

SÜLEYMAN DEMIREL ÜNİVERSİTESİ

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

BİTİRME PROJESİ

Cyclens

İnsanları Analiz Edip Sınıflandırılmasını Sağlayan Modüler Yapay Zeka Yazılımı

Furkan TÜRKAL

Metin UR

Proje Danışmanı

Dr. Öğretim Üyesi Fatih GÖKÇE

ver. 0.7.0

June 21, 2019

1

ÖZET

İnsanoğlu var olduğundan bugüne dek, 21. Yüzyıl'da en akıllı (teknolojik) hayatını yaşamaktadır. Hayatımıza, geçen yıllarda, Endüstri 4.0'dan sonra Japon bilim adamlarının ve sosyologların ortaya çıkardığı çok önemli bir kavram girmiştir: Toplum (Society) 5.0 [1]; toplumu Süper Akıllı hale getirmek. Bunun için farklı türlerden teknolojilerin ve sistemlerin birleşmesiyle oluşturulmuş yeni bir yapıyı hayatı geçirmek gerekmektedir. Projemizde Süper Akıllı Toplum kapsamına bağlı kalarak, yeni teknolojilere (5G, VR, AR, XR, IoT vb.) ve çevre birimlerine tam uyumlu çalışacak, gelecekte üretilcek yeni nesil Akıllı Lens konseptini [14,15,16,17] hayatı geçirmek için ilk yazılım adımlarını atmayı hedeflemekteyiz.

Giyilebilir Akıllı Lens'ler, gelecekte çok önemli bir yere sahip olacaklardır. İnsanların hem göz numaralarını otomatik olarak ayarlayabilecek, hem de diğer akıllı cihazlar ile senkronize olarak bağlantı kurarak, bildirimleri gösterebileceklerdir. Diğer insanların, yüzlerini analiz edip tanımakla kalmayıp, vücut dillerini ve mimiklerini de anlayıp, kişilik analizlerini çıkarabileceklerdir [2]. Akıllı Lensler veya onları simüle edecek donanımlar, yeterli işlem gücüne (şimdilik) sahip olmadıklarından, ekran kartı (GPU) ile beslenen sunucularda işlemler gerçekleştirilecektir. Bu projede oluşturacağımız, yapay zeka servisleri, uzak veya yakın sunucuya bağlanarak, sunucudaki yapay zeka algoritmalarını işletip, işlem gücü düşük olan bir istemciye (client) veriler gönderilecektir. Lens/Ekran üzerinde Yapay Zeka tarafından işlenmiş ve hazırlanmış olan GUI (Arayüz) oluşturulacaktır.

Yapılması asıl hedeflenen akıllı Lens teknolojisine henüz sahip olmadığımız için, bu teknoloji mobil bir cihaz kullanılarak, VR (Sanal Gerçeklik) gözlüğü ve AR (Arttırılmış Gerçeklik) teknolojileri ile simüle edeceğiz. Gelecekte üretilmesi muhtemel olan Akıllı Lens'ler için tamamen uyumlu olarak çalışacak bu projede modüler olarak tasarlanmış yapay zeka servislerine sahip olmayı hedeflemekteyiz.

2

MOTİVASYON

Günümüze kadar yapılmış konusu Yapay Zeka ve Görüntü İşleme temalı bilim kurgu filmleri ve kitapları, birçok araştırmmanın, çalışmanın ve projenin temel motivasyon kaynağını oluşturmaktadır [3]. Büyük firmalar tarafından bu iş için geliştirilmiş yapay zeka kütüphaneleri ve eklentileri, projelerin gelişim sürecini hızlandırarak, daha ölçülebilir ve uygun geliştirme ortamını sunmaktadır.

Teknolojinin gelişimiyle öncümüzdeki yıllarda hayatımıza gireceğini öngördüğümüz Akıllı Lens teknolojisi ile ilgili bu güne kadar yapılan çalışmalar henüz böyle bir ürünün hayatımıza girmesini sağlayacak düzeye ulaşmamış olup, piyasada şu anda -bildiğimiz kadarıyla- henüz satışı bulunmamaktadır. Akıllı Lens'ler ile ilgili ARGE ve Tasarım sürecinde olan şirketler için de ilham kaynağı sağlayabilecek bir prototip yazılımı geliştirmeyi amaçlamaktayız. İnsanlar için gelecekte son derece etkin ve kaliteli göz deneyimini sağlayacak ürünlerin yanı sıra, Lens ile iletişimde olacak işletim sistemimiz sayesinde, vurguladığımız "Akıllı Yaşam" tabirini, gözlerimiz için üst plana taşımayı hedeflemekteyiz. Proje sonunda beklenen başarı elde edilip, ciddi bir ilerleme sağlanırsa, projenin hedeflediğimiz "Akıllı Modüler Sistem" mimarisi sayesinde geliştirilmeye ve uygulanmaya oldukça açık olacak bir ilk prototip ortaya konulacaktır.

3

PROBLEM, ÇALIŞMA VE YENİLİK

3.1 PROBLEM TANIMI

İnsanları bazen tanımak, analiz etmek, insan gözünün ve zekasının anlayamayacağı kadar zor olabiliyor. Bu durumda bizim yerimize bir şey geçiyor: Yapay Zeka. Bir kişi ile yüz yüze görüşürken, akıllı aracımızı kullanarak, insanların birbirine olan mimiksel tepkilerini ölçebilmek, o insana ait kişisel özelliklerini çıkarıp veritabanına ekleyebilmek, ağız hareketlerinden ve ses seviyesinden psikolojik durum raporunu çıkarabilmek gibi özellikler gerçekten hayatımıza oldukça ciddi seviyede etki edebilirdi. Karşımızdaki insanların "Kişisel Veritabanı" ni oluşturabilmek için bu projenin temelini atıyoruz.

3.2 ÇALIŞMANIN AMACI

Projenin asıl amacı, kullanacağımız Yapay Zeka ve Görüntü İşleme tekniklerini tek bir çatı altına toplayarak, insanları analiz etmek, duygularını anlamak, yaşıını tahmin etmek, cinsiyetini öğrenmek, eylem ve hareketlerini anlayarak bir "Kişisel Veritabanı" ortaya çıkarmaktır. Bu özellikleri birleştirerek insana özel benzersiz bir ID (Kimlik Numarası) oluşturarak, bu ID sayesinde insanların veritabanından çekilipli hatırlanması işlemi veya veritabanına bir insanı kaydetme işlemi yapılabilir [4].

İnsanları uzaktan analiz eden bu sistemde, oluşturulan veritabanı sayesinde, derinlemesine ve detaylı olarak veri madenciliği teknikleri kullanılarak, proje türüne özgün, özel amaçlı sistemler de oluşturulabilir. Projenin en başta modüler olarak tasarılanması da tam olarak bunu amaçlamaktadır.

3.3 YENİLİK UNSURU

Toplum 5.0 ile gelecekteki yeni teknolojiler ile insanlar artık daha da iç içe yaşıyor olacaktır [5]. Bu durumu daha deneyimsel hale getirebilmek için farklı türde yeni nesil yazılımlar denemeye devam etmekteyiz. İnsanların hayatlarına uzun süre eşlik eden, fiziksel kusurları düzeltten donanımlar (Gözlük, Lens, vb.) da buna örnek olabilir. Eğer bu tür donanımlara, hayatı daha da kolaylaştırabilecek ve zevkli hale getirebilecek özellikler verebilirsek, hem Toplum 5.0 hedefimize bir adım daha yaklaşmakla kalmayıp, hem de yaşam kalitemizi bir üst seviyeye çıkartabiliriz.

Sadece kameralar ve Akıllı Lens'ler ile etkileşimde olan yapay zeka modülleri sayesinde, çok amaçlı ve modüler yönlü kullanılabilirlik özelliği, bir çok farklı yeni projenin temellerini oluşturacaktır.

4

PROJEDE KULLANILAN YÖNTEM VE METHODLAR

Yapılması planlanan projede kullanacağımız Arch Linux İşletim Sistemi, Yapay Zeka ve Görüntü İşleme algoritmaları ile birlikte eş zamanlı olarak senkronize bir şekilde çalışarak sistem bütünlüğünü oluşturacaktır. Bu nedenle, işletim sistemini seçmemizin nedeni ayrı ayrı programları farklı PID Tree şeklinde çalıştmak yerine, işletim sisteminin bizlere sağlamış olduğu Systemd servislerinden faydalananmaktadır [6]. İşletim sistemi ilk başlatıldığında (Boot) olduğu zaman, bütün çekirdek (Core) modüller çalışmaya başlayacak ve kullanıma hazır hale getirilecektir. Bu nedenle, İşletim Sistemini bir amaç olarak tasarlamak yerine, bir araç olarak tasarlayıp modüler bir yapı kazandırıyoruz.

Bu servislerimize güç verecek olan işletim sistemi Arch Linux 'u seçmemizin asıl sebebi ise; içerdiği zengin içerikli Paket Yöneticisi (PACMAN, AUR) ve Arch Topluluk Forumları bu sebepler arasında yer alıyor. [7] Bir sorun ile karşılaşmamız durumunda, hem topluluk tarafından yardım alabilir, hem de deneyimlerimizi paylaşabiliriz. Bunların yanında güvenilirliği, sade tasarımlı ve performansı bizler için bu seçimi yapmaktaki ana kriterler arasındaydı. Sunucunun yalnızca 256MB Ram istemesi ve NVIDIA kütüphanelerinin FTP ile kolay bir şekilde sağlanmasını da bu sebepler arasında gösterebiliriz.

4.1 KÜTÜPHANELER VE YAZILIMLAR

- **Tensorflow:** Projede kullanacağımız açık kaynaklı Makine Öğrenmesi kütüphanesi olan Tensorflow, Google Brain ekibi tarafından açık-kaynak olarak topluluk desteğiyle geliştirilmektedir. Bu kütüphaneyi seçmemizin asıl sebebi, güçlü bir yapıya ve güncel bir topluluğa sahip olmasıdır. [8]

Kullanım Yeri: Projede yapılandıracağımız *Yüz Tanıma* modülü için Tensorflow'un, Hub çatısı altında bulunan, *image processing* sistemine gücünü veren, *retrain.py* [20] isimli derin öğrenme script dosyasını kullanacağız.

Eğitim (Train) aşaması için kullanacağımız model ise TensorFlow Research tarafından oluşturulan *Inception v3* [21] öğrenim sistemidir. *Inception v3* modelini, diğer *AlexNet* veya *ResNet* modellerine tercih etmemizin sebebi, benchmark testlerine göre [22], oldukça hızlı eğitilmesi ve diğer modellere oranla daha hızlı cevap vermesidir. Ayrıca hala aktif bir araştırma ve geliştirme içerisinde olması, bu durumu destekleyen diğer unsurlardır.

- **Keras:** Kullanacağımız diğer bir nöral ağ kütüphanesi olan Keras ise, Python ile yazılmış olup, çapraz platform desteğine, Tensorflow ve Theano gibi kütüphanelere yeni beceriler ve kabiliyetler katmamıza olanak sağlıyor. [9] Araştırmalarımız için hızlı denemeler yapmak ve sonuca hemen ulaşabilmek bizlere oldukça zaman kazandıracaktır. Yüksek seviye Önişleme (Preprocessing) ve Katman (Layer) desteği ile hız konusunda da bize avantaj sağlıyor. Kullanacağımız diğer Aktivasyon (Activations) hesaplamlarını ve Regularizers katman ceza optimizasyonlarını ise ana modülü tasarlarken kullanacağız.

Kullanım Yeri: Keras, projede kullanacağımız *Ifade Tanıma* ve *Cinsiyet Tanıma* modüllerine gücünü verecektir. *keras.models* çatısı altında bize sunulan *load model* motoru, *Hierarchical Data Format* desteği [23] sayesinde, önceden öğretilmiş (pre-trained) modelleri en hızlı ve optimize edilmiş yoldan direkt olarak hafızaya yükleyememize imkan tanıyor. Bu sayede sınıflandırıcılarımız (classifiers); Keras tarafından bize sunulan, Tensorflow ve Theano çatılarının üzerine oluşturulmuş yüksek seviyeli fonksiyonlar sayesinde, *predict* fonksiyonu ile öğrenme işlemini en basit ve en verilmiş bir şekilde yapmış olacağız.

- **OpenCV:** Bu projede olmazsa olmaz baş kütüphaneleri arasında yer alıyor. Projemizde gerçek zamanlı görme işlemlerini bu kütüphaneyi kullanarak yapacağız. Intel tarafından desteklenip, açık- kaynak kodlu olması, bize geliştirilebilir ve özelleştirilebilir imkanlar sağlamaktadır. [10]

Kullanım Yeri: Genel amaçlı görüntü işleme kütüphanesi olarak kullanacağımız OpenCV; modüllerin Process aşamasından bir önceki Boru Hattı (Pipeline) olan PreProcess aşamasında işlemlerini sürdürdürecektil. Bu işlemler; *haarcascade* modeli ile yüzün koordinatlarını (*x*, *y* ve ölçüğünü (*width*, *height*) bulmak, yüzleri *resize* fonksiyonu ile işlemeye hazır hale getirip optimize etmek, derin öğrenme aşaması için gerekecek olan maskeleme işlemlerini yapmak ve yüz tanıma sisteminin ana parçasını oluşturmak gibi görevlere sahip olacaktır.

- **Theano:** Görüntü işlemede matematiksel işlemleri optimize bir şekilde değerlendirmek ve çok boyutlu dizileri verimli bir şekilde hesaplayabilmek için Theano kütüphanesinden

faydalanaçğız. GPU desteğinin olması, hız ve ölçeklenebilir özellikler sağlamaç, dinamik olarak C kodu üretebilmesi, bu kütüphaneyi seçmemizin asıl sebeplerindendir. [11]

- **Flask:** Python için geliştirilmiş, Werkzeug (WSGI, Python ile HTTP NGINX/Apache sunucusu arasındaki haberleşme kütüphanesi) üzerine yapılandırılmış bir mikro web kütüphanesidir. [24] Django'ya göre tercih etmemizin asıl sebepleri; oldukça basit entegrasyonu, *RESTful request dispatching* sistemi, güclü bir dökümantasyonu ve diğer birçok özelliğe sahip olmasıdır.

Kullanım Yeri: İstemci (Client) tarafından Sunucuya (Server) istek yapılabilmesi ve geriye yanıt dönmesi için bu kütüphaneyi kullanacaçğız. Tanımladığımız *Route* yollarına gelen istekler; modüller tarafından sırayla işlenecek ve geriye bir *JSON* veri yapısını HTTP hata kodu ile API'ye istekte bulunan kişi için geriye yanıt dönecektir.

- **Tornado:** Facebook tarafından satın alınan FriendFeed şirketinin geliştirdiği, ölçeklenebilir ve non-blocking I/O yapısını bizlere sağlayan yüksek performanslı bir web kütüphanesidir. [25]

Kullanım Yeri: Kullanacaçığımız Flask kütüphanesi *blocking* bir yapıda olup, bu yapıyı *non-blocking* hale getirebilmek için Tornado'nun bizlere sunmuş olduğu *HTTPServer* ve *WSGIContainer* desteklerini kullanarak, Flask web sunucumuza *IOLoop* örneklemi (instance) yapısı oluşturabiliriz. [26, 27]

- **NumPy:** Çok boyutlu diziler (array), çeşitli türetilmiş nesneler (maskelenmiş diziler ve matrisler gibi) ve bir sürü yüksek seviye, matematiksel, mantıksal, şekil manipülasyonu, sıralama, seçme, ayrik Fourier de dahil olmak üzere diziler üzerinde hızlı işlemler yapmamızı sağlayan Python kütüphanesidir. [28]

Kullanım Yeri: Client tarafından API sunucusuna gönderilen resim için; işlenmeden önce optimize edilerek ve işlemlerin daha hızlı olmasını sağlayabilmek amacıyla, *uint8* dizisine dönüştürülme aşamasında kullanacaçğız. Hem bu sayede, kütüphaneler arasında tip dönüşüm problemlerinin önüne geçilecek, hem de hafızada daha verimli yer tutacak, hem de diğer birçok işlemleri yapabilmek için NumPy'nin bize sunduğu fonksiyonlardan faydalanaçebileceçiz.

- **DLIB:** Çapraz-platform makine öğrenmesi algoritmaları içeren C++ ve Boost tabanlı bir Python kütüphanesidir. *HoG* ve *CNN* yüz bulma modellerini destekler ve *Face Location* verilerini işleyebilir.

4.2 GENEL SİSTEM MİMARİSİ

Yapacağımız mimari genel olarak Sunucu-İstemci (Server-Client) ilişkisine bağlı kalmaktadır. Aradaki bilgi ve görüntü akışını sağlayabilmek için Uygulama Programlama Arayüzü (API) kullanacağız. Bu sayede sistemimizi kontrol etmek hem daha da kolaylaşacak, hem de daha anlaşılır bir veri akışı sağlanacaktır.

- **Go:** API’yi yapacağımız programlama dilini Go olarak seçmemizin başlıca nedeni, Sunucu ve İstemci haberleşme performansının ciddi derecede hızlı ve efektif kullanımına sahip olmasıdır. [12] Google tarafından açık-kaynak olarak geliştiriliyor olması da artı bir etkidir.
- **ReactNative:** Yapacağımız istemcinin görevi, sunucu tarafında belirlediğimiz bir değer ile (Rate) gerçek zamanlı olarak, kamera tarafından yakalananmiş görüntülerini API’ye göndermektir. Diğer bir görevi ise sunucunun görüntütüyü işledikten sonra API aracılığı ile bir cevap geldiğinde, gelen cevabı ekrana çizmektir. İstemciyi tasarlamak için Facebook tarafından geliştirilen, çapraz (cross) platform destekli ReactNative’i kullanacağız. [13]
- **Apache Solr:** Bütün yüz tanıma işlemleri yüksek matematiksel işlem gücü ve hızlı arama yetenekleri istediginden dolayı, bu görevde en uygun olabilecek veritabanını seçmemiz gerekmektedir. Bu sebeple *Search Engine* tabanlı, kurumsal seviyede açık-kaynak bir veritabanı olan Apache Solr ’ı kullanacağız. Gerçek zamanlı indeksleme, dinamik kümeleme, veritabanı entegrasyonu ve NoSQL özellikleri, sunucuya oldukça hız kazandıracak olup; yüzbinlerce yüz verisi içeren (her satır için 128 kolon) bir veritabanında, aradığımız 128 adet parametre içeren yüz verisi, bütün diğer veriler ile tek tek 128 parametreli *Euclidean* hesabının yapılması milisaniyeler içerisinde bizlere sonuç verecektir. Veritabanına erişim sadece sunucu tarafından sağlanacak olup, diğer hiçbir modülün erişimi sağlanamayacaktır.

Yüzler arasındaki mesafeleri ölçmen için kullandığımız formül ise;

$$d(q, p) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + (q_3 - p_3)^2 + \dots + (q_n - p_n)^2}$$

Figure 4.1: Euclidean Hesabı

Python tarafından Apache Solr'a, hesapladığımız 128 adet Encoding değerlerini göndermemiz gerekmektedir. Bize donecek olan mesafelerin en küçüğü, o yüze en yakın olan kişinin değeridir. Bu nedenle göndermemiz gereken SQL Query;

```

SELECT * FROM face_encodings
ORDER BY
    sqrt(
        power(e1 - ENCODING_VALUE_0, 2) +
        power(e2 - ENCODING_VALUE_1, 2) +
        power(...) +
        ...
        power(e128 - ENCODING_VALUE_128)
    )

```

4.3 GENEL İLETİŞİM MİMARİSİ

Sistemimizin genel iletişim mimarisi, bir istemci (Client) cihazın, Ağ (Network) üzerinden API sunucusuna bağlanarak, sürekli iletişim halinde olma prensibine dayanıyor. API bu iletişim yapabilmek için hem yapay zeka modül sunucularını, hem de istemciden gelecek olan verileri dinliyor ve uygun yönlendirmeyi (Route) gerçekleştiriyor.

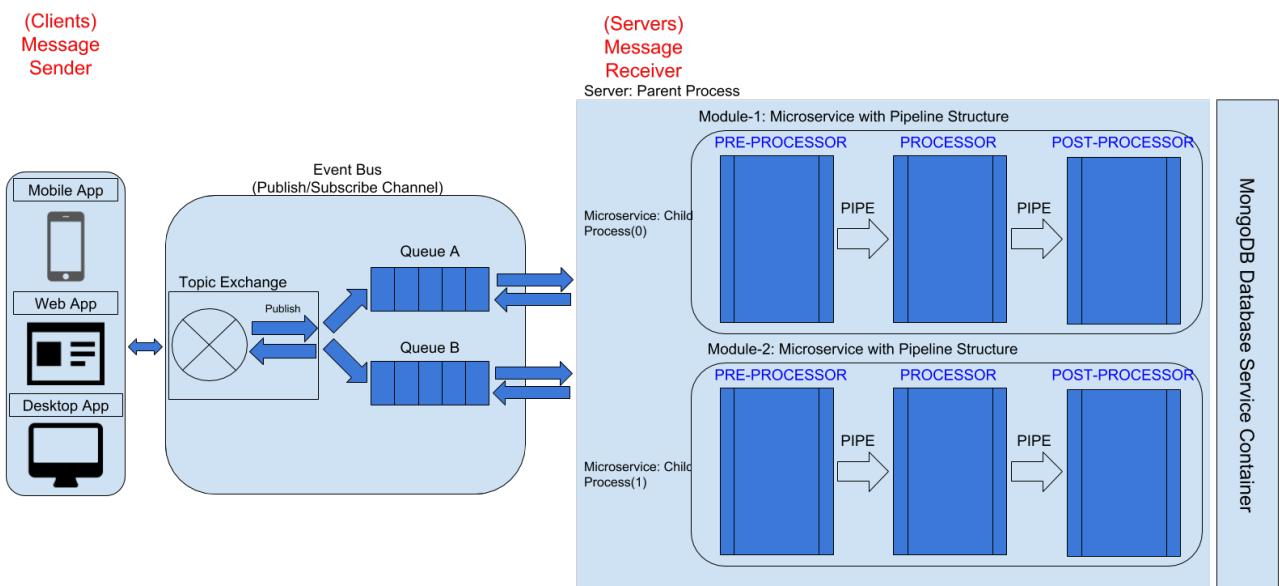


Figure 4.2: Genel İletişim Mimarisi

4.4 UYGULAMA ÇALIŞMA PRENSİBİ

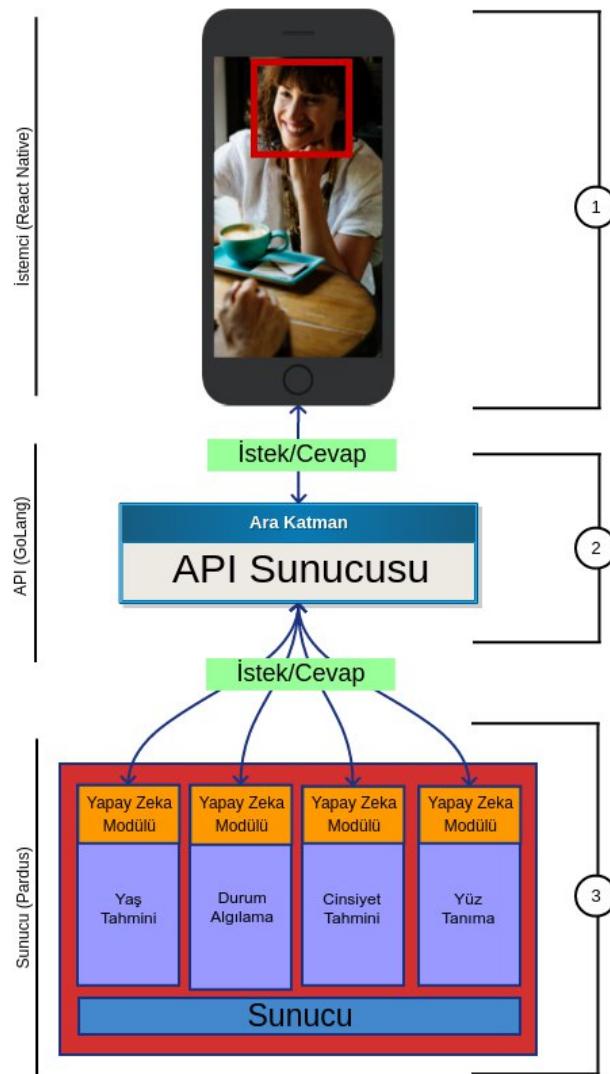


Figure 4.3: Genel Çalışma Mimarisi

4.4.1 İSTEMCI (CLIENT)

İstemcinin görevlerinden biri, elde ettiği görüntüde olabildiğince hızlı bir şekilde yüz tespiti yapıp, görüntüdeki yüzü takip etmektir. Yüz tespit ve takibini sunucu yerine istemcide yapmamızın nedeni, sunucuya gönderilen verinin iletilip geri dönmesi arasında geçen süre yüzünden, görüntüde kaymalar ve gecikmelere sebep olabilmesidir. İstemcinin diğer görevi ise, API'a resim göndererek ondan donecek cevaba göre ekrana (GUI) gerekli bilgileri çizdirmektir. API cevap döndürdüktan sonra, istemci yeni bir resim göndererek yeni bilgiler isteyecektir. Böylelikle, ekrana gerçek zamanlı olan, sürekli güncellenerek değişen bilgiler çizilecektir.

1. Program Client'de başlatılır

- - Yaş, Cinsiyet, Yüz İfadesi, Hareket durumu alanları boştur
- - Client'in hangi modüller için çalışacağı ayarı başlangıçta (varsayılan) hepsidir.
- - Kullanıcı, çalıştırılmak istediği modülleri aktif/pasif edebilir

2. Client için ana akış sistemi başlatılır

- Client sürekli olarak durmadan real-time yüz aramasına başlar
- Client yüz tespit ettiği an, o anda aktif olan modüller için API'a Normal öncelikte istek yapar

3. Client, Server'a bilgileri gönderir

- İçereceği bilgiler: Frame, Modül, İşlenme Önceliği, İşleme Geçmiş, İşlenecek yerin konumu, boyutu, margin değerleri, Timestamp



Figure 4.4: Client Arayüz Taslağı

4.4.2 API

API, istemci ile yapay zeka modülleri arasında köprü görevi görecektir. İstemcide belirlediğimiz veri hızına (rate) göre, istemciden gelen resmi (frame) uygun modüle yönlendirecek ve ardından, modül tarafından gönderilen işlenmiş JSON verisini istemciye gönderecektir.

Listing 4.1: API Request

```
1 {
2   "requestID": "uuid",
3   "metadata": {
4     "width": 1600,
5     "height": 900,
6     "format": "jpeg"
7   },
8   "modules": [
9     { "type": "face", "priority": 1 },
10    { "type": "state", "priority": 2 },
11    { "type": "age", "priority": 3 },
12    { "type": "gender", "priority": 4 }
13  ],
14  "people": [
15    {
16      "id": "id",
17      "position": {
18        "x": 1,
19        "y": 1
20      },
21      "size": {
22        "width": 500,
23        "height": 500
24      },
25      "quaternion": {
26        "pitch": 1,
27        "roll": 1,
28        "yaw": 1
29      },
30    },
31    {
32      "id": "id",
33      "position": {
34        "x": 1,
35        "y": 1
36      },
37      "size": {
38        "width": 500,
39        "height": 500
40      },
41      "quaternion": {
42        "pitch": 1,
43        "roll": 1,
44        "yaw": 1
45      }
46    }
47  ]
48 }
```

Listing 4.2: API Response

```

1 {
2   "requestID": "uuid",
3   "metadata": {
4     "width": 1600,
5     "height": 900,
6     "format": "jpeg"
7   },
8   "frame": {
9     "isBW": false,
10    "color": {
11      "dominantForeground": "grey",
12      "dominantBackground": "black",
13      "dominantColors": [ "grey", "black" ],
14      "accentColor": "333333"
15    },
16    "blur": { "level": "normal", "value": 0.0 },
17    "exposure": { "level": "normal", "value": 0.0 },
18    "noise": { "level": "normal", "value": 0.0 }
19  },
20  "people": [
21    {
22      "id": "uuid",
23      "age": 1,
24      "gender": "male",
25      "face": {
26        "position": {
27          "x": 1,
28          "y": 1
29        },
30        "size": {
31          "width": 500,
32          "height": 500
33        },
34        "quaternion": {
35          "pitch": 1,
36          "roll": 1,
37          "yaw": 1
38        },
39        "attributes": {
40          "headHair": {
41            "bald": 0.1,
42            "invisible": false,
43            "colors": [
44              { "color": "brown", "confidence": 0.98 },
45              { "color": "black", "confidence": 0.45 }
46            ]
47          },
48          "facialHair": {
49            "moustache": 0.1,
50            "beard": 0.1,
51            "sideburns": 0.1
52          },
53          "emoticon": {
54            "anger": 0.0,
55            "contempt": 0.0,
56            "disgust": 0.0,
57            "fear": 0.0,
58            "happiness": 0.0,
59            "neutral": 0.0,
60            "sadness": 0.0,
61            "surprise": 0.0
62          },
63          "accessories": [
64            { "type": "glasses", "confidence": 1.0 }
65          ]
66        }
67      }
68    }

```

4.4.3 SUNUCU (SERVER)

Sunucu, projede tüm analizleri ve görüntü işlemelerini yapan yapay zeka modüllerinin bulunduğu sistemdir. API'ın resim gönderdiği modül asenkron olarak görüntüyü işler ve istemcinin anlayacağı şekilde bir JSON verisi oluşturur. Oluşturduğu bu veriyi tekrar API'a göndererek istemciye iletilmesini sağlar.

1. Her Microservice'de Pipeline mimarisi vardır

- Pipeline Mimarisi: Pre-Process, Process, Post-Process
- Her Microservice'nin sahip olduğu Pipeline mimarisi sayesinde, hiç bir işlem birbirini beklemez
- Pipeline sadece istenilen modül için özel olarak çalışır, diğer modüllerin aşamalarını etkilemez
- Örnek: 4 Çekirdek - 8 Thread bir işlemci: her çekirdek bir modül; her 2 thread, modüle ait bir pipeline
- Her Microservice'nin her bir Pipeline aşaması, KESİNLİKLE boş kalmamalıdır
- Frame'de 1 adet yüz ve 2 adet Yüz işleme Microservice'si varsa, birisi boş kalabilir
- Hiçbir Microservice'in (Modül) birbirleri ile iletişim yoktur ve birbirlerinden haberdar değildir

2. Queue tarafından iletilen Frame ve gerekli bilgiler Pre-Process aşamasına gönderilir

- Microservice Queue mimarisi sayesinde, Pre-Process çok uzun bile süurse, diğer modüller çalışmaya devam eder
- Her Pre-Process, modüle göre özel olarak farklı tekniklerle çalışır
- 1. Aşamada; Her modül için ortak çalıştırılacak teknikleri (Netleştirme, Contrast, Gamma, HDR, vb.) uygular
- 2. Aşamada; Her modüle özel uygulanması gereken teknikleri uygular
- Belirlenen zamanı aşmış ise, istenileni bulamazsa, Frame kötü ise vb. Frame drop edilir

3. Pre-Process aşamasında tamamlanan iş, Process aşamasına iletilir

- Pre-Process tarafından işleme aşamasına gönderilen Frame artık hazırır

- Process aşamasında Frame nasıl işlenmesi gerekiyorsa o şekilde işlenir ve bir JSON çıktısı (Process Time, Timestamp, ID) elde edilir
- İşlenmiş Frame, (varsayıncı önceki değerler) Post-Process'e gönderilir

4. Post-Process'e alınan Frame'ler kontrol edilir

- Process'in ürettiği değerleri kontrol eder ve Client'e gönderilip gönderilmeyeceğine karar verir
- Bu kararları yaparken bazı kurallara dikkat eder: İşlenmiş Frame'de istenilen şey bulunmuş mu, Yüz, Yaş, vb. gibi değerler için Tolerans aralığı (değerlerin doğruluğu), Yüz ve Yaş gibi kritik değerler bir çok kez işletilip, tolerans aralığından fazla olması sağlanır
- Post-Process söylemediği sürece, Microservice bu işin bittiğini bilemez
- Pre-Process'den Post-Process'e kadar işlem çok uzun sürüp Tolerans değerini aşmışsa, işlem Drop edilir
- Bütün işlemler tam yapılmış ve doğru ise bilgiler Client'e gönderilir
- Client, aldığı JSON verisine göre anlık olarak, yüzün konumuna göre bilgileri ekrana çizdirir
- Eğer Client'de yüz artık yoksa, o bilgiler Drop edilir
- Yeni kişi(ler) gelmişse, o kişi için istek yapmaya devam edilir ve iletişim akışı sağlanmış olur

4.4.4 VERİTABANI (DATABASE)

Veritabanı, modüller tarafından işlenmiş sonuçları kaydetmek için kullanılacaktır. Her işlenen veri için bir adet UUID tanımı yapılmıştır. Bu 36-bit UUID Primary (Birincil) anahtara göre her seferinde yeniden işlenen veriler karıştırılır. Eğer eşleşme olmasa durumda son bilgiler getirilir, eşleşme olmaması durumda ise yeni bir değer olarak girdilir kayıt edilir.

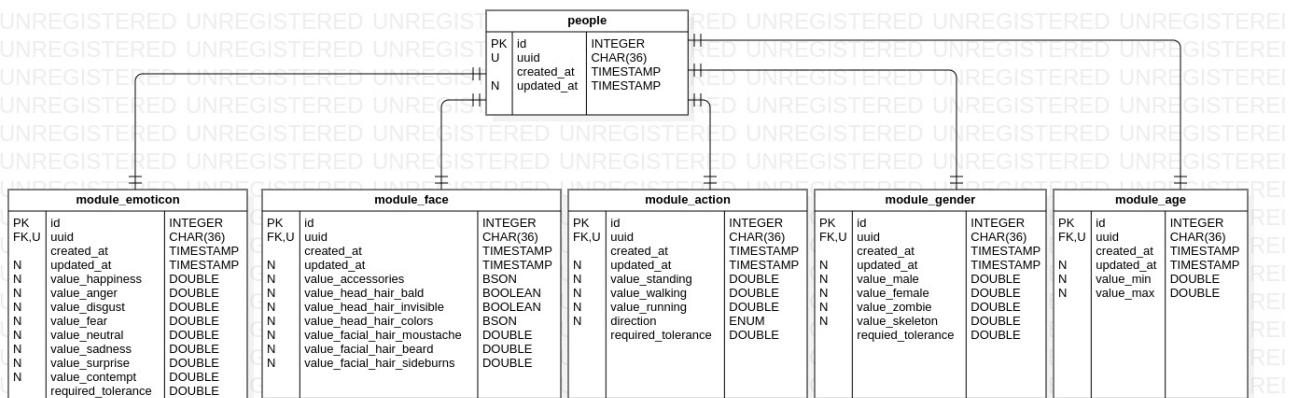


Figure 4.5: Veritabanı Diyagramı

4.4.5 SINIFLAR (CLASSES)

Sınıflar, projede en iyi şekilde kullanılabilmesi için OOP (Nesne-Yönelik-Programlama) mantığı izlenerek tasarlanmıştır.

1. Her Modül yapısı, Module.py Abstract sınıfından türer
2. Her Modüle ait 3 adet Process yapısı vardır
3. Her Process yapısı, Process.py Abstract sınıfından türer

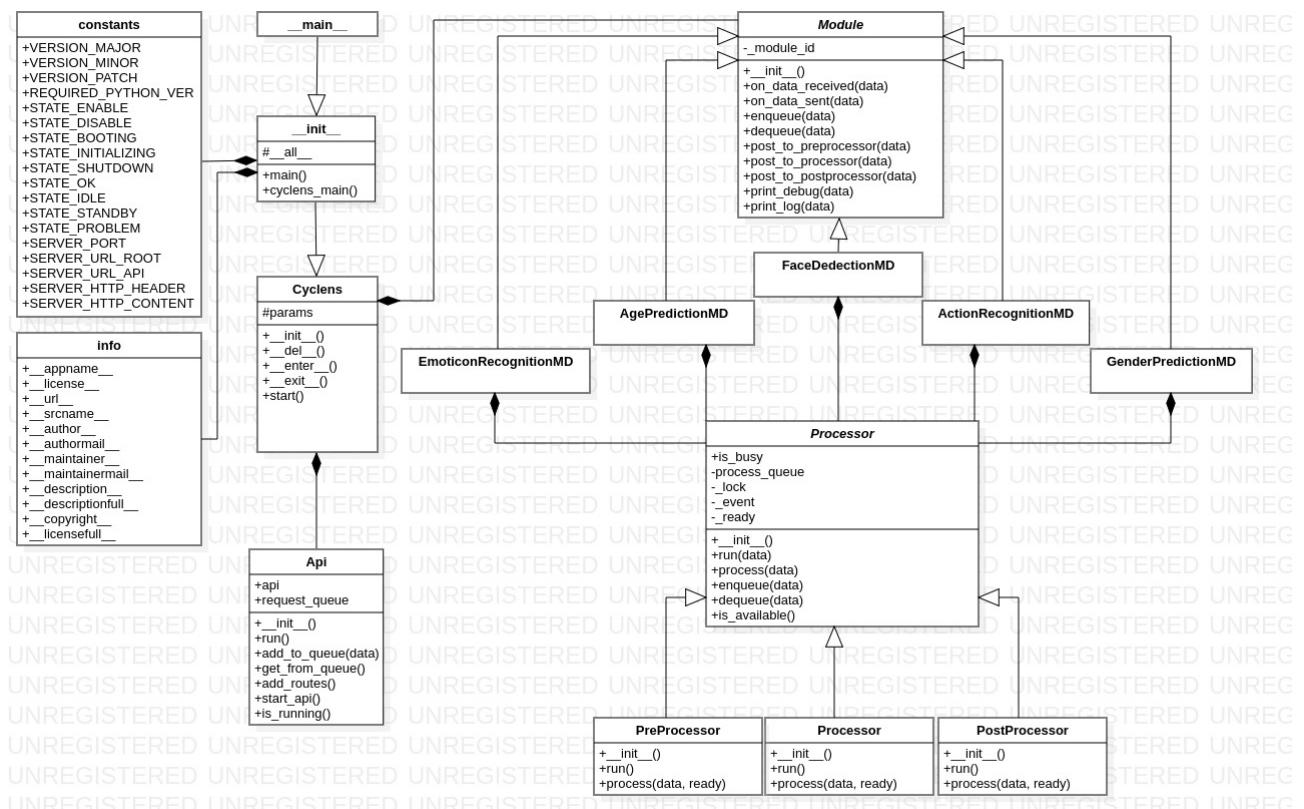


Figure 4.6: Sınıf Diyagramı

4.5 KULLANILACAK ARAÇLAR

Projede asıl yapılması gereken sistemi daha iyi simüle edebilmek için VR gözlük kullanacağız. Tasarlayacağımız sistemin gözümüz için en iyi deneyimi ve gerçekliği yaşatmasını beklediğimiz için proje arayüzünü Sanal Gerçeklik Gözlüğü ile görmeyi hedefliyoruz.

5

PROJE İŞ-ZAMAN PLANI

Projenin *İş-Zaman Planı* ([Figure 5.1](#))'na göre, projeye ilk olarak gereksinim analizlerinin tasarılanması ile başlanacaktır. Bize ihtiyaç olan gerekli kütüphanelerin çalışma prensiplerini öğrenerek ön bilgiye sahip olunması hedeflenmektedir. Kodlama aşamasına geçmeden önce, bize gerekli olan sistem diyagramlarını ve mimarisini hazırlamamız gerekmektedir.

İlk olarak Sunucuya göndereceğimiz gerçek-zamanlı (real-time) kamera verilerini işlememiz gerektiğinden, önce istemcinin (Client) yapımına başlayacağız. İstemci üzerinde kullanacağımız yazılımın planlamasını yaptıktan sonra kodlama aşamasına geçilip, son olarak durum raporunu almamız gerekiyor. Daha sonra Sunucu (Server) tarafında çalışmaya başlayacağız.

Bu aşamadan sonra projenin modüler sistem tasarımlarını ve iskelet mimarisini oluşturmamız gerekmektedir. Modüler sistemi ve mimari diyagramını tasarladıkten sonra, her module uygun olan gereksinim analizleri hazırlanıp, kodlama aşamasına geçilecektir. Kodlama aşamasından sonra ise Birim Testleri (Unit Test) ve değerlendirme raporu hazırlanacak, daha sonra tekrar yapılandırma aşamasına geçilecektir. Bütün testler tamamlandıktan sonra her modül için sonuç raporu hazırlanacak ve bir sonraki modülün yapımına geçilecektir. Projede yapacağımız her modül için aynı işlemler tekrar devam edecektir.

Sunucu ve istemci taraflı bütün modüller ve tasarımlar yapıldıktan sonra, modül servis testleri ve sonuç raporu hazırlanacaktır.

GÖREV AŞAMALARI	BAŞLANGIÇ TARİHİ	BİTİŞ TARİHİ	SÜREÇ (GÜN)	ELYÜL	EKİM	KASIM	ARALIK	OCAK	ŞUBAT	MART	NİSAN	MAYIS	HAZİRAN
				2018	2018	2018	2018	2019	2019	2019	2019	2019	2019
PLANLAMA													
Proje Planının Oluşturulması	09/23/18	09/30/18	7										
ANALİZ													
Gereksiniin Analizi	10/01/18	10/31/18	30										
Alt Sistem Modüllerinin Analizi	10/07/18	10/22/18	15										
İstemci ve Sunucu Analizi	10/22/18	10/31/18	9										
TASARIM													
Sistem Tasarımının Hazırlanması	11/01/18	12/25/18	54										
Program Tasarımının Hazırlanması	12/01/18	12/16/18	15										
Tasarım Raporunun Hazırlanması	12/16/18	12/23/18	7										
KODLAMA													
İstemci (Kamera) Kodlanması	12/24/18	01/31/19	37										
Yapay Zeka Modüllerinin Kodlanması	01/01/19	05/20/19	139										
Sunucu Modüllerinin Birleştirilmesi	03/10/19	05/25/19	75										
TEST													
Unit (Birim) Test	02/01/19	05/20/19	109										
Entegrasyon ve Sistem Testi	04/01/19	05/25/19	54										
Kabul Testi	05/25/19	06/30/19	35										
SONUÇ													
Proje Sonuçlarının Değerlendirilmesi	07/01/19	07/07/19	6										

Figure 5.1: Proje İş-Zaman Planı Diyagramı

5.1 PROJE TARİHLERİ

Proje Başlangıç Tarihi Eylül 2018
 Proje Bitiş Tarihi Temmuz 2019

6

TESTLER

Tasarladığımız projenin her durumda doğru çalıştığını ve hatalı ve test edilmemiş fonksiyon sayısını iyice azaltmak ve ortadan kaldırmak için *Birim (Unit) Testlerine* ihtiyacımız olacaktır.

6.0.1 BİRİM (UNIT) TESTLERİ

Yaptığımız birim testlerindeki asıl amaç; modüllerin döndürdüğü sonuçları değil, fonksiyonların doğru çalıştığını kontrol etmektir. Test ettiğimiz fonksiyonlar ve sonuçları;

MODULE-AP-CASCADE-NULL-DEGIL	PASSED
MODULE-ER-CASCADE-NULL-DEGIL	PASSED
MODULE-GP-CASCADE-NULL-DEGIL	PASSED
MODULE-ER-EMOTION-SAYISI-7	PASSED
MODULE-GP-GENDER-SAYISI-2	PASSED
MODULE-AP-TOPLAM-YUZ-1-VE-BASARILI	PASSED
MODULE-ER-TOPLAM-YUZ-1-VE-BASARILI	PASSED
MODULE-GP-TOPLAM-YUZ-1-VE-BASARILI	PASSED
PREPROCESSOR-YUKLE	PASSED
PREPROCESSOR-CASCADE-KONTROL	PASSED
PREPROCESSOR-NULL-KONTROL	PASSED
PREPROCESSOR-DATA-KONTROL	PASSED
PREPROCESSOR-YUZ-KONTROL	PASSED
PREPROCESSOR-COZUNURLUK-KONTROL	PASSED
POSTPROCESSOR-YUKLE	PASSED
POSTPROCESSOR-NULL-KONTROL	PASSED
POSTPROCESSOR-DATA-KONTROL	PASSED
POSTPROCESSOR-RESULT-KONTROL	PASSED
POSTPROCESSOR-KULLANILMAYANLARI-KALDIR	PASSED

6.1 BENCHMARK TESTLERİ

Tasarladığımız projede, sürümlere göre yaptığımız *Benchmark Testleri*, projenin hızını ve verimliliğini izlemek için önemli bir rol oynamaktadır. Aşağıda yaptığımız sürümlere göre sonuçlar verilmiştir.

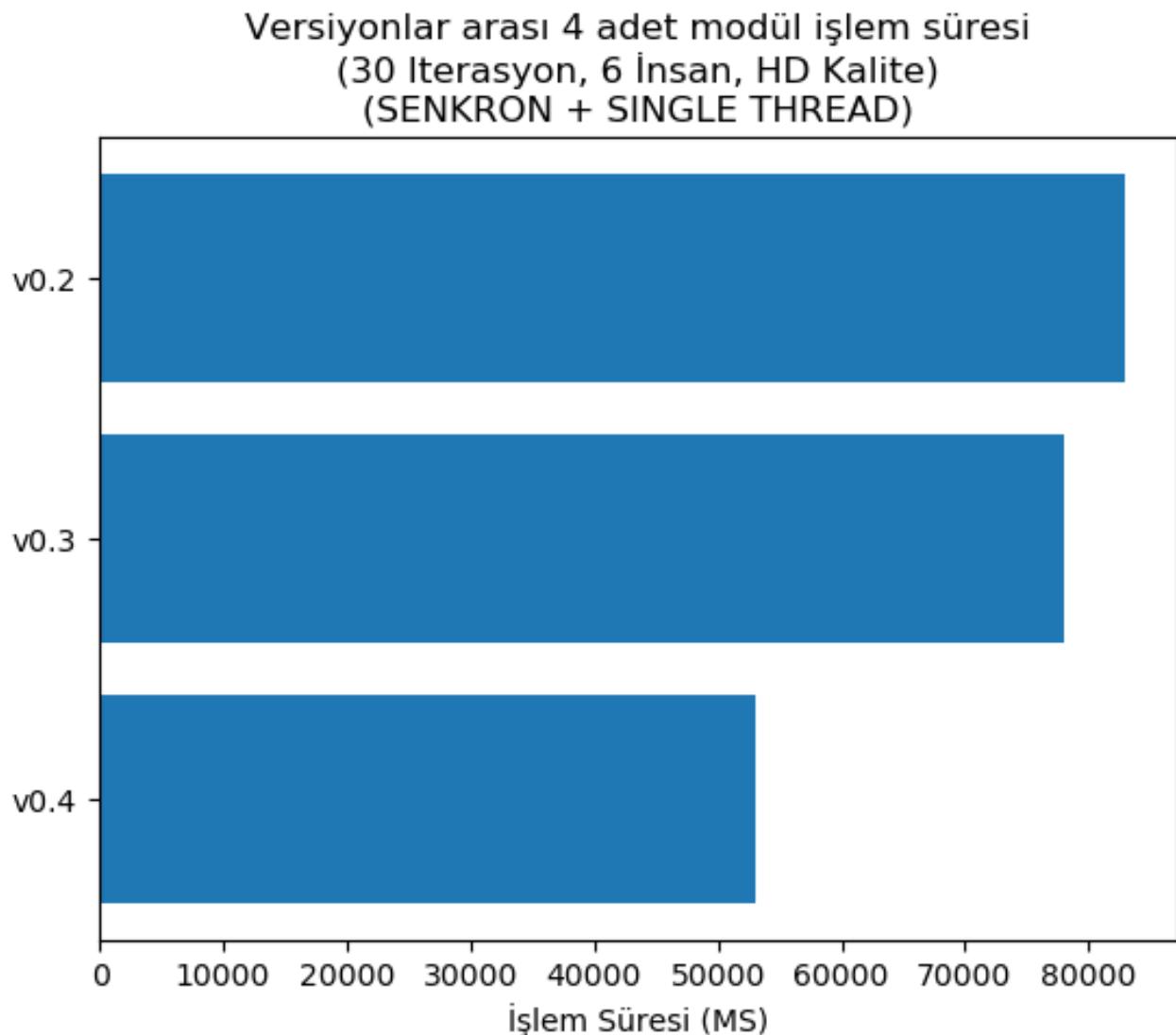


Figure 6.1: SENKRON + SINGLE THREAD Testi

Versiyonlar arası 4 adet modül işlem süresi
(30 Iterasyon, 6 İnsan, HD Kalite)
(ASENKRON + MULTI PROCESS)

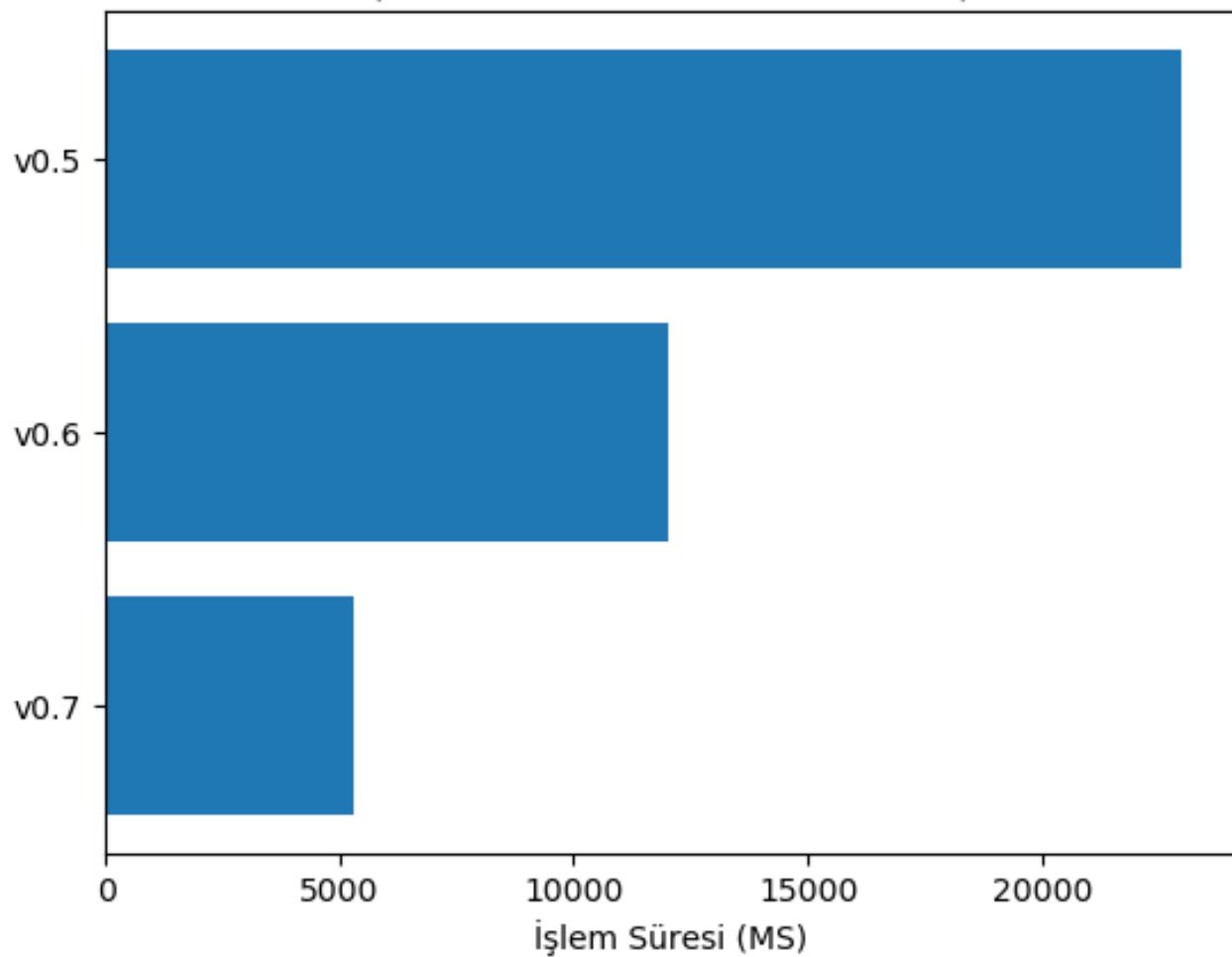


Figure 6.2: ASENKRON + MULTI PROCESS Testi

Versiyonlar arası 4 adet modül gerçek zamanlı FPS değeri
(Webcam Kalite)

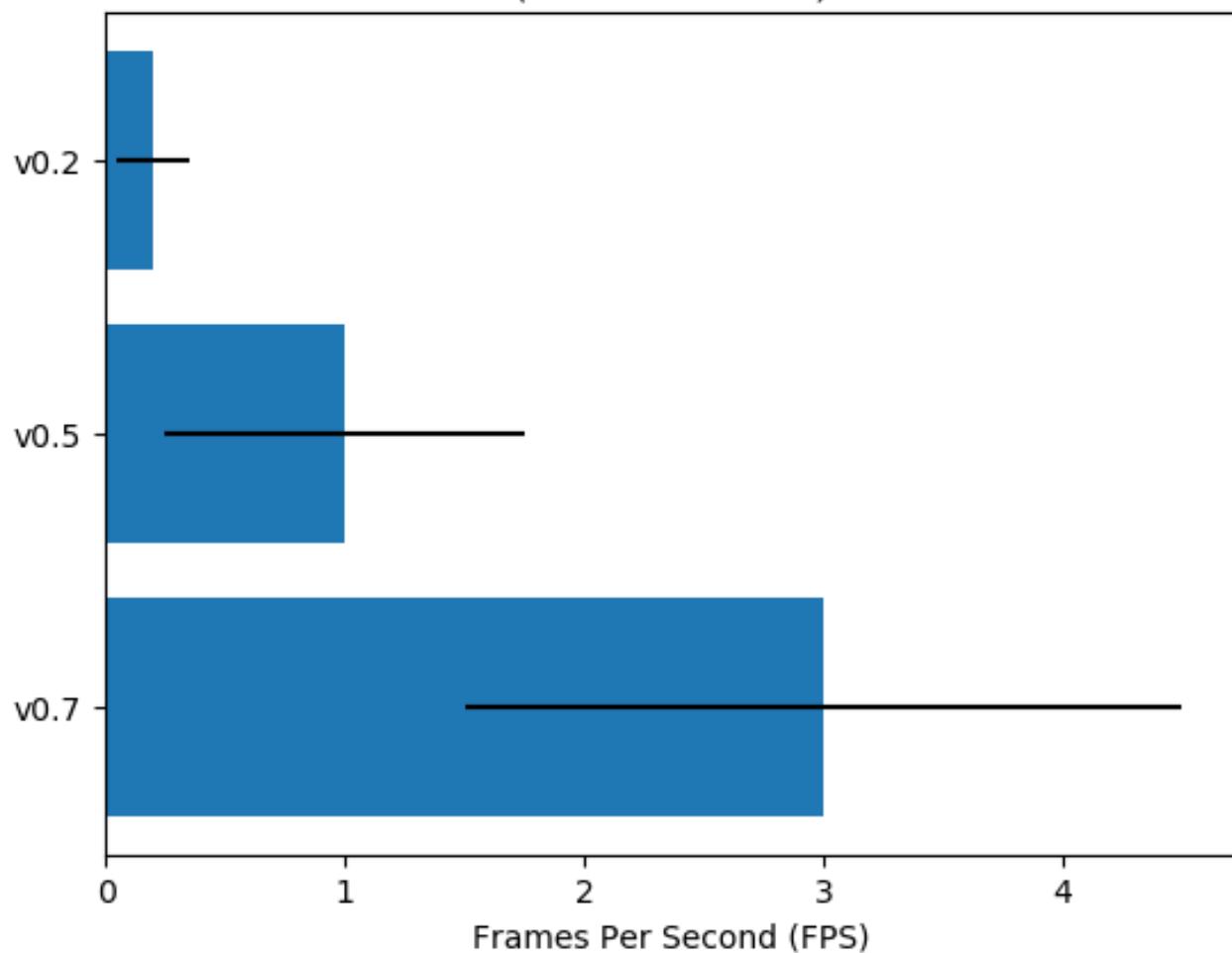


Figure 6.3: Gerçek Zamanlı Webcam Testi

SONUÇ

Tasarladığımız bu proje sayesinde, insanlar üzerinde Yapay Zeka ve Görüntü İşleme metotları ile birçok analiz ve sınıflandırma işlemi yapabiliriz. Açık kaynak olarak geliştireceğimiz projeye birçok kişi katkıda bulunabilir ve isteyen herkes bu projeye destek olup, yeni özelliklerin eklenmesinde yardımcı olabilir. Modüler yapıdaki servisler sayesinde birçok farklı türde yeni projeler geliştirilebilir ve yeni fikirlerin kapılarını aralayabiliriz. Hedefler doğrultusunda Akıllı Lens'ler için tasarladığımız bu yazılım mimarisi, hem genel, hem ticari, hem askeri, hem de özel alanda kullanım yerlerine sahip olabilir.

Bibliography

- [1] http://www8.cao.go.jp/cstp/english/society5_0/index.html
- [2] <https://www.nanalyze.com/2017/03/smart-contact-lenses/>
- [3] https://en.wikipedia.org/wiki/Science_fiction_film
- [4] https://en.wikipedia.org/wiki/Facial_recognition_system
- [5] <http://www.gazete32.com.tr/isparta/sdu/insanligin-degisimi-simdi-basliyor.html>
- [6] <https://wiki.debian.org/systemd>
- [7] <https://en.wikipedia.org/wiki/TensorFlow>
- [8] <https://keras.io/>
- [9] <https://opencv.org/about.html>
- [10] <http://deeplearning.net/software/theano/introduction.html>
- [11] <https://golang.org/project/>
- [12] <https://facebook.github.io/react-native/>
- [13] <https://www.youtube.com/watch?v=N15PczCiN04>
- [14] <https://www.youtube.com/watch?v=C7YycopxoKSw>
- [15] <https://www.youtube.com/watch?v=JWLti-oVwuw>
- [16] <https://www.nanalyze.com/2017/03/smart-contact-lenses/>
- [17] <https://developer.nvidia.com/cudnn>
- [18] <https://www.geforce.com/hardware/technology/cuda>
- [19] https://github.com/tensorflow/hub/blob/master/examples/image_retraining/retrain.py

- [20] <https://github.com/tensorflow/models/tree/master/research/inception>
- [21] <https://www.tensorflow.org/guide/performance/benchmarks>
- [22] <https://keras.io/models/about-keras-models/>
- [23] <http://flask.pocoo.org/>
- [24] <http://flask.pocoo.org/docs/1.0/quickstart/#routing>
- [25] <http://www.tornadoweb.org/en/stable/>
- [26] <https://stackoverflow.com/questions/13163990/why-use-tornado-and-flask-together>
- [27] <https://stackoverflow.com/questions/8143141/using-flask-and-tornado-together>
- [28] <https://en.wikipedia.org/wiki/NumPy>