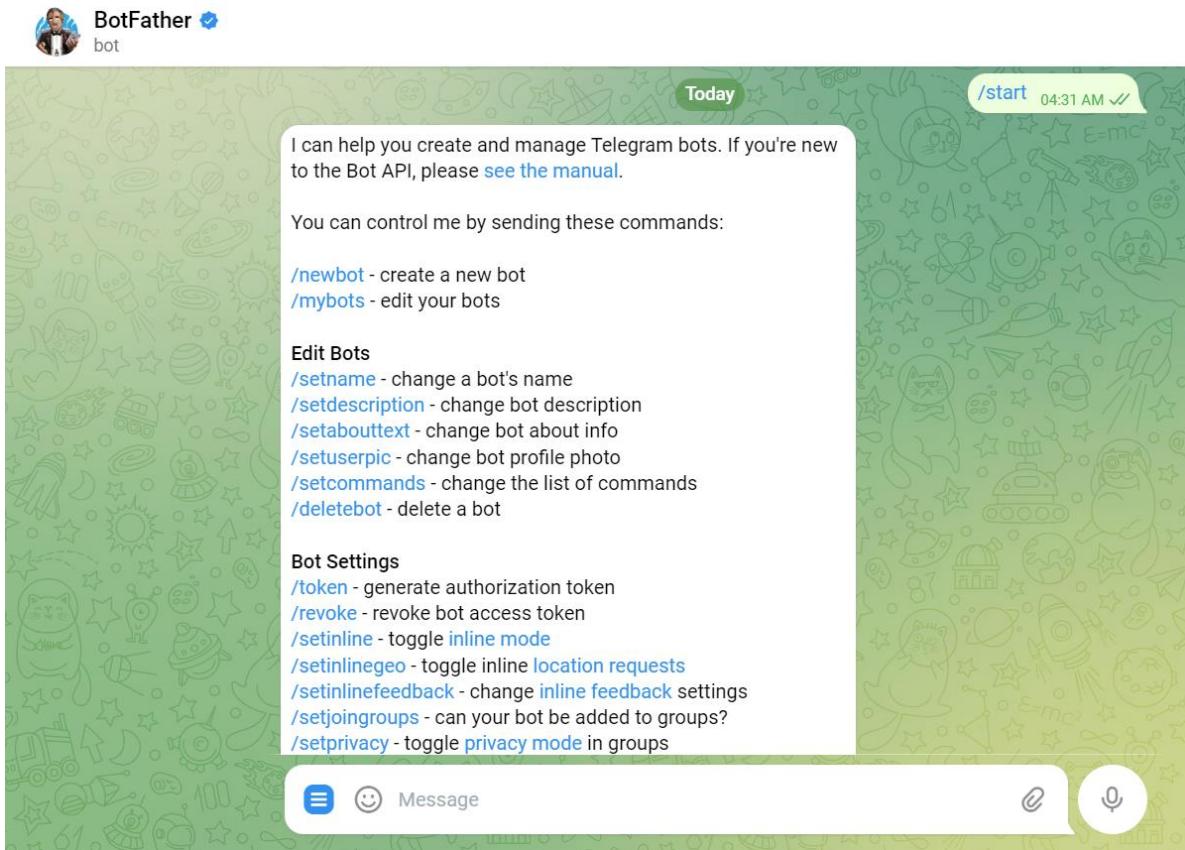


2. ADIM: KEŞFET

Öğrencilere Telegram ortamında bot oluşturma adımlarına yönelik bilgiler verilir.

Telegram Sisteminde Bot Oluşturma

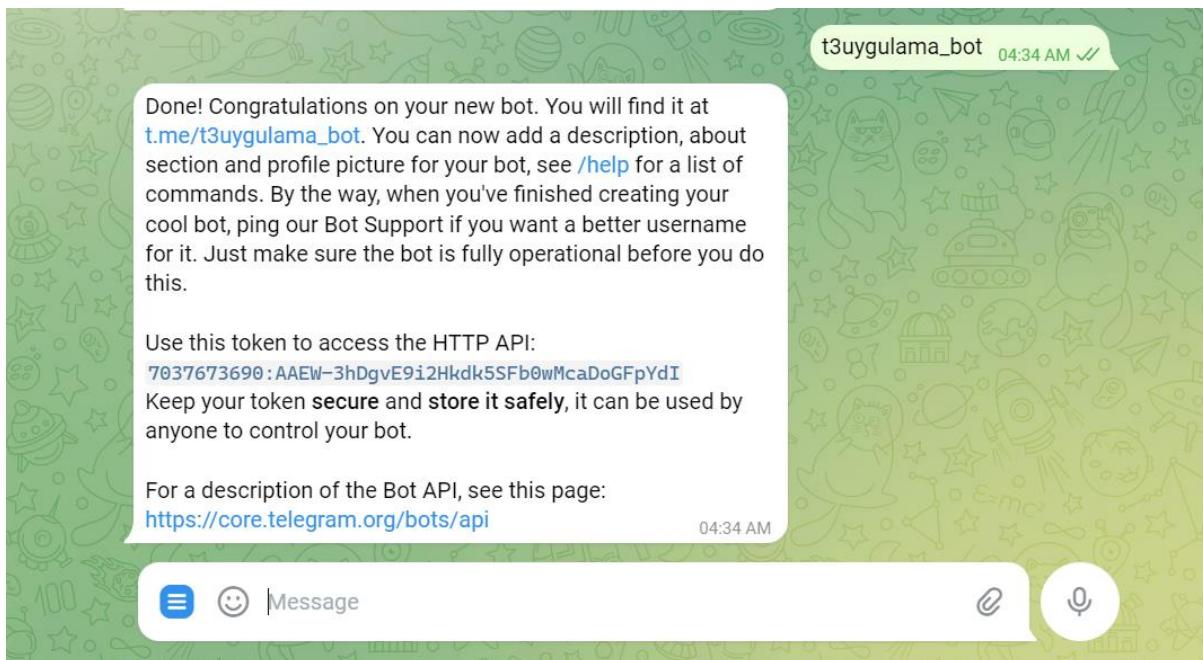
Öğrencilere sesli asistan çözümlerinin geliştirilmesi noktasında Telegram bot çözümleri ile Google Text-To-Speech hizmeti arasında bağlantı oluşturan bir uygulama geliştirileceği açıklanarak, öncelikli olarak Telegram sistemi üzerinden özel bir bot aracı oluşturulması gerektiği belirtilir. Bu amaçla öncelikli olarak Telegram hesabı olmayan öğrencilerin Telegram uygulamasını yükleyerek sisteme dahil olmaları istenir. Telegram sistemine dahil olunduktan sonra bot oluşturma için **BotFather** adındaki, “mavi tik” ibaresine sahip, Telegram’ın resmi bot yönetim hesabını arayıp bulmaları ve hesap ile mesajlaşma ekranını görüntüledikten sonra **Start** düğmesine tıklayarak etkileşimi başlatmaları sağlanır. BotFather öncelikli olarak bot oluşturma, düzenleme ve yönetimi konusunda kullanılabilecek komutları ve Telegram Bot API ile etkileşim kurabilmek için sunulan rehbere dahil link bilgilerini paylaşır (Şekil 55).



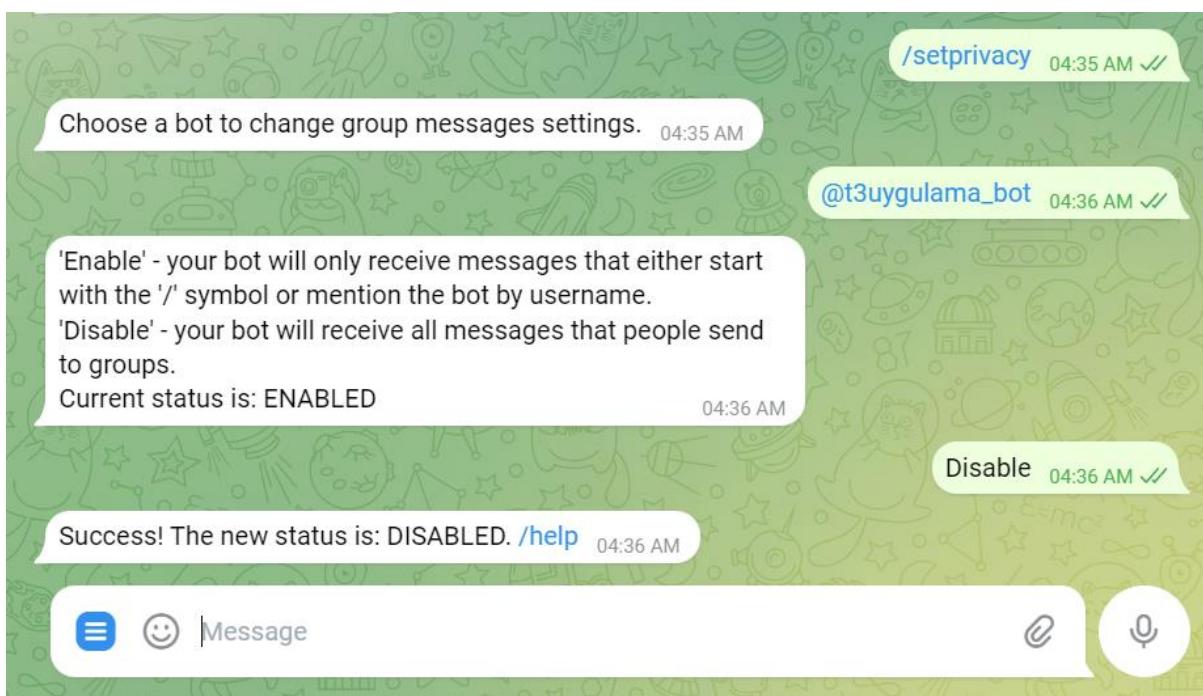
Şekil 55. Telegram ortamında BotFather hesabının açılış (start) arayüzü

BotFather’ın sunduğu komutlardan yola çıkılarak **/newbot** mesajı gönderilmek suretiyle yeni bot hesabı oluşturma süreci başlatılır. Söz konusu mesaj iletildiginde BotFather sırasıyla **bot hesabının genel ismi** ve Telegram ortamında görünecek **bot kullanıcı ismi** (sonu bot ibaresi ile biten) bilgilerini talep eder. İlgili bilgiler verildikten sonra Telegram sistemi bot hesabını otomatik olarak oluşturur ve bot için oluşan Telegram linki, tekrar düzenlemeler konusunda izlenebilecek adımlar ve en önemli bota ulaşmak için kullanılacak HTTP API **token** bilgisini (bir tür erişim key bilgisi) mesaj olarak bildirir (Şekil 56a). Öğrencilerden söz konusu token bilgisini uygulamalarda kullanabilmek adına not almaları ve geliştirme amaçları dışında başkalarıyla paylaşmamaları vurgulanır.

Öğrencilerin son olarak Şekil 56b’de gösterilen adımları izleyerek bot hesaplarına isim ve komut üzerinden genel erişimi kısıtlamaları istenir (Şekil 56b). Söz konusu düzenleme bot hesaplarının daha denetimli ve amaca yönelik kullanımında aktif hale getirilebilecektir. Öğrencilere artık bot hesaplarının oluşturulduğu ve bot etkileşim fonksiyonlarının geliştirilmeye hazır olduğu ifade edilir.



(a)



(b)

Şekil 56. (a) Oluşturulan bot hesabına yönelik link, token bilgileri ve diğer bildirimlerin görüntülenmesi

(b) Bot hesabının isim ve komut üzerinden genel erişiminin kısıtlanması



3. ADIM: ÜRET

Öğrencilerle birlikte Python ortamında Google Text-To-Speech tabanlı sesli asistan Telegram bot uygulamasına yönelik kütüphane çağrıları ve ön tanımlamalar gerçekleştirilir.

Kütüphane Çağrıları ve Ön Tanımlamaların Gerçekleştirilmesi

Öğrencilerin Python ortamından Telegram bot hesabına erişerek düzenlemeler yapabilmesi için öncelikli olarak Telegram bot kütüphanesini yüklemeleri gerekmektedir. Bu amaçla aşağıda belirtilen yükleme komutu ile ilgili kütüphanenin konsol ortamından yüklenmesi gerçekleştirilecektir:

pip install python-telegram-bot

Kütüphanenin başarılı bir şekilde yüklenmesinden sonra öğrencilerden Python ortamında Şekil 57'de gösterilen kütüphane çağrılarını yapmaları istenir. Yapılan çağrılar Telegram bot bileşenleri ile birlikte sesli asistan kullanım süreçlerini de destekleyecek genel kütüphaneler arasında yer alan **os** ve **io**'un çağrımasına yönelikdir. Özellikle **telegram.ext** altından çağrılan bileşenler, Telegram bot hesabı etkileşim komutlarının tanımlanması ve bu komutlar üzerinden kendisine gelen verileri (mesajlar, dosyalar...vs.) ele almasına yönelik fonksiyonları içermektedir.

```
7 import logging
8 from telegram import ForceReply, Update
9 from telegram.ext import Application, CommandHandler, ContextTypes, MessageHandler, filters
10
11 import os
12 import io
```

Şekil 57. Telegram kütüphanesi ve genel kütüphane bileşenlerinin çağrımasına dair kodlar

Uygulamada Google Speech-To-Text üzerinden sesli asistan desteği oluşturulacağı için devam eden satırlarda Speech-To-Text kütüphane çağrıları ve “client” değişken tanımlamasını yapılması sağlanır. Öğrencilere Speech-To-Text hizmetinden faydalananın için daha önce oluşturulan JSON dosyasını (Bkz. 3. Hafta – 5. Gün, 1., 2. Dersler) mevcut uygulama dizinine kopyalayarak kullanması istenir. Speech-To-Text bileşenlerinin tanımlanmasından sonra uygulamanın Telegram etkileşimi esnasında döngü sorunlarıyla karşılaşmaması ve yine Telegram ortamına oturum açabilmesi (logging) için gerekli olan bazı destekleyici kodlar da yazılır (Şekil 58).

```

14 from google.cloud import speech
15 os.environ['GOOGLE_APPLICATION_CREDENTIALS'] = 'key.json'
16 client = speech.SpeechClient()
17
18 import nest_asyncio
19 nest_asyncio.apply()
20
21 logging.basicConfig(
22     format='%(asctime)s - %(name)s - %(levelname)s - %(message)s', level=logging.INFO
23 )
24 logging.getLogger("httpx").setLevel(logging.WARNING)
25
26 logger = logging.getLogger(__name__)

```

Şekil 58. Google Speech-To-Text tanımlamaları ve diğer destekleyici kodlar



4. ADIM: İLERLET

Öğrencilerin Telegram bot fonksiyonları ve Google Text-To-Speech entegrasyonuna yönelik fonksiyonları kodlayarak uygulamayı çalıştırırları, sonuçları gözlemlemeleri ve gruplar halinde çalışarak genel denemeler / düzenlemeler yapmaları sağlanır.

Bot Fonksiyonları ve Sesli Asistan Entegrasyonları

Öğrencilere uygulamanın geriye kalan kodlama bölümünde akişin sırasıyla Telegram botuna ait genel fonksiyonların tanımlanması, Speech-To-Text için sesli komut dosyası alımı ve tahmin fonksiyon bloğunun tanımlanması ve ana program bloğunun tanımlanması şeklinde olacağı açıklanır.

Öğrenciler Telegram botu için Şekil 59'da gösterilen etkileşim fonksiyonlarını kodlar. **async def** ifadesi ile başlayan her bir fonksiyon, bot hesaba gelen komutlar karşısında sağlanacak döntürtleri içermektedir. **async def start** fonksiyonu, bot ile Start düğmesi tıklanarak ya da /start komutu gönderilerek etkileşime başlanması durumunda tetiklenecek fonksiyondur. Söz konusu fonksiyon içerisinde kullanıcıya ismiyle hitap edilerek genel bir karşılama mesajı görüntülenmektedir. Öğrencilere bot hesabının kullanıcı tarafına dönütte bulunurken **await update.message.reply_text** fonksiyon tanımlamasının kullanıldığı ifade edilir.

```

28     async def start(update: Update, context: ContextTypes.DEFAULT_TYPE) -> None:
29         user = update.effective_user
30         await update.message.reply_html(
31             rf"Merhabalar {user.mention_html()}! Ben T3 örnek uygulama botuyum. Tanıştığuma memnun oldum!",
32             reply_markup=ForceReply(selective=True),
33         )
34
35     async def yardım(update: Update, context: ContextTypes.DEFAULT_TYPE) -> None:
36         await update.message.reply_text("Yardım menüsü. Kullanabileceğin komutlar sunlar:")
37         await update.message.reply_text("Yardım için (Zaten buradasın (: ) : /yardım")
38         await update.message.reply_text("Yazdıklarımı tekrar etmem için: /yankı")
39         await update.message.reply_text("T3 Hakkında: /hakkında")
40
41     async def yankı(update: Update, context: ContextTypes.DEFAULT_TYPE) -> None:
42         await update.message.reply_text("Söylediklerin yankı yaptı:", update.message.text)
43
44     async def hakkında(update: Update, context: ContextTypes.DEFAULT_TYPE) -> None:
45         await update.message.reply_text("Sayfamızı ziyaret edebilirsin: https://www.t3vakfi.org/")
46

```

Şekil 59. Telegram botunun etkileşim fonksiyonlarının kodlanması

Telegram botunun Speech-To-Text üzerinden kullanıcının gönderdiği sesli mesajları algılayarak etkileşim içerişine girmesi için Şekil 60'da gösterilen fonksiyon yazılır. Söz konusu fonksiyon diğer bot fonksiyonlarıyla benzer şekilde kodlanmış olsa da, özel olarak kullanıcının ses dosyasını okuyup sesli komutun Speech-To-Text Yapay Zeka modeli üzerinden tahmin edilmesini sağlayan ve tahmin edilen komuta göre daha önce tanımlanmış etkileşim fonksiyonlarının çağrılmasını mümkün kıلان kodlar içermektedir. **49.52. kod satırları** arasında gerçekleştirilen ses dosyası okuma işlemleri ardından **53.-58. kod satırları** arasında Speech-To-Text sunucunu üzerinden tahmin / tespit süreci işletilmektedir. Son olarak **60.-64. kod satırları** arasında tespit edilen komuta göre bot fonksiyonları yönlendirmeleri yapılmakta, eğer komut anlaşılmaz ise kullanıcıya uygun mesaj iletilmektedir.

```

47     async def sesli_komut(update, context):
48         await update.message.reply_text("Sesli komut için teşekkür ederim (:")
49         with io.BytesIO() as fh:
50             fh = await context.bot.get_file(update.message.voice.file_id)
51             fh = fh.toString('base64')
52             audio = speech.RecognitionAudio(content=fh)
53             config = speech.RecognitionConfig(
54                 encoding=speech.RecognitionConfig.AudioEncoding.LINEAR16,
55                 enable_automatic_punctuation=True,
56                 audio_channel_count=1,
57                 language_code="tr-TR")
58             response = client.recognize(request={"config": config, "audio": audio})
59
60             for result in response.results:
61                 if (result.alternatives[0].transcript == "yardım"):
62                     yardım(update, context)
63                 else:
64                     update.message.reply_text("Maalesef komutunuzu anlayamadım :(")

```

Şekil 60. Telegram sesli komut algılama fonksiyon aşamalarının kodlanması

Ana Program Bloğu

Öğrenciler son bölümde uygulamanın Şekil 61'de gösterilen kodlar yardımıyla ana program blogunu kodlar. Söz konusu kodlar öncelikli olarak Telegram bot hesabına erişimi sağlayacak

application değişkeninin tanımlanması ile başlamakta, ilgili satırda kod içerisinde daha önce temin edilmiş olan **token bilgisi** belirtilmektedir. **Application.builder().token** kod tanımlaması, token bilgisi eşliğinde tanımlı durumda Telegram bot hesabına erişim bilgilerinin tanımlanmasını, kod satırı sonundaki **build()** fonksiyonu da erişim değişken yapısının oluşturulmasını sağlamaktadır.

Telegram botuna yönelik daha önceki satırlarda tanımlanan her bir fonksiyon, ana program bloğu devamında yer verilen **application.add_handler** kod satırları ile birlikte bot uygulama değişkenine eklenmektedir.

Öğrencilere sesli komut etkileşim fonksiyonunun diğer fonksiyonlardan farklı bir şekilde tanımlama içерdiği (**application.add_handler(MessageHandler(filters.VOICE, sesli_komut))**) özellikle vurgulanır. Benzer şekilde **application.add_handler(MessageHandler(filters.TEXT & ~filters.COMMAND, yanlı))** kod yapılanması da kullanıcidan gelen mesajı tekrar iletme adına kendine has bir yapıya sahiptir.

Ana program bloğunun son satırında yer alan **application.run_polling(allowed_updates=Update.ALL_TYPES)** kodu, Telegram botunun aktif hale getirilerek çalışmaya başlamasını sağlamaktadır.

```
66 if __name__ == "__main__":
67     application = Application.builder().token("token_bilgisi_buraya_girilmelidir").build()
68
69     application.add_handler(CommandHandler("start", start))
70     application.add_handler(CommandHandler("yardim", yardım))
71     application.add_handler(MessageHandler(filters.TEXT & ~filters.COMMAND, yanlı))
72     application.add_handler(CommandHandler("hakkında", hakkında))
73     application.add_handler(MessageHandler(filters.VOICE, sesli_komut))
74     application.run_polling(allowed_updates=Update.ALL_TYPES)
```

Şekil 61. Ana program bloğa ait kodlar

Sonuçların Gözlemlenmesi

Kodlanan Python uygulaması öğrencilerle birlikte çalıştırılır ve Telegram tarafına geçirilerek bot hesabı ile etkileşim başlatılır. Şekil 62'de gösterildiği gibi oluşturulan bot standart kullanıcı komutlarına dönütler ürettiği gibi gönderilen sesli komutlara da dönüt oluşturmaktadır.



Şekil 62. Geliştirilen bot hesabıyla Telegram ortamında gerçekleştirilen etkileşimler

Öğrencilerin oluşturdukları bot hesaplarını genel hatlarıyla kontrol ettikten sonra gruplar halinde çalışarak yeni düzenlemeler yapmaları istenir. Bu etkinlik için 10 dakikalık bir süre ayrılrı.

UYGULAMANIN PYTHON KODLARI

```
import logging
from telegram import ForceReply, Update
from telegram.ext import Application, CommandHandler, ContextTypes, MessageHandler,
filters

import os
import io

from google.cloud import speech
os.environ['GOOGLE_APPLICATION_CREDENTIALS'] = 'key.json'
client = speech.SpeechClient()
```

```
import nest_asyncio
nest_asyncio.apply()

logging.basicConfig(
    format="%(asctime)s - %(name)s - %(levelname)s - %(message)s", level=logging.INFO
)
logging.getLogger("httpx").setLevel(logging.WARNING)

logger = logging.getLogger(__name__)

async def start(update: Update, context: ContextTypes.DEFAULT_TYPE) -> None:
    user = update.effective_user
    await update.message.reply_html(
        rf"Merhabalar {user.mention_html()}! Ben T3 örnek uygulama botuyum. Tanışlığıma
memnun oldum!",
        reply_markup=ForceReply(selective=True),
    )

async def yardim(update: Update, context: ContextTypes.DEFAULT_TYPE) -> None:
    await update.message.reply_text("Yardım menüsü. Kullanabileceğin komutlar şunlar:")
    await update.message.reply_text("Yardım için (Zaten buradasın (: ): /yardim")
    await update.message.reply_text("Yazdıklarını tekrar etmem için: /yankı")
    await update.message.reply_text("T3 Hakkında: /hakkında")

async def yanki(update: Update, context: ContextTypes.DEFAULT_TYPE) -> None:
    await update.message.reply_text("Söylediklerin yankı yaptı:",update.message.text)

async def hakkında(update: Update, context: ContextTypes.DEFAULT_TYPE) -> None:
    await update.message.reply_text("Sayfamızı ziyaret edebilirsin:
https://www.t3vakfi.org/")

async def sesli_komut(update, context):
    await update.message.reply_text("Sesli komut için teşekkür ederim (:")
```

```
with io.BytesIO() as fh:
```

```
    fh = await context.bot.get_file(update.message.voice.file_id)
    fh = fh.tostring('base64')
    audio = speech.RecognitionAudio(content=fh)
    config = speech.RecognitionConfig(
        encoding=speech.RecognitionConfig.AudioEncoding.LINEAR16,
        enable_automatic_punctuation=True,
        audio_channel_count=1,
        language_code="tr-TR")
    response = client.recognize(request={"config": config, "audio": audio})
```

```
for result in response.results:
```

```
    if (result.alternatives[0].transcript == "yardım"):
        yardım(update, context)
    else:
        update.message.reply_text("Maalesef komutunuzu anlayamadım :")
```

```
if __name__ == "__main__":
```

```
    application = Application.builder().token("token_bilgisi_buraya_girilmelidir").build()
    application.add_handler(CommandHandler("start", start))
    application.add_handler(CommandHandler("yardim", yardım))
    application.add_handler(MessageHandler(filters.TEXT & ~filters.COMMAND, yankı))
    application.add_handler(CommandHandler("hakkında", hakkında))
    application.add_handler(MessageHandler(filters.VOICE, sesli_komut))
    application.run_polling(allowed_updates=Update.ALL_TYPES)
```



5. ADIM: DEĞERLENDİRME

Derste edinilen bilgilerden hareketle soru-cevap aşaması gerçekleştirilerek sesli asistan ve Telegram bot entegrasyonuna yönelik edinimler pekiştirilir.

Öğrencilere şu sorular sorulabilir:

- Ders kapsamında gerçekleştirdiğiniz Telegram bot geliştirme süreçlerine dair düşünceleriniz nelerdir?

- Google Text-To-Speech ve Telegram bot entegrasyonu konusunda düşüneleriniz nelerdir?
- Geliştirdiğiniz sesli asistan ve Telegram bot destekli çözümü hangi faaliyet alanlarına ya da problemlere uyarlamak isterdiniz?
- Bot tabanlı çözümlerin avantaj ve dezavantajları ile birlikte gelecekteki potansiyelleri hakkında görüşleriniz nelerdir?