T.C.
ERCİYES ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
ELEKTRİK ELEKTRONİK MÜHENDİSLİĞİ
ANABİLİM DALI

GÖRÜNTÜ İŞLEME YÖNTEMLERİNİ KULLANARAK KİMYASAL İDRAR GÖRÜNTÜLERİNİN İNCELENMESİ

Hazırlayan
Ahmet TAŞKIRAN

Danışman
Doç. Dr. Enis GÜNAY

Yüksek Lisans Tezi

Temmuz 2019
KAYSERİ

T.C.
ERCİYES ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
ELEKTRİK ELEKTRONİK MÜHENDİSLİĞİ
ANABİLİM DALI

GÖRÜNTÜ İŞLEME YÖNTEMLERİNİ KULLANARAK KİMYASAL İDRAR GÖRÜNTÜLERİNİN İNCELENMESİ

(Yüksek Lisans Tezi)

Hazırlayan
Ahmet TAŞKIRAN

Danışman
Doç. Dr. Enis GÜNAY

Temmuz 2019
KAYSERİ

BİLİMSEL ETİĞE UYGUNLUK

Bu çalışmadaki tüm bilgilerin, akademik ve etik kurallara uygun bir şekilde elde edildiğini beyan ederim. Aynı zamanda bu kural ve davranışların gerektirdiği gibi, bu çalışmanın özünde olmayan tüm materyal ve sonuçları tam olarak aktardığımı ve referans gösterdiğimi belirtirim.

Ahmet TAŞKIRAN

"Görüntü İşleme Yöntemlerini Kullanarak Kimyasal İdrar Görüntülerinin İncelenmesi" adlı Yüksek Lisans tezi, Erciyes Üniversitesi Lisansüstü Tez Önerisi ve Tez Yazma Yönergesi' ne uygun olarak hazırlanmıştır.

Hazırlayan

Ahmet TAŞKIRAN

Danışman

Doç. Dr. Enis GÜNAY

Elektrik Elektronik Mühendisliği ABD Başkanı

Prof. Dr. Necmi TAŞPINAR

Doç. Dr. Enis GÜNAY danışmanlığında **Ahmet TAŞKIRAN** tarafından hazırlanan **"Görüntü İşleme Yöntemlerini Kullanarak Kimyasal İdrar Görüntülerinin İncelenmesi"** adlı bu çalışma jürimiz tarafından Erciyes Üniversitesi Fen Bilimleri Enstitüsü **Elektrik-Elektronik Mühendisliği** Anabilim Dalında **yüksek lisans** tezi olarak kabul edilmiştir.

05 / 08 / 2019

JÜRİ:

Danışman : Doç. Dr. Enis GÜNAY .....

Üye : Doç. Dr. Ahmet Turan ÖZDEMİR .....

Üye : Dr. Öğr. Üyesi Ahmet DOĞAN .....

ONAY:

Bu tezin kabulü Enstitü Yönetim Kurulunun 20/08/2019. tarih ve 2019/48-10 sayılı kararı ile onaylanmıştır.

20/08/2019

Prof. Dr. Mehmet AKKURT

Enstitü Müdürü

TEŞEKKÜR

Bana çalışmalarım süresince her türlü yardımı ve fedakârlığı sağlayan, danışmanım Doç. Dr. Enis GÜNAY' a teşekkürlerimi borç bilirim.

Ahmet TAŞKIRAN

Temmuz 2019, KAYSERİ

GÖRÜNTÜ İŞLEME YÖNTEMLERİNİ KULLANARAK KİMYASAL İDRAR GÖRÜNTÜLERİNİN İNCELENMESİ

Ahmet TAŞKIRAN

Erciyes Üniversitesi, Fen Bilimleri Enstitüsü
Yüksek Lisans Tezi, Temmuz 2019
Danışman: Doç. Dr. Enis GÜNAY

ÖZET

Bu tez çalışmasında idrar analiz yöntemlerinden kimyasal idrar analizi konusunda incelemeler yapılmıştır. Çalışma kimyasal idrar analiz yöntemlerinden birisi olan strip yöntemini temel almaktadır. Günümüzde kullanılan strip inceleme yöntemlerinden farklı olarak görüntü işleme tabanlı bir yaklaşım geliştirilmiştir. Bu yaklaşımın yazılım tabanlı olması, algoritmik geliştirme kolaylığı, detaylı sonuç üretebilme yeteneği gibi birçok avantajıyla ön plana çıkacağı öngörülmektedir. Strip yöntemle elde edilen sonuçlar görüntülenerek sayısal verilere dönüştürülmüştür. Bu görüntüler görüntü işleme algoritmalarıyla düzenlenmiştir ve optimizasyon sürecinde Hücresel Sinir Ağları yaklaşımından faydalanılmıştır. Elde edilen görüntü veri seti bir uzman yardımıyla incelenmiş ve etiketlenmiştir. Etiketlenen görüntüler kümeleme algoritmaları ve derin öğrenme yaklaşımları kullanılarak sınıflandırılmıştır. Sonuçlar tablolar halinde karşılaştırılmıştır. Kümeleme algoritmalarının bu çalışmada yetersiz kaldığı ve derin öğrenme algoritmalarına olan ihtiyaç ortaya konulmuştur.

Anahtar Kelimeler: İdrar Analizi, Hücresel Sinir Ağları, Yapay Sinir Ağları, Kümeleme, Derin Öğrenme,

INVESTIGATION OF CHEMICAL URINE IMAGES USING IMAGE PROCESSING METHODS

Ahmet TAŞKIRAN

Erciyes University, Graduate School of Natural and Applied Sciences
Master Thesis, July 2019
Supervisor: Assoc. Prof. Dr. Enis GÜNAY

ABSTRACT

In this thesis, chemical urine analysis, one of the urine analysis methods, has been examined. The study is based on the strip method which is one of the chemical urine analysis methods. Unlike the strip analysis methods used today, an image processing based approach has been developed. It is foreseen that this approach will come to the forefront with many advantages such as being software based, ease of algorithmic development, and ability to produce detailed results. The results obtained by strip method were displayed and converted into numerical data. These images are arranged with image processing algorithms and the optimization process is based on the Cellular Neural Networks approach. The obtained image data set was examined and labeled with the help of an expert. The tagged images are classified using clustering algorithms and deep learning approaches. The results were compared in tables. Clustering algorithms were insufficient in this study and the need for deep learning algorithms was revealed.

Keywords: Urine Analysis, Cellular Neural Networks, Artificial Neural Networks, Clustering, Deep Learning

İÇİNDEKİLER

GÖRÜNTÜ İŞLEME YÖNTEMLERİNİ KULLANARAK KİMYASAL İDRAR GÖRÜNTÜLERİNİN İNCELENMESİ

5. BÖLÜM

DERİN ÖĞRENME

6. BÖLÜM

TEST ÇALIŞMASI ve PRATIK UYGULAMA

7. BÖLÜM

TARTIŞMA, SONUÇ ve ÖNERİLER

KISALTMALAR

CNN : Konvolüsyonel Sinir Ağı

YSA : Yapay Sinir Ağı

HSA : Hücresel Sinir Ağı

MLP : Çok Katmanlı Algılayıcılar

TBK : Tam Bağımlı Katman

POOL : Pooling Katmanı

RNN : Tekrarlayan Sinir Ağları

RBM : Sınırlı Boltzmann Makineleri

DBN : Derin İnanç Ağları

DSN : Derin Yığın Ağları

TABLOLAR LİSTESİ

ŞEKİLLER LİSTESİ

GİRİŞ

Akademik ve teknolojik gelişmelerin en büyük ilham kaynağı kuşkusuz doğadır. Geçmişten günümüze birçok çalışma, merak içerisinde olan insanın çevresini analiz etmesi ve araştırmasıyla ortaya çıkmıştır. Yine insanın kendini tanımaya olan merakı, yaşamını daha kaliteli hale dönüştürmekte ve teknolojiye yön vermektedir. İnsan fizyolojisi henüz tam olarak çözümlenememiş yapısıyla merak uyandırmakta ve derin araştırmalara konu olmaya devam etmektedir.

İnsan vücudunda oluşabilecek hastalıkların teşhisi yani klinik tanıda patolojik durumların belirlenmesi hayati bir öneme sahip olmaktadır. Bu amaçla kullanılan en yaygın tespit yöntemlerinden birisi de idrar analizidir. İdrar aracılığı ile vücuttan atılan maddelerin oranlarındaki artış ve azalışlar, normalde idrarda bulunmaması gereken patolojik olguların tespiti, tedavinin etkinliğinin veya komplikasyonlarının belirlenmesi rutin idrar analizi sayesinde tespit edilmekte ve böylece hastalıkların tanı ve tedavisinde etkili bir şekilde kullanılmaktadır [1-3].

Rutin idrar analizi fiziksel, kimyasal ve mikroskobik olmak üzere 3 farklı analiz yönteminin birleşimiyle gerçekleştirilir. Bu çalışmada idrarın kimyasal analizi ele alınacaktır. Bu kapsamda en yaygın olarak kullanılan idrar striplerinin idrara daldırılması ve ya strip üzerine idrar damlatılmasıyla gerçekleşen tepkime sonucu oluşan renk değişimleri tespit edilecektir. Günümüzde bu renk değişimleri laborantlar tarafından manuel olarak ve ya optik okuyucular vasıtasıyla değerlendirilmektedir.

Sunduğumuz çözüm tekniği striplerin görüntü işleme algoritmalarıyla değerlendirilmesine yöneliktir. Böylece analiz detayı artırılabilecek, elde edilen dijital veri ileride tekrar kullanılmak üzere saklanabilecek, değerlendirme işlemini yapan doktora görsel bir sonuç verilebilecek, taşınabilir bir sistem tasarlanarak esneklik sağlanacak, uzaktan erişim özelliği sayesinde hasta evde analiz yapıp doktoruna bilgi aktarabilecektir.

Tezin birinci bölümünde; yapılacak çalışma hakkında genel bilgiler verilecek ve geçmişte yapılan akademik çalışmalar incelenecektir.

Tezin ikinci bölümünde; idrar analizi hakkında bilgiler verilecek, idrar analiz yöntemleri açıklanacaktır. İdrar analiz yöntemlerinden birisi olan ve bu tez çalışmasında incelenecek olan kimyasal idrar analizi üzerine açıklamalar yapılacaktır.

Tezin üçüncü bölümünde; klasik görüntü işleme yöntemleri anlatılacaktır.

Tezin dördüncü bölümünde; yapay sinir ağları, hücresel sinir ağları ve konvolüsyonel sinir ağları hakkında bilgiler verilecektir.

Tezin beşinci bölümünde; derin öğrenme algoritmaları açıklanacaktır ve tez çalışmasında kullanılan AlexNet mimarisi detaylı şekilde anlatılacaktır.

Tezin altıncı bölümünde; yapılan tüm çalışma ve uygulanan yöntemler açıklanacak, geliştirilen pratik uygulama hakkında bilgiler verilecektir.

Tezin yedinci bölümünde; tartışma, sonuç ve öneriler kısımlarına yer verilecektir.

1. BÖLÜM

GENEL BİLGİLER ve LİTERATÜR ÇALIŞMASI

1.1. Konunun Belirlenmesi

Bu tez çalışmasında, görüntü işleme teknikleri kullanılarak kimyasal idrar analizi sonuçlarının üzerinde araştırmalar yapılacaktır. Bu incelemelerde idrar içerisinde bulunan partiküllerin tespit işlemi bir renk skalasının araştırılması problemi olarak ele alınacaktır. Böylece idrarın kimyasal analizi, renk tespiti tekniklerinin kullanıldığı görüntü işleme algoritmalarından faydalanılarak yapılacaktır.

Bu çalışmada hastalık tanısında yardımcı bir analiz olan idrar analizinin, kimyasal idrar analizi kapsamında faaliyetlerin yürütülmesi planlanmaktadır. Kimyasal idrar analizi için kullanılan ve farklı kimyasalları bulunduran test çubukları (strip) üzerine idrar damlatılarak ve ya çubukların idrara daldırılarak incelemesi alanında daha başarılı partikül tespiti yapılması amaçlanmaktadır. Bu kapsamda klasik görüntü işleme teknikleri, sınıflandırma/kümeleme algoritmaları, Hücresel Sinir Ağı (HSA) ve Derin Sinir Ağı yaklaşımları kullanılacaktır. Çalışmanın endüstriyel bir cihaza dönüştürülebilmesi için Arm tabanlı işlemci mimarisine sahip olan Raspberry Pi kartı üzerinde de analizler yapılacaktır.

1.2. Araştırmanın Yöntemi

Bu tez çalışmasında kimyasal idrar analiz yöntemleri incelenecek ve bunlardan strip yöntemi detaylandırılacaktır. Strip yönteminin analiz teknikleri incelenip, uygulanacak olan farklı tekniklerle elde edilecek sonuçlar kıyaslanacaktır.

Uygulamada takip edilecek işlem adımları aşağıdaki gibidir:

- Kullanılmamış ve pozitif değerlere sahip olan strip örnekleri temin edilecektir.

- Kullanılmamış stripler kalibrasyon yazılımının geliştirilmesi için kullanılacaktır.
- Pozitif striplerden renk sahaları ayrıştırılarak görüntü veri seti oluşturulacaktır.
- Elde edilen veri seti farklı algoritmalarla değerlendirilerek görüntülerin sınıflandırılması ve ya değerlendirilmesi sağlanacaktır.
- Bu sonuçlarının nicel değerlere dönüştürülebilmesi için gerekli dönüşüm tablosu ve yazılım için gerekli parametreler hesaplanacaktır.
- Sonuçların tutarlı olup olmadığının kontrolü için referans bir analiz cihazı ve ya birkaç cihaz kullanılarak sistemin kararlılık analizi yapılacaktır.
- Sonuçlar, Tartışma ve Öneriler verilecektir.

1.3. Literatür Taraması

1880'li yılların başlarında bazı eczacılar ve bu alanda çalışan diğer uygulayıcılar karmaşık işlem adımlarına sahip bazı kimyasal prosedürleri günümüzdeki 'strip' yani 'kuru kimya' tekniğine dönüştürmeye çalıştılar. Bu teknik test kâğıdı veya test şeridi tekniği olarak da adlandırılabilmektedir.

Şeker ve albumin için ilk popüler test kağıdı 1883'te İngiltere'de ortaya çıkmıştır[4]. Bu yüzyılın başlarından beri idrarda kan bulunması durumunu tespit etmek için geliştirilmiş kuru teknikler de mevcuttu.

1930'lara kadar strip tekniğini 'modern' teknik olarak sunan birçok ticari kuruluş ortaya çıktı. Böylece yeni bir analitik kimya alanı oluşmuş oldu. İkinci Dünya Savaşı döneminde Avrupa'daki bu gelişim durdu ve 1956'dan sonra Amerikan endüstrisi test kağıtlarının geliştirilmesi ve pazarlanması konusunda lider konuma geldi. Bu konudaki lider firma "Clinistix" (Ames Company, bugünkü Bayer Diagnostic) oldu [5].

Strip teknolojisinin gelişmesi stripleri otomatik olarak değerlendirecek cihazların gereksinimini ortaya çıkarmıştır. Bu sebeple strip okumak için yarı otomatik, tam otomatik cihazlar ortaya çıkmıştır. İlerleyen yıllarda da hem kimyasal hem de mikroskobik idrar analizini birlikte yapabilen tam idrar analiz cihazları piyasaya sürülmüştür [6-9].

Firma ve strip çeşitliliğinin artmasıyla test şeritleri arasında kıyaslama çalışmaları literatürde daha çok yer almaya başlamıştır.

1987 yapılan bir çalışmada idrarda lökosit aktivitesinin saptanması için Ames firmasının geliştirdiği Leukostix test şeritlerinin değerlendirmesi için Bio-Dynamics firmasının Chemstrip LN test sonuçları ile karşılaştırma yapılmıştır. Bu çalışmada 412 idrar analiz edilmiştir ve sonuç olarak kararlı ve tekrarlanabilir veriler elde edilmiştir. Böylece firmanın bu ürününün mikroskobik incelemeye uygun bir alternatif olduğu sonucu çıkmıştır [10].

1994 deki başka bir çalışmada kırmızı kan hücrelerinin ve lökositlerin saptanması için 5486 idrar, otomatik bir strip tekniği ile analiz edilmiştir. 3127 idrar çubuk üzerinde tamamen olumsuz ve 2359 olumlu bir bulgu göstermiştir. Bu çalışmada iki görevli 2359 pozitif ve 456 negatif idrarı farklı günlerde mikroskobik olarak analiz etmiştir. Stripte negatif olan idrarlar mikroskobik analizle örtüşen sonuçlar vermiştir. Sonuçta stripte pozitif olan idrarlar için kesin yargıya ulaşabilmek adına mikroskobik analizin gerektiği belirtilmiştir [11].

Günümüzde strip okuyucu cihazların giderek küçülmesi ve mobil sağlık çalışmalarındaki ihtiyacın artması bu alanda yapılması gereken güncel çalışmaların gerekliliğini ortaya koymaktadır. 2010 yılında Kore'de yapılan bir çalışmada cep boyutunda bir kolorimetrik okuyucuyu (13.5 × 6.5 × 2.5 cm3) ve piyasada bulunan 10 parametreli kağıt şeritleri (glukoz, protein, glukoz, bilirubin, ürobilinojen, ketonlar, nitrit, pH) birleştiren yeni bir mobil sağlık platformu geliştirilmiştir. Bu sistemde kalorimetrik okuyucu bir akıllı telefonla haberleşerek verileri internet ortamına açabilmektedir. Geliştirilen okuyucu, üç renkli ışık yayan diyotlar, silikon fotodiyotlar ve yeni bir poli (metil metakrilat) (PMMA) optik ayırıcıdan oluşan yeni bir kolorimetrik çoklu algılama modülü içerir. Sinyal verilerinin (kırmızı, mavi ve yeşil) dönüşümlerini ton (H) renk haritasına veya Y model verisine dönüştürerek veri okuma yöntemlerini kullanır. Sonuçları sayısal verilere dönüştürmek için de eğri uydurma yöntemini kullanır. Okuyucu, pille çalışan, ucuz, hafif ve analizde çok hızlıdır. Binlerce insan idrar örneklerinde uygulanmıştır ve güvenilir şekilde üriner glukoz ve protein miktarının tespiti sağlanmıştır [12].

Strip analizi konusunda yapılan başka bir çalışma ise striplerin akıllı telefonlarla incelenmesine yöneliktir. Uzaktan sağlık izleme sistemleri için geliştirilen bu yöntem oldukça pratik bir yaklaşım sunmaktadır. Bu çalışmalarla evde kan ve kimyasal idrar testi yapmak mümkün hale gelmektedir. Strip analizi konusunda yapılan çalışmada akıllı telefon için geliştirilen bir uygulama sayesinde şeritlerin fotoğrafı çekilmekte ve görüntü işleme algoritmaları uygulanarak sonuç tespit edilmektedir. Burada fotoğrafın çekildiği ortam ışığı, fotoğrafı çeken kişi ve telefonun kamera kalitesi sonucu etkileyen etkenler arasındadır. Bu nedenlerle uygulama erken teşhisi kolaylaştırabilecek bir mobil platform olarak ortaya çıkmıştır [13]. Sistemi geliştiren firma bu yöntemi patentlemiştir [14].

Bu konudaki başka bir çalışma da hastanelerin acil departmanları için geliştirilmiştir. İdrarda bulunan lökosit ve nitritin hızlı ve doğru bir şekilde taranması için akıllı telefon tabanlı bir idrar şerit okuyucu geliştirilmiştir. Geliştirilen okuyucu 81 idrar örneği ile değerlendirilmiştir. Bu çalışmada okuyucunun lökosit ve nitrit için algılama performansını ve güvenilirliğini değerlendirmek üzere bir fotometrik analiz cihazı (US-3100R Plus, Eiken Chemical, Ltd., Tokyo, Japonya) referans olarak kullanılmıştır [15].

Bu tez çalışması kapsamında geliştirilecek yöntemle reflaktans fotometri tekniğine görüntü işleme methoduyla çalışan bir alternetif sunulması amaçlanmaktadır. Aynı zamanda görüntü işleme teknikleriyle çalışan sistemlerdeki; kamera farkı, fotoğraf kalitesi, ışık, açı gibi problemlerinin de önüne geçilecektir. Bilgisayar ortamında geliştirilen algoritmalar tam idrar analiz cihazlarına entegre edilerek sistem maliyetlerinin azaltılması da bu çalışmanın pratik uygulamalardaki katkısı olarak ortaya çıkacaktır.

2. BÖLÜM

İDRAR ANALİZİ

İnsan vücudunun her iki yanında ve kaburgaların altında konumlanan böbrekler; kanın filtrelenmesi sağlayarak vücuttaki su miktarının düzenlenmesi ve sodyum potasyum miktarının dengelenmesi işini gerçekleştirmektedir. Böbreklerin bu filtrasyon sonrası oluşturduğu su, üre, keratin gibi vücutta fazla bulunan gereksiz tüm metabolik atıkları içeren sıvı idrar olarak adlandırılmaktadır. Diğer bir tanıma göre idrar, böbreklerden ürinasyon işlemiyle salgılanan ve vücudun yan ürünü olan bir sıvıdır. Üretra vasıtasıyla vücuttan atılmaktadır [16]. Sağlıklı bir insan idrarı tamamen sterildir. Yetişkin bir birey günlük ortalama 600-1800ml idrar çıkarır. İnsanoğlu anne karnındaki yaşam döngüsünde kendi idrarı içerisinde yüzer.

Sağlıklı bir idrar genellikle sarı renkli ve açık duru halde bulunur. Ancak idrarın içerisinde bulunan etkenlere göre rengi, miktarı, konsantrasyonu ve içeriği değişebilir. Bu değişimler glukoz, protein, bilirubin, eritrosit, lökosit, kristaller ve bakteri gibi bileşenlerin anlamlı derecede artışı ve ya azalışı ile olur. İdrarda bulunan bu anormal miktarlardaki bileşenler böbrek ve idrar yolu enfeksiyonlarının tanısında ve incelenmesinde önemli rol oynamaktadır. İdrar yollarında meydana gelen bir enfeksiyon, üreme organlarını da yakından ilgilendirmektedir [1-3].

İnsan vücudunda oluşabilecek hastalıkların teşhisi hayati bir öneme sahiptir. İdrar metabolik bir atık olması ve tıbbi bir operasyona gerek duymadan kolay bir şekilde elde edilebilmesi nedenleriyle diğer analiz yöntemlerine göre idrar analizi daha basit ve zararsızdır. Bu amaçla kullanılan en yaygın tespit yöntemlerinden birisidir. İdrar analizi hasta için bir sağlıklılık testi olarak, gebelik değerlendirme testi olarak veya planlanmış bir cerrahi operasyon öncesi yapılabilir.

Rutin idrar analizi fiziksel, kimyasal ve mikroskobik olmak üzere 3 farklı analiz yönteminin birleşimiyle gerçekleştirilir. Bu 3 farklı idrar analiz yönteminin birleşimine tam idrar analizi adı verilmektedir. Bu incelemelerde idrar içerisinde bulunan normal ve ya normal olmayan metabolik atıkların, bakterilerin, ilaç kalıntılarının ve hücresel kalıntıların miktarının belirlenmesi amaçlanmaktadır.

2.1.Tarihte İdrar Analizi

İdrar analizi tıpta uygulanan en eski klinik laboratuvar testtir. M.Ö. 400 yıllarında Hipokrat, idrar incelemesinin önemine değinmiş ve idrarın genel görünümünü, yoğunluğunu, rengini ve idrar çökeltisinin miktarını incelemiştir. Yani Hipokrat idrarı fiziksel olarak incelemiş ve bazı yorumlar yapmıştır. Hipokrat, Aforizmalar eserinde idrarın üzerinde köpük bulunmasının böbrek hastalığının habercisi olduğunu belirtmiştir. Böbrek hastalıklarında idrarla birlikte atılan protein miktarı artmaktadır ve bu köpüğe neden olmaktadır [17].

500 yıllarında yaşayan Ayus Veda şeker hastalarının idrarlarını inceleyerek idrarın sinekleri cezbettiğini gözlemlemiştir. Bunun üzerine diabetli hastaların ballı idrar çıkardıklarını tarif etmiştir [18].

Böbrek hastalığının tespiti konusunda farklı yaklaşımlardan birisi 7. yy. da yaşamış Theophilus tarafından ortaya atılmıştır. Theophilus proteinleri belirlemek için idrarın ısıtılmasını önermektedir. Isınan proteinlerin yapısı bozularak çökelti oluştururlar ve böylece böbrek hastalığı olup olmadığı tespit edilebilir. Bu konudaki başka bir yaklaşımsa Paracelsus (1493-1541) tarafından ortaya atılmıştır. İdrara karıştırılan sirke nedeniyle de proteinler tepkimeye girerek çökelti oluşturur ve böylece böbrek hastalığı olup olmadığı tespit edilebilir [19].

Ebubekir Muhammed bin Zekeriya el Razi (854-932) (Rhases) nin idrar incelemeleri sonucu böbrek dokusu iltihabı hastalığının tanısını koyup tedavi ettiği bilinmektedir.

Ebu Ali el-Hüseyin bin Abdullah bin Sina yani bilinen adıyla İbni Sina (980-1037) önemli bir tıbbi kaynak olan "Kanunu" eserinde idrar hakkında bilgiler vermiştir. İdrarda bulunan kanın böbrek hastalıklarını işaret ettiğini ve yangısal iltihabik bir etkiye neden olduğunu ifade etmiştir. Karaciğer ve mesane hastalıklarında idrar kokusunun

ağırlaştığından da bahsetmiştir. Sarılık durumunda hastanın idrarının kıyafetlerde renk değişimine neden olduğunu da söylemiştir.

Orta çağlarda idrar inceleme üroskopi adını almıştır. Bu dönemlerde bilgi eksikliği ve günümüzdeki gibi teknolojik gelişmelerin olmaması nedeniyle kendilerini şifacı olarak tanıtan insanlar tarafından idrar analizi kötüye kullanılmıştır. Bu durum öyle bir boyuta ulaşmıştır ki idrardan geleceği gördüğünü iddia eden kişiler bile ortaya çıkmıştır.

Thomas Willis 1674'de yayımlanan "İdrar İncelemesi" adlı kitabında üroskopi yöntemini teorikleştirmiş ve bilime kazandırmıştır. 18. ve 19. yüzyıllarda bilimsel metodların uygulanması ve mikroskobun icadı modern çağ idrar mikroskobisinin gelişmesine katkı sağlamıştır. Kimyasal idrar analizi alanında 20. yüzyılın ilk yarısında geliştirilen metodlar bu analizin bir parçası halini almıştır. Böylece günümüzde hastalarda kullanılan önemli testlerden birisi oluştur [20].

2.2. Fiziksel İdrar Analizi

Fiziksel incelemede idrar; miktarı, görünüm, kıvam, renk, koku, dansite (özgül ağırlık), pH'sı yönünden analiz edilir. Bunlar enfeksiyon hakkında fikir verir.

Normal bir idrar sarı renktedir. İdrar içerisindeki maddelerin oksijenle kimyasal tepkimeye girmesi sonucu idrar renginde değişim oluşur. Renksiz, sarı, turuncu, kırmızı, yeşil, mavi, kahverengi ve siyah renkte idrar olabilir. Bu renk değişimleri kullanılan ilaçlardan, tüketilen yiyeceklerden, fiziksel aktivite ve stresten kaynaklı olabilir. Örneğin idrarın çay rengi tonlarında olması bilirubin miktarının fazlalığını işaret eder. Sarılık hastalığında idrarda bilirubin görülür. Her renk tonu farklı bir hastalığın işaretçisi olabilir. İdrarda bulunan kristallerin ve amorf yapıların artması berraklığın azalmasına sebep olur. İdrardaki bakteri sayısının artması idrar kokusunun artışına sebep olur. Bu bakteriler idrarda bulunan üreyi amonyağa parçalar. İdrar içerisinde koku yapan bileşen amonyaktır. Bu parçalama işlemi aynı zamanda idrar pH'ının artmasına da yol açar. Böylece pH ölçümü ile idrarın ne kadar asidik veya alkali olduğu belirlenebilir. Normal bir idrar pH'sı altıdır. Şeker hastalarının idrarlarındaki ketondan dolayı idrarları meyvemsi kokulara sahiptir. Üreme sistemindeki bir enfeksiyon idrarın pis kokmasına neden olur. İdrardaki koku çok keskin olmadığı sürece rutin idrar analizinde değerlendirilmez.

Normal yetişkin bir insanın idrar dansitesi(özgül ağırlık) 1015-1025 g/ml aralığında değişmektedir. Bol su tüketilen durumlarda ve sıcaklığın düşük olduğu durumlarda dansite düşük çıkar [21].

Bu parametreler ile bazı hastalıkların tanısı konabilirken bazı tedavi takipleri de etkin bir şekilde yapılabilmektedir.

2.3. Mikroskobik İdrar Analizi

İdrarın mikroskobik (sediment) analizindeki en temel yöntem, idrarın mikroskopta bir operatör tarafından incelenmesidir. Bu mikroskop görüntülerindeki eritrosit, lökosit, lökosit kümeleri, silendirler, epitel hücreleri, bakteri, maya, kristal partiküllerinin sınıflandırılıp sayımı gerçekleştirilir.

Normal idrar sedimentinde her mikroskop sahasında 2-5lökosit, 0-1 eritrosit, 1-2 epitel, hücresi bulunabilir. Sağlıklı insan idrarında çeşitli kristaller ve silendir yapıları da bulunabilir. İdeal idrar sedimentinde bakteri olması beklenmez. Eritrositin çok olması taş, kanama tümör eğilimli durumların habercisidir. Sedimentte çok sayıda lökosit olması idrar yolu enfeksiyonunu düşündürür. Ateş, toksinler ve enfeksiyon sonucu epitel hücreeler artar. İdrarda kristallerin incelenmesi böbrek taşı, kalıtsal bozukluklar ve şüpheli ilaç kullanımında önemlidir.

2.4. Kimyasal İdrar Analizi

Kimyasal idrar analizinde idrarın; dansite (yoğunluk), pH, protein, nitrik, glikoz, bilirubin, keton cisimcikleri, ürobilinojen, eritrosit ve lökosit miktarları hakkında araştırma yapılır. Bunların dışında özel durumlarda okzalat, ürobilinojen, sitrat, fenol, hippürik asit, tca, ağır metaller (pb, hg, cd), amino asit profili gibi testler de çalışılmaktadır. Bu parametreler ile bazı hastalıkların tanısı konabilirken bazı tedavi takipleri de etkin bir şekilde yapılabilmektedir. Örneğin, protein ve glikoz ölçümü ile bazı böbrek hastalıklarının tespiti yapılabilmektedir.

2.4.1. Kullanılan Yöntemler

İdrarda aranılan her bir bileşen için farklı analiz yöntemleri geliştirilmiştir. Bu analizler klinik açıdan yapılırken iki kriter dikkate alınmaktadır. Nitel (kalitatif) analizde bir

karışımın içindekilerin neler olduğu saptanmaya çalışılır. Nicel (kantitatif) analizde ise bir karışımdaki maddelerin miktarı saptanır. Klinik bir teşhis koyabilmek için çoğu durumda nitel inceleme yeterli olsa da bazen nicel incelemeler de gerekmektedir.

Bu bölümde idrardaki farklı bileşiklerin tespiti için kullanılan farklı yöntemler hakkında kısa bilgiler verilecektir ve bu çalışmanın ana konusu olan strip yöntemi detaylı şekilde incelenecektir.

İdrarda protein analizi yapılırken proteinlerin çöktürülmesi prensibi kullanılır. Bu prensibi uygulamak için triklorasetik asit (TCA) ile çöktürme yöntemi, proteinleri kaynatarak yani asetik asitle çöktürme yöntemi, proteinleri yoğunlaştırılmış nitrik asit ile çöktürme yöntemi (Heller'in Halka Deneyi), organik çözücüler ile çöktürme deneyi gibi birçok kimyasal süreç vardır. Bu yöntemler idrar içerisindeki protein olup olmadığının belirlenmesinin sağlar. Bazı durumlarda ise nicel inceleme gerekir ve protein miktarının tespit edilmesi için Purdy Metodu [22] gibi yöntemlerden faydalanılır.

İdrarda glikoz varlığı tespit edilmeye çalışılırken Fehling Metodu [23], idrarda bilirubin varlığı belirlenirken metilen mavisi yöntemi [24], Rosin yöntemi [25] gibi yöntemler kullanılır.

2.4.2. Strip Yöntemi

Rutin kimyasal analizde kısa sürede ve birçok testi aynı anda yapmaya olanak sağlayan stripler tanı amaçlı kullanılır. Bu stripler 3-4 mm eninde ve 10-12 cm uzunluğunda bir kâğıdın üzerine farklı kimyasal bileşiklerin yerleştirilmesiyle elde edilir. Nitekim Şekil 1'de örnek bir test strip yer almaktadır. Bu çubuklar üzerinde lökosit, eritrosit, protein, nitrit, glukoz, keton, pH, dansite, bilirubin ve ürobilinojen gibi maddelerin varlığının belirlenmesini ve miktarı hakkında bilgi vermesini sağlayan kimyasal karışımlar içerir. Bu çubuklar ticari olarak üretilmiş kitlerdir. Ticari kaygılar nedeniyle her firma kendine özgü stripler üretmektedir. Uygulanan bu yöntemin yapısı gereği kuru sistem stripler olarak da ifade edilmektedir [26].

Şekil 2.1. İdrar analizi için kullanılan bir strip örneği

İdrarın stripe uygulanmasında kullanılan temel yöntem bir tüp içerisindeki idrara stripin daldırılması şeklindedir. İdrarla buluşturulan strip üzerinde kimyasal tepkimeler gerçekleşerek renk değişimleri olur. Bu renk değişimleri bir tablo yardımıyla değerlendirilerek sonuçlar elde edilir. Ancak bu yöntemde ortaya çıkan strip üzerindeki kimyasalların birbirlerini etkilemesi gibi bir dezavantaj vardır. Bunun sonucunda da oluşan renk değişimleri yanıltıcı olabilmektedir. Bu problemin önüne geçmek için idrar pipetleme yöntemleri kullanılarak strip üzerine damlalar şeklinde bırakılabilir.

Strip yönteminin avantajları kısa sürede sonuç alabilmek ve tek bir adımda birden çok testi gerçekleştirebilmek olarak ifade edilebilir [27]. Strip yönteminin dezavantajlarına bakıldığında striplerin kapalı kutularda ve rutubetsiz ortamlarda saklanma ihtiyacı, daldırma yönteminde sonuçların yanıltıcı olabilme durumu ve ticari nedenler dolayısıyla her firmanın farklı strip üretmesi bu nedenle de standart bir sıralamanın olmaması ifade edilebilir. Strip yöntemi kullanılarak tespit edilen parametreler:

- Özgül Ağırlık: Bu değer böbreklerin idrarı yoğunlaştırma yeteneğini tayin etmektedir. Örnek alınacak bireyin 12 saat sıvı tüketmemesi gerekmektedir. Bu

koşullar altında alınan örnekte 1025 g/L çıkması durumunda böbreklerin normal çalıştığı anlaşılır.

- pH: Hasta olmayan bir bireyin taze idrarında pH değeri 5-6 arasındadır. Bakteriyel bir enfeksiyon olması durumunda 8 ve üzeri değerler çıkabilir. İdrarın görüntüsünden ve kokusunda da bu durum gözlemlenebilir.
- Lökosit: Üriner sistemde meydana gelen bir enfeksiyonunun belirtecidir.
- Nitrit: Üriner sistemde meydana gelen bir enfeksiyonunun belirtecidir. Üriner yolların enfeksiyonunu gösterir. Bakteri kaynaklı enfeksiyon durumlarında bakteriler redüktaz enzimi salgılayabilirler. Bu enzim ile nitratı nitrite dönüşür ve idrardaki miktarı artar. Böylece idrarda nitrit bulunması bakteriyel bir enfeksiyon olduğuna işarettir.
- Eritrosit: Eritrositlerin hemoliz olması sonucu ortaya çıkan hemoglobini gösterir.
- Keton
- Bilirubin: Sarılık hastalığının tespiti için erken teşhis olanağı sağlar. Gözakı sararmadan önce idrarda 4 mg/dL'den daha yüksek tespit edilmesi halinde sarılık olma ihtimalinin üksek olduğunu gösterir.
- Ürobilinojen: Bağırsaklarda bulunan bakteri faunasının ürettiği ürobilinojen normal şartlarda idrarda bulunur. Ürobilinojenin artması safra tıkanıklığı giderildikten sonra karşılaşılan bir durumdur.
- Glukoz
- Protein
- Hemoglobin

2.4.2.1. Striplerin Değerlendirilmesi

En temel değerlendirme şekli analizi yapan laborantın strip tablosunu kullanarak renk değişimlerini gözlemlemesi ve sonuçları not alması prensibine dayanır.

İkinci ve yaygın şekilde kullanılan yöntemse reflektans fotometre prensibiyle çalışan analiz cihazlarına striplerin yerleştirilerek sonuç bilgisinin alınmasıdır. Reflektans fotometre yönteminde farklı dalga boylarında ışık yayan ledler kullanılarak striplerden yansıyan ışıklar optik alıcılar yardımıyla değerlendirilir. Bu prensiple çalışan basit ve ya tam otomatik sistemler hastanelerde aktif olarak kullanılmaktadır.

Şekil 2.2. Reflektans fotometre yöntemi ile çalışan strip okuyucu örneği

Bu çalışma kapsamında önerilen değerlendirme yöntemi ise striplerin görüntü işleme prensipleriyle incelenmesini konu edinmektedir.

3. BÖLÜM

GÖRÜNTÜ İŞLEME

3.1 Görüntü İşlemenin Temelleri

Görüntü; gerçekte var olmayan bir görünümün insan gözünün görebilmesi için bir araçla oluşturulmasıdır. Fotoğraf ise optik ve ya kimyasal süreçlerin uygulanarak kalıcı bir yüzey üzerine görüntünün aktarılmış halidir. Dijital fotoğraf ise piksel değerlerinden oluşan elektronik bir fotoğraftır. Burada bir örnekle bu tanımları açıklamak gerekirse görüntü gölgedir ve ışık tarafından oluşturulur. Bir fotoğraf makinesi yardımıyla gölgenin kaydedilmesi sonucu da saklanabilir olan fotoğraf ortaya çıkmış olur. Fotoğraf Yunanca ışık ve çizmek anlamlarına gelen photos ve graphos sözcüklerinden türetilmiştir. 1840 yıllarında ilk kez bu isimden John F. W. Herschel (1792-1871) bahsetmiştir [28].

Dijital görüntü temel anlamda quantalama ile elde edilir. Yani gerçek dünyadaki analog ve sonlu olmayan değerler sonlu değerlere indirgenerek dijital (sayısal) olarak ifade edilirler. Elde edilen sonlu ve dijital veriler kümesi dijital görüntü olarak adlandırılır. Dijital veriler bitlerle temsil edilir. Genellikle 12 bitten daha uzun veri blokları kullanılmaz. İnsan gözünün ayırt edebileceği renk aralığı 12 bitle temsil edilen piksel değerlerinden çok daha azdır.

1975 yılında ilk dijital fotoğraf makinesinin keşfi, Kodak çalışanı Steve Sasson tarafından olmuştur. Çektiği siyah beyaz fotoğrafları kasete kaydederek saklayan bu makine 23 saniyede bir fotoğraf oluşturabiliyordu. Her fotoğraf 0.01 megapiksel çözünürlüğe sahipti [29-30].

İlk dijital kameranın keşfedilmesiyle başlayan dijital görüntü havuzunun her geçen gün genişlemesi ve bu görüntüler içerisinden anlamlar çıkarma gereksinimi, görüntü işleme

algoritmalarının/uygulamalarının gelişmesini sağlamıştır. Böylece geçen yıllar içerisinde birçok çalışma yapılmış ve literatürdeki yerini almıştır.

Görüntüler en az 2 boyutlu olurlar. Bu tip görüntüler siyah ve beyazdan meydana gelir. 3 boyutlu görüntüleri de RGB format olarak değerlendirebiliriz. 4 boyutlu görüntülere uydu görüntüleri ve modern ultrason görüntüleri örnek olarak verilebilir.

Bu açıklamalardan sonra görüntü işleme şu şekilde tanımlanabilir: Kaydetme ya da ölçme yoluyla elde edilen dijital görüntü verilerini girdi olarak kullanarak bir işlemci birimi üzerinde donanımsal ya da yazılımsal olarak ölçme ve değerlendirme işlemlerine tabi tutmak, amaca uygun şekilde değiştirmek ya da farklı bir biçime dönüştürmek gibi uygulanan süreçlerin tamamıdır.

3.2. Görüntü işleme ile ilgili bazı terim ve tanımlar

- Ölçeklendirme: Görüntünün orantılı ya da orantısız olarak büyültülmesi ve ya küçültülebilmesi işlemine denir.

- Döndürme: Görüntünün yatay ve dikey olarak ya da kullanıcının istek ve ihtiyacına bağlı olarak, kendi ekseni etrafında, belli bir açı ile çevrilmesi işlemidir.

- Yansıtma: Görüntünün yatay ve dikey olarak, aynadaki görüntüye benzer şekilde ters çevrilmesi işlemidir.

- Renk düzeltmesi: Görüntü üzerindeki ışık ve renk tonlarının değiştirilmesi sürecidir.

- İşaretleme: Resim üzerinde bir işlem yapmak için belli bir bölümünün seçilmesine işaretleme denir.

- Katman: Genellikle görüntü işleme yazılımlarında kullanılan ve alttaki nesnenin görünmesini engellemeyen saydam sanal yüzeylere katman denir. Katmanlar özellikle montaj işlemlerinde bolca kullanılmaktadır.

- Örtüleme/Maskeleme: Görüntünün bazı bölümlerini çalışmanın dışında tutmak için kapatılabilir. Bu işleme örtme ya da maskeleme denir. Maskeler görüntünün verilen komuttan etkilenmemesini veya bazı bölümlerin gizlenmesini sağlar.

- Filtre: Bir görüntüde uygulandığında orijinal görüntüde olmayan eklemeler ya da çıkarmalar sağlayan araçlardır.

- Dönüştürme: Bir görüntü formatını, başka bir formata dönüştürmek için kullanılan yazılımların yaptığı işlemdir [31].

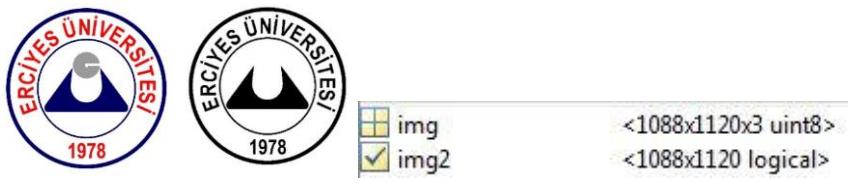
3.3. Sayı Tipleri

Double sayı tipi 8 byte (64bit) veriden oluşmaktadır. Daha önce açıklandığı gibi görüntülerin 12 bitten fazla tutulması insan gözü için anlamlı değildir. Double veri tipi 2^{64}=18.446.744.073.709.551.616 renk aralığını kapsamaktadır. Görüldüğü üzere görüntülerin double olarak depolanması gereksizdir.Uint8 Sayı tipi 1 byte (8bit) veriden oluşur. Genellikle kullanılan veri uzunluğudur. 2^8 = 256 farklı rengi temsil edebilir. Uint16 sayı tipi 2 byte (16bit) veriden oluşur. 2^{16}=256x256=65536 farklı rengi temsil edebildiği için Uint8 ile kıyaslandığında 256 kat daha fazla renk içerir denilebilir.

3.4. Görüntü Biçimleri

- İkili (Binary/Logical) Görüntüler

İkilik(Binary/Logical) bir görüntüde her piksel 0 (siyah) veya 1 (beyaz) değerine sahiptir.



Şekil 3.1. Erciyes Üniversitesi logosu, logonun ikili görüntüsü, açıklamalar

- Yoğunluk görüntüleri

Yoğunluk görüntüleri, elemanları belli bir skaladaki aydınlanma değerlerini gösteren matrislerdir. Değerleri 0 siyahı, 1 beyazı belirtmek üzere 0-1 aralığındaki gerçel sayılardır. Örneğin unit8, değerleri 0 ile 255 arasında değişen 8 bitlik gri seviyeli bir resmin sınıfıdır.

Şekil 3.2. Mühendislik F. logosu, logonun yoğunluk görüntüsü, açıklamalar

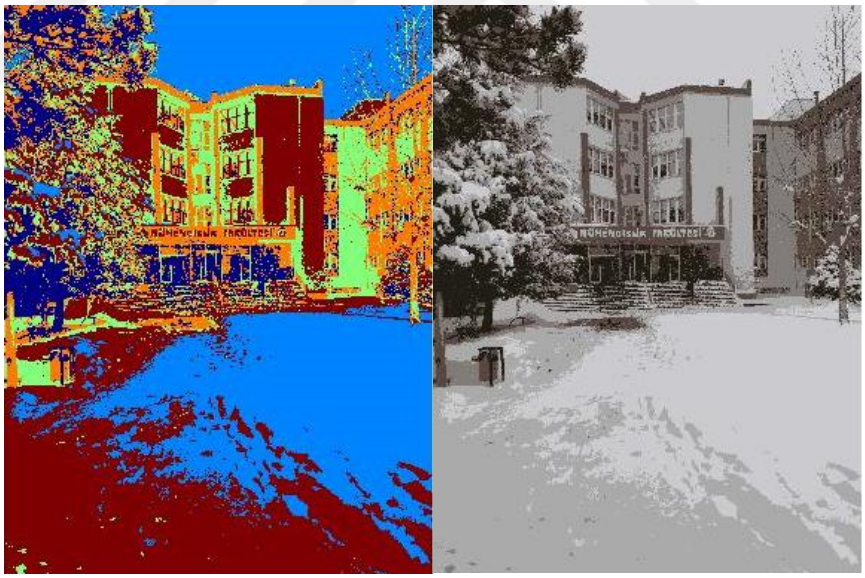
- Gerçek Renkli Görüntüler

Gerçek renkli görüntülerde yer alan renkler üç ana rengin belli oranlarda karışımından oluşur. Işıktaki ana renkler kırmızı, yeşil ve mavidir. Kırmızı (R), yeşil (G) ve mavi (B)'nin kısaltması olarak da RGB karşımıza çıkar. Gerçek renkli resimler 3 farklı matrisin birleşimi şeklindedir. Bu matrislerin boyutları birbirine eşittir. MxNx3 şeklinde ifade edilebilir.

Şekil 3.3. Mühendislik Fakültesi logosu, açıklama

Her piksel için ona ait üç değeri değil de sadece bir değer gönderme yöntemine indeksleme denilmektedir. Renkleri çözümleyebilmek için bir anahtara (eşleştirme haritasına) ihtiyaç duyulacaktır. İşte bu anahtara renk haritası, değerlerden oluşan bu matrise de indekslenmiş görüntü diyoruz.

Şekil 3.4. Mühendislik Fakültesi ve renklerin 5 merkezli sınıflandırılması



Şekil 3.5. Jet renk paletiyle ve merkezlerin ağırlıklarına göre boyama sonuçları

3.5. Ön İşlem

3.5.1. Cebirsel (Aritmetik) İşlemler

Piksel değerleri üzerinde uygulanan toplama, çıkarma, çarpma bölme işlemlerini ifade etmektedir. Her bir piksel değeri sabit bir sayı ile toplama işlemine tabi tutulursa renklerin beyaza doğru yaklaştığı görülecektir. Görüntü işlemede çıkarma işlemi

genellikle iki görüntü arasındaki farkların bulunması amacıyla kullanılır. İkili formatta yapılan çıkarma işlemi görüntüyü daha da koyu bir hale dönüştürecektir. Çarpma işlemi 1' den daha büyük bir sayıyla piksellerin çarpılması anlamına gelir. Piksel değerleri artacağı için beyaza doğru daha hızlı bir geçiş elde edilir. Burada toplamadan farklı olarak siyah pikseller sıfır değerinde olacağı için hangi sabitle çarparsak çarpalım sonuç yine sıfır olacaktır. Bu nedenle siyah piksellerde değişim olmayacaktır. Bölme işlemi de çarpma işleminin tam tersidir denilebilir. Piksel değerleri sabit bir değere bölündüğünde küçülecek böylece siyaha doğru daha hızlı yaklaşacaktır.

Şekil 3.6. Orijinal logo, 100 ile toplama ve çıkarma işlemleri sonuçları

3.5.2. Mantıksal(Lojik) İşlemler

Bu işlemler ikili görüntüler üzerinde uygulanmaktadır. Aşağıda ki tablolardan anlaşılacağı gibi; piksel AND sıfır = sıfır, piksel AND bir = piksel, piksel OR sıfır = piksel, piksel OR bir = bir sonuçları verir. İkili görüntü için sıfır siyah rengi, bir beyaz rengi temsil etmektedir. Not işleminde ise sıfırın tümleyeni bir, birin tümleyeni sıfırdır. Bu nedenle görüntüdeki pikseller tersleriyle yer değiştirmiştir denilebilir.

Tablo 3.1. AND, OR ve NOT işlemlerine ait tablolar

AND		
A	B	Q
1	0	0
0	1	0
1	0	0
1	1	1

OR		
A	B	Q
0	0	0
0	1	1
1	0	1
1	1	1

NOT	
A	Q
0	1
1	0

Şekil 3.7. Orijinal logo, ikili logo, ikili logonun tümleyeni

3.5.3. Gri Seviye Azaltma

Bu işlem yoğunluk görüntüsü, indeksli görüntü yada RGB görüntü üzerinde uygulanabilir. Örneğin uint8 türündeki bir RGB görüntü alınarak 0-255 skalasındaki piksel değerleri ikili dizilere dönüştürülür. Burada 0=00000000, 1=00000001, … 255=11111111 olacak şekilde dönüşüm gerçekleştirilir. Elde edilen bu diziler yine aynı uzunluktaki sabit ikili dizilerle (ÖR: 00000011 ile) AND yada OR işlemine tabi tutulur. İşlem sonucunda bulunan değerler tekrar 0-255 arası piksel değerlerine dönüştürülür ve resim tekrar elde edilir. Bu işlem aslında kuantalama işlemidir. Elde edilen görüntüde bir tür kontrast açma işlemi uygulanmış olur.

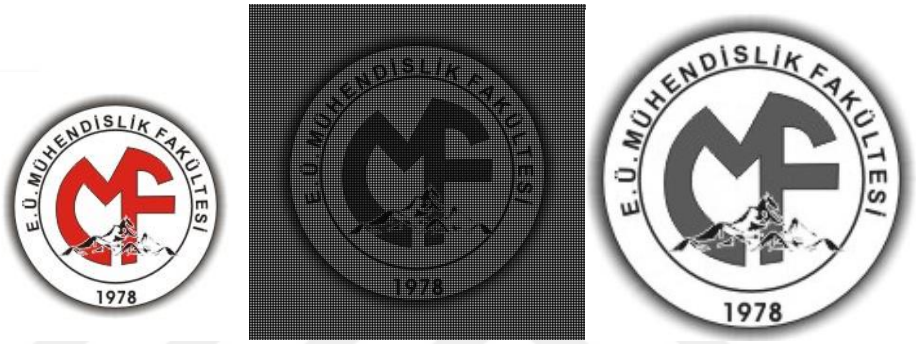
Şekil 3.8. Orijinal görüntü, gri seviye azaltma işleminden elde edilen görüntü

3.5.4. Temel Önişlemler

Orijinal görüntünün bir kısmının koparılarak yeni bir görüntü oluşturulma işlemine parça çıkarma(kesme) denilmektedir.

Çözünürlük değiştirilmeden araya piksel ekleme yöntemleri kullanılarak görüntü boyutunun arttırılması işlemine büyütme (zoom) denilmektedir. Büyütme için kullanılan bazı yöntemler aşağıda verilmiştir.

- Genişletme ve Ortalama Alma
- Sıfır Ekleme ve Birinci Dereceden Konvolüsyon Uygulama
- Sıfır Ekleme ve Sıfırıncı Dereceden Konvolüsyon Uygulama



Şekil 3.9. Orijinal görüntü, genişletilmiş görüntü, büyütülmüş görüntü

Piksel değerleri üzerinde değişiklik yapılmadan matris üzerinde uygulanan öteleme yapma işlemine kaydırma (shift) denilmektedir. Burada r0 öteleme miktarı olmak üzere aşağıdaki formülle temsil edilir.

Ryeni= R+r0 (1)

Aşağıdaki formülün görüntüye uygulanmasıyla saat yönünün tersine döndürme işlemi yapılır.

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \quad (2)$$

Şekil 3.10. Orijinal görüntü, döndürülmüş görüntü

3.5.5. Yapısal (Morfolojik Filtreler)

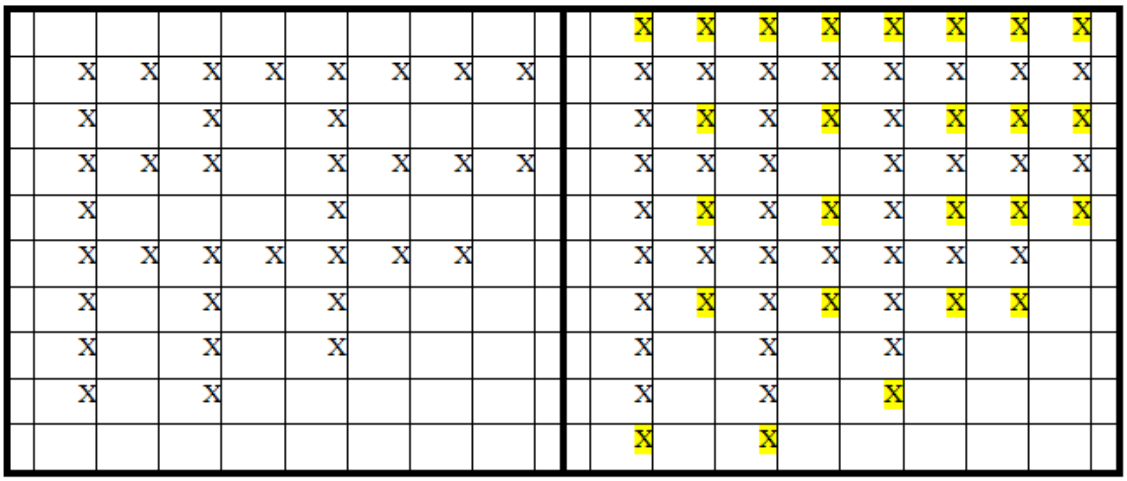
Görüntü üzerinde bir yapılandırma elemanı kullanılarak bazı yapısal işlemler gerçekleştirilir. Bu işlemler sonucunda bölgesel geliştirmeler elde edilir. Kullanılan yapılandırma elemanı orijinal görüntü üzerindeki tüm pikselleri gezerek işleme tabi tutulur. Yapılandırma elemanının merkezi referans olacak şekilde sonuçlar kaydedilir. Yeni görüntü elde edilir. Bu işlemler bir örnek üzerinde açıklanacaktır.

X
X
X

Şekil 3.11. Yapılandırma elemanı (X: lojik1=Beyaz renk), Merkez=2. Piksel

- Yayma (Dilation) İşlemi

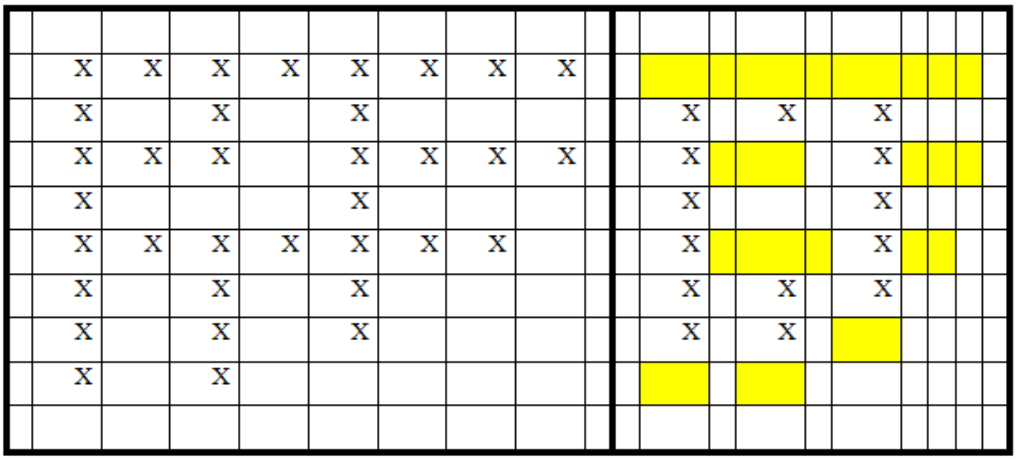
Yapılandırma elemanının merkezinde bulunan piksel eğer beyazsa (Lojik 1 ise) çevredeki pikseller yapılandırma elemanı ile yer değiştirilir.



Şekil 3.12. Orijinal görüntü ve yayma işlemi sonucu

- Aşındırma (Erosion) İşlemi

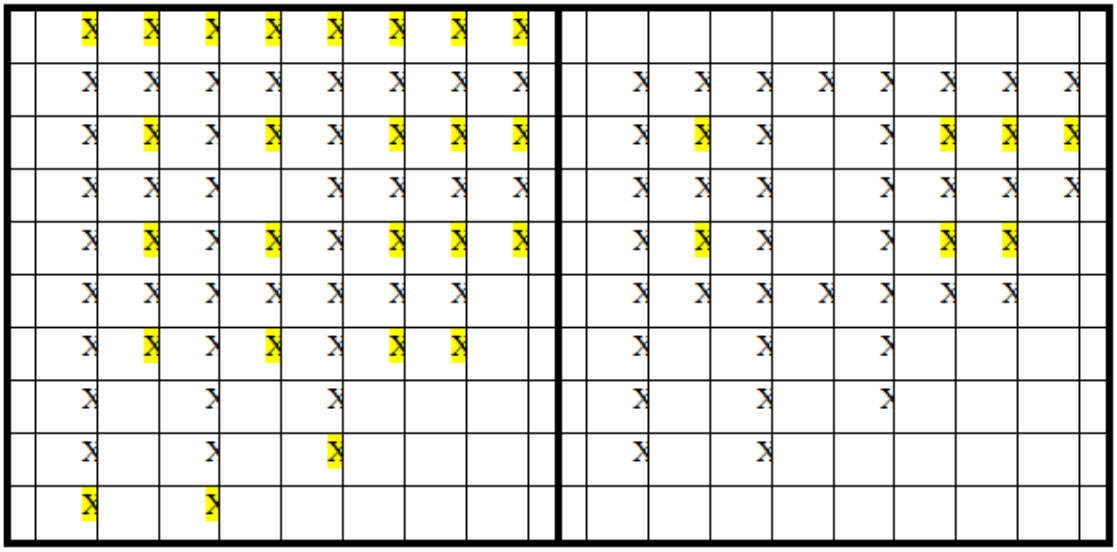
Yapılandırma elemanını orijinal görüntüde birebir örtüşme sağlıyorsa sadece yapılandırma elemanının merkezindeki piksek korunur diğerleri silinir.



Şekil 3.13. Orijinal görüntü ve aşındırma işlemi sonucu

- Kapatma (Close) İşlemi

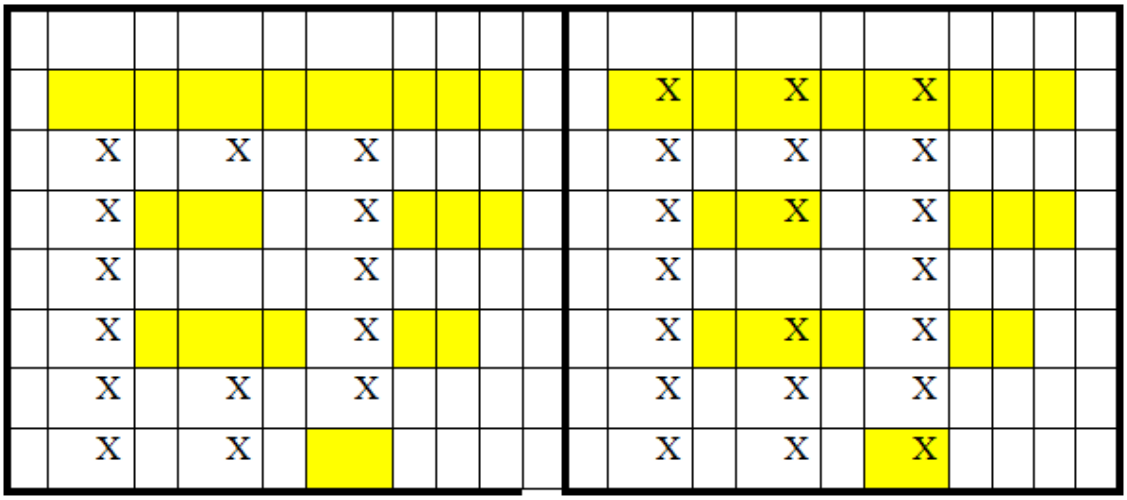
Önce yayma sonra aşındırma işlemlerinin arka arkaya uygulanmasıdır. Yayma görüntüsü üzerine aşındırma uygulayalım.



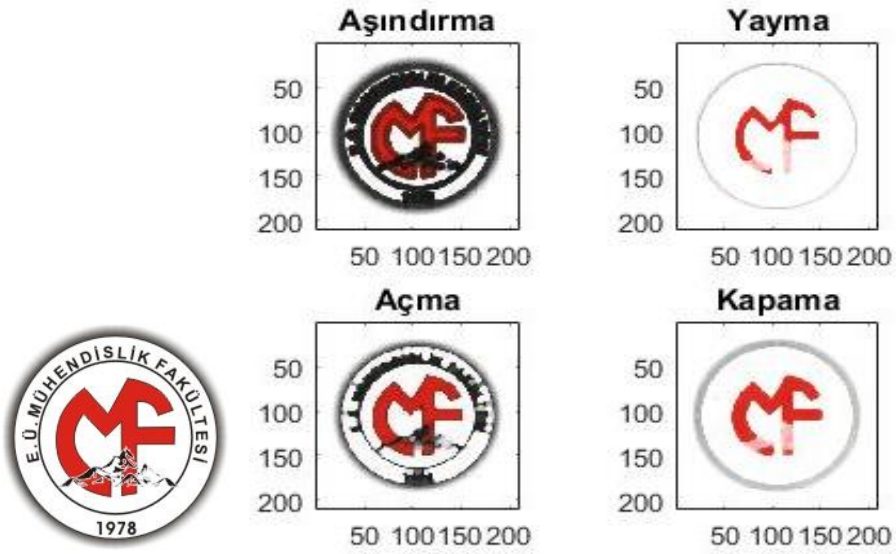
Şekil 3.14. Yayma görüntüsü ve üzerinde uygulanan aşındırma işlemi sonucu

- Açma (Open) İşlemi

Önce aşındırma sonra yayma işlemlerinin arka arkaya uygulanmasıdır. Açma görüntüsü üzerine aşındırma uygulayalım.



Şekil 3.15. Aşındırma görüntüsü ve üzerinde uygulanan yayma işlemi sonucu



Şekil 3.16. Orijinal görüntü üzerinde uygulanan tüm işlemlerin sonuçlar

4. BÖLÜM

SİNİR AĞLARI

4.1. Giriş

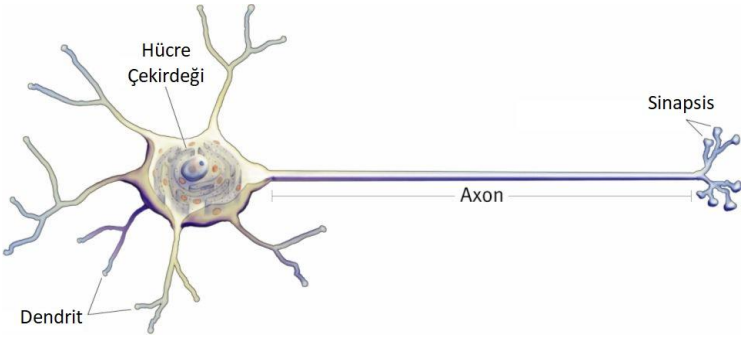
Bu bölümde, tez çalışmasının asıl çalışma konusu olan Konvolüsyönel Sinir Ağları (CNN) yaklaşımının temel taşı olan yapay sinir ağları hakkında teorik bilgiler verilecektir. İlerleyen bölümlerdeyse Konvolüsyönel Sinir Ağları detaylı şekilde açıklanacaktır.

4.2.Yapay Sinir Ağları (YSA)

Yapay sinir ağı; insan beyninin sinir hücrelerinden oluşmuş katmanlı ve paralel olan yapısının tüm fonksiyonlarıyla beraber yazılımsal olarak ve ya donanım üzerinde modellenmesidir.

Burada belirtilen fonksiyonlar, insan beyninin bilgileri yorumlayabilmesi, yeni bilgiler türetebilmesi ve bu işlemi bir yardım almadan otomatik olarak gerçekleştirmesi olarak açıklanabilir.

Şekil 19' da biyolojik olarak bir sinir hücresi gösterilmiştir. Gerçek bir sinir hücresinin çalışma sistemi incelenerek, yapay sinir ağlarının nasıl modellendiği açıklanacaktır.

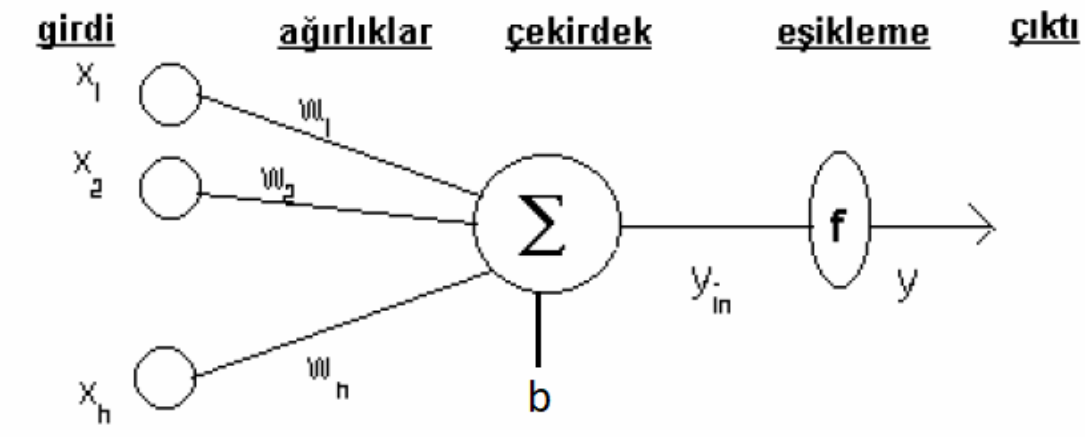


Şekil 4.1. Sinir Hücresi

Dendritin görevi diğer sinir hücrelerinden alınan sinyalleri, kendi sinir hücresinin çekirdeğine iletmektedir. Dendritler aynı zamanda sinyal seçiciliğini de sağlamaktadır. Hücre çekirdeği ile her bir dendrit arasında farklı bir iletişim vardır. Böylece iletişime göre dendrit bazen ağırlıklı pay sahibi olurken bazen de pasif olduğu gözlemlenmiştir. Dentritlerin bu işlevi, yapay sinir ağında girişlerin ağırlıklandırılması olarak modellenmiştir. Buradan yola çıkılarak yapay sinir ağındaki "öğrenme" işlemi ağırlık katsayılarının güncellenmesi olarak ifade edilebilmektedir.

Hücre çekirdeği, dendritler yoluyla iletilen tüm sinyalleri alıp toplayan merkezdir. Hücre çekirdeği, girişindeki sinyali diğer sinir hücrelerine ulaştırabilmek için aksona iletir. Hücre çekirdeğinin işlevi, yapay sinir ağında toplam fonksiyonu olarak modellenmiştir.

Akson, hücre çekirdeğinden gelen sinyali kendinden sonraki sinir hücresine iletmekle görevlidir. Akson ucunda sinapsis denilen birimlere bilgiyi aktarır. Sinapsis, aksondan gelen toplam bilgiyi belli bir eşik değerine göre değiştirerek işlemden geçirdikten sonra diğer sinir hücrelerinin dendritlerine iletmekle görevlidir. Böylece toplam sinyal, belli bir aralığa indirgenerek diğer sinir hücrelerine iletilmiş olunur. Böylece gelen sinyaller ile iletilen sinyaller arasında bir ilişki oluşturulur. Sinapsin bu işlevi, yapay sinir ağında aktivasyon fonksiyonu olarak modellenmiştir.



Şekil 4.2. Yapay sinir ağı modeli

Yukarıdaki YSA modelinde:

x1, x2, … xn ile ifade edilen girişlerdir. w1, w2, .. wn ile ifade edilen ağırlık katsayılarıdır. b ile ifade edilen biastır. x1w1, x2w2, .. xnwn çiftleri her bir dendriti temsil etmektedir. yin ile ifade edilen toplam fonksiyonu çıkışıdır.

Burada dentritleri ve biası giriş olarak alan ve yin çıkışını üreten çekirdektir. Burada bias her toplam işlemine katılan bir sabittir. f ile ifade edilen aktivasyon fonksiyonunu yani sinapsisi göstermektedir. y ile ifade edilen aktivasyon fonksiyonunu çıkışıdır.

Özetle YSA; xn girdilerine karşılık y çıktı sinyalini oluşturur ve bu sinyali diğer hücrelere iletir. wn ve y ler arasındaki ilişki wn ağırlıkları ile ayarlanır. Bu ayarlama işlemine öğrenme ve ya eğitim denir. Eğitim esnasında wn ağırlıklarındaki değişim izlenir ve yeterli hata oranına ulaşıldıktan sonra öğrenme tamamlanmıştır denilebilir [32].

4.2.1.Tarihsel Gelişimi

1943 yılında Warren McCulloch ile Walter Pitts tarafından ilk yapay sinir ağı modeli gerçekleştirilmiştir. McCulloch ve Pitts, insan beyninin hesaplama yeteneğinden esinlenerek, elektrik devreleriyle basit bir sinir ağı modellemişlerdir [33].

1949 yılında Hebb öğrenme algoritmasını bilgisayarlar tarafından gerçekleştirilecek şekilde geliştirmiştir. Hebb kuralında her sinapsisin yeni ağırlığı basitçe eski ağırlık değeri ile giriş ve çıkışın çarpımının toplamı olarak hesaplanır. Hebb öğrenme kuralı, yapay sinir ağları için en eski ve en basit öğrenme kuralıdır. Algotitması aşağıdaki gibidir [34].

Adım-1: Tüm ağırlıklara ilk değerini ata; w(i)=0 (i=1 to n)

Adım-2:Tüm eğitim vektörleri (s) ve hedef vektörleri (t) için aşağıdaki adımları uygula:

Adım-2.1: Giriş birimlerine (x) eğitim vektörlerini al; x(i)=s(i) (i=1 to n)

Adım-2.2: Çıkış birimlerine (y) hedef vektörlerini al; y=t

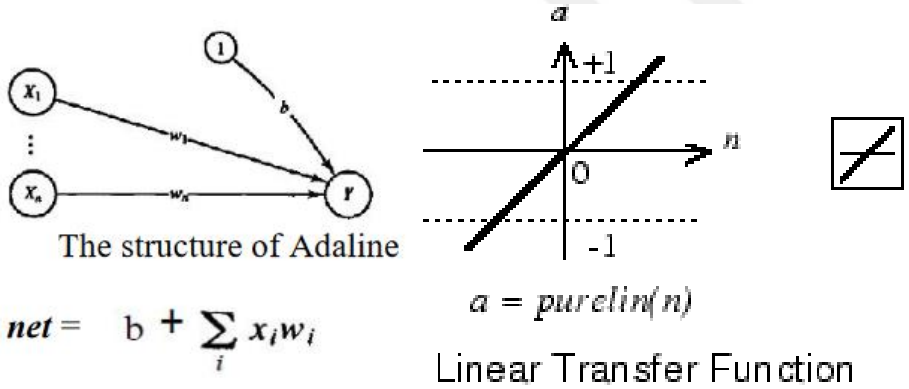
Adım-2.3: Ağırlıkları (w) ve biası (b) güncelle;

w(i)_yeni=w(i)_eski+x(i)*y (i=1 to n)

b_yeni=b_eski+y

1957 yılında geliştirilen Perceptron en eski sinir ağlarından biridir. Bir sinir hücresinin birden fazla girdiyi alarak bir çıktı üretmesi prensibiyle çalışır. Bu yapısı, doğrusal bir fonksiyonla iki parçaya bölünebilen problemlerde kullanılmasına olanak sağlar. Bu problemlere AND, OR, NOT durumları örnek olarak verilebilir [35].

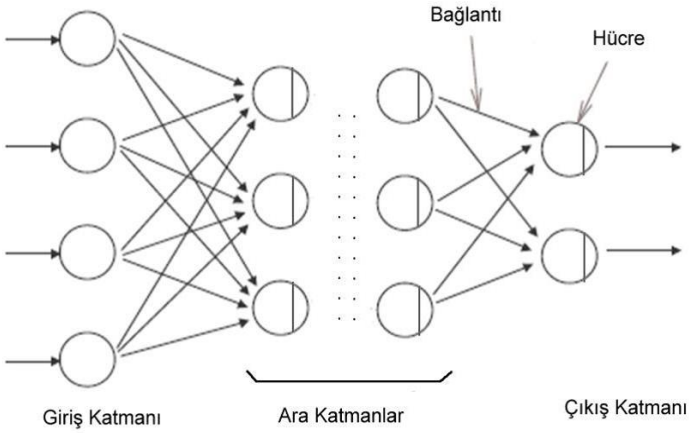
1960- Widrow ve Hoff ADALINE (Adaptive Linear Neuron - Uyarlanabilir Doğrusal Nöron) öğrenme algoritmasını geliştirmiştir. Adaline giriş sinyalleri ve hedef çıktıları için (-1, +1) aralığında aktivasyon fonksiyonu kullanır. Günümüzde Matlab' da "purelin" fonksiyonu ile temsil edilmektedir. Girişlere etki eden ağırlıklar ayarlanabilir. Ağırlıkların ayarlanması işlemine eğitim denir. Bu işlem en küçük karelerin ortalması (LMS), Widrow–Hoff kuralı gibi yöntemlerle yapılır. Adaline'da sadece bir çıkış birimi vardır [36].



Şekil 4.3. Adaline

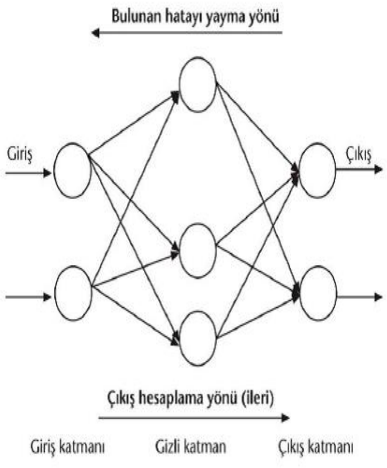
Adaline üzerine birçok çalışma yapılsa da zamanla tek katmanlı sinir ağının yetersizliği gündeme gelmiştir ve onun iki katmanlı hali olan "Madaline" ortaya çıkmıştır. Widrow, telefon hatları üzerindeki gürültüleri elimine etmeye yarayan adaptif filtreleri geliştirmek için adaline kullanmıştır. Böylece ilk defa YSA'lar gerçek bir probleme uygulanmıştır. Ancak adaline birçok uygulama için oldukça iyi çalışmasına rağmen lineer(doğrusal) problem uzayıyla sınırlıdır. Bu nedenle doğrusal olmayan çözümler üretebilmek için mimaride ve algoritmada değişikliklere gidilerek Çok Katmanlı Algılayıcılar (MLP) ağı önerilmiştir.

MLP ile çok katmanlı ağ yapısı, doğrusal olmayan aktivasyon fonksiyonları ve geriye yayılım adı verilen öğrenme sistemi kullanılmıştır. Böylece eğitim çeşidine göre ileri beslemeli ve geri beslemeli ağlar olarak iki grup YSA ortaya çıkmıştır [37].



Şekil 4.4. İleri Beslemeli Sinir Ağlara Örnek

İleri beslemeli sinir ağları sadece ileri yönlü sinyal akışını gerçekleştirebilir. Üç katmanlı ileri beslemeli sinir ağı bir örneği Şekil 21'de gösterilmiştir. İleri beslemeli yapay sinir ağları katmansal mimariye sahiptir. Yani ilgili katmandaki hücrelerin çıkışları sonraki katmana giriş olarak uygulanır. Giriş katmanı, aldığı bilgileri işlemeden kendinden sonraki katmana yani ara(gizli) katmana iletir. Bilgi, ara ve çıkış katmanında hesaplanır ve sonuç üretilir.



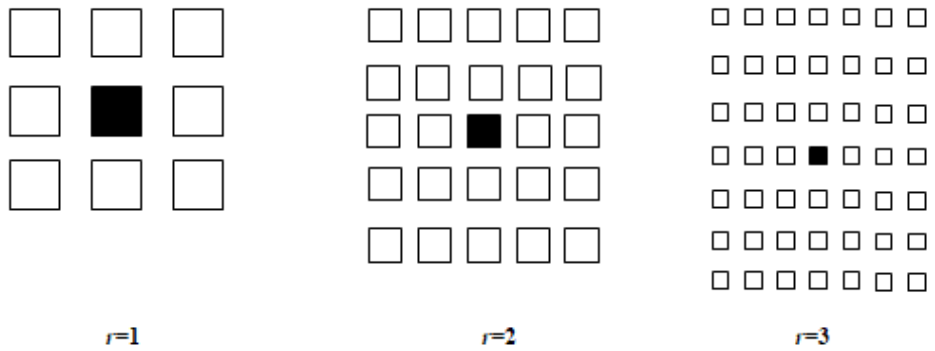
Şekil 4.5. Geri Beslemeli YSA

Geri beslemeli yapay sinir ağlarında, geri besleme işlemi iki şekilde gerçekleştirilebilir. İlgili katmanda bulunan hücreler arasında olabilir. İlgili katman ile sonraki katmanlar arasındaki hücrelerle olabilir. Geri besleme işlemi bir geciktirme elemanı üzerinden gerçekleştirilir. Bu yapısı nedeniyle geri beslemeli yapay sinir ağları, doğrusal olmayan dinamik bir davranış sergiler. Yukarıda bulunan Şekil 22'de iki katmanlı ve çıkışlarından giriş katmanına geri beslemeli bir YSA yapısı görülmektedir [38].

4.3.Hücresel Sinir Ağları (HSA)

HSA, doğrusal olmayan bir davranış gösteren gerçek zamanlı sistemler bütünüdür. İlk kez Chua ve Yang tarafından analog işaretleri işleyen ve doğrusal olmayan bir analog devre olarak ortaya konulmuştur [39-40]. Bu çalışmada HSA' nın görüntü işleme alanındaki uygulamalarından faydalanılmıştır. HSA' nın görüntü işlemedeki görevi, giriş görüntüsünü istenilen amaca uygun olarak çıkış görüntüsüne dönüştürmektir. Çoklu gri seviyeye sahip giriş görüntüsünün siyah beyaz olarak çıkışta üretilmesi olarak da ifade edilebilir [41].

HSA'nın temel yapı taşı hücrelerdir. Yapay sinir ağlarıyla HSA'yı ilişkilendirecek olursak bölgesel ilişki içerisinde düzgün olarak dağılmış hücreler YSA'daki nöronlara karşılık gelir denilebilir. Hücreler arasında yakın komşular birbirleri arasında bilgi alışverişi yaparak bilginin birimler arasında paylaşılmasına olanak sağlamaktadır. Komşuluk r ile ifade edilerek aşağıdaki şekilde örneklendirilmiştir [42].



Şekil 4.6. Bir HSA hücresinin r=1, r=2 ve r=3 komşulukları.

Diğer sinir ağları yaklaşımlarındakine benzer şekilde eşik, geri besleme ve kontrol katsayıları bir hücreyi oluşturan alt birimlerdir. Bu katsayılar literatürde önceden

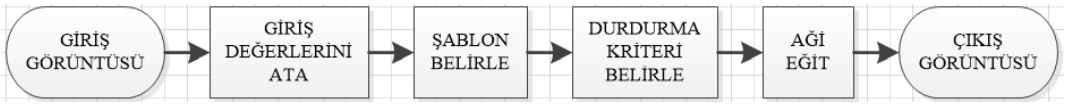
yapılan çalışmalarla problemlere özgü olarak belirlenmiştir. Ancak uygulanacak probleme uygun katsayıların olmaması durumunda optimizasyon algoritmalarından faydalanılarak yeni katsayılar da üretilebilir. Buradaki optimizasyon işlemi yapay sinir ağlarındaki eğitim işlemine karşılık gelmektedir. Bu tez çalışması kapsamında yatay ve dikey hatlar hücresel sinir ağları yardımıyla tespit edilmeye çalışılmıştır. Çalışma yapılırken Genetik Algoritma kullanılarak optimizasyon uygulanmış ve probleme has katsayılar tespit edilmiştir. Ancak bu katsayılar MatCNN aracında bulunan EDGE yani kenar katsayılarından daha iyi sonuçlar üretememiştir. MatCNN kenar katsayıları aşağıda verilmiştir.

- Komşuluk: 1
- Geri besleme katsayıları:

  [0 0 0;
  0 2 0;
  0 0 0];
- Kontrol Katsayıları :

  [-0.25 -0.25 -0.25;
  -0.25 2.0 -0.25;
  -0.25 -0.25 -0.25];
- Bias = -1.5

Bu tez çalışmasında HSA' nın bir sinir ağı gibi kullanıldığı ve görüntü işleme problemlerine uygulandığı MatCNN aracı kullanılmıştır. MatCNN onlarca hücresel ilişkiyi içerisinde bulunduran şablon kütüphanesine sahip ve literatürde birçok çalışmada yer almış bir araçtır [43]. Giriş görüntüsü, önceden belirlenmiş hücrelerden oluşan şablonlar ile çalıştırılarak bir denge durumuna yakınsandığında elde edilen son görüntü sistemin çıktısıdır. Burada görüntüdeki her bir piksel hücresel sinir ağlarında bir hücreye karşılık gelmektedir. Aşağıdaki şekilde MatCNN de uygulanan algoritma verilmiştir.



Şekil 4.7. HSA uygulama algoritması

4.4.Konvolüsyenel Sinir Ağları (CNN)

CNN, gelişmiş bir yapay sinir ağı modelidir. Klasik YSA' lardan farklı olarak CNN, minimum önişlem ile görüntülerden görsel desenleri doğrudan tanımak için tasarlanmıştır.

CNN öncesindeki görüntü sınıflandırma çalışmaları, görüntünün ön incelemesi yapıldıktan sonra özniteliklerin çıkarılması temeline dayanmaktaydı. Bu çıkarılan öznitelikler farklı işlem ve kıyaslamalara tabi tutularak sistemden cevap alınabiliyordu. Görüntü sınıflandırma çalışmaları genellikle bir ve ya birkaç kategoride ve birbirinden rahatça ayırt edilebilen uygulamalarda başarılı olmaktaydı. Bu çalışmalarda çıkarılan öznitelikler ve nesneleri birbirinden ayırt etmeyi sağlayacak matematiksel işlemlerin ve algoritmaların doğru tercih edilmesi sistem başarısını doğrudan etkiliyordu. Bu nedenle klasik görüntü işleme tekniklerinin iyi bilinmesi ve doğru yöntemlerin uygulanması sistem cevabı acısından önem taşımaktaydı.

Yine CNN öncesinde klasik yapay sinir ağları kullanılarak yapılan görüntü sınıflandırma çalışmaları özniteliklerin sinir ağlarıyla değerlendirilmesi prensibine dayanmaktaydı. Bu çalışmalarda da özniteliklerin çıkarılması işi sistem başarısını etkileyen en büyük etkendi.

Gün geçtikçe artan veri miktarı, bu veri havuzundan anlamlı olanların çıkarılması gerektiği sonucunu doğurdu. Klasik yöntemlerle bu havuzu anlamlandırabilmek büyük hesaplama zorlukları doğurmaktaydı. Böylece görüntü sınıflandırma, nesne bulma, nesne takip etme, doğal dil işleme gibi problemlerin çözümü için yeni bir yaklaşıma ihtiyaç duyulmaktaydı. Bu ihtiyaçlara cevap verecek olan yapay sinir ağı tabanlı CNN mimarisi Yann LeCun tarafından ortaya atıldı.

CNN, aşırı değişkenliğe sahip desenleri (el yazısı gibi), basit geometrik dönüşümleri ve bozulmaları tanıyabilmektedir [44].

Çok katmanlı yapısı gereği CNN öznitelik çıkarma işlemini kendi içerisinde otomatik olarak yapmaktadır. Böylece klasik görüntü işleme adımları kullanılarak yapılan öznitelik çıkarma işlemi daha az adımda daha hızlı ve daha doğru şekilde yapılabilmektedir. CNN bu yapısıyla diğer tüm yöntemlerden ayrılmaktadır.

4.4.1.Katmanlar

Katmansal yapının anlaşılabilmesi için öncelikle katmanlardaki işlemlerin gerçekleştirilmesinde kullanılan temel kavramlar açıklanacaktır.

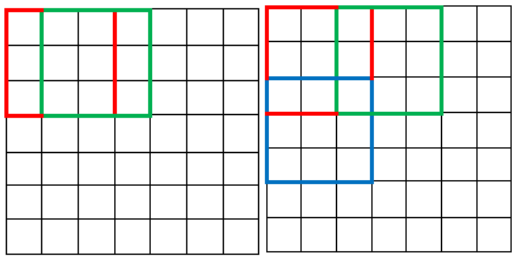
- Filtre (Maske, Kernel)

Görüntü işlemede filtreleme kavaramı; görüntü içerisinden bir bölümün kesilip çıkarılması yöntemiyle elde edilmesi olarak ifade edilebilir. Buradan anlaşılacağı üzere de filtre kesilip çıkarılacak alanı temsil etmektedir. Tüm görüntülerde olduğu gibi filtreler de matris formatında ve piksellerden/sayılardan oluşmaktadır. Özetle filtre, görüntü üzerinde işlem yapılmasına aracılık eden görüntüler ve ya matrislerdir.

Şekil 4.8. Orijinal Görüntü, Filtre, Filtrelenmiş Görüntü

- Kaydırma Adım Mesafesi (Stride)

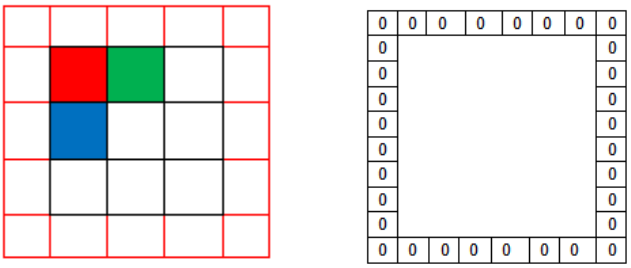
Bu değer bir filtrenin görüntü üzerinde kaç piksellik adımlarla kaydırılacağının bilgisini verir. Aşağıda 3x3 lük bir filtrenin 7x7 görüntü üzerinde adım mesafesi “bir” olarak (Şekil 2-5-1) ve adım mesafesi “iki” olarak dolaştırılmasını gösteren şekiller bulunmaktadır [45].



Şekil 4.9. Kaydırma adım mesafesi örneği

- Piksel Ekleme (Padding)

Görüntü/matris boyutu işlem yapmak için uygun değilse eğer piksel ekleme işlemi yapılır. Bu işlem ihtiyaca göre sıfır doldurma, bir doldurma, en yakındaki piksel değeriyle doldurma gibi yöntemler uygulanarak yapılır.



Şekil 4.10. Piksel ekleme işlemi, Sıfır doldurma işlemi

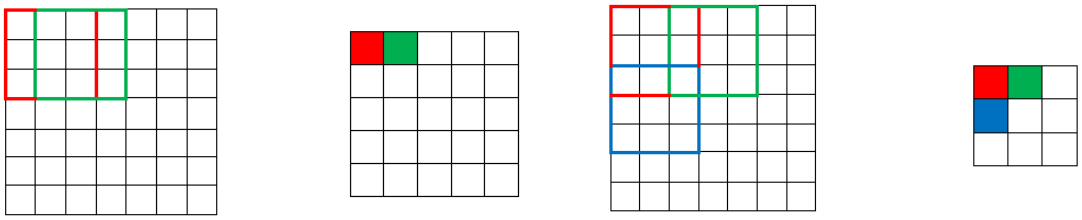
4.4.2.Konvolüsyon Katmanı

CNN' deki ilk katman her zaman konvolüsyon katmanıdır. Bu katmanın amacı görüntüden özellikler çıkarmak olarak ifade edilebilir. Giriş görüntüsü küçük parçalara ayrılarak ele alınır ve her bir parçacık matematiksel işlemlerden geçirilerek bir özet değeri hesaplanır. Bu özet değerleri bir sonraki katmanın yapısına uygun formata getirilerek çıkış görüntüsü(matrisi) elde edilmiş olur. Bu çıkış bir sonraki katmana giriş olarak uygulanacaktır.

Daha teknik bir dille ifade etmek gerekirse bu katmanda, giriş görüntüsü üzerinde bir filtre belirlenen adım mesafesine göre kaydırılarak ve her adımda görüntünün ilgili bölümü ile filtre değerleri noktasal çarpım işlemine tabi tutulup tüm değerler toplanmak

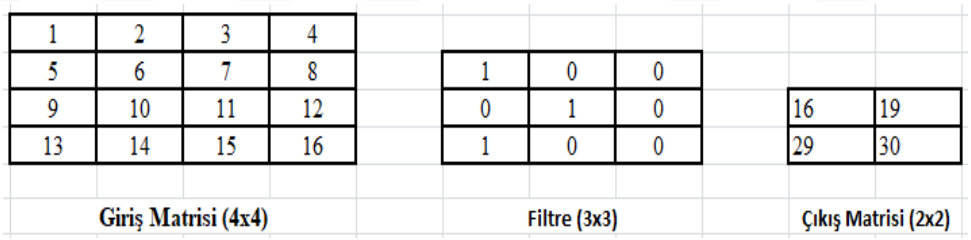
suretiyle çıkış özet matrisi elde edilir. Bu çıkış matrisi özellik haritası (Feature Map) olarak da ifade edilmektedir.

Aşağıdaki şekillerde 7x7 giriş matrisi üzerinde 3x3 filtrenin adım mesafesi "bir" ve "iki" olarak gezdirilmesiyle elde edilen çıkış matrisleri gösterilmektedir.



Şekil 4.11. 7x7 Giriş Görüntüsü, 5x5 Çıkış Görüntüsü, 7x7 Giriş Görüntüsü, 3x3 Çıkış Görüntüsü

Bu katmandaki matematiksel işlemlerin daha kolay anlaşılabilmesi için aşağıdaki örnekte adım mesafesi bir olarak uygulanarak özellik haritası yani çıkış matrisi oluşturulmuştur.



Şekil 4.12. Konvolüsyon işlemi örneği

Tablo 4.1 Konvolüsyon işlemi örneği

1*1+0*2+0*3+0*5+1*6+0*7+1*9+0*10+0*11=16	1*2+0+3+0*4+0*6+1*7+0*8+1*10+0*11+0*12=19
1*5+0*6+0*7+0*9+1*10+0*11+1*13+0*14+0*15=29	1*6+0*7+0*8+0*10+1*11+0*12+1*14+0*15+0*16=30

Çıkışın Hesaplanması

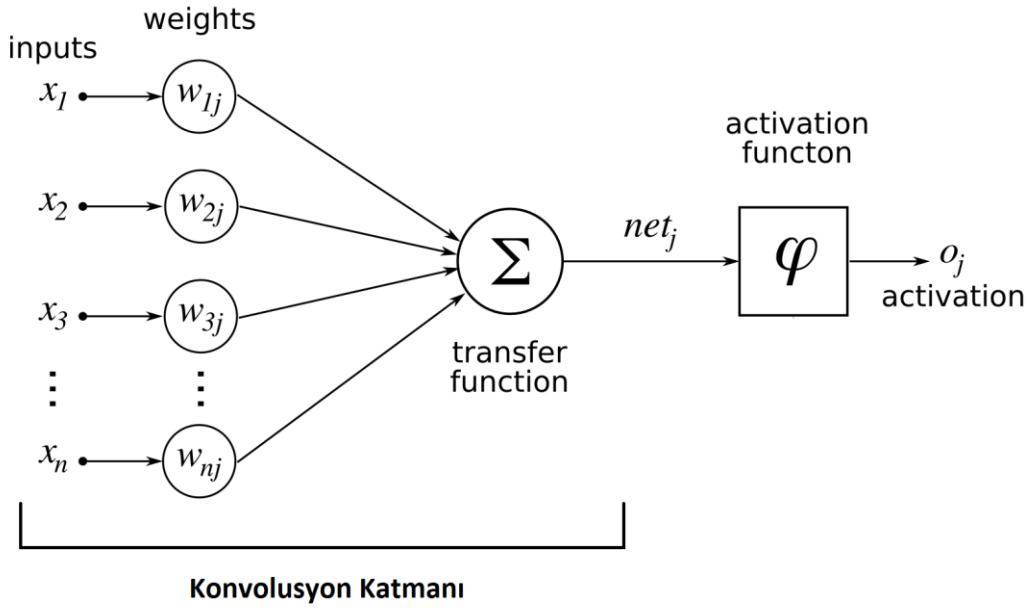
Çıkış matrisinin boyutu belirlenirken aşağıdaki formülden faydalanılır.

$$Çıkış\ Matrisi\ Boyutu = Giriş\ Matrisi\ Boyutu - Filtre\ Boyutu + 1 \quad (3)$$

Yukarıdaki örnek için; Giriş = (4x4), Filtre =(3x3) için Çıkış = (4-3+1) x (4-3+1)=2x2 oldu.

Konvolüsyon katmanı ile yapay sinir ağlarını ilişkilendirmek gerekirse, burada filtrenin uygulandığı her bir adım içerisindeki giriş değerleri yapay sinir ağındaki giriş değerlerine, filtre içerisindeki değerler ise yapay sinir ağındaki ağırlık değerlerine karşılık gelmektedir. Bu nedenle filtre ağırlık matrisi olarak da ifade edilebilir.

Filtre ilk adımdayken x1=1, x2=2, x3=3, x4=5, x5=6, x6=7, x7=9, x8=10, x9=11, w1=1, w2=0, w3=0, w4=0, w5=1, w6=0, w7=1, w8=0, w9=0 olarak yapay sinir ağı işlem yapar ve her adım için bu yöntemle çıkış değerleri oluşturulur.



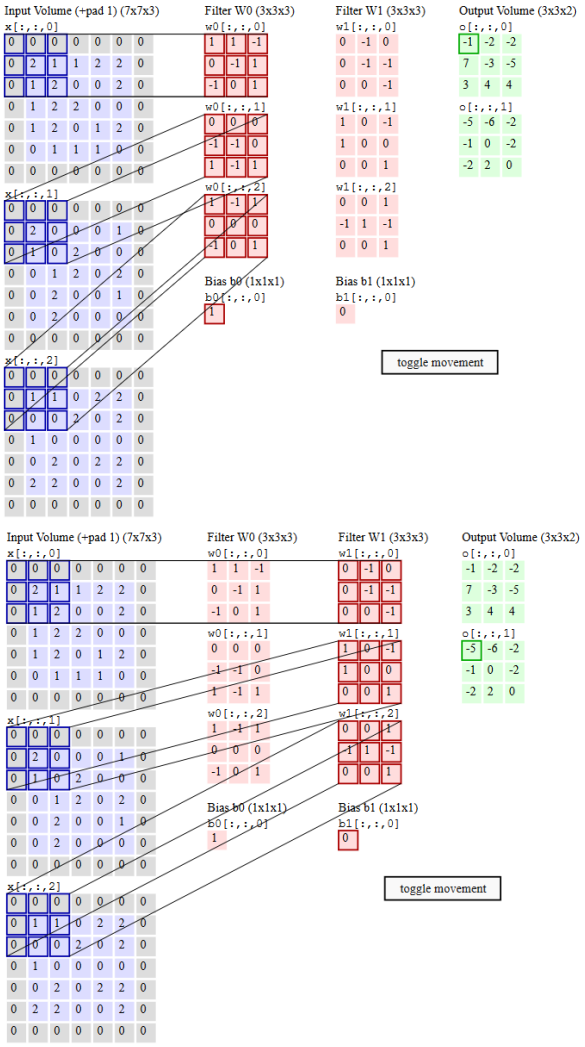
Şekil 4.13. Konvolüsyon

Konvolüsyon katmanında elde edilen çıkış matrisi bir sonraki katmana giriş olarak uygulanacaktır. Ancak burada boyut uyumsuzlukları yaşanabilir. Bu nedenle giriş görüntüsüne piksel ekleme uygulanarak konvolüsyon işlemine tabi tutulur. Piksel

ekleme işlemi uygulanarak elde edilen görüntü matrisi boyutu aşağıdaki formülle hesaplanır.

$$Çıkış\ Görüntüsü\ Boyutu = \frac{Giriş - Filtre\ Boyutu + 2xPikselEkleme}{Adım\ mesafesi} + 1 \qquad (4)$$

Şuana kadar verilen tüm örnekler iki boyutlu görüntüler üzerinde uygulanan işlemleri göstermekteydi. Ancak renkli görüntüler kırmızı-yeşil-mavi (RGB) 3 kanallı yapıdadır. Bu nedenle renkli görüntülerle çalışırken aynı işlemler kanal sayıları dikkate alınarak uygulanacaktır.



Şekil 4.14. Renkli görüntüde çıkış matrisinin oluşturulması

Burada; W1görüntü genişliği, H1 görüntünün yüksekliği, D1 görüntünün derinliği yani kanal sayısı olmak üzere giriş görüntüsü W1xH1xD1 olarak ifade edilebilir. Bu

doğrultuda çıkış görüntüsü ise W2xH2xD2 olacaktır. Burada adım mesafesi S, filtre boyutları FW ve FH, filtre sayısı K, piksek ekleme P olmak üzere Formül 4' den yararlanarak W2=(W1-FW+2*P)/S+1, H2=(H1-FH+2*P)/S+1 ve D2=K şeklinde bulunur. Yani W2=(7-3+2x1)/3+1=3, H2=(7-3+2x1)/3+1=3 ve D2=2 olarak hesaplanacaktır. Böylece çıkış matrisi boyutu 3x3x2 olur.

Son olarak her filtredeki ağırlık sayısı FWxFHxD1 formülüyle, toplam ağırlık sayısı FWxFHxD1xK formülüyle ve bias sayısı K'ya eşit olarak hesaplanır [46].

4.4.3. Aktivasyon Katmanı

Konvolüsyon katmanından sonra bir aktivasyon katmanını uygulanır. Bu katman doğrusal olmayan bir fonksiyon ihtiva eder. YSA'lar için tanh, sigmoid gibi fonksiyonlar tercih edilirdi. CNN uygulamalarında daha hızlı ve yakın doğrulukta sonuçlar alınmasıyla işlem maliyetini düşürdüğü için ReLU fonksiyonu tercih edilmeye başlandı. ReLU tüm negatif değeleri sıfıra eşitlerken pozitif değerleri değiştirmez [47].

$$ReLU(x) = max\,(0, x) \qquad (5)$$

-5	0	5
6	3	-1
2	-2	2

0	0	5
6	3	0
2	0	2

Şekil 4.15. ReLU

4.4.4.Görüntü Küçültme (Pooling) Katmanı

Görüntü boyutu sonraki katman için büyükse eğer yapılan boyut küçültme işlemine pooling denir. Alt örnekleme olarak da ifade edilebilir. Bu katmanda bir öğrenme işlemi yapılmaz. Hesaplama karmaşıklığını azaltmak için görüntü boyutu küçültülür. Burada görüntü içerisindeki bazı önemli bilgilerinde kaybolması da olasıdır. Bu nedenle ağ başarısının azaldığı yönünde eleştirilere neden olmaktadır. Pooling katmanı yerine başka çözüm önerilerinin geliştirildiği çalışmalar mevcuttur [48].

En yaygın olarak kullanılan pooling çeşidi maksimum alma işlemidir. Adım aralığı ve filtre boyutu belirlenir. Maksimum pooling uygulanırken konvolüsyon katmanındakine

benzer şekilde adım adım gezilerek ilgili komşuluk içerisindeki en büyük piksel değeri tespit edilir. Bu değer çıkış matrisine yazılır.

120	180	150	100
60	80	20	80
1	0	5	10
100	25	75	50

180	150
100	75

Şekil 4.16. 2x2 filtre ve adım mesafesi 2 olan örnek

Burada; W1görüntü genişliği, H1 görüntünün yüksekliği, D1 görüntünün derinliği yani kanal sayısı olmak üzere giriş görüntüsü W1xH1xD1 olarak ifade edilebilir. Bu doğrultuda çıkış görüntüsü ise W2xH2xD2 olacaktır. Burada adım mesafesi S, filtre boyutları FW ve FH olmak üzere Formül 4' den yararlanarak W2=(W1-FW)/S+1, H2=(H1-FH)/S+1 ve D2=D1 şeklinde bulunur. Yani W2=(4-2)/2+1=2, H2=(4-2)/2+1=2 ve D2=1 olarak hesaplanacaktır. Böylece çıkış matrisi boyutu 2x2x1 olur.

4.4.5.Tam Bağımlı Katman (Full Connected Layer)

Bu katman ağın son katmanıdır ve sınıf sayısı kadar çıkış verecek şekilde ayarlanır. Yani CNN'in sınıflandırmayı yapan katmanıdır. Bu katmanda çıkışı elde edebilmek için girişteki kategori sayısı ile aynı boyutta olan bir filtre kullanılır. Kategori sayısı ağa uygulanacak filtre sayısına(K) eşittir. Böylece bu katmanın çıkışı 1x1xK boyutunda olur.

Tam bağımlı katmanı bir örnekle açıklayalım. Ağı kedi, köpek, masa, insan ve at fotoğraflarıyla eğitelim. Ağırlıklar belirlendikten sonra test aşamasına geçelim. Giriş görüntüsü insan fotoğrafı olsun. 5 sınıflı bir ağın çıkış matrisi [0 .1 .1 .75 .05] benzer şekilde olacaktır. Burada sınıf-1 %0, sınıf-2 %10, sınıf-3 %10, sınıf-4 %75, sınıf-5 %5 ihtimalle olacağı anlaşılır. Yani %75 ihtimalle insan fotoğrafıdır sonucu çıkar. Burada 5 kategoride işlem yapıldığı için çıkış matrisinin boyutu da 1x1x5 olur.

Tüm bu katmanların birleşmesiyle konvolüsyon katmanı "KONV", aktivasyon fonksyonu "ReLU", pooling katmanı "POOL" ve tam bağımlı katman "TBK" ile ifade edilmek üzere genelleştirilmiş bir CNN;

[Giriş – KONV –ReLU – POOL - TBK]

şeklinde oluşturulabilir. Bu yapı bir katmanlı yapay sinir ağına benzetilebilir. Böylece CNN, orijinal görüntüyü değerlendirip görüntüden özellikler çıkararak kategorilere olan benzerliklerini tespit eder yani sınıflandırır denilebilir.

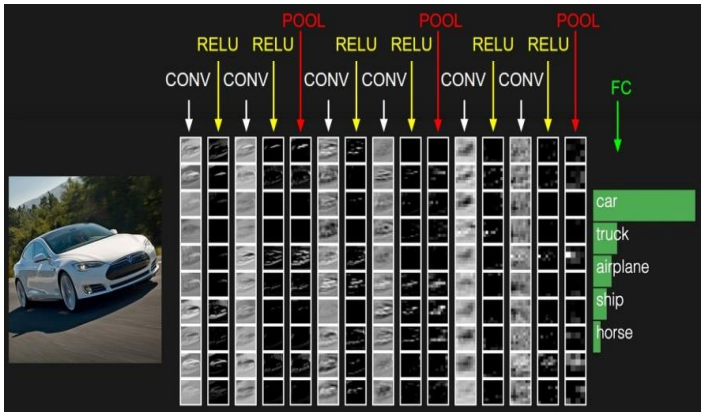
4.4.6.Katmansal Mimari

Şu ana kadar olan bölümde genelleştirilmiş bir CNN yapısı anlatıldı. CNN'ın çok katmanlı mimarisinden ve öznitelik çıkarmadaki başarısından birçok kez bahsedildi. Bu kısımda katmansal mimariyle birlikte CNN'in bu "derin" yapısı açıklanacaktır.

Genelleştirilmiş CNN yapısı tek katmanlı yapay sinir ağı olarak düşünülürse, katmansal mimariyle birlikte CNN çok katmanlı yapay sinir ağı olarak ifade edilebilir. Burada ilk katman basit öznitelikler çıkaracak, ikinci katman ilk katmandaki öznitelikleri kullanarak daha karmaşık öznitelikleri bulacaktır. Katman sayısının artmasıyla bulunan özniteliklerin karmaşıklıkları da artar ve böylece görüntüden çok detaylı bir analiz elde edilmiş olur.

N pooling olmayan katman sayısı ve M pooling olan katman sayısı olmak üzere aşağıda çok katmana sahip bir CNN mimarisi gösterilmiştir.

INPUT => [[KONV -> RELU]*N => POOL?]*M => [TBK -> RELU]*K => TBK



Şekil 4.17. Örnek CNN Uygulaması

CNN, çok katmanlı yapıda bilgi işleme yapması nedeniyle alt düzeydeki veri katmanlarından üst düzey öznitelik öğrenebilen hiyerarşik bir sistemi oluşturur.

4.4.7.Maliyet Fonksiyonu (Loss Function) Katmanı

Maliyet fonksiyonu ağın hata oranını ya da başka bir ifadeyle ağın başarısını ölçen fonksiyondur. Genellikle derin sinir ağlarının son katmanı maliyet fonksiyonu olarak tanımlanmaktadır. Bu fonksiyonun temel işlevi ağın yaptığı tahminle gerçek değer arasındaki farkı hesaplamaktır. Sinir ağının eğitimi tamamlandıktan sonra girdinin hangi kategoriye ne kadar benzediğini gösteren etiket vektörü oluşur. Maliyet fonksiyonu, bu eğitim etiket vektörüyle gerçek sonuçların bulunduğu vektör arasındaki farkı tüm değerler için hesaplar ve elde edilen hataları toplayarak sonucu bulur. Yani bu fonksiyon eğitim sonucunda elde edilen ağırlık ve bias değerlerinin problemin çözümü için ne kadar uygun olduğunu hesaplar. Bu nedenle ağ eğitimi başarılı olmazsa gerçek değerle tahmin edilen değer arasındaki fark büyük olacaktır.

Bu değeri hesaplayan birden çok lineer fonksiyon türü mevcuttur. En sık kullanılan fonsiyonlar Sigmoid, Multiclass Support Vector Machine (SVM), Softmax, Cross Entropy Likelihoo vb. dir. Sigmoid fonksyonu genellikle iki sınıftan oluşan problemlerin çözümünde kullanılır. Diğer fonksiyonlarsa çok sınıflı problemlerin çözümünde kullanılmaktadır.

SVM, orjinal boyutunda linear olarak çözülemeyecek problemleri daha büyük boyuta taşıyarak lineer olarak kolayca çözülebilecek hale getirmektedir. SVM bu özelliğini kullanarak model tarafından üretilen skoru normalize eder ve hata oranını hesaplar. SVM sonucunda en küçük değer sıfırken en büyük değer sonsuz olabilir.

Softmax fonksiyonu yapay sinir ağı tarafından üretilen skor değerlerini kullanarak olasılık temelli maliyet değeri üretmektedir. Softmax işlemi sonucunda girişin her bir sınıfa ait benzerliğini ifade eden olasılık değeri üretilir. Ağ modelinin çıktıları normalize edilmemiş değerlerdir. Softmax fonksyonu bu değerleri normalize ederek olasılık değerlerine dönüştürmektedir. Bu işlem istatistik yöntemlerinden birisi olan en çok olabilirlik (maximum likelihood) fonksiyonuyla yapmaktadır. Softmax sonucunda en küçük değer sıfırken en büyük değer bir olur. Bu nedenle derin sinir ağlarında daha çok tercih edilmektedir [49].

$$softmax(n) = exp(n)/sum(exp(n)) \quad (6)$$

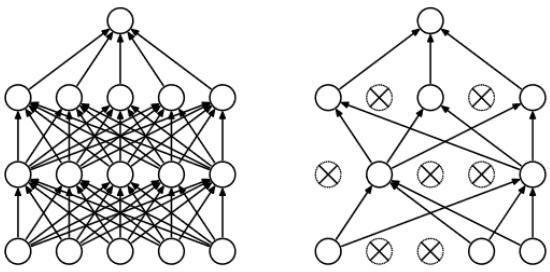
4.4.8.Nöron Seyreltme (Dropout Layer) Katmanı

Ara katmanlardan belli kurallara göre belli nöronların kaldırılmasıyla genel ağ yapısında iyileştirme yapılması işlemidir. Bu teknik genellikle tam bağlı katmanlardan sonra kullanılır. Bir eşik değer belirlenerek ve ya rastgele olarak bir sonraki katmandaki nöron sayısı bu değere göre azaltılmaktadır.

Bu yönteme 2014 yılındaki bir çalışmada "dropout" ismi verilmiştir. Ancak bu yöntem 2012 yılındaki AlexNet çalışmasında kullanılmıştır [50-51].

Bu yöntem bir örnekle şu şekilde açıklanabilir. Bir fabrika montaj hattında çalışan 20 çalışan olduğunu ve hepsinin ürün üzerinde farklı işlemler yaptığı bir senaryo olsun. Çalışanlar sadece kendi iş akışlarını yapacak şekilde eğitilmiş olsunlar. Bu durumda çalışanlardan birinin izin alması durumunda hattın devamı duracaktır ve iş ilerleyişi bu durumdan büyük oranda etkilenecektir. Ancak çalışanların diğer iş akışlarını da asgari seviyede öğrenmesi halinde gelmeyen (dropout yapılan) kişi yerine, diğerleri de bakarak iş akışını devam ettirebilir. Böylece izin alan kişi yerine başka bir kişi idareten bakarak genel iş ilerleyişinin durması engellenmiş olur.

Sinir ağında yapılan dropout işleminde ise birbirine bağlı olan nöronların ağırlık güncellemesi işleminde güncellenecek nöronun diğerinden az etkilenmesi sağlanır. Güncellenecek nöronun kendinden öncekine bağımlı olması durumu ortadan kaldırılmış olur. Böylece nöronlar birbiri hakkında az bilgiye sahip olurlar ve genel ağ daha tutarlı çalışabilir. Aynı zamanda her katmanda farklı kombinasyonlarla dropout olacağı için öğrenme başarısı da artmaktadır. Bu katman sayesinde zaman ve başarı açısından daha iyi performans almak mümkündür. Bu yöntem, derin öğrenme uygulamalarında en sık kullanılan iyileştirme yöntemlerinden biridir.



Şekil 4.18. Dropout

5. BÖLÜM

DERİN ÖĞRENME

5.1. Derin Öğrenme Hakkında

Büyük miktarlardaki etiketlenmiş eğitim verileriyle beslenen ve bu verileri yorumlayan, onlardan sonuçlar yani özellikler çıkarabilen çok katmanlı yapıdaki ileri seviye sinir ağları yapısına derin sinir ağı denir. Burada genellikle çok katmanlı mimari, hiyerarşik bir düzen içerisinde çalışmakta ve uyguladığı dönüşümlerle verileri anlamlandırmaktadır.

Derin sinir ağlarının kullanıldığı yapıların eğitim işlemine de derin öğrenme denir. Derin öğrenme aynı zamanda yapay öğrenmenin bir dalı olarak da sınıflandırılmaktadır. Başka bir tanıma göre derin öğrenme, çoklu işlem katmanlarından oluşan hesaplamalı modellerin çoklu soyutlama seviyeleri ile veri özelliklerini öğrenmesini sağlayan bir makine öğrenme algoritması sınıfıdır [52].

Bu bölümde son yıllarda literatürde ki en popüler derin öğrenme ağ mimarileri hakkında bilgiler verilecek ve çalışmada kullanılacak olan CNN mimarileri detaylı olarak incelenecektir.

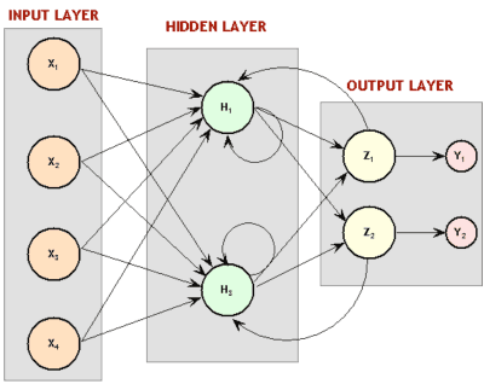
Derin sinir ağları konusunda yapılan literatür taramasına göre makine birimleri arasındaki her bağlantının bir daire oluşturduğu bir tür yapay sinir ağı olan Tekrarlayan sinir ağları konuşma ve el yazısı tanımada tercih edilmektedir. Yine el yazısı tanıma, konuşma tanıma, hareket tanıma, resimden altyazı tanımada uzun mesafeli bağımlılıkları öğrenebilmek için bilgileri depolayan özel bir RNN türü olan Uzun Kısa Vadeli Hafıza Ağları tercih edilir. Boltzmann Makinalarının bir türü olan Sınırlı Boltzmann Makineleri kendi katmanındaki nöronlarla bağlantısı olmayan hızlı ve girişler üzerinde olasılık dağılımını öğrenebilen bir rastgele yapay sinir ağıdır. Boyut

küçültme, sınıflandırma, filtreleme, özellik öğrenme ve konu modelleme çalışmalarında tercih edilir. Sınırlı Boltzmann Makinelerindeki her bir katmanın önceki ve sonraki katmanlarla bağlantılı olduğu özelleşmiş bir Boltzmann makinesi olan Derin İnanç Ağları etiketlenmemiş verilerin kümelendirilmesini ve bu kümelerin etiketlenmesiyle de sınıflandırma yapılmasını sağlar. NASA yüksek çözünürlüklü, çok çeşitli uydu görüntülerini sınıflandırmak için Derin İnanç Ağları ve CNN ile çalışmalar yapmıştır. Derin inanç ağları, görüntü tanıma, bilgi alımı, doğal dil anlayışı, hata tahmini alanlarında tercih edilir. Derin sinir ağlarının hesaplama maliyeti problemini çözmek için yalnızca bir gizli katmanı olan küçük alt ağlardan (modül) oluşan ve herhangi bir modülün çıktısı kıvrımlı bir mimari oluşturmak için daha üst katmanlara gönderilebilen böylece modüllerin tek tek ve ya paralel olarak eğitilebildiği Derin Yığın Ağları sürekli konuşma tanıma alanında tercih edilir.

5.2. Tekrarlayan Sinir Ağları (Recurrent Neural Network – RNN)

RNN, makine birimleri arasındaki her bağlantının bir daire oluşturduğu bir tür yapay sinir ağıdır. Geriye yayılım algoritmasının ortaya çıkmasından sonra keşfedilmiştir.

İnsanlar düşüncelerine her saniye sıfırdan başlamazlar. Bu çalışmayı okuduğunuzda, ilk kez okuduğunuz bir kelimeyi daha önceden öğrendiğiniz kelimeler yardımıyla anlayabiliyorsunuz. Yani daha önce öğrenilen bilgiler hafızamızda depolanıyor ve yeni şeyler öğrenirken bu bilgilerden faydalanıyoruz. Geleneksel sinir ağların bu özelliğin eklenmesiyle RNN elde edilmiştir. RNN içerisinde bulundurduğu döngüsel yapısıyla bilginin önceki bilgiler kullanılarak devam etmesini sağlamaktadır. RNN çoğunlukla konuşma ve el yazısı tanımada tercih edilmektedir [53].



Şekil 5.1. RNN

Şu anki görevi yerine getirmek için yalnızca son bilgilere bakmanın yeterli olduğu bir senaryoda ilgili bilgi ile ihtiyaç duyulan arasındaki mesafenin azdır. RNN' ler mesafenin az olduğu senaryolarda geçmiş bilgileri kullanmayı öğrenebilir.

Ancak iki kelime arasındaki mesafenin artması RNN' lerin bilgiyi öğrenememesi problemini ortaya çıkarmıştır. Bu problemi çözebilmek için RNN in özel bir hali olan Uzun Kısa Vadeli Hafıza Ağları (Long short-term memory, LSTM) fikri ortaya çıkmıştır [54].

Bu ağ uzun mesafeli bağımlılıkları öğrenebilmek için bilgileri depolayan özel bir RNN türüdür. RNN deki tekrarlama modülünün 4 adımlı bir yapı kullanılarak özelleştirilmesiyle elde edilmiştir. Hochreiter & Schmidhuber (1997) tarafından tanıtılmıştır. LSTM, el yazısı tanıma, konuşma tanıma, hareket tanıma, resimden altyazı tanımada tercih edilir.

5.3. Sınırlı Boltzmann Makineleri (Restricted Boltzmann Machines - RBM)

Sınırlı Boltzmann makineleri, 1986'da Harmonium adıyla ilk kez ortaya çıkmıştır. RBM, Boltzmann Makinalarının bir türüdür [55]. Ancak 2006'da Geoffrey Hinton ve arkadaşları tarafından yapılan çalışmayla hızlı bir öğrenme algoritması olarak literatürdeki yerini almıştır. RBM, boyut küçültme, sınıflandırma, filtreleme, özellik öğrenme ve konu modelleme çalışmalarında tercih edilir.

RBM' ler, derin inanç ağlarının yapı taşlarını oluşturan ve iki katmana sahip olan sinir ağlarıdır. RBM' nin ilk katmanı görünür veya giriş katmanı, ikinci katmanıysa gizli katman olarak adlandırılır. Katmanlar içerisinde bulunan nöronlar kendinden sonraki katmanla ilişkilidir. Kendi katmanındaki diğer nöronlarla bağlantıya sahip değildir. Diğer bir deyişle, katman içerisinde iletişim yoktur. Bu Sınırlı Boltzmann Makinesi'ndeki sınırlamadır. Ancak Kısıtlamasız Boltzmann Makineleri gizli birimler arasında da bağlantıya sahiptir.

RBM'de her nöron girişi işleyen bir hesaplama birimidir ve bu girdiyi iletip iletmeyeceğine dair stokastik (rastgele belirlenmiş) kararlar verir. Bu durumda girdileri değiştiren ağırlık katsayıları da rasgele başlatılır. Yani RBM, girişler üzerinde olasılık dağılımını öğrenebilen bir rastgele yapay sinir ağıdır [56].

5.4. Derin İnanç Ağları (Deep Belief Networks – DBN)

Derin inanç ağı; RBM' deki her bir katmanın önceki ve sonraki katmanlarla bağlantılı olduğu özelleşmiş bir Boltzmann makinesidir. Ancak burada da RBM' deki gibi aynı katmandaki nöronlar arası bağlantı yoktur. Derin inanç ağı RBM yığını olarak da adlandırılmaktadır. 2006'da Geoff Hinton ve öğrencileri tarafından tanıtılmıştır [57].

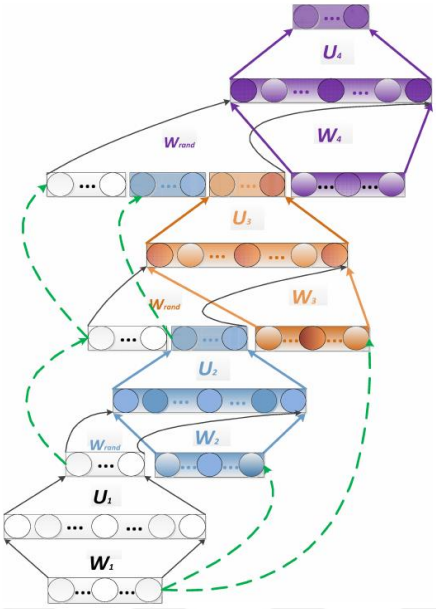
DBN, etiketlenmemiş verilerin kümelendirilmesini ve bu kümelerin etiketlenmesiyle de sınıflandırma yapılmasını sağlar. NASA yüksek çözünürlüklü, çok çeşitli uydu görüntülerini sınıflandırmak için DBN ve CNN ile çalışmalar yapmıştır [58].

Derin inanç ağları, görüntü tanıma, bilgi alımı, doğal dil tanıma, hata tahmini alanlarında tercih edilir.

5.5. Derin Yığın Ağları (Deep Stacking Networks – DSN ve ya Deep Convex Net – DCN)

Derin Sinir Ağı' nın en büyük problemlerinden biri daima zor öğrenme prosedürü olmuştur. Birden fazla gizli katmandan oluşan bir sinir ağı eğitimi ile ilgili hesaplama gücü, her ek katmanla birlikte katlanarak artar. Dolayısıyla, bu hesaplama maliyeti bir problem teşkil etmektedir. Bu kısıtlamayı ortadan kaldırmak için Deng ve Yu, derin sinir ağlarının orijinal yaklaşımından biraz farklı bir mimari ile 2011'de Derin Yığın Ağlarını tanıttılar [59].

Derin Yığın Ağı; derin bir mimari olarak kabul edilmekle birlikte, aslında yalnızca bir gizli katmanı olan küçük alt ağlardan oluşur. Burada her alt ağ bir modül olarak isimlendirilir. Herhangi bir modülün çıktısı kıvrımlı bir mimari oluşturmak için daha üst katmanlara gönderilebilir. Yapısı gereği her bir modül tek tek ve ya paralel olarak eğitilebilir. Hiçbir modül bir önceki modülün çıkışına bağlı değildir. Bu yapı sayesinde paralel işlemler GPU'lar üzerinde kolayca gerçekleştirilebilir. Derin Yığın Ağları, sürekli konuşma tanıma alanında tercih edilir [60].



Şekil 5.2. Derin Yığın Ağı

5.6. Konvolüsyonel Sinir Ağları (Convolutional Neural Networks - CNN)

Önceki bölümde CNN hakkında detaylı bilgi verilmiştir. Burada farklı CNN mimarileri tanıtılacaktır.

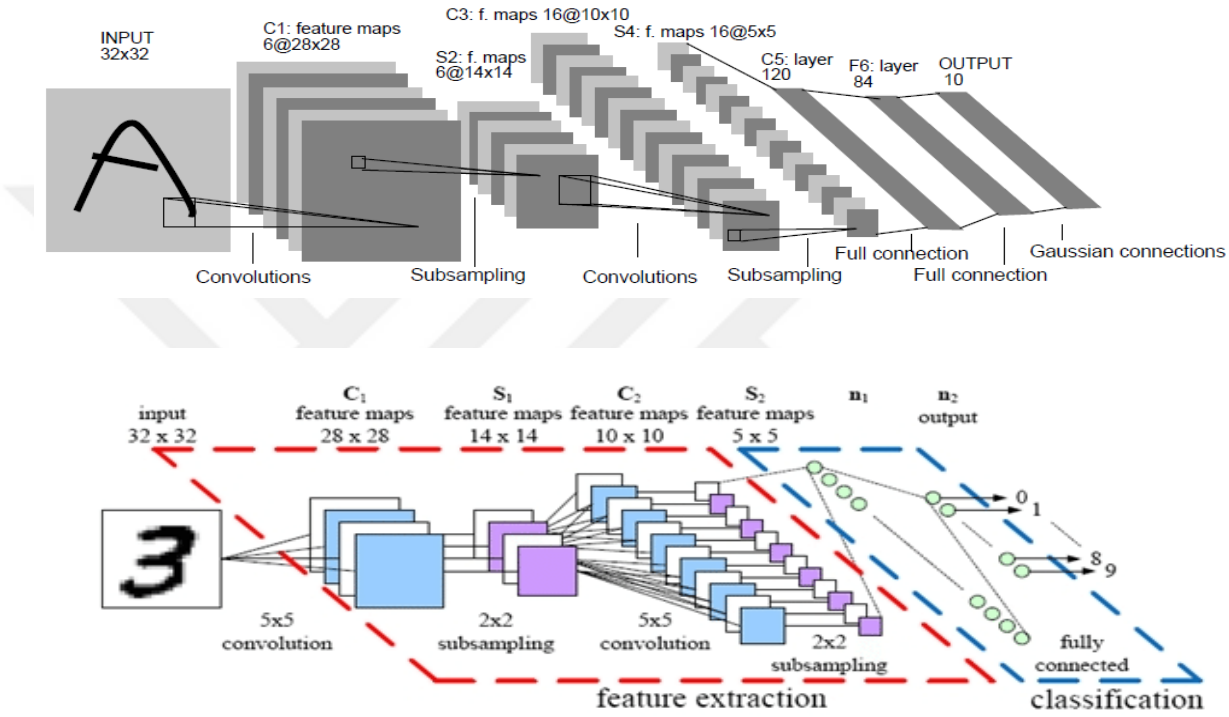
5.6.1.LeNet5

LeNet5, Yann LeCun ve arkadaşlarının 1998'de yayınladıkları makale ile ortaya çıkmıştır. Bu çalışma konvolüsyonel mimariyi yedi katmanlı(giriş hariç) bir ağ yapısı halinde karakter tanıma problemine uygulamıştır. Bu karmanlar; giriş-konvolüsyon-pooling-konvolüsyon-pooling-tam bağımlı katman - tam bağımlı katman - gaussian bağımlı katman olarak sıralanabilir. Giriş görüntüsü 32x32 pikseldir. Bu çalışmada MNIST veri tabanını kullanarak karakter tanıma işlemi yapılmıştır. MNIST el yazısı tanıma için aynı boyuta getirilmiş ve bazı önişlemler uygulanmış 60000 eğitim ve 10000 test görüntüsü içeren NIST'ten temin edilmiş en yaygın kullanılan veri tabanlarından biridir [61].

Birinci konvolüsyonel katmanda 6 farklı 5x5 filtre adım mesafesi 1 olarak uygulanarak 28x28 piksellik 6 adet özellik haritası elde edilir. Birinci pooling katmanında 2x2 lik filtre kullnılır ve 6 adet 14x14 piksellik çıkış matrisleri elde edilir.

İkinci konvolüsyonel katmanda 16 farklı 5x5 filtre adım mesafesi 1 olarak uygulanarak 10x10 piksellik 16 adet özellik haritası elde edilir. İkinci pooling katmanında 2x2 lik filtre kullnılır ve 16 adet 5x5 piksellik çıkış matrisleri elde edilir.

Birinci tam bağımlı katman çıkışında 1x120, ikinci tam bağımlı katman çıkışında 1x84 ve gaussian bağlı katman çıkışında 1x10 boyutunda çıkış matrisi elde edilir.

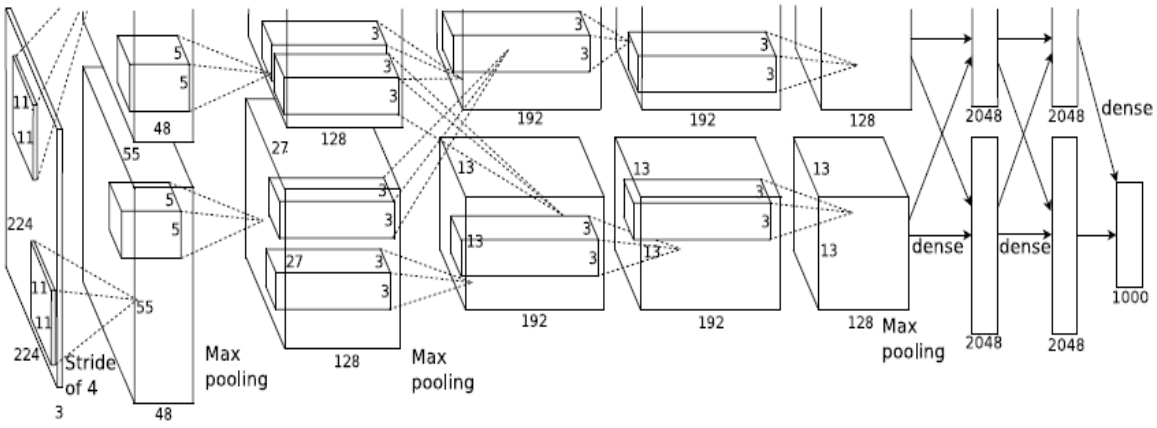


Şekil 5.3. LeNet

5.6.2. AlexNet (2012)

2012 de Alex Krizhevsky, Ilya Sutskever ve Geoffrey Hinton, 2012 ILSVRC'yi kazanmak için AlexNet adında derin bir konvolüsyonel sinir ağı tasarladılar. AlexNet, %15.4' lük test hatası oranıyla yarışmanın kazananı oldu [51].

Alexnet, 25 katmanlı bir CNN ağ yapısıdır ve 1000 farklı kategorideki yüksek çözünürlüklere sahip 1.200.000 görüntüyü sınıflandırmak için çalıştı. Yapısı içerisinde 60.000.000 parametre kullanmakta ve 650.000 nöron bulundurmaktadır. Beş konvolüsyon katmanı ve 1000 kategori için üç tam bağımlı katman içermektedir. Ağın eğitimi için GPU kullanılmıştır. Tam bağımlı katmanlarda dropout yöntemi uygulanmıştır.



Şekil 5.4. AlexNet

Bu katmanlı mimari aşağıdaki gibidir.

1. Giriş katmanı. 227x227x3 boyutuna dönüştürülen görüntü bu katmana uygulanır.

2. Konvolüsyon katmanı. Adım mesafesi 4 olmak üzere 96 adet 11x11x3 filtre kullanılarak 55x55x96 boyutlu çıkış matrisi üretilir. Bu katmanda 55*55*96=290400 nöron, 11*11*3*96=34848 ağırlık değeri, 96 adet bias değeri yer almaktadır.

3. ReLU katmanı.

4. Normalizasyon katmanı. Eleman başına 5 kanal ile çapraz kanal normalizasyonu uygulanır.

5. Maksimum pooling katmanı. 3x3 filtre ile adım mesafesi 2 olarak uygulanır.

6. Konvolüsyon katmanı. 256 adet 5x5x48 boyutlu filtre adım sayısı 1 olarak uygulanmıştır. Burada [2 2] piksel ekleme uygulanır. Bu katmanda 27*27*256=186624 adet nöron, 5*5*48*256 = 307200 ağırlık ve 256 adet bias değeri vardır.

7. ReLU katmanı.

8. Normalizasyon katmanı. Eleman başına 5 kanal ile çapraz kanal normalizasyonu uygulanır.

9. Maksimum pooling katmanı. adım sayısı 2 olarak 3x3 filtreyle maksimum pooling uygulanır.

10. Konvolüsyon katmanı. 1 adım sayısı ve 1 piksel ekleme uygulanarak 384 adet 3x3x256 alt eleman elde edilir. Bu katmanda 13*13*384=64896 adet nöron, 3*3*256*384=884736 adet ağırlık ve 384 adet bias değeri vardır.

11. ReLU katmanı.

12. Konvolüsyon katmanı. 1 adım mesafesi ve 1 piksel ekleme uygulanarak 384 adet 3x3x192 alt eleman elde edilir. Bu katmanda 13*13*384=64896 adet nöron, 3*3*192*384=663552 ağırlık ve 384 bias değeri vardır.

13. ReLU katmanı.

14. Konvolüsyon katmanı. [1 1] adım sayısı ve [1 1] doldurma uygulanarak 256 adet 3x3x192 alt eleman elde edilir. Bu katmanda 13*13*256=43264 adet nöron, 3*3*192*256=442368 ağırlık ve 256 bias değeri vardır.

15. ReLU katmanı.

16. Maksimum pooling katmanı. [2 2] adım sayısı ve [0 0] doldurma uygulanarak 3x3 maksimum pooling uygulanır.

17. Tam bağımlı katman.9216 giriş uygulanarak 4096 çıkış elde edilir.. 4096*9216=37748736 ağırlık ve 4096 bias değeri vardır.

18. ReLU katmanı.

19. Dropout katmanı. %50 dropout oranı uygulanır.

20. Tam bağımlı katman.4096 giriş uygulanarak 4096 çıkışelde edilir. 4096*4096=16777216 ağırlık, 4096 bias değeri vardır.

21. ReLU katmanı.

22. Dropout katmanı. %50 dropout oranı uygulanır.

23. Tam bağımlı katman.4096 giriş uygulanarak 1000 çıkış elde edilir. 1000*4096=4096000 ağırlık, 1000 bias değeri vardır.

24. Softmax katmanı.

25. Çıkış katmanı.

6. BÖLÜM

TEST ÇALIŞMASI ve PRATIK UYGULAMA

6.1. Çalışma Hakkında

Bu çalışma adımlarını 3 ana başlık altında toplayabiliriz. İlk adım idrar striplerine ait görüntülerin elde edilmesi, ikinci adım bu görüntülerin klasik görüntü işleme adımlarından geçirilerek düzenlenmesi ve böylece veri setinin elde edilmesi, üçüncü adımsa veri setinin farklı algoritmalarla test edilmesi şeklinde ifade edilebilir.

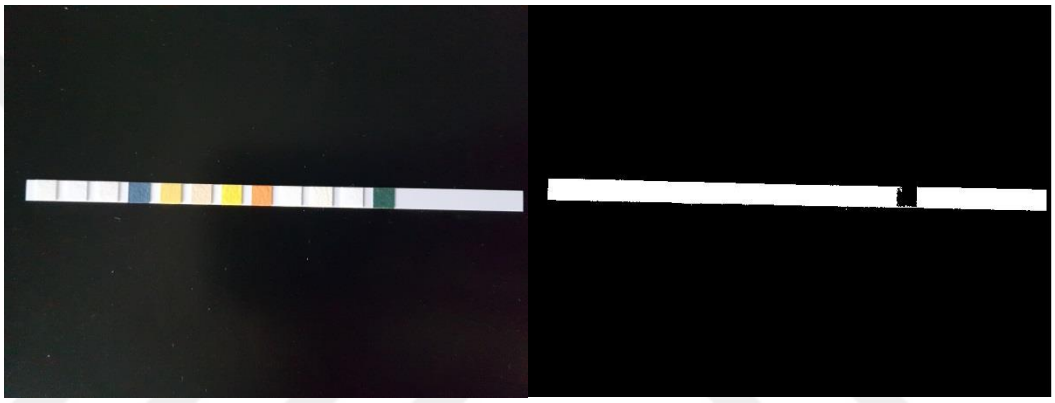
İlk adımı gerçekleştirmek için hazırlanan yazılım öncelikle boş striple çalıştırılarak renk sahaları için kalibrasyon işlemi sağlanır. Bu işlemin amacı şu an bulunulan ortam, kamera ve ışık şartlarında elde edilen strip görüntüsünün test görüntülerinde doğru sonuçlar vermesini sağlamaktır. Kalibrasyon işlemi aynı zamanda kullanılan görüntüdeki açısal dönüklükleri ve renk sahaları arasındaki mesafenin ölçümünü yapmak için de kullanılmaktadır. Böylece test aşamasında kullanılmak üzere tüm idrarlı strip görüntülerinden renk sahaları kesilerek veri seti oluşturulacaktır.

Görüntünün elde edilmesi aşamasında zıtlık oluşturması nedeniyle siyah renkte bir arka plan kullanılacaktır. Böylece stripin zeminden ayrılması işlemi kolayca yapılabilmektedir.

Kalibrasyon işleminde kamera yardımıyla elde edilen renkli görüntü öncelikle kırmızı yeşil ve mavi renk kanallarına ayırt edilir. Bunlardan mavi renk kanalı alınarak bulunduğu zeminden ayırt edilmek üzere gri seviyeli bir görüntüye dönüştürülür. Bu gri görüntüde uygulanan morfolojik işlemler ve belli bir eşik değer yardımıyla zeminle strip birbirinden keskin şekilde ayırt edilir.

Sadece stripi içeren yeni görüntü eğer açısal olarak bir dönüklüğe sahipse döndürülerek başlangıç noktası hep aynı yöne gelecek şekilde durması sağlanır.

Kalibrasyon işleminde strip üzerindeki küçük renkli alanların tespiti renk algılama algoritmaları kullanılarak yapılır ve iki renkli alan arasındaki boşluğun mesafesi piksel cinsinden hesaplanır. Burada strip üzerindeki iki mavi renkteki bölgelerden faydalanılmıştır. Böylece birkaç renkli alan kullanılarak tüm renkli bölgelerin koordinatları belirlenmiş olur/ Bu koordinat bilgileri ile bölgelerde bulunan renk yoğunluklarının ortalama değerleri bir dosya içerisine kalibrasyon bilgileri olarak kaydedilir ve bir sonraki kalibrasyon yapılana kadar bu değerler kullanılır.



Şekil 6.1. Kameradan alınan orjinal görüntü, Stribi arkaplandan ayırmak için kullanılacak olan iki seviyeli görüntü

Şekil 6.2. Kesme işleminden sonra elde edilen strip görüntüsü, Mavi bölgelerin tespiti

Şekil 6.3. Strip üzerinde analiz yapılan tüm bölgeler

Tablo 6.1. Kalibrasyon Görüntüsü Ortalama Değerleri

SIRA	Kırmızı	Yeşil	Mavi	Genel
1	58,9448	102,1793	58,9448	85,7191
2	251,8887	248,1373	247,3899	249,1386
3	243,3386	249,0450	249,9628	247,4488
4	252,2508	251,6149	252,60244	252,1567
5	112,4604	180,9373	252,6576	182,0185
6	95,5589	240,7614	253,58201	196,6335
7	208,9220	234,4034	251,8497	231,7250
8	142,1486	229,5436	253,0107	208,2343
9	164,6593	136,4583	102,9162	134,6779
10	247,7617	248,3914	249,2896	248,4809
11	251,9732	248,3527	247,0201	249,1153
12	245,5549	247,1131	246,0907	246,2529

Kalibrasyon işleminden sonra yeni örneklerde yazılım çalıştırılır ve strpiler arka plandan ayrılıp açısal dönüklükleri düzeltilerek kaydedilir. Bu adımlarda da kalibrasyondaki temel görüntü işleme adımları benzer şekilde uygulanır.

Strip elde edildikten sonra renkli alanların tespiti aşamasına geçilir ve burada hücresel sinir ağlarından faydalanılır. Kalibrasyon ve test görüntüsü arasındaki görüntü farklılıklarının sonucu etkilememesi için kenar tespiti yapılır. Burada HSA' lardan faydalanılmasındaki temel amaç kamera titremesi, odak kayması, mesafe değişimi gibi nedenlerle görüntüde oluşan farklılıkların dezavantajlarını ortadan kaldırmaktır. Sonucu

negatif etkileyen bölgeler inceleme dışı bırakılacaktır. Böylece kalibrasyon adımında tespit edilen bölge büyüklükleri ve koordinatları kullanılır ve test görüntüsündeki ilgili alanlar HSA' ya kenarların tespiti için sunulur. Kenarların kaymasındaki bir başka neden de görüntü kesme işleminin sadece tamsayılara karşılık gelen piksel değerleriyle yapılmasıdır. Kayan noktalı yerlere isabet eden koordinatlar ötelenmek zorundadır. Bu işlem sonucunda kenar bölgeler kesilir ve saf strip bölgeleri elde edilmiş olur.

Şekil 6.4. HSA Girişi, HSA Çıkışı

Renkli bir görüntünün HSA' ya giriş olarak sunulabilmesi için bazı dönüştürme işlemlerine ihtiyaç duyulmuştur. Öncelikle görüntü kırmızı, yeşil ve mavi kanallara ayrılmıştır. Bu matrisler double sayı tipine dönüştürülerek yeniden ölçeklenmiştir. Yeni ölçekleme [-1, 1] aralığında uygulanmıştır. Bu görüntüler ayrı ayrı HSA ya sunulmuştur. HSA otuz tekrar çalıştırılmıştır. Sonuçta elde edilen 3 kanal görüntü matris toplama işlemine tabi tutulmuştur. Bu adımdan sonra oluşan yeni görüntüde temel morfolojik işlemeler uygulanmış ve kenarlar net bir şekilde tespit edilmiştir. Bu kenarlar orijinal bölgelerden çıkarılarak saf strip bölgeleri elde edilmiştir. Böylece tüm görüntülere bu işlemler uygulanarak veri seti oluşturulmuştur.

6.2. Test Yöntemi 1

En temel test etme uygulaması olarak kalibrasyon işleminde elde edilen renk sahalarının ortalama değerleri, idrar damlatılmış renk sahalarından elde edilen renk sahalarının ortalama değerleriyle kıyaslanmasını kapsamaktadır. Bu işlem yapılırken fark değerlerinin mutlak değeri kullanılır. Strip üzerindeki her bir bölge için renk farkları toplamı bulunur. Bölgesel toplamların toplamı yani genel toplam elde edilerek bölge sayısına bölünür ve ortalama toplam renk farkı elde edilir. Ortalama değerin üzerinde olan bölgesel toplam değerleri o bölgedeki renk değişiminin bulunduğunu ispatlar.

Tablo 6.2. Test görüntüsü için elde edilen sonuçlar

SIRA	Test Ortalaması	Kalibrasyon Ortalaması	Mutlak Fark
1	162,7637	85,7191	77.0446
2	232,8929	249,1386	16.2457
3	216,6866	247,4488	30.7622
4	222,0550	252,1567	30.1017
5	198,2889	182,0185	16.2704
6	213,7547	196,6335	17.1212
7	226,6936	231,7250	5.0314
8	204,9788	208,2343	3.2555
9	208,7902	134,6779	74.1123
10	221,3575	248,4809	27.1234
11	198,2717	249,1153	50.8436
12	185,9636	246,2529	60.2893

Bu analizde toplam fark değeri 898 olarak ve ortalama toplam fark değeri ise 408,201/12=34,01678 olarak elde edilmiştir. 34,01678 üzerinde olan toplam farklar tabloda işaretlenmiştir. Bu bölgelerde analiz pozitif olarak sonuç vermiştir.

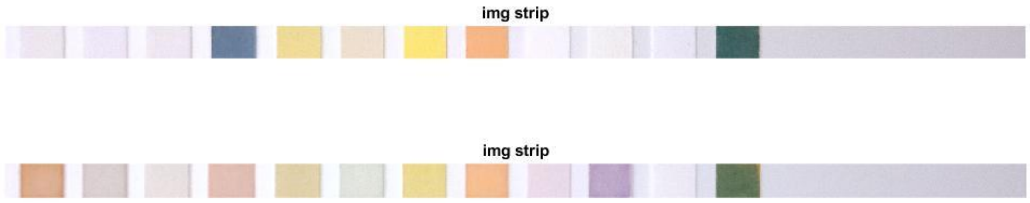
Şekil 6.5. Orijinal strip görüntüsü, Test stripi görüntüsü

Örnek iki:

Tablo 6.3. İkinci örnek için elde edilen sonuçlar

SIRA	Test Ortalaması	Kalibrasyon Ortalaması	Mutlak Fark
1	102.1028	81.5159	20.5869
2	239.5217	238.0471	1.4746
3	186.9699	241.4725	54.5026
4	229.3324	249.5449	20.2125
5	193.3261	188.6576	4.6685
6	201.0244	202.5799	1.5555
7	225.5699	223.6719	1.898
8	202.2329	204.3021	2.0692
9	199.5624	130.8456	68.7168
10	227.8332	237.6741	9.8409
11	210.7921	240.0839	29.2918
12	177.8366	236.0692	58.2326

Bu analizde ortalama toplam fark değeri ise 273.0499/12= 22.75416 olarak elde edilmiştir. 22.75416 üzerinde olan toplam farklar tabloda işaretlenmiştir. Bu bölgelerde analiz pozitif olarak sonuç vermiştir.



Şekil 6.6. Boş strip ve idrar uygulanmış strip

Pozitif olan değerleri kimyasal parametrelere dönüştürmek için strip üretici firmanın sağlamış olduğu renk aralığı tablosu kullanılacaktır [62].

Şekil 6.7. Renk Tablosu

Elde edilen sayısal değerlerin doktora sunulacak olan sözel ifadelere (bir pozitif, iki pozitif, üç pozitif ve ya dört pozitif) dönüştürmek için aşağıdaki dönüşüm tablosu kullanılacaktır.

Tablo 6.4. Parametre dönüşüm tablosu

	Parameters	neg.	+	++	+++	++++		
12	Bilirubin (mg/dl)	neg.	1	3	6			
11	Urobilinogen (mg/dl)	norm	2	4	8	12		
10	Ketones (mg/dl)	neg.	15	50	150			
9	Ascorbic acid (mg/dl)	neg.	20	40				
8	Glucose (mg/dl)	norm	50	150	500	1000		
7	Protein (mg/dl)	neg.	30	100	500			
6	Blood (Ery/ml)	neg.	ca. 5-10	ca. 50	ca. 300			
5	pH	5	6	7	8	9		
4	Nitrite	neg.	pos.					
3	Leukocytes (Leu/ml)	neg.	ca. 25	ca. 75	ca. 500			
2								
1	Specific Gravity	1000	1005	1010	1015	1020	1025	1030

Tablo 6.5. Örnek iki için doktora gönderilecek nihai sonuç:

20.5869	neg
1.4746	neg
54.5026	+++
20.2125	neg
4.6685	neg
1.5555	neg
1.898	neg
2.0692	neg
68.7168	+++
9.8409	neg
29.2918	+
58.2326	+++

Uygulanan bu yöntem pratik bir uygulamaya dönüştürülerek sonraki bölümde aktarılmıştır.

6.3. Test Yöntemi 2

Strip görüntülerinden HSA yardımıyla renk sahalarının çıkarılmasıyla veri seti oluşturulmuştur. Bu veri seti kullanılarak lökosit hücrelerinin yer aldığı 448 adet test stribi görüntüleriyle bir çalışma gerçekleştirilmiştir. Biyokimya Uzmanı Dr. Latif ERBAY ile birlikte 448 görüntü gözle analiz edilerek her birisi etiketlenmiştir. Bu etiketleme işlemi strip üretici firmanın sunduğu renk tablosundakine uygun olarak

lökosit bölümü için 4 farklı kategoride uygulanmıştır. Burada yapılan çalışma temel olarak bir kümeleme işlemini ifade etmektedir. 448 görüntü analiz edilerek 4 farklı grupta toplanmıştır. Her bir görüntü sadece bir kümeye aittir. Bu yaklaşımı kümeleme olarak ifade edilmektedir.

Bir veri setinde benzer özellikler gösteren verilerin gruplara ayrılması işlemi kümeleme olarak tanımlanabilir. Bir kümeleme işleminde aynı küme içinde benzerliklerin fazla, kümeler arası benzerliklerin az olması hedeflenmektedir. K-Means ve Fuzzy C-Means algoritmaları müşteri segmentasyonu, pazar segmentasyonu, bilgisayarlı görü gibi alanlarda sıkça kullanılırlar.

K-Means, 1967 yılında J.B. MacQueen tarafından geliştirilmiştir. Alogritmanın isminde yer alan k harfi küme sayısını belirtmektedir. Algoritma, hata hesaplamada yaygın olarak kullanılan Karasel Hata Fonksiyonunu en aza indirgemeyi hedeflemektedir. Böylece küme benzerliği kümedeki değerlerin ortalamaya yakınlıkları ile ifade edilmektedir. Bu ortalama kümenin ağırlık merkezidir.

K-Means algoritması ilk adımda k adet rastgele merkezleri belirler. Tüm elemanlar merkezlere olan mesafelerine göre kümelendirilir. Oluşan kümeler için yeni merkezler hesaplanır ve yeni merkezlere göre tekrar kümeleme yapılır. Bu şekilde istenilen hata oranı yakalanıncaya kadar algoritma çalıştırılır.

K-Means algoritması 448 görüntü üzerinde çalıştırılmış ve 4 grup elde edilmiştir. Bu dört grup için toplam 263 görüntü doğru tespit edebilmiştir. Yani K-Means la yapılan gruplama işleminde 448 görüntünün %58,7 si olması gereken grupta bulunabilmiştir.

Fuzzy Cmeans algoritması ilk olarak Dunn (1974) tarafından önerilmiş ve Bezdek (1981) tarafından geliştirilmiştir [63].

Bu yöntem, nesnelerin iki veya daha fazla kümeye ait olabilmesine izin verir. Bulanık mantıkta her bir veri, kümelerin hepsine [0,1] arasında değişen üyelik değeri ile bağlıdır. Bir veri için bağlı olduğu her bir sınıfın üyelik değerleri toplamı 1 olmalıdır. Böylece ilgili veri için en yakın olduğu küme merkezine ait olma olasılığı diğerlerinden daha yüksek olacaktır. Amaç fonksiyonun belirlenen minimum ilerleme değerine yakınsaklaşmasıyla kümeleme işlemi tamamlanır.

Fuzzy Cmeans algoritması da 448 görüntü üzerinde çalıştırılmış ve 244 tanesini doğru tespit edebilmiştir. Bu yöntemle yapılan gruplama işleminde 448 görüntünün %54,46 si olması gereken grupta bulunabilmiştir.

Bir kümeleme problemi olarak ele alınan strip görüntülerindeki renk sahalarının sınıflandırılması işleminde bu yaklaşımların yeterli doğrulukta sonuçlar veremediği gözlemlenmiştir.

6.4. Test Yöntemi 3

Konvolüsyonel sinir ağlarıyla yapılan çalışmalarda %90 ve üzeri ağ eğitim başarısı elde edilmiştir. Bu başarı değeri mimarinin test işlemi için %30luk rastgele test görüntüsü seçmesi nedeniyle değişkenlik gösterebilir. %91,67 başarıyla eğitilen bir ağ için; 134 test görüntüsüyle yapılan bir çalışma sonucunda elde edilen ortalama doğruluk yani ağın test başarısı verileri yüzde olarak aşağıdaki tabloda verilmiştir.

Tablo 6.6. CNN ile elde edilen ortalama doğruluk değerleri

Gerçek Kategoriler	Tahmini Kategoriler		
	Kategori 1	Kategori 2	Kategori 3
Kategori 1	%95.12	%4.88	%0.0
Kategori 2	%3.66	%89.02	%7.32
Kategori 3	%0.0	%9.15	%90.85

Bu çalışma sonuçları: Aralık 2018 ‘de Samsun ‘da gerçekleştirilen 2. Uluslararası Bilimsel Çalışmalarda Yenilikçi Yaklaşımlar Sempozyumu ‘nda sunulmuştur ve bildiri olarak yayınlanmıştır [64].

Derin sinir ağlarının bu probleme uygulanmasında stripte bulunan 12 bölgeden idrar analizi için anlamlı olan 11 renk bölgesi ele alınmıştır. Bu bölgelerin her biri, bir sınıflandırma problemi olarak değerlendirilmiş ve 11 farklı derin sinir ağı yapısı tasarlanmıştır. Ancak veri setinin yeterli olmaması nedeniyle tüm alanlarda şu an için başarılı öğrenme skorları sağlanamamıştır. Bir sistem halinde bu çalışmanın başarıya

ulaşabilmesi için her bir renk bölgesi için eğitim aşamasından başarılı sonuç alabilecek kadar görüntüye ihtiyaç duyulmaktadır. Bu çalışma ancak bir hastane ortamında tüm örneklerin toplanmasıyla birkaç yılda temin edilebilir.

Konvolüsyonel sinir ağlarıyla yapılan çalışmada veri seti AlexNet mimarisinin eğitim işlemi için kullanılacaktır. Görüntü çeşitliliğinin çok olması öğrenme kalitesini doğrudan etkilemektedir. Bu nedenle üretici firmanın sağlamış olduğu renk aralığı tablosundaki her bir kategori için onlarca farklı görüntüye ihtiyaç duyulmaktadır. Ancak bu kadar çok çeşitlilikte ve sonucu pozitif olan idrar örneği bulmak mümkün olmamıştır. Bu nedenle bu sınıflandırma çalışması en çok temin edilebilen lökosit hücrelerinin pozitif olduğu sonuçlar için uygulanacaktır.

Lökosit değerleri pozitif çıkan stripler için 4 bölgede sınıflandırma çalışması yapılacaktır. Negatif, mikrolitrede 25 lökosit, mikrolitrede 75 lökosit, mikrolitrede 500 lökosit olarak 4 renk sahasında sonuçlar verilecektir.

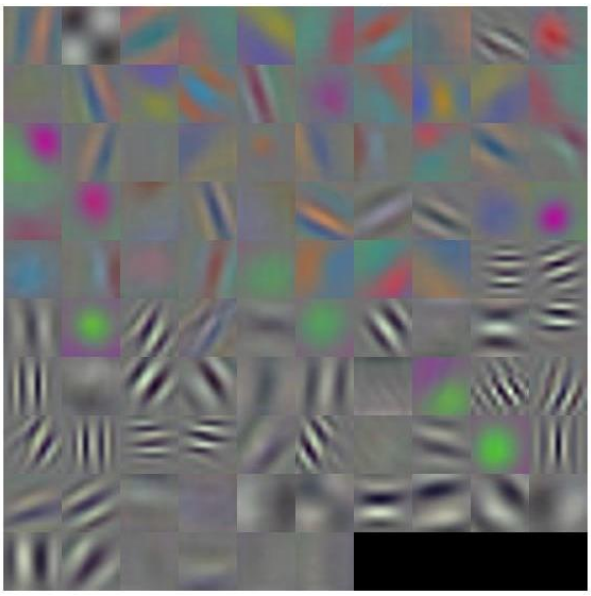
Bu çalışma sonucunda ağ eğitim kalitesi %90 ve üzeri olarak hesaplanmıştır. Bu kalite değeri hesaplanırken kullanılmak üzere veri setinin %30 luk kısmı eğitime başlamadan önce test işlemi için ayrılmıştır. Eğitim sonunda elde edilen sınıflandırma sonuçlarıyla gerçek sınıflandırma sonuçlarının kıyaslanması sonucu bu değer elde edilir. Yani doğru tahmin edilen görüntülerle yanlış tahmin edilen görüntülerin yüzdelik değerlerinin ortalaması elde edilir. Buda ağın genel başarı skorunu ortaya çıkarır.

Hücresel sinir ağları ve derin sinir ağları mimarilerinin hızlı çalışabilmesi için Intel Core i7 işlemciye, Nvidia 840M grafik işlemciye ve 8GB Ram bellek kapasitesine sahip bir bilgisayarda çalışmalar yürütülmüştür. Derin sinir ağları algoritmaları zamandan tasarruf etmek adına genelde grafik işlemcilerde çalıştırılmaktadır. Bu çalışma kapsamında Matlab paket programında algoritmalar hazırlanmış ve çalıştırılmıştır. Hücresel sinir ağlarını kullanmak için MatCnn paketinden, derin öğrenme algoritmalarını kullanmak içinde AlexNet paketinden faydalanılmıştır.

AlexNet in çalıştırılması için hazırlanan veri seti ve gerçek sonuçları içerecek şekilde alt klasörler halinde etiketlenmiştir. Bu etiketleme işlemini yaparken biyokimya uzmanlarından ve laborantlardan yardım alınmıştır. Bu etiketleme işlemleri sonucunda elde edilen görüntüler %30 oranında test amaçlı olarak ayrılmıştır. Bu ayırma işlemi de

rastgele olarak uygulanmıştır. Bu nedenle aynı veri tabanıyla yapılan farklı zamanlardaki eğitim işlemleri farklı başarılarda sonuçlar vermektedir. 24 katmanlı yapısıyla AlexNet derin öğrenme sağladıktan sonra son katmanda da sınıflandırma yapmakta ve sonuç vermektedir. Çok sınıflı lineer bir sınıflandırıcı olan SVM kullanılarak eğitim işlemi uygulanmıştır. Eğitim sonucunda elde edilen ağırlık değerleriyle test için ayrılan %30 'luk veri çalıştırıldığında sistem başarısı tespit edilmektedir. Böylece yüzde olarak ağın başarısı matematiksel olarak ifade edilebilmektedir. Burada işleme alınan her bir kategori için test verilerinde doğru tahmin edilen ve yanlış tahmin edilen sonuç sayısını içeren bir karşılaştırma matrisi elde edilir. Tüm sınıflar için bu karşılaştırma matrisleri doğruluk oranları hesaplanarak genel ortalama doğruluk değeri elde edilir.

Katmanlarda çok fazla giriş çıkış ve ağırlık parametreleri yer almaktadır. Örneğin ağın ikinci katmanı olan ilk konvolüsyon katmanında 11x11x3x96, ağın altıncı katmanı olan ikinci konvolüsyon katmanında 5x5x48x256 boyutunda bir ağırlık matrisi kullanılmaktadır. Bu mimarinin derin olmasını sağlayan çok parametreli ve çok katmanlı yapısıdır. Aşağıda ikinci katmanda elde edilen 96 adet 11x11x3 boyutundaki ağırlıkların görsel olarak gösterimi yer almaktadır. Diğer katmanlar boyutundan kaynaklı olarak AxBx3 şeklinde gösterimi mümkün olmadığı için görsellere dönüştürülememiştir.



Şekil 6.8. İkinci konvolüsyon katmanının görselleştirilmiş hali

6.5. Pratik Uygulama

Sistemin çalışacağı donanımsal ve yazılımsal altyapı hakkında genel bilgiler verilecektir.

Raspberry Pi tek baskı devre kartı üzerine yerleştirilmiş mini bir bilgisayardır. İngiltere de bulunan Raspberry Pi Vakfı tarafından desteklenir. Bu ürün 2009 yılından beri geliştirilmektedir ve ilk satışı 29 Şubat 2012'de başlamıştır. O zamandan beri birçok modeli çıkmıştır. Bu kartlar farklı firmalar tarafından da kopyalanarak değişik isimlerde piyasaya sürülmüştür.

Bu kart temel bir bilgisayarın ihtiyacı olan tüm birimleri üzerinde bulundurmaktadır. Arm firmasına ait işlemcileri kullanmaktadır. Hdmi girişi sayesinde görüntü aktarma yeteneğine sahiptir. Ayrıca Display Serial Interface (DSI) soketi yardımıyla da görüntü aktarımı sağlanabilir. Farklı modellerde sayısı değişmekle birlikte çevre birimlerinin bağlantısı için Usb girişleri bulunan ve dahili giriş çıkış birimlerine sahip olan kredi kartı büyüklüklerinde bir karttır. Giriş çıkış birimleri fonksiyonel olarak programlanabilmekte ve böylece UART, I²C, SPI gibi haberleşme protokollerinin de desteklemektedir. Genel amaçlı olarak sunulan giriş çıkış birimleri buton kontrol etmek, motor sürmek, röle anahtarlama gibi birçok işlemde kullanılabilir.

Raspberry Pi kartları Linux tabanlı Debian, Fedora, Arch Linux ve türevleri gömülü sistem işletim sistemlerini çalıştırma yeteneğine sahiptir. Yeni modeller Windows IoT ile de uyumlu şekilde çalışmaktadır. Kart işletim sistemini çalıştırmak ve depolama birimi olarak kullanmak için Sd kart girişi kullanmaktadır. Minimum alanı 4gb olan ve sınıf 4 Sd kart kullanılması tavsiye edilmektedir. Bunun yanında ağ bağlantısı için Rj45 standardındaki Ethernet girişi de dahili olarak gelmektedir. Yeni modellerde kablosuz bağlantı da desteklemektedir. The Camera Serial Interface (CSI) soketi sayesinde de Raspberry için geliştirilmiş 5 megapiksellik kameraya doğrudan bağlanarak görüntü alabilmektedir.

Kart üzerinde durum bildirimleri vermesi için ledler yerleştirilmiştir. Kart aynı zamanda dâhili olarak analog ses çıkışına da sahiptir. Kart beslemesi için mikro Usb girişi kullanılmaktadır. Uygulamaya göre farklı işletim sistemleri ve farklı çevre birimlerinin

kullanılmasına izin verecek şekilde tasarlandığı için esnek bir karttır. Bu da popüler olarak kullanılmasını sağlayan etkenlerden birisidir.

Bu kartın çalışmada tercih edilme sebebi kompakt bir cihaz tasarlanabilmesi için küçük ama yeterli performansta çalışabilen bir karta ihtiyaç duyulmasıdır. Ayrıca kullanılacak programlama dilini desteklemesi, kütüphanelerle uyumlu şekilde çalışabilmesi ve yeterince basit olması hedeflenmiştir. Rasbian dağıtımı Python 2 ve Opencv 2 kütüphanelerine destek vermektedir.

Bu çalışmada Raspberry Pi kartları için geliştirilmiş Debian tabanlı Rasbian işletim sistemi tercih edilecektir. Usb kamera yardımıyla strip görüntüleri elde edilecek ve Python programlama dilinde OpenCv kütüphaneleri kullanılarak görüntü işleme algoritmaları geliştirilecektir [65-66].

Python, Guido Van Rossum adlı Hollandalı bir programcı tarafından 90'lı yılların başında geliştirilmeye başlanmış C, C++, Perl, Ruby ve benzerleri gibi bir programlama dilidir. Python' un diğer dillere kıyasla basit ve temiz bir söz dizimine sahip olması program yazmayı ve başkasının yazdığı bir programı okumayı kolaylaştırır. Python programlama dilinin diğer bir çok dile göre en büyük avantajı, derlenmeye gerek olmadan çalıştırılabilmesidir. Yani derleme işlemine ihtiyaç olmadığı için program geliştirme süreci daha hızlıdır. Günümüzde Google, Dropbox, YouTube ve Yahoo gibi birçok büyük şirket Python dilinde kod geliştirmektedir [67].

NumPy kütüphanesi, Python' da bilimsel hesaplamalarda kullanılan bir pakettir. Çok boyutlu diziler oluşturarak bu diziler üzerinde matematiksel, mantıksal, sıralama, seçme, ayrık Fourier dönüşümü, temel cebir işlemleri, temel istatistik işlemler, rassal simülasyon gibi bir çok işlemi hızlı bir şekilde yapmayı sağlar. NumPy dizileri oluşturulurken sabit bir boyutta olur ve aynı veri türüne sahip olması gerekir. Bu kütüphane görüntülerin matrisler şeklinde saklanmasını ve kolayca işlem yapılmasını sağlayacaktır [68].

OpenCV (Open Source Computer Vision) 1999 yılında Intel tarafından geliştirilmeye başlanmış olan bir görüntü işleme kütüphanesidir. Açık kaynak kodlu olması sebebiyle ücretsiz olarak kullanılabilen platform bağımsız bir kütüphanedir. Windows, Linux, Android, MacOS ve iOS gibi birçok platformda çalışabilmektedir. Günümüzde

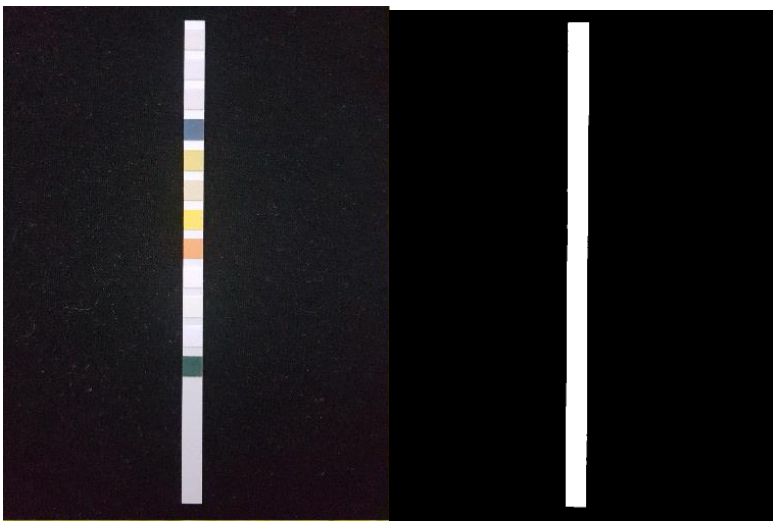
kütüphanenin büyük kısmı C++ dilindedir. Görüntü işleme ve makine öğrenmesi alanında 2500'den fazla algoritmaya sahiptir. Nesne sınıflandırma ve tespit etme, plaka tanıma, yüz tanıma, karakter tanıma, üç boyutlu görüntüler üzerinde işlem yapma gibi birçok uygulamada aktif olarak kullanılmaktadır [69].

6.6. Algoritma

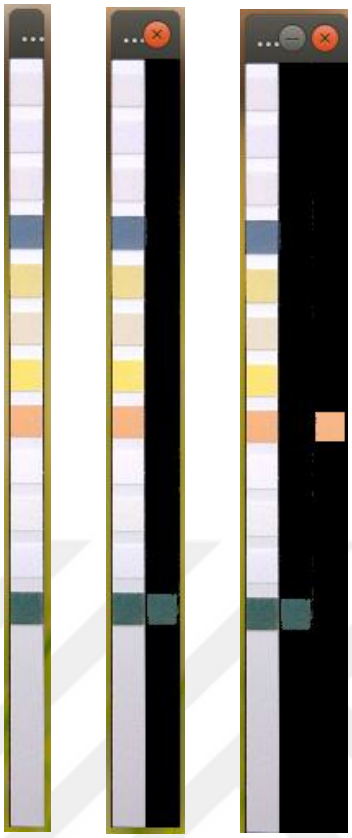
Hazırlanan yazılım kalibrasyon ve test olmak üzere iki aşamadan oluşmaktadır. Kabibrasyon adımında kullanılmamış strip renk verileri tespit edilir. Bu renk değerleri idrar damlatılan stripteki renk verileriyle kıyaslanacaktır. Görüntünün elde edilmesi aşamasında zıtlık oluşturması nedeniyle siyah renkte bir arka plan kullanılacaktır. Böylece stripin zeminden ayrılması işlemi kolayca yapılabilmektedir.

Kalibrasyon işlemi için boş strip sisteme konularak kalibrasyon programı çalıştırılır. Usb kamera yardımıyla elde edilen renkli görüntü öncelikle bulunduğu zeminden ayırt edilmek üzere gri seviyeli bir görüntüye dönüştürülür. Bu gri görüntüde uygulanan morfolojik işlemler ve belli bir eşik değer yardımıyla zeminle strip birbirinden keskin şekilde ayırt edilir.

Sadece stripi içeren yeni görüntü ihtiyaç duyulması halinde açısal olarak döndürülerek dikey bir şekilde durması sağlanır. Dikey hale getirmek algoritmanın çalışmasını daha tutarlı ve hızlı hale getirmektedir.

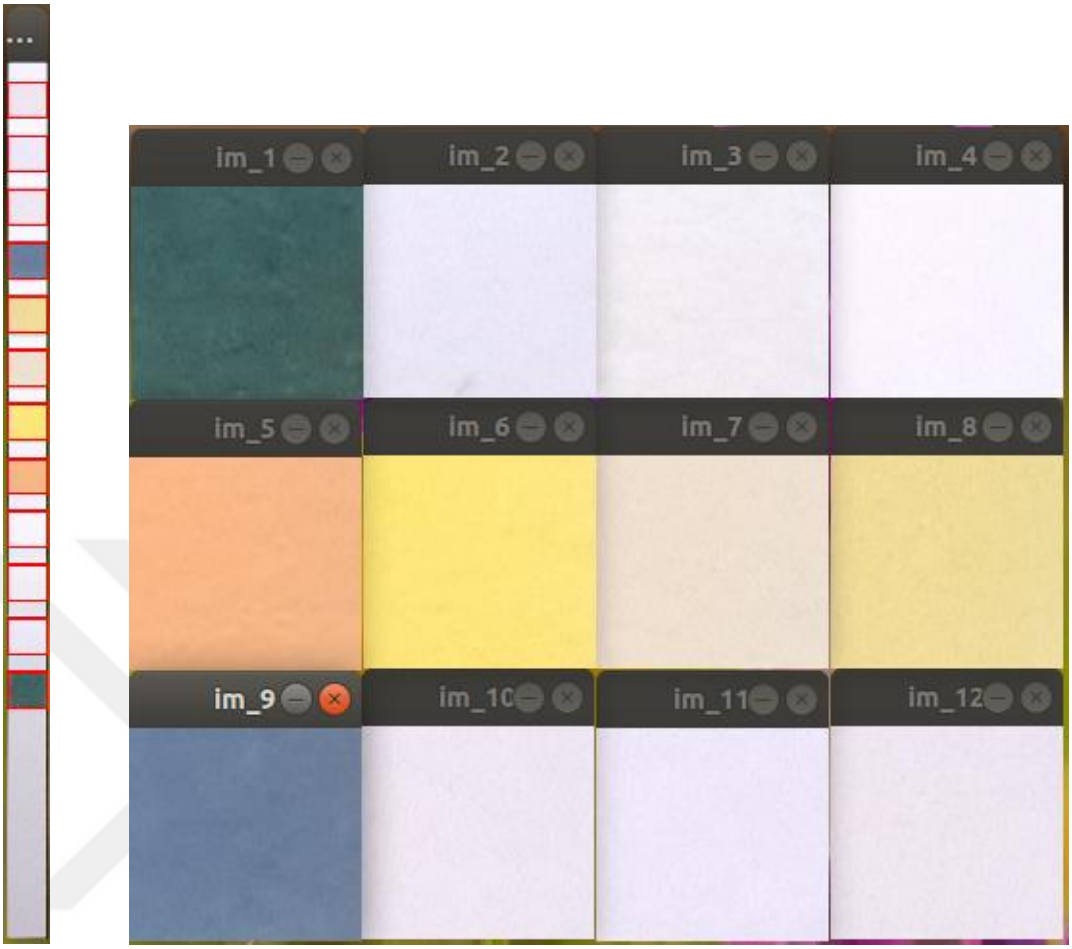


Şekil 6.9. Kameradan alınan orjinal görüntü, Stribi arkaplandan ayırmak için kullanılacak olan gri seviyeli görüntü



Şekil 6.10. Kesme işleminden sonra elde edilen strip görüntüsü, Mavi bölgenin tespiti, Turuncu bölgenin tespiti

Bu adımdan sonra strip üzerindeki küçük renkli alanların tespiti ve iki renkli alan arasındaki boşluğun mesafesi piksel cinsinden hesaplanır. Burada strip üzerindeki mavi ve turuncu renkteki bölgelerden faydalanılmıştır. Böylece birkaç renkli alan kullanılarak tüm renkli bölgelerin koordinatları belirlenmiş olur. Bu koordinat bilgileri ile bölgelerde bulunan renk yoğunluklarının ortalama değerleri bir dosya içerisine kalibrasyon bilgileri olarak kaydedilir ve bir sonraki kalibrasyon yapılana kadar bu değerler kullanılır.



Şekil 6.11. Strip üzerinde analiz yapılan tüm bölgeler

Kalibrasyon işleminden sonra yeni örneklerde analiz işlemi yapılabilir. Analiz adımlarında da kalibrasyondaki görüntü işleme adımları benzer şekilde uygulanır. Elde edilen ortalama renk değerleri kalibrasyon görüntüsündeki değerlerle kıyaslanarak renk değişimleri yani farklar tespit edilir. Bu işlem yapılırken fark değerlerinin mutlak değeri kullanılır. Strip üzerindeki her bir bölge için renk farkları toplamı bulunur. Bölgesel toplamların toplamı yani genel toplam elde edilerek bölge sayısına bölünür ve ortalama toplam renk farkı elde edilir. Ortalama değerin üzerinde olan bölgesel toplam değerleri o bölgedeki renk değişiminin bulunduğunu ispatlar.

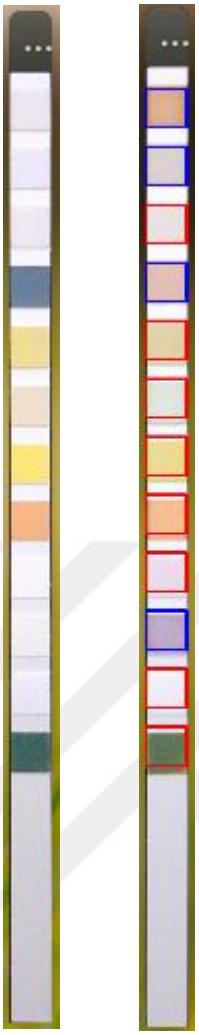
Tablo 6.7. Kalibrasyon görüntüsü ortalama değerleri

SIRA	B	G	R
1	97	100	69
2	246	231	235
3	243	237	242
4	254	244	249
5	136	183	246
6	127	230	253
7	208	224	238
8	155	220	239
9	156	127	108
10	242	231	239
11	249	232	238
12	240	229	237

Tablo 6.8. Pozitif sonuçlara sahip olan bir test örneği

SIRA	B	G	R	FARK B	FARK G	FARK R	Toplam Fark
1	88	115	96	9	15	27	44
2	246	236	239	0	5	4	9
3	202	170	192	41	67	50	158
4	234	220	236	20	24	13	57
5	147	191	246	11	8	0	19
6	146	218	236	19	12	17	48
7	222	229	226	14	5	12	31
8	169	209	224	14	11	15	40
9	182	192	227	26	65	119	210
10	227	226	234	15	5	5	25
11	209	208	219	40	24	19	83
12	140	172	220	100	57	17	174

Bu analizde toplam fark değeri 898 olarak ve ortalama toplam fark değeri ise 898/12=74.83 olarak elde edilmiştir. 74,83 üzerinde olan toplam farklar tabloda işaretlenmiştir. Bu bölgelerde analiz pozitif olarak sonuç vermiştir.



Şekil 6.12. Orijinal strip görüntüsü, Analiz görüntüsü yani doktor kontrolüne sunulacak olan sonuç görüntüsü (Kırmızı işaretli bölgeler negatif, Mavi işaretli bölgeler pozitif sonuçlar)

Pozitif olan değerleri kimyasal parametrelere dönüştürmek için strip üretici firmanın sağlamış olduğu renk aralığı tablosu kullanılacaktır. Elde edilen sayısal değerlerin doktora sunulacak olan sözel ifadelere (bir pozitif, iki pozitif, üç pozitif ve ya dört pozitif) dönüştürmek için dönüşüm tablosu kullanılacaktır. Bu işlemler sonucunda doktorun kontrolüne sunulacak olan nihai sonuç tablosu aşağıdaki gibi olacaktır.

Tablo 6.9. Sonuç tablosu

174	++
83	+
25	neg
210	+++
40	neg
31	neg
48	neg
19	neg
57	neg
158	+++
9	neg
44	neg

7. BÖLÜM

TARTIŞMA, SONUÇ ve ÖNERİLER

7.1.Tartışma

Lökosit hücrelerinin yer aldığı 448 adet test stribi görüntüleriyle yapılan çalışmalarda: Kmeans algoritması 448 görüntü üzerinden 263 tanesini (%58.7) doğru tespit edebilmiştir. Fuzzy Cmeans algoritması da 448 görüntü üzerinden 244 tanesini (%54.46) doğru tespit edebilmiştir. Konvolüsyonel sinir ağlarıyla yapılan çalışmalarda %90 ve üzeri ağ eğitim başarısı elde edilmiştir. Bu başarı değeri mimarinin test işlemi için %30 'luk rastgele test görüntüsü seçmesi nedeniyle değişkenlik göstermiştir. %91,67 başarıyla eğitilen bir ağ için; 134 test görüntüsüyle yapılan bir çalışma sonucunda elde edilen ortalama doğruluk yani ağın test başarısı verileri yüzde olarak birinci kategori için %95.12, ikinci kategori için %89.02 ve üçüncü kategori için %90.85 olarak elde edilmiştir.

Bu kıyaslama çalışmasında, CNN tabanlı mimarilerin sınıflandırma problemleri için uygun olduğu ve başarılı sonuçlar ürettiği sonucuna varılmıştır.

Pratik uygulamada elde edilen idrar analiz sonuçları 77 Elektronika marka LabUMat & UriSed model tam otomatik idrar analizörü cihazı ile kıyaslanmıştır. Bu kapsamda yapılan çalışmada 30 pozitif test sonucu geliştirilen sistemle analiz edilmiştir. Bu 30 analiz içerisinde bilirubin pozitif olan 3 testte pozitif sonuç verilmiştir ancak bir tanesinde miktar doğru tespit edilememiştir. Urobilinogen pozitif olan 3 testte pozitif sonuç verilmiştir ancak bir tanesinde miktar doğru tespit edilememiştir. 30 test içerisinde keton bulunan vaka ile karşılaşılmamıştır. Absorbik asit pozitif çıkan 5 testten birinde elde edilen renk değerleri tablodaki değerlerle benzerlik göstermemiştir. Toplam renk farkı kontrolü yapıldığı için o testte pozitif olarak sonuç verilmiştir. Glikoz pozitif olan 7 testte doğru sonuçlar alınmıştır. Kan pozitif olan 6 testte ilk 3 renk skalası için doğru değerler alınırken sonraki 3 skala için hatalı sonuçlar elde edilmiştir. Bu aşamada

fark ortalaması değerleri doğru sonuçlar vermemiştir. Ph testinde elde edilen sonuçlarda tutarsızlıklar gözlemlenmiştir. Bu nedenle renk aralıkları daha fazla idrar test edilerek yeniden düzenlenmelidir. Nitrit bulunan örnekle karşılaşılmamıştır. Bu kapsamda daha fazla pozitif idrar örneğine ihtiyaç duyulmaktadır. 12 adet lökosit pozitif idrar analiz edilmiştir ve bunlardan 2 tanesi çok düşük miktarda olduğundan negatif sonuç alınmıştır. Bu nedenle lökositler için daha kapsamlı bir renk skalası geliştirilmesi gerekmektedir. Graviti değeri pozitif olan 7 örneğe rastlanmıştır. Renk skalası en geniş olan parametre olması nedeniyle miktar hesabında 3 hatalı sonuç verilmiştir.

Buradan ortalama alma prensibiyle çalışan pratik sistem başarısının yeterli olmadığı anlaşılmıştır. Ancak sistem başarısının arttırılması konusunda 30 analiz sayısının yeterli olmadığı da görülmektedir. Bu çalışmanın daha başarılı olabilmesi için yüzlerce pozitif örnekle değerlendirme yapmak ve yazılımda kullanılan renk tablosunu güncellemek gerekmektedir.

7.2.Sonuç ve Öneriler

Strip tekniğiyle kimyasal idrar görüntülerinin incelendiği bu çalışma kapsamında görüntülerin sınıflandırılması problemi konvolüsyonel sinir ağlarıyla yüksek doğrulukta sonuçlanmıştır.

Telefonla yapılan değerlendirme işlemlerinde her telefonun farklı kamera ve donanımsal özelliklere sahip olması, fotoğrafın çekildiği andaki ışık gibi çevresel etkenler ve hatta fotoğrafı çeken kişiden kaynaklı bile sonuçlarda tutarsızlıklar yaşanabilir. Geliştirilen kalibrasyon yazılımı, HSA tabanlı gürültü giderme katmanı ve CNN tabanlı sınıflandırma yapısıyla sistemde bu tarz dezavantajların olmadığı gözlemlenmiştir.

Günümüzde uygulanan kimyasal idrar analiz yöntemlerine alternatif bir yaklaşım sunulmuştur. Kullanılan reflektans fotometri yönteminden tamamen farklı olan bu yaklaşımla analizlerin daha detaylı ve anlaşılır, sistemlerin daha az alan kaplayan ve taşınabilir olması hedeflenmiştir. Sistem bileşenlerinin yazılımsal olarak bir sınırının bulunmaması yeni fikirlerin hayata geçirilmesine olanak sağlayacaktır. Bunlar uzaktan erişim, sonuçların uzaktan değerlendirilmesi, çevrimiçi olarak sonuçların sunucularda saklanabilmesi ya da hastane sistemine entegrasyon gibi çalışmalar olabilir.

Günümüzde kullanılan strip analiz cihazları markadan markaya farklılık göstermektedir. Geliştirilen sistem, farklı marka stripler için de devreye alınabilecek yeteneğe sahiptir. Böylece tüm farklı stripler için tek bir analiz cihazı ortaya çıkarılabilir.

Tam idrar cihazları mikroskobik idrar analizi yapmak için görüntü işleme tabanlı bir sistem, kimyasal analizi yapabilmek içinse reflektans fotometri tekniğini kullanmaktadır. Bu cihazlar kamera ve görüntü işleme donanımlarına mikroskobik analizi yapabilmek için zaten sahiptir. Bu yaklaşımla mikroskobik ve kimyasal analiz makineleri tek bir donanımla iki prosesi de gerçekleştirebilecek yeteneğe sahip olabilir. Böylece sistem maliyeti ve kapladığı alan da otomatik olarak küçülecektir.

Çalışma sonucunda elde edilen analiz sonuçlarının doğruluğunun yükseltilebilmesi için çalışmaların sürdürülmesi gerekmektedir. Farklı markalara ait analiz cihazlarıyla sistem kıyaslanmalı ve doğruluk ve kararlılık değerlerini yükseltmek için çalışmalara devam edilmelidir.

Usb kamera, mini bilgisayar ve görüntü işleme algoritmaları kullanılarak strip tekniğiyle kimyasal idrar analizi gerçekleştirebilen pratik uygulama geliştirilmiştir. Ancak mini bilgisayarın donanımsal sınırları nedeniyle tüm algoritma entegre edilememiş ve ortalama alma yöntemiyle sonuç üretilmiştir. Ortalama alma yöntemiyle elde edilen sonuçların yeterli olmadığı görülmüştür.

KAYNAKÇA

1. Simerville, J. A., Maxted, W. C. , Pahira, J. J., 2005. Urinalysis: a comprehensive review. **American Family Physician., 71** (6): 1153–62.

2. Harper, D. 26 September 2011. Urinalysis. **Online Etymology Dictionary**.

3. MedlinePlus Medical Encyclopedia, 2009. Fractional excretion of sodium. (Web sayfası: **https://medlineplus.gov/ency/article/003602.htm**), (Erişim tarihi: Ocak 2019).

4. Banauch, D., Koller, P. U., Bablok, W., 1983. Evaluation of diabur-test 5000: a cooperative study carried out at 12 diabetes centers. **Diabetes Care, 6** (3): 213-8.

5. Voswinckel, P., 1994. A marvel of colors and ingredients. The story of urine test strip. **Kidney Int Suppl., 47**:S3-7

6. Peele, J. D., Gadsden, R. H., Crews, R., 1977. Semi-automated vs. visual reading of urinalysis dipsticks. **Clin Chem., 23** (12): 2242-6.

7. Ben-Ezra, J., Bork, L., McPherson, R. A., 1998. Evaluation of the Sysmex UF-100 automated urinalysis analyzer. **Clin Chem., 44** (1):92-5.

8. Budak, Y. U, Huysal, K., Yilmaz, S., 2011. Comparison of three automated systems for urine chemistry and sediment analysis in routine laboratory practice. **Clin. Lab., 57** (1-2):47-52.

9. Yüksel, H., Kaplan, I., Dal, T., Kuş, S., Toprak, G., Evliyaoğlu, O., 2014. İdrar kültürü testi gerekliliğini öngörmede tam otomatik idrar analizi sonuçlarının Performansı. **Journal of Clinical and Experimental Investigations.**

10. Scheer, W. D., 1987. The detection of leukocyte esterase activity in urine with a new reagent strip. **Am J Clin Pathol., 87** (1): 86-93.

11. Bonnardeaux, A., Somerville, P., Kaye, M., 1994. A study on the reliability of dipstick urinalysis. **Clin Nephrol., 41** (3):167-72.

12. Lee, D. S., Jeon, B. G., Ihm, C., Park, J. K., Jung, M. Y., 2010. A simple and smart telemedicine device for developing regions: a pocket-sized colorimetric reader. **Lab on Chip, 11** (1):120-6.

13. Velikova, M., Ruben, L., Scheltinga, J., Peter, J. F., Spaanderman, M., 2014. Smartphone-based analysis of biochemical tests for health monitoring support at home. **Healthc Technol Lett, 1** (3): 92–97.

14. Stephen, C., Kapoor, N., Jonathan, L., 2014. Method and apparatus for performing color-based reaction testing of biological materials. US Patent No: USOO8655009B2 Feb. 18, 2014.

15. Choi, K., Chang, I., Lee, J. C., Kim, K., Noh, S., Ahn, H., Cho, J. H., Kwak, Y. H. , Kim, S., Kim, H. C., 2016. Smartphone based urine reagent strip test in the emergency department. **Telemed J E Health, 22** (6): 534-540.

16. Wikipedi. 2015. İdrar. (Web sayfası: **http://tr.wikipedia.org/wiki/İdrar**), (Erişim tarihi: Ocak 2019).

17. Rosner, F., Muntner, S., 1969. Moses Maimonides' aphorisms regarding analysis of urine. **Ann. Intern. Med., 71** (1): 217-220.

18. Winsten, S., 1969. The skeptical chemist**. Clin. Chem., 15** (8):737-744.

19. Doğan, Y., 2013. Klinik biyokimyanın doğuşunda idrar analizlerinin önemi. (Web sayfası: **http://e-kutuphane.teb.org.tr/pdf/mised/mayis02/13.pdf**), (Erişim tarihi: Ocak 2019).

20. Caraway, W. T., 1973. The scientific development of clinical chemistry to 1948. **Clin Chem., 19** (4): 373-383.

21. Milli Eğitim Bakanlığı., 2011., Tıbbi laboratuvar idrarın fiziksel ve kimyasal Analizi.(Web sayfası: **http://www.megep.meb.gov.tr/mte_program_modul/moduller_pdf**), (Erişim tarihi: Ocak 2019).

22. Atasağungıl M., 1966. Biokimya Pratik Dersleri, Ankara Universitesi Eczacılık Fakültesi, Güzel Istanbul Matbaası Ankara 129/9.

23. Free A.H., Free H.M., 1975. Urine sugar testing state of the art. **Laboratory Medicine, 6** (2): 23-29

24. Kurt İ., Serdar M.A., Abughoush M.W., 2001. İdrar stikleri ile idrarın kimyasal analizi her zaman yeterli mi?. **Türk Biyokimya Dergisi, 26** (4): 143-156

25. Nebioğlu S., Burat., M.K., Büyükbingöl Z., 1996. Biyokimya Pratikleri, Ankara Üniversitesi Eczacılık Fakültesi Yayınları Ankara, 86/63

26. Özer B., Söğüt S., Duran N., Özer C., Kuvandık G., Çetin M., 2007. Üriner sistem infeksiyonlarında laboratuvar testlerinin tanı değerleri. **Türk Mikrobiyol Cem Derg., 37** (3): 152-156

27. Roggeman S., Zaman Z., 2001. Safely reducing manual urine microscopy analyses by combining urine flow cytometer and strip results. **American Journal of Clinical Pathology, 116** (6): 872–878

28. Herschel J.F.W., 1997. On the chemical action of the rays of the solar spectrum on preparations of silver and other substances, both metallic and non-metallic; and on some photographic processes. **Philos. Trans. R. Soc. Lond., 4** (13): 1-59

29. Zhang M., 2007. The world's first digital camera by Kodak and Steve Sasson. (Web sayfası: **https://petapixel.com/2010/08/05/the-worlds-first-digital-camera-by-kodak-and-steve-sasson**), (Erişim tarihi: Haziran 2018).

30. Sasson S., 2007. We had no idea. (Web sayfası: **https://wettengl.info/Blog/Dokumente/D080-SteveSasson-WeHadNoIdea.pdf**), (Erişim tarihi: Haziran 2018).

31. Yıldırım K.S., Ince C., Kalaycı T.E., 2003. Görüntü İşleme.

32. Büyükarıkan B., 2018. Artıficıal neural networks-ann (yapay sinir ağları-ysa). (Web sayfası: **erkanulker.kisisel.selcuk.edu.tr/.../Artificial_neural_networks_yapay_sinir_aglari.pptx**), (Erişim tarihi: Ocak 2019).

33. McCulloch W.S., Pitts W., 1943. A logical calculus of the ideas immanent in nervous activity. **The bulletin of mathematical biophysics, 5** (4): 115–133

34. Öğücü M. O., 2006. Yapay Sinir Ağları İle Sistem Tanıma İstanbul Teknik Üniversitesi Fen Bilimleri Enstitüsü, Yüksek Lisans Tezi, İstanbul

35. KABALCI E., Çok katmanlı algılayıcı (multilayer perceptron). (Web sayfası: **https://docplayer.biz.tr/42742276-Cok-katmanli-algilayici-multilayer-perceptron-doc-dr-ersan-kabalci.html**), (Erişim tarihi: Ocak 2019).

36. Widrow B., Hoff M.E., 1960. Adaptive switching circuits. **Stanford Universitesi, 241** (551): 1553-1

37. Chauvin Y., Rumelhart D.E., 1995. Backpropagation: Theory, Architectures, and Applications. Psychology Press, 556

38. KABALCI E., Yapay sinir ağları (Web sayfası: **https://ekblc.files.wordpress.com/2014/02/ysa.pdf**), (Erişim tarihi: Ocak 2019).

39. Chua, L.O., Yang, L., 1988. Cellular neural networks: theory. **IEEE Transaction on Circuits and Systems, 35** (10): 1257-1272

40. Chua, L.O., Yang, L., 1988. Cellular neural networks theory. **IEEE Transaction on Circuits and Systems, 35** (10): 1273-1290

41. Tolluoğlu O., KENT S., Kargın S., Ucan O.N., 2004. Hücresel yapay sinir ağlarında kümeleme yöntemi ile gürültü temizleme uygulaması. **Türkiye 2004 Bilimsel Kongresi ve Ulusal Genel Kurulu**

42. Günay E., 2002. Hücresel Sinir Ağları İle Medikal Görüntü İşleme. Erciyes Üniversitesi Elektrik Elektronik Mühendisliği, Yüksek Lisans Tezi, Kayseri

43. Szabo V., Varga B., 2010. matCNN 2.0 Algorithmic users guide. **Pazmany Peter Catholic University**

44. Lecun Y., Lenet. (Web sayfası: **http://yann.lecun.com/exdb/lenet/**), (Erişim tarihi: Ocak 2019).

45. Kara O., Başcan O., Tecir Y., 2018. Görüntü İşleme İle Kavrulmuş Fındık Üzerindeki Kabuk Ve Çürüklerin Belirlenmesi Sakarya Üniversitesi Teknoloji Fakültesi, Bitirme Tezi, Sakarya

46. Deshpande A., 2019. A beginner's guide to understanding convolutional neural networks. (Web sayfası: **https://adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks-Part-2**), (Erişim tarihi: Ocak 2019).

47. Karpathy A., 2019. Convolutional neural networks (CNNs / ConvNets). (Web sayfası: **http://cs231n.github.io/convolutional-networks**), (Erişim tarihi: Ocak 2019).

48. Springenberg J.T., Dosovitskiy A., Brox T., Riedmiller M., 2015. Striving for simplicity: the all convolutional net. **arXiv**:1412.6806

49. Çarkacı N., 2017. Derin öğrenme uygulamalarında temel kavramlar : perceptron, skor fonksiyonu ve hata hesaplaması(loss function). (Web sayfası: **https://tr.linkedin.com/pulse/derin-%C3%B6%C4%9Frenme-uygulamalar%C4%B1nda-temel-kavramlar-skor-ve-%C3%A7arkac%C4%B1**), (Erişim tarihi: Ocak 2019).

50. Srivastava N., Hinton G., Krizhevsky A., Sutskever I., Salakhutdinov R., 2014. Dropout: a simple way to prevent neural networks from overfitting. **Journal of Machine Learning Research 15**: 1929-1958

51. Krizhevsky A., Sutskever I., Hinton G., Alexnet : ımagenet classification with deep convolutional neural networks. **Neural Information Processing Systems, 4824**

52. Krizhevsky A., Sutskever I., Hinton G., ImageNet classification with deep convolutional neural networks. (Web sayfası: **https://codeburst.io/deep-learning-what-why-dd77d432f182**), (Erişim tarihi: Ocak 2019).

53. Moocarme M., 2019. Country lyrics created with recurrent neural networks (Web sayfası: **http://www.mattmoocar.me/blog/RNNCountryLyrics**), (Erişim tarihi: Ocak 2019).

54. Hochreiter S., Schmidhuber J., 1997. Long short-term memory. **Neural Computation, 9** (8): 1735-1780

55. Hinton G.E., Salakhutdinov R. R., 2006. Reducing the dimensionality of data with neural networks. **Science, 313** (5786): 504–507

56. Deeplearning4j., 2018. Restricted Boltzmann machine. (Web sayfası: **https://deeplearning4j.org/restrictedboltzmannmachine**), (Erişim tarihi: Ocak 2019).

57. Hinton G.E., Teh YW., 2006. A fast learning algorithm for deep belief nets. **Neural Computation 18** (7): 1527-54

58. Basu S., Ganguly S., Mukhopadhyay S., DiBiano R., Karki M., Nemani R., 2015. DeepSat - a learning framework for satellite ımagery. **Association for Computing Machinery.**

59. Deng L., Yu D., 2011. Deep Convex Net: A scalable architecture for speech pattern classification. **International Speech Communication Association**

60. Khamparia A., Gupta D., Nhu N.G., Khanna A., 2019. Sound classification using convolutional neural network and tensor deep stacking. **Network IEEE Access, PP**(99):1-1

61. LeCun Y., Bottou L., Bengio Y., Haffner P.. 1998. Gradient-based learning applied to document recognition. **IEEE**

62. Labstrip U11 plus. (Web sayfası: **https://www.pocd.com.au/General-Supplies/Labstrip-U-11-Plus**), (Erişim tarihi: Ocak 2019).

63. Işık M., Çamurcu A.Y., 2007. K-means, k-medoıds ve bulanık C-means algoritmalarının uygulamalı olarak performanslarının tespiti. **İstanbul Ticaret Üniversitesi Fen Bilimleri Dergisi, 6** (11): 31-45

64. Taşkıran A., Günay E., (2018). Classification of Chemical Urine Analysis Images. **SETSCI Conference Indexing System, 3**: 115-115

65. Richardson M., Wallace S.P., 2012. Getting Started With Raspberry Pi, Maker Media Inc. USA, 177

66. Zelle J.M., 2017. Python Programming: An Introduction To Computer Science. Franklin Beedle & Associates Inc. USA, 525

67. Pedregosa F., Varoquaux G., Gramfort A., Michel V., Thirion B., Scikit-learn: machine learning in python, mach. learn. **Journal of Machine Learning Research, 12** (2011): 2825-2830

68. Lutz M., 2013. Learning python: powerful object-oriented programming. O'Reilly Media Inc. USA, 1540

69. Laganière R., 2014. OpenCV Computer Vision Application Programming Cookbook Second Edition. Packt Publishing Ltd İngiltere, 374

EKLER

EK 1. Pratik Uygulama Kalibrasyon Yazılımı Kodları

```
import numpy as np
import argparse
import cv2
def flipImage(img):
    cv2.imshow("img", img)
    rimg=img.copy()
    fimg=img.copy()
    rimg=cv2.flip(img,1)
    fimg=cv2.flip(img,0)
    cv2.imshow("fimg", fimg)
    return fimg
def cropArea(img, x1,x2,y1,y2):
    center = (int((x1+x2)/2), int((y1+y2)/2))
    size = (int(1.2*(x2-x1)),int(1.2*(y2-y1)))
    cropped = cv2.getRectSubPix(img, size, center)
    return cropped
def crop_minAreaRect(img_box, rect, box):
    W = rect[1][0]
    H = rect[1][1]
    Xs = [i[0] for i in box]
    Ys = [i[1] for i in box]
    x1 = min(Xs)
    x2 = max(Xs)
    y1 = min(Ys)
```

```
        y2 = max(Ys)
        rotated = False
        angle = rect[2]
        if angle < -45:
                angle+=90
                rotated = True
        center = (int((x1+x2)/2), int((y1+y2)/2))
        size = (int(1.2*(x2-x1)),int(1.2*(y2-y1)))
        #cv2.circle(img_box, center, 10, (0,255,0), -1)
        M = cv2.getRotationMatrix2D((size[0]/2, size[1]/2), angle, 1.0)
        cropped = cv2.getRectSubPix(img_box, size, center)
        cropped = cv2.warpAffine(cropped, M, size)
        croppedW = W if not rotated else H
        croppedH = H if not rotated else W
        croppedRotated = cv2.getRectSubPix(cropped, (int(croppedW), int(croppedH)),
(size[0]/2, size[1]/2))
        return croppedRotated
ap = argparse.ArgumentParser()
ap.add_argument("-i", "--image", help = "path to the image")
args = vars(ap.parse_args())
img_orj = cv2.imread(args["image"])
height, width = img_orj.shape[:2]
im1 = cv2.resize(img_orj,(width/8,height/8))
cv2.imshow("image original", im1)
img_gray1 = cv2.cvtColor(img_orj, cv2.COLOR_BGR2GRAY)
kernel = np.ones((20,20),np.uint8)
img_open1 = cv2.morphologyEx(img_gray1, cv2.MORPH_OPEN, kernel)
```

```
ret,img_thresh1 = cv2.threshold(img_open1,50,255,cv2.THRESH_BINARY)

im2, contours, hierarchy = cv2.findContours(img_thresh1,cv2.RETR_TREE,cv2.CHAIN_APPROX_SIMPLE)

height, width = img_thresh1.shape[:2]

img2 = cv2.resize(img_thresh1,(width/8,height/8))

cnt=contours[0]

rect = cv2.minAreaRect(cnt)

box0 = cv2.boxPoints(rect)

box0 = np.int0(box0)

# crop and rotate

img_crop1 = crop_minAreaRect(img_orj,rect,box0)

img_strip = img_crop1.copy()

height, width = img_crop1.shape[:2]

print(width)

print(height)

#mavi renkteki alani tespit et

dancite_low = [80, 80, 50]

dancite_up =[130, 140, 90]

lower = np.array(dancite_low, dtype = "uint8")

upper = np.array(dancite_up, dtype = "uint8")

mask1 = cv2.inRange(img_crop1, lower, upper)

output1 = cv2.bitwise_and(img_crop1, img_crop1, mask = mask1)

img_gray2 = cv2.cvtColor(output1, cv2.COLOR_BGR2GRAY)

kernel = np.ones((10,10),np.uint8)

img_open2 = cv2.morphologyEx(img_gray2, cv2.MORPH_OPEN, kernel)

kernel1 = np.ones((30,30),np.uint8)

img_close2 = cv2.morphologyEx(img_open2, cv2.MORPH_CLOSE, kernel1)
```

```
height, width = img_crop1.shape[:2]
img_res3 = cv2.resize(img_crop1,(width/8,height/8))
img_cift1=np.hstack([img_crop1, output1])
height, width = img_cift1.shape[:2]
img_res4 = cv2.resize(img_cift1,(width/8,height/8))
cv2.imshow("asdfadf", img_res4)
cv2.waitKey(0)
im3, contours, hierarchy = cv2.findContours(img_close2,cv2.RETR_TREE,cv2.CHAIN_APPROX_SIMPLE)
alan=0
count=contours[0]
for cnt in contours:
        if alan < cv2.contourArea(cnt) :
                alan=cv2.contourArea(cnt)
                count=cnt
print(alan)
rect = cv2.minAreaRect(cnt)
box1 = cv2.boxPoints(rect)
box1 = np.int0(box1)
cv2.drawContours(img_crop1,[box1],0,(0,0,255),10)
height, width = img_crop1.shape[:2]
img_res5 = cv2.resize(img_crop1,(width/8,height/8))
dancite_low = [100, 140, 220]  # B G R
dancite_up =[150, 210, 255]  # B G R
lower = np.array(dancite_low, dtype = "uint8")
upper = np.array(dancite_up, dtype = "uint8")
mask2 = cv2.inRange(img_crop1, lower, upper)
```

```
output2 = cv2.bitwise_and(img_crop1, img_crop1, mask = mask2)
img_gray10 = cv2.cvtColor(output2, cv2.COLOR_BGR2GRAY)
kernel2 = np.ones((10,10),np.uint8)
img_open10 = cv2.morphologyEx(img_gray10, cv2.MORPH_OPEN, kernel2)
kernel3 = np.ones((30,30),np.uint8)
img_close10 = cv2.morphologyEx(img_open10, cv2.MORPH_CLOSE, kernel3)
height, width = img_close10.shape[:2]
res2 = cv2.resize(img_close10,(width/8,height/8))
img_cift3=np.hstack([img_cift1, output2])
height, width = img_cift3.shape[:2]
img_res10 = cv2.resize(img_cift3,(width/8,height/8))
cv2.imshow("img_res10", img_res10)
cv2.waitKey(0)
im3, contours, hierarchy = cv2.findContours(img_close10,cv2.RETR_TREE,cv2.CHAIN_APPROX_SIMPLE)
alan=0
count=contours[0]
for cnt in contours:
        if alan < cv2.contourArea(cnt) :
                alan=cv2.contourArea(cnt)
                count=cnt
print(alan)
rect = cv2.minAreaRect(cnt)
box2 = cv2.boxPoints(rect)
box2 = np.int0(box2)
cv2.drawContours(img_crop1,[box2],0,(0,0,255),10)
print(box1)
```

```
print(box2)
print(image)
b1Xs = [i[0] for i in box1]
b1Ys = [i[1] for i in box1]
b1x1 = min(b1Xs)
b1x2 = max(b1Xs)
b1y1 = min(b1Ys)
b1y2 = max(b1Ys)
print(b1x1,b1x2,b1y1,b1y2)
b2Xs = [i[0] for i in box2]
b2Ys = [i[1] for i in box2]
b2x1 = min(b2Xs)
b2x2 = max(b2Xs)
b2y1 = min(b2Ys)
b2y2 = max(b2Ys)
print(b2x1,b2x2,b2y1,b2y2)
imMesafe=(b2y2-b1y2)/4
print(imMesafe)
bim11=img_crop1
file = open('kalibrasyon.txt','w')
file.write(str(b1x1)+'\n')
file.write(str(b1x2)+'\n')
file.write(str(b1y1)+'\n')
file.write(str(b1y2)+'\n')
file.write(str(b2x1)+'\n')
file.write(str(b2x2)+'\n')
file.write(str(b2y1)+'\n')
```

```
file.write(str(b2y2)+'\n')

file.close()

if imMesafe >0 :

bim1=cv2.rectangle(img_crop1,((b1x1+b2x1)/2,b1y1+imMesafe),((b1x2+b2x2)/2,b1y2
+imMesafe),(0,0,255),10)bim2=cv2.rectangle(bim1,((b1x1+b2x1)/2,b1y1+imMesafe*2
),((b1x2+b2x2)/2,b1y2+imMesafe*2),(0,0,255),10)

bim3=cv2.rectangle(bim2,((b1x1+b2x1)/2,b1y1+imMesafe*3),((b1x2+b2x2)/2,b1y2+i
mMesafe*3),(0,0,255),10)

bim5=cv2.rectangle(bim3,((b1x1+b2x1)/2,b1y1+imMesafe*5),((b1x2+b2x2)/2,b1y2+i
mMesafe*5),(0,0,255),10)bim6=cv2.rectangle(bim5,((b1x1+b2x1)/2,b1y1+imMesafe*6
),((b1x2+b2x2)/2,b1y2+imMesafe*6),(0,0,255),10)

bim7=cv2.rectangle(bim6,((b1x1+b2x1)/2,b1y1+imMesafe*7),((b1x2+b2x2)/2,b1y2+i
mMesafe*7),(0,0,255),10)

bim8=cv2.rectangle(bim7,((b1x1+b2x1)/2,b1y1+imMesafe*8),((b1x2+b2x2)/2,b1y2+i
mMesafe*8),(0,0,255),10)

bim9=cv2.rectangle(bim8,((b1x1+b2x1)/2,b1y1+imMesafe*9),((b1x2+b2x2)/2,b1y2+i
mMesafe*9),(0,0,255),10)

bim10=cv2.rectangle(bim9,((b1x1+b2x1)/2,b1y1+imMesafe*10),((b1x2+b2x2)/2,b1y2
+imMesafe*10),(0,0,255),10)

bim11=cv2.rectangle(bim10,((b1x1+b2x1)/2,b1y1+imMesafe*11),((b1x2+b2x2)/2,b1y
2+imMesafe*11),(0,0,255),10)

if imMesafe <0 :

bim1=cv2.rectangle(img_crop1,((b1x1+b2x1)/2,b1y1+imMesafe),((b1x2+b2x2)/2,b1y2
+imMesafe),(0,0,255),10)

bim2=cv2.rectangle(bim1,((b1x1+b2x1)/2,b1y1+imMesafe*2),((b1x2+b2x2)/2,b1y2+i
mMesafe*2),(0,0,255),10)

bim3=cv2.rectangle(bim2,((b1x1+b2x1)/2,b1y1+imMesafe*3),((b1x2+b2x2)/2,b1y2+i
mMesafe*3),(0,0,255),10)

bim5=cv2.rectangle(bim3,((b1x1+b2x1)/2,b1y1+imMesafe*5),((b1x2+b2x2)/2,b1y2+i
mMesafe*5),(0,0,255),10)

bim6=cv2.rectangle(bim5,((b1x1+b2x1)/2,b1y1+imMesafe*6),((b1x2+b2x2)/2,b1y2+i
mMesafe*6),(0,0,255),10)

bim7=cv2.rectangle(bim6,((b1x1+b2x1)/2,b1y1+imMesafe*7),((b1x2+b2x2)/2,b1y2+i
mMesafe*7),(0,0,255),10)
```

```
bim8=cv2.rectangle(bim7,((b1x1+b2x1)/2,b1y1+imMesafe*8),((b1x2+b2x2)/2,b1y2+imMesafe*8),(0,0,255),10)

bim9=cv2.rectangle(bim8,((b1x1+b2x1)/2,b1y1+imMesafe*9),((b1x2+b2x2)/2,b1y2+imMesafe*9),(0,0,255),10)

bim10=cv2.rectangle(bim9,((b1x1+b2x1)/2,b1y1+imMesafe*10),((b1x2+b2x2)/2,b1y2+imMesafe*10),(0,0,255),10)

bim11=cv2.rectangle(bim10,((b1x1+b2x1)/2,b1y1+imMesafe*11),((b1x2+b2x2)/2,b1y2+imMesafe*11),(0,0,255),10)

en = (b1x2-b1x1)/5

boy= (b1y2-b1y1)/5

print b1x1, b1x2, b1y1, b1y2

print  b2x1, b2x2, b2y1, b2y2

print en,boy

im_1 = cropArea(img_strip,(b1x1+b2x1)/2+en,(b1x2+b2x2)/2-en,b1y1+boy,b1y2-boy)

im_2 = cropArea(img_strip,(b1x1+b2x1)/2 +en ,(b1x2+b2x2)/2 - en,b1y1+imMesafe +boy,b1y2+imMesafe- boy)

im_3 = cropArea(img_strip,(b1x1+b2x1)/2 +en ,(b1x2+b2x2)/2 - en,b1y1+imMesafe*2 +boy,b1y2+imMesafe*2- boy)

im_4 = cropArea(img_strip,(b1x1+b2x1)/2 +en ,(b1x2+b2x2)/2 - en,b1y1+imMesafe*3 +boy,b1y2+imMesafe*3- boy)

im_5 = cropArea(img_strip,(b1x1+b2x1)/2 +en ,(b1x2+b2x2)/2 - en,b1y1+imMesafe*4 +boy,b1y2+imMesafe*4- boy)

im_6 = cropArea(img_strip,(b1x1+b2x1)/2 +en ,(b1x2+b2x2)/2 - en,b1y1+imMesafe*5 +boy,b1y2+imMesafe*5- boy)

im_7 = cropArea(img_strip,(b1x1+b2x1)/2 +en ,(b1x2+b2x2)/2 - en,b1y1+imMesafe*6 +boy,b1y2+imMesafe*6- boy)

im_8 = cropArea(img_strip,(b1x1+b2x1)/2 +en ,(b1x2+b2x2)/2 - en,b1y1+imMesafe*7 +boy,b1y2+imMesafe*7- boy)

im_9 = cropArea(img_strip,(b1x1+b2x1)/2 +en ,(b1x2+b2x2)/2 - en,b1y1+imMesafe*8 +boy,b1y2+imMesafe*8- boy)

im_10 = cropArea(img_strip,(b1x1+b2x1)/2 +en ,(b1x2+b2x2)/2 - en,b1y1+imMesafe*9 +boy,b1y2+imMesafe*9- boy)
```

```
im_11 = cropArea(img_strip,(b1x1+b2x1)/2 +en ,(b1x2+b2x2)/2 -
en,b1y1+imMesafe*10 +boy,b1y2+imMesafe*10- boy)

print(im_1.mean())

print(im_2.mean())

print(im_3.mean())

print(im_4.mean())

print(im_5.mean())

print(im_6.mean())

print(im_7.mean())

print(im_8.mean())

print(im_9.mean())

print(im_10.mean())

print(im_11.mean())

height, width = bim11.shape[:2]

res3 = cv2.resize(bim11,(width/8,height/8))

cv2.imshow("image", res3)

cv2.imshow("im_1", im_1)

cv2.imshow("im_2", im_2)

cv2.imshow("im_3", im_3)

cv2.waitKey(0)
```

EK 2. Pratik Uygulama Test Yazılımı Kodları

```
import numpy as np

import argparse

import cv2

def flipImage(img):

        cv2.imshow("img", img)
```

```
        rimg=img.copy()
        fimg=img.copy()
        rimg=cv2.flip(img,1)
        fimg=cv2.flip(img,0)
        cv2.imshow("fimg", fimg)
        return fimg
def cropArea(img, x1,x2,y1,y2):
        center = (int((x1+x2)/2), int((y1+y2)/2))
        size = (int(1.2*(x2-x1)),int(1.2*(y2-y1)))
        cropped = cv2.getRectSubPix(img, size, center)
        return cropped
def crop_minAreaRect(img_box, rect, box):
        W = rect[1][0]
        H = rect[1][1]
        Xs = [i[0] for i in box]
        Ys = [i[1] for i in box]
        x1 = min(Xs)
        x2 = max(Xs)
        y1 = min(Ys)
        y2 = max(Ys)
        rotated = False
        angle = rect[2]
        if angle < -45:
                angle+=90
                rotated = True
        center = (int((x1+x2)/2), int((y1+y2)/2))
        size = (int(1.2*(x2-x1)),int(1.2*(y2-y1)))
```

```
        M = cv2.getRotationMatrix2D((size[0]/2, size[1]/2), angle, 1.0)
        cropped = cv2.getRectSubPix(img_box, size, center)
        cropped = cv2.warpAffine(cropped, M, size)
        croppedW = W if not rotated else H
        croppedH = H if not rotated else W
        croppedRotated = cv2.getRectSubPix(cropped, (int(croppedW), int(croppedH)), (size[0]/2, size[1]/2))
        return croppedRotated
ap = argparse.ArgumentParser()
ap.add_argument("-i", "--image", help = "path to the image")
args = vars(ap.parse_args())
img_orj = cv2.imread(args["image"])
height_orj, width_orj = img_orj.shape[:2]
im1 = cv2.resize(img_orj,(width_orj/8,height_orj/8))
cv2.imshow("image original", im1)
img_gray1 = cv2.cvtColor(img_orj, cv2.COLOR_BGR2GRAY)
kernel = np.ones((20,20),np.uint8)
img_open1 = cv2.morphologyEx(img_gray1, cv2.MORPH_OPEN, kernel)
ret,img_thresh1 = cv2.threshold(img_open1,50,255,cv2.THRESH_BINARY)
im2, contours, hierarchy = cv2.findContours(img_thresh1,cv2.RETR_TREE,cv2.CHAIN_APPROX_SIMPLE)
height, width = img_thresh1.shape[:2]
img2 = cv2.resize(img_thresh1,(width/8,height/8))
cnt=contours[0]
rect = cv2.minAreaRect(cnt)
box0 = cv2.boxPoints(rect)
box0 = np.int0(box0)
```

```
img_crop1 = crop_minAreaRect(img_orj,rect,box0)
img_strip = img_crop1.copy()
file = open('kalibrasyon.txt','r')
b1x1=int(file.readline())
b1x2=int(file.readline())
b1y1=int(file.readline())
b1y2=int(file.readline())
b2x1=int(file.readline())
b2x2=int(file.readline())
b2y1=int(file.readline())
b2y2=int(file.readline())
file.close()
print b1x1
print b1x2
print b1y1
print b1y2
print b2x1
print b2x2
print b2y1
print b2y2
height_crop1, width_crop1 = img_crop1.shape[:2]
w=(b1x2+b1x2)/2+8 #175
h= height_crop1 #3834
imMesafe=(b2y2-b1y2)/4
print(imMesafe)
bim11=img_crop1
img_crop2 = cv2.resize(img_crop1,(w,h))
```

```
if imMesafe >0 :

        bim0=cv2.rectangle(img_crop2,((b1x1+b2x1)/2,b1y1),((b1x2+b2x2)/2,b1y2),(0,
0,255),10)

        bim1=cv2.rectangle(bim0,((b1x1+b2x1)/2,b1y1+imMesafe),((b1x2+b2x2)/2,b1y
2+imMesafe),(0,0,255),10)

        bim2=cv2.rectangle(bim1,((b1x1+b2x1)/2,b1y1+imMesafe*2),((b1x2+b2x2)/2,b
1y2+imMesafe*2),(0,0,255),10)

        bim3=cv2.rectangle(bim2,((b1x1+b2x1)/2,b1y1+imMesafe*3),((b1x2+b2x2)/2,b
1y2+imMesafe*3),(0,0,255),10)

        bim4=cv2.rectangle(bim3,((b1x1+b2x1)/2,b1y1+imMesafe*4),((b1x2+b2x2)/2,b
1y2+imMesafe*4),(0,0,255),10)

        bim5=cv2.rectangle(bim4,((b1x1+b2x1)/2,b1y1+imMesafe*5),((b1x2+b2x2)/2,b
1y2+imMesafe*5),(0,0,255),10)

        bim6=cv2.rectangle(bim5,((b1x1+b2x1)/2,b1y1+imMesafe*6),((b1x2+b2x2)/2,b
1y2+imMesafe*6),(0,0,255),10)

        bim7=cv2.rectangle(bim6,((b1x1+b2x1)/2,b1y1+imMesafe*7),((b1x2+b2x2)/2,b
1y2+imMesafe*7),(0,0,255),10)

        bim8=cv2.rectangle(bim7,((b1x1+b2x1)/2,b1y1+imMesafe*8),((b1x2+b2x2)/2,b
1y2+imMesafe*8),(0,0,255),10)

        bim9=cv2.rectangle(bim8,((b1x1+b2x1)/2,b1y1+imMesafe*9),((b1x2+b2x2)/2,b
1y2+imMesafe*9),(0,0,255),10)

        bim10=cv2.rectangle(bim9,((b1x1+b2x1)/2,b1y1+imMesafe*10),((b1x2+b2x2)/
2,b1y2+imMesafe*10),(0,0,255),10)

        bim11=cv2.rectangle(bim10,((b1x1+b2x1)/2,b1y1+imMesafe*11),((b1x2+b2x2)
/2,b1y2+imMesafe*11),(0,0,255),10)

if imMesafe <0 :

        bim0=cv2.rectangle(img_crop2,((b1x1+b2x1)/2,b1y1),((b1x2+b2x2)/2,b1y2),(0,
0,255),10)

        bim1=cv2.rectangle(bim0,((b1x1+b2x1)/2,b1y1+imMesafe),((b1x2+b2x2)/2,b1y
2+imMesafe),(0,0,255),10)

        bim2=cv2.rectangle(bim1,((b1x1+b2x1)/2,b1y1+imMesafe*2),((b1x2+b2x2)/2,b
1y2+imMesafe*2),(0,0,255),10)

        bim3=cv2.rectangle(bim2,((b1x1+b2x1)/2,b1y1+imMesafe*3),((b1x2+b2x2)/2,b
1y2+imMesafe*3),(0,0,255),10)
```

```
        bim4=cv2.rectangle(bim3,((b1x1+b2x1)/2,b1y1+imMesafe*4),((b1x2+b2x2)/2,b
1y2+imMesafe*4),(0,0,255),10)

        bim5=cv2.rectangle(bim4,((b1x1+b2x1)/2,b1y1+imMesafe*5),((b1x2+b2x2)/2,b
1y2+imMesafe*5),(0,0,255),10)

        bim6=cv2.rectangle(bim5,((b1x1+b2x1)/2,b1y1+imMesafe*6),((b1x2+b2x2)/2,b
1y2+imMesafe*6),(0,0,255),10)

        bim7=cv2.rectangle(bim6,((b1x1+b2x1)/2,b1y1+imMesafe*7),((b1x2+b2x2)/2,b
1y2+imMesafe*7),(0,0,255),10)

        bim8=cv2.rectangle(bim7,((b1x1+b2x1)/2,b1y1+imMesafe*8),((b1x2+b2x2)/2,b
1y2+imMesafe*8),(0,0,255),10)

        bim9=cv2.rectangle(bim8,((b1x1+b2x1)/2,b1y1+imMesafe*9),((b1x2+b2x2)/2,b
1y2+imMesafe*9),(0,0,255),10)

        bim10=cv2.rectangle(bim9,((b1x1+b2x1)/2,b1y1+imMesafe*10),((b1x2+b2x2)/
2,b1y2+imMesafe*10),(0,0,255),10)

        bim11=cv2.rectangle(bim10,((b1x1+b2x1)/2,b1y1+imMesafe*11),((b1x2+b2x2)
/2,b1y2+imMesafe*11),(0,0,255),10)

en = (b1x2-b1x1)/6

boy= (b1y2-b1y1)/2

print b1x1, b1x2, b1y1, b1y2

print  b2x1, b2x2, b2y1, b2y2

print en,boy

im_1 = cropArea(img_strip,(b1x1+b2x1)/2+en,(b1x2+b2x2)/2-en,b1y1+boy,b1y2-boy)

im_2 = cropArea(img_strip,(b1x1+b2x1)/2 +en ,(b1x2+b2x2)/2 - en,b1y1+imMesafe
+boy,b1y2+imMesafe- boy)

im_3 = cropArea(img_strip,(b1x1+b2x1)/2 +en ,(b1x2+b2x2)/2 - en,b1y1+imMesafe*2
+boy,b1y2+imMesafe*2- boy)

im_4 = cropArea(img_strip,(b1x1+b2x1)/2 +en ,(b1x2+b2x2)/2 - en,b1y1+imMesafe*3
+boy,b1y2+imMesafe*3- boy)

im_5 = cropArea(img_strip,(b1x1+b2x1)/2 +en ,(b1x2+b2x2)/2 - en,b1y1+imMesafe*4
+boy,b1y2+imMesafe*4- boy)

im_6 = cropArea(img_strip,(b1x1+b2x1)/2 +en ,(b1x2+b2x2)/2 - en,b1y1+imMesafe*5
+boy,b1y2+imMesafe*5- boy)
```

```
im_7 = cropArea(img_strip,(b1x1+b2x1)/2 +en ,(b1x2+b2x2)/2 - en,b1y1+imMesafe*6 +boy,b1y2+imMesafe*6- boy)

im_8 = cropArea(img_strip,(b1x1+b2x1)/2 +en ,(b1x2+b2x2)/2 - en,b1y1+imMesafe*7 +boy,b1y2+imMesafe*7- boy)

im_9 = cropArea(img_strip,(b1x1+b2x1)/2 +en ,(b1x2+b2x2)/2 - en,b1y1+imMesafe*8 +boy,b1y2+imMesafe*8- boy)

im_10 = cropArea(img_strip,(b1x1+b2x1)/2 +en ,(b1x2+b2x2)/2 - en,b1y1+imMesafe*9 +boy,b1y2+imMesafe*9- boy)

im_11 = cropArea(img_strip,(b1x1+b2x1)/2 +en ,(b1x2+b2x2)/2 - en,b1y1+imMesafe*10 +boy,b1y2+imMesafe*10- boy)

im_12 = cropArea(img_strip,(b1x1+b2x1)/2 +en ,(b1x2+b2x2)/2 - en,b1y1+imMesafe*11 +boy,b1y2+imMesafe*11- boy)

b1B=97

b1G=100

b1R=69

b2B=246

b2G=231

b2R= 235

b3B =243

b3G =237

b3R= 242

b4B= 254

b4G =244

b4R =249

b5B =136

b5G =183

b5R =246

b6B =127

b6G =230
```

```
b6R =253
b7B =208
b7G =224
b7R =238
b8B =155
b8G =220
b8R =239
b9B =156
b9G =127
b9R =108
b10B =242
b10G =231
b10R =239
b11B =249
b11G =232
b11R =238
b12B =240
b12G =229
b12R =237
a=0
toplam=np.array([0,0,0])
while a<im_1.shape[2] :
        b=0
        while b< im_1.shape[1] :
                toplam[a]=toplam[a]+im_1[0,b,a]
                b=b+1
        a=a+1
```

```
print 'b1B', toplam[0]/im_1.shape[1],'b1G', toplam[1]/im_1.shape[1],'b1R', toplam[2]/im_1.shape[1]

print 'fark', b1B-toplam[0]/im_1.shape[1], b1G-toplam[1]/im_1.shape[1], b1R-toplam[2]/im_1.shape[1]

a=0

toplam=np.array([0,0,0])

while a<im_2.shape[2] :

        b=0

        while b< im_2.shape[1] :

                toplam[a]=toplam[a]+im_2[0,b,a]

                b=b+1

        a=a+1

print 'b2B',toplam[0]/im_2.shape[1],'b2G', toplam[1]/im_2.shape[1],'b2R', toplam[2]/im_2.shape[1]

print 'fark', b2B-toplam[0]/im_2.shape[1], b2G-toplam[1]/im_2.shape[1], b2R-toplam[2]/im_2.shape[1]

a=0

toplam=np.array([0,0,0])

while a<im_3.shape[2] :

        b=0

        while b< im_3.shape[1] :

                toplam[a]=toplam[a]+im_3[0,b,a]

                b=b+1

        a=a+1

print 'b3B', toplam[0]/im_3.shape[1],'b3G', toplam[1]/im_3.shape[1],'b3R', toplam[2]/im_3.shape[1]

print 'fark', b3B-toplam[0]/im_3.shape[1], b3G-toplam[1]/im_3.shape[1], b3R-toplam[2]/im_3.shape[1]

a=0
```

```
toplam=np.array([0,0,0])

while a<im_4.shape[2] :

        b=0

        while b< im_4.shape[1] :

                toplam[a]=toplam[a]+im_4[0,b,a]

                b=b+1

        a=a+1

print 'b4B', toplam[0]/im_4.shape[1],'b4G', toplam[1]/im_4.shape[1],'b4R', toplam[2]/im_4.shape[1]

print 'fark', b4B-toplam[0]/im_4.shape[1], b4G-toplam[1]/im_4.shape[1], b4R-toplam[2]/im_4.shape[1]

a=0

toplam=np.array([0,0,0])

while a<im_5.shape[2] :

        b=0

        while b< im_5.shape[1] :

                toplam[a]=toplam[a]+im_5[0,b,a]

                b=b+1

        a=a+1

print 'b5B', toplam[0]/im_5.shape[1],'b5G', toplam[1]/im_5.shape[1],'b5R', toplam[2]/im_5.shape[1]

print 'fark', b5B-toplam[0]/im_5.shape[1], b5G-toplam[1]/im_5.shape[1],b5R-toplam[2]/im_5.shape[1]

a=0

toplam=np.array([0,0,0])

while a<im_6.shape[2] :

        b=0

        while b< im_6.shape[1] :
```

```
                toplam[a]=toplam[a]+im_6[0,b,a]

                b=b+1

        a=a+1

print 'b6B', toplam[0]/im_6.shape[1],'b6G', toplam[1]/im_6.shape[1],'b6R', toplam[2]/im_6.shape[1]

print 'fark',b6B- toplam[0]/im_6.shape[1], b6G-toplam[1]/im_6.shape[1],b6R-toplam[2]/im_6.shape[1]

a=0

toplam=np.array([0,0,0])

while a<im_7.shape[2] :

        b=0

        while b< im_7.shape[1] :

                toplam[a]=toplam[a]+im_7[0,b,a]

                b=b+1

        a=a+1

print 'b7B', toplam[0]/im_7.shape[1],'b7G', toplam[1]/im_7.shape[1],'b7R', toplam[2]/im_7.shape[1]

print 'fark', b7B-toplam[0]/im_7.shape[1], b7G-toplam[1]/im_7.shape[1],b7R-toplam[2]/im_7.shape[1

a=0

toplam=np.array([0,0,0])

while a<im_8.shape[2] :

        b=0

        while b< im_8.shape[1] :

                toplam[a]=toplam[a]+im_8[0,b,a]

                b=b+1

        a=a+1

print 'b8B', toplam[0]/im_8.shape[1],'b8G', toplam[1]/im_8.shape[1],'b8R', toplam[2]/im_8.shape[1]
```

```
print 'fark', b8B-toplam[0]/im_8.shape[1],b8G-toplam[1]/im_8.shape[1], b8R-toplam[2]/im_8.shape[1]

a=0

toplam=np.array([0,0,0])

while a<im_9.shape[2] :

        b=0

        while b< im_9.shape[1] :

                toplam[a]=toplam[a]+im_9[0,b,a]

                b=b+1

        a=a+1

print 'b9B', toplam[0]/im_9.shape[1],'b9G', toplam[1]/im_9.shape[1],'b9R', toplam[2]/im_9.shape[1]

print 'fark', b9B-toplam[0]/im_9.shape[1], b9G-toplam[1]/im_9.shape[1], b9R-toplam[2]/im_9.shape[1]

a=0

toplam=np.array([0,0,0])

while a<im_10.shape[2] :

        b=0

        while b< im_10.shape[1] :

                toplam[a]=toplam[a]+im_10[0,b,a]

                b=b+1

        a=a+1

print 'b10B', toplam[0]/im_10.shape[1],'b10G', toplam[1]/im_10.shape[1],'b10R', toplam[2]/im_10.shape[1]

print 'fark',b10B- toplam[0]/im_10.shape[1], b10G-toplam[1]/im_10.shape[1],b10R-toplam[2]/im_10.shape[1]

a=0

toplam=np.array([0,0,0])

while a<im_11.shape[2] :
```

```
        b=0
        while b< im_11.shape[1] :
                toplam[a]=toplam[a]+im_11[0,b,a]
                b=b+1
        a=a+1
print 'b11B', toplam[0]/im_11.shape[1],'b11G', toplam[1]/im_11.shape[1],'b11R', toplam[2]/im_11.shape[1]
print 'fark', b11B-toplam[0]/im_11.shape[1],b11G- toplam[1]/im_11.shape[1], b11R-toplam[2]/im_11.shape[1]
a=0
toplam=np.array([0,0,0])
while a<im_12.shape[2] :
        b=0
        while b< im_12.shape[1] :
                toplam[a]=toplam[a]+im_12[0,b,a]
                b=b+1
        a=a+1
print 'b12B', toplam[0]/im_12.shape[1],'b12G', toplam[1]/im_12.shape[1],'b12R', toplam[2]/im_12.shape[1]
print 'fark', b12B-toplam[0]/im_12.shape[1],b12G- toplam[1]/im_12.shape[1], b12R-toplam[2]/im_12.shape[1]
height, width = bim11.shape[:2]
res3 = cv2.resize(bim11,(width/8,height/8))
cv2.imshow("image", res3)
cv2.waitKey(0)
```

ÖZGEÇMİŞ

KİŞİSEL BİLGİLER

Adı Soyadı: Ahmet TAŞKIRAN
Uyruğu: Türkiye (T.C)
Doğum Tarihi ve Yeri: 12.10.1992 - Nevşehir
Medeni Durum: Bekar
e-mail: tataskiran@gmail.com
Yazışma Adresi:

EĞİTİM

Derece	**Kurum**	**Mezuniyet Tarihi**
Yüksek Lisans	Erciyes Üniversitesi Elektrik Elektronik Mühendisliği	2018
Lisans	Erciyes Üniversitesi Elektrik Elektronik Mühendisliği, Bilgisayar Mühendisliği (Çift Anadal)	2015
Lise	2000 Evler Anadolu Lisesi, Nevşehir	2010

İŞ DENEYİMLERİ

Yıl	**Kurum**	**Görev**
2016-Halen	Femaş	3 Yıl
2015-2016	Chronos Teknoloji	1 Yıl

YABANCI DİL

İngilizce

YAYINLAR

1.TAŞKIRAN A., GÜNAY E., (2018). Classification of Chemical Urine Analysis Images. *SETSCI Conference Indexing System, 3* : 115-115