

**TÜRKİYE CUMHURİYETİ
YILDIZ TEKNİK ÜNİVERSİTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ**



EL YAZISINI DİJİTAL METNE DÖNÜŞTÜRME

16011059 – Ahmet Onur AKMAN

BİLGİSAYAR PROJESİ

Danışman
Dr. Öğr. Üyesi Göksel BİRİCİK

Haziran 2020

TEŞEKKÜR

İlk olarak proje süreci boyunca her ihtiyacım olduğunda yardımcı olan, projenin danışmanı Sayın Dr. Öğr. Üyesi Göksel BİRİCİK'e ve bugüne kadar bana kattıklarıyla bu proje üzerine çalışmamı mümkün kılmış olan bütün Yıldız Teknik Üniversitesi Bilgisayar Mühendisliği Bölümü akademisyenlerine teşekkür ederim.

Programlamaya ilgi duyan herhangi birinden deneyimli bir bilgisayar mühendisine kadar, derin öğrenme çalışma alanını herkes için erişilebilir kılan, bu proje boyunca kütüphanelerinden faydalandığım sektör devlerine teşekkür ederim.

Son olarak, bütün hedef ve isteklerim için bana her zaman desteklerini hissettiren, 2020 Covid-19 salgını sebebiyle İstanbul'dan Eskişehir'deki evime döndüğümde derslerimi kaçırımadam ve bu proje üzerinde çalışabilmem için bana uygun bir çalışma ortamı ayıran aileme, sadece teşekkür etmek yeterli olmayacak olsa da, teşekkür ederim.

Ahmet Onur AKMAN

İÇİNDEKİLER

KISALTMA LİSTESİ	v
ŞEKİL LİSTESİ	vi
TABLO LİSTESİ	viii
ÖZET	ix
ABSTRACT	x
1 GİRİŞ	1
2 ÖN İNCELEME	4
3 FİZİBİLİTE	8
3.1 Teknik Fizibilite	8
3.1.1 Yazılım Fizibilitesi	8
3.1.2 Donanım Fizibilitesi	9
3.2 İş Gücü Ve Zaman Fizibilitesi	9
3.3 Yasal Fizibilite	9
3.4 Ekonomik Fizibilite	9
4 SİSTEM ANALİZİ	10
4.1 Görüntü İşlemleri	11
4.1.1 Görüntünün Ayırtılmaya Uygun Hale Getirilmesi	11
4.1.2 Harflerin Ayırıtırlması	11
4.1.3 Harflerin Tahmin Aşaması İçin Uygun Hale Getirilmesi	11
4.2 Çıktı Kontrolü	11
4.2.1 Metin İçerigindeki Boşlukların Kontrolü	11
4.2.2 Tahmin Hatalarının Telafi Edilmesi	12
4.3 Sinir Ağları Modeli	12
4.3.1 Veri İhtiyacı	12
4.3.2 Sinir Ağları Mimarisi	13
4.3.3 Keras	14

4.3.4	Başarı Değerlendirmesi	14
5	SİSTEM TASARIMI	16
5.1	Yazılım Tasarımı	16
5.1.1	Görüntü Üzerine İşlemler	17
5.1.2	Harf Ayırma	20
5.1.3	Model Çıktısında Hata Düzeltme ve Kelime Tespiti	21
5.1.4	Yapay Sinir Ağlı Mimarisi	22
5.2	Veri Seti Kullanımı	30
5.3	Girdi ve Çıktı Tasarımı	31
6	UYGULAMA	32
6.1	Fotoğraf Filtresi	32
6.2	Satır ve Harf Algılama	32
6.3	Model Tarafından Tahmin Yürüttülecek Görüşlerin Input Formatına Uyarlanması	33
6.4	Harf Tanımlama ve Metin Oluşturma	34
6.5	Çıktı Metninde Hata Azaltma Çalışmaları	35
7	DENEYSEL SONUÇLAR	37
8	PERFORMANS ANALİZİ	40
9	SONUÇ	41
	Referanslar	43
	Özgeçmiş	44

KISALTMA LİSTESİ

CNN	Convolutional Neural Network
ReLU	Rectified Linear Unit
EMNIST	Extended Modified National Institute of Standards and Technology
LSTM	Long Short Term Memory
MNIST	Modified National Institute of Standards and Technology
NIST	National Institute of Standards and Technology

ŞEKİL LİSTESİ

Şekil 2.1 Google Translate, kamera destekli yanında çeviri özelliği	4
Şekil 3.1 Gantt Chart	9
Şekil 4.1 Proje için hazırlanan Use Case Diyagramı	10
Şekil 4.2 EMNIST veri seti sınıflarının içeriği. [5]	13
Şekil 5.1 Sistem için hazırlanan Sequence Diyagramı	16
Şekil 5.2 Sistemin yapısını açıklamak için oluşturulmuş şema	16
Şekil 5.3 Kullanılacak görselin orijinal hali	18
Şekil 5.4 Filtre değerleri: 10 ve 11	18
Şekil 5.5 Filtre değerleri: 30 ve 40	19
Şekil 5.6 Önce binerizasyon, daha sonra gürültü azaltma uygulanmış görsel.	19
Şekil 5.7 Önce gürültü azaltma, sonra binerizasyon uygulanmış görsel. . .	20
Şekil 5.8 Aynı görselin iki farklı metotla küçültülmesi.	21
Şekil 5.9 pyenchant.github.io/pyenchant linkinde bulunan Enchant kullanım snippeti	22
Şekil 5.10 İncelenen CNN Modelleri	23
Şekil 5.11 Farklı Dropout değerleri için alınan accuracy değerleri	26
Şekil 5.12 Learning Rate Schedulers kullanımının loss grafiklerinde yarattığı değişim	26
Şekil 5.13 Data Augmentation kullanımının accuracy ve loss grafiklerinde yarattığı değişim	27
Şekil 5.14 Farklı epoch sayılarıyla yapılan training sonucunda oluşan accuracy ve loss grafikleri.	28
Şekil 5.15 Farklı batch size değerleriyle yapılan training sonucunda oluşan accuracy ve loss grafikleri.	29
Şekil 5.16 EMNIST verilerinin oluşturulma şekli[5]	30
Şekil 5.17 Balanced sınıfı içerisindeki veri örneği dağılımı[5]	30
Şekil 5.18 Kullanıcı için tasarlanmış tkinter yapısı.	31
Şekil 5.19 Kullanıcının görsel seçmesi için sağlanan yol.	31
Şekil 6.1 Örnek bir fotoğraf ve filtrelemeden sonraki hali.	32
Şekil 6.2 Harf algılama algoritmasının görsellerdeki performansına bir örnek.	33
Şekil 6.3 Harf algılama algoritmasının görsellerdeki performansına bir örnek.	33

Şekil 6.4 Resize edilmiş bazı harf görselleri	34
Şekil 6.5 Algılanan harfin bir görsel halinde kırılmasından sonra istenen input formuna dönüştürme aşamaları	34
Şekil 6.6 Tek karakter çizimleriyle modelden alınan bazı pozitif sonuçlar .	35
Şekil 6.7 Tek karakter çizimleriyle modelden alınan negatif sonuçlar	35
Şekil 6.8 Son çıktıda hata azaltma çalışmaları	36
Şekil 7.1 Örnek input ile programın yaptığı okuma işleminin sonucu. . . .	37
Şekil 7.2 Örnek input ile programın yaptığı okuma işleminin sonucu. . . .	37
Şekil 7.3 Örnek input ile programın yaptığı okuma işleminin sonucu. . . .	38
Şekil 7.4 Örnek input ile programın yaptığı okuma işleminin sonucu. . . .	38
Şekil 7.5 Örnek input ile programın yaptığı okuma işleminin sonucu. . . .	38
Şekil 7.6 Örnek input ile programın yaptığı okuma işleminin sonucu. . . .	38
Şekil 7.7 Örnek input ile programın yaptığı okuma işleminin sonucu. . . .	39
Şekil 7.8 Örnek input ile programın yaptığı okuma işleminin sonucu. . . .	39
Şekil 7.9 Örnek input ile programın yaptığı okuma işleminin sonucu. . . .	39

TABLO LİSTESİ

Tablo 3.1 Maliyet Tablosu	9
Tablo 8.1 Input Özelliklerine Göre Okuma Süresi	40

ÖZET

EL YAZISINI DİJİTAL METNE DÖNÜŞTÜRME

Ahmet Onur AKMAN

Bilgisayar Mühendisliği Bölümü
Bilgisayar Projesi

Danışman: Dr. Öğr. Üyesi Göksel BİRİCİK

Bu çalışma insan eliyle oluşturulmuş bir el yazısı metnin dijital metne dönüştürülmesini işlevinde bir program ortaya koymayı amaçlar. Günlük hayatı ortaya çıkan verilerin dijitalleştirimesi için yapılan çalışmaların bir parçası olan karakter/dijit tanımlama çalışmaları sayesinde, üretilen yazılı belgelerin daha iyi ve daha düzenli saklanabilmesi mümkün olmuştur. Bu proje ile bu çalışma alanı üzerine araştırma yapılmış ve bu alana yönelik bir çalışma ortaya konmuştur.

Program temel olarak görüntü işleme, karakter ayıklama, karakter tanımlama ve metin işleme olmak üzere 4 farklı bölümden oluşmaktadır. İnsan eliyle yazılmış bir metnin fotoğrafı, bu program içerisinde önce işlenir, sonra metindeki karakterler ayırtırılır, ayırtırılan karakterler EMNIST dataseti ile eğitilmiş olan CNN mimarisinde bir yapay sinir ağları modeli ile sınıflandırılır ve sınıflandırılan karakterler dijital metin haline getirilir. Bu metin kullanıcıya sunulmadan önce bir dizi hata ayıklama işleminden geçer. Kullanıcı en sonunda sağladığı görseldeki metni çıktı olarak alır.

Programın belirtilen modülleri farklı yaklaşımların denenmesi sonucu elde edilen en optimal yapının seçilmesi ile oluşturulmuştur.

Anahtar Kelimeler: El yazısı metnin dijital metne dönüştürülmesi, Yapay sinir ağları, CNN, karakter tanımlama

ABSTRACT

CONVERSION OF HANDWRITING TO TEXT

Ahmet Onur AKMAN

Department of Computer Engineering
Computer Project

Advisor: Dr. Göksel BİRİCİK

This study aims to present a program in the function of converting handwritten text into digital text. Thanks to character / digit identification studies, which are a part of the studies for digitizing the data that emerge in daily life, it has been possible to store the written documents better and more organized. By this project, research has been done on this field of study and a functioning product has been put out.

The study basically consists 4 different parts: image processing, character extraction, character identification and text processing. In this program, the photograph of a handwritten text is processed, then the characters in the text are extracted, the parsed characters are classified by the artificial neural network model in CNN architecture, trained with the EMNIST dataset and the classified characters are converted into digital text. This text goes through a series of correction before it is presented to the user. The text which was obtained from the given image is presented as the output to the user.

The specified modules of the program were created by selecting the most optimal structure obtained by trying different approaches.

Keywords: Conversion of handwritten to text, Artifical Neural Network, CNN, character identification

1 GİRİŞ

İçinde bulunduğuuz çağda nüfus artışı ve teknolojinin hızla gelişmesi sonucu veri ve verinin yönetimi önem kazanmıştır. Öyle ki; artık hayatımızın her yerinde olan sensörler, kullandığımız elektronik cihazlar, internette paylaştığımız veya öne çıkardığımız içerikler ve bunun gibi birçok etmen sayesinde var olan veri her gün katlanarak artmaktadır. Bunun yanında, verinin değer kaybetme hızının yeni veri oluşum hızına göre çok daha yavaş olması da verinin birikmesine sebep olmuştur. Böylesine hızla artan ve çeşitlenen bu olguyu depolamak ve yönetmek her geçen gün daha zor hale gelmektedir. Bu sebeple veri depolama ve veri yönetme teknolojileri üzerinde harcanan para artmış ve veri yönetimi teknolojileri için yapılan çalışmalar önem kazanmıştır.

Verinin her geçen gün çoğalması ve çeşitlenmesi ile eski veri depolama metodlarımız artık yetersiz kalmış, teknolojik cihazlar bu amaçla kullanılmaya başlamış ve birkaç on yıl içerisinde belli bir miktardaki veriyi saklamak için gereken fiziksel alan git gide küçülmüştür. Fiziksel boyutu ve maliyeti azalan veri depolama birimleri sayesinde her gün oluşan verileri yakalamak ve onları kayıt altında tutmak mümkün olmuştur.

Veri depolama birimlerinin gelişmesi ve ucuzlaşması her ne kadar saklanabilen veri miktarını artırsa da, saklanan verinin yönetilmesi ve değerlendirilmesi de bir o kadar zorlaşmıştır. Günümüzde bireysel kullanım amacıyla piyasaya sürülen bilgisayarların hafızaları bile birkaç terabyte veri saklayabilmektedir. Herkesin elinin altında bunca büyük veri depolama imkanının olması, ticari kuruluşların veri teknolojilerine çok büyük yatırımlar yapması ve devlet yapılaşmalarının bile dijitalleşmeye başlaması ile kaydedilen ve işlenmesi gereken veri devasa boytlara ulaşmıştır. Ancak saklanan veri kullanışlı olmadıkça bir anlam ifade etmemektedir. Sadece gerekli verilerin saklanması ve saklanan bu verileri düzenli, hızlı erişilebilir ve kolay kullanılabilir olması bilişim teknolojilerinin en büyük hedeflerinden biridir. Bahsettiğimiz veri miktarı ve karmaşıklığı bu uğraş için insan beyini yetersiz bırakmıştır. Bu yetersizlik sonucu veri yönetimi teknolojileri ortaya çıkmış ve bu teknolojiler günümüzün en hızlı gelişen teknolojileri arasındadır.

Veri depolama ve veri işleme teknolojilerinin gelişimi her ne kadar hızlı olsa da, bu teknolojilerin ihtiyaçlarını tam olarak karşılayacak kapasitede olduğunu söylemek doğru değildir. Bazı alanlar üzerine yapılan çalışmalar yetersizdir, bazı alanlarda yapılan çalışmalar yeterli olsa bile ortaya çıkan ürünler henüz herkes tarafından kolayca erişilebilir hale gelmemiştir. Dünyadanın birçok yerindeki birçok gelişmiş ülke dijitalleşme sürecinde ilerleyememiş, bu da eldeki verilerin hala zor erişilebilir olmasına yol açmıştır. Zor erişilebilir olan veriler günlük hayatın yavaşlamasına ve gereken insan gücünün artmasına neden olmaktadır.

Bu eksikliklerin yanı sıra, bu teknolojileri halihazırda kullanmasına karşın hâlâ bu teknolojilerin sunabileceği kolaylıklardan tam olarak faydalananı bekleyen yapılar da mevcuttur. Buna örnek olarak toplumsal emniyet/asayiş birimleri verilebilir. Gelişmiş birçok ülkedeki polis departmanları işlenen suçları inceleme ve sorumluları tespit etme amacıyla elinde biriktirdiği verilerden faydalnamaktadır. İncelenen ve sonuca bağlanan her olay da yeni veriler doğurmaktadır. Ancak bu verileri depolama ve işleme çalışmaları tam olarak dijitalleşmemiştir ve insan beyinin yapacağı hata oranını sıfıra indirmek maalesef mümkün olmayacağından.

Günümüzde teknolojinin geldiği yer bize insan eliyle oluşturulan fiziksel verileri daha iyi işleyerek çok daha verimli sonuçlar elde etme imkanı sunmuştur. Bunlara örnek olarak yüz, yazı ve ses algılama/tanımlama temelli çalışmalar verilebilir. Günümüzde sahip olduğumuz bilgisayarların işlem gücü ve saklayabileceği veri miktarı düşünülürse bu çalışmalarının potansiyelinin çok büyük olduğu sonucuna varılacaktır.

Bu çalışma alanlarının en popüler uğraşlarından biri de insan eliyle yapılan çizimlerin analiz edilmesidir. Bir insan beyni başka bir insan tarafından yazılan yazıyı (eğer yazı okunaklıysa) okuyabilme ve tanıdığı insanların el yazısıyla karşılaştırarak bu yazıyı yazan kişiyi tespit etme potansiyeline sahiptir. Bahsettiğimiz uğraşların en büyük amacı, bu beceriyi dijital ortamda oluşturmak, hata oranını olabildiğince azaltmak ve bilgisayarların saklayabileceği veri miktarından faydalananarak geniş çaplı hizmet veren ürünler ortaya koymaktır.

Bu projede odaklanacağımız uğraş alanı, insan eliyle bir kağıda yazılmış bir metni algılamak ve tanımlamak olacaktır. Bu ve bu tip çalışmalar, şüphesiz ki verileri daha verimli depolama ve o verilere daha kolay erişme konusunda bize yardımcı olma potansiyeline sahiptir.

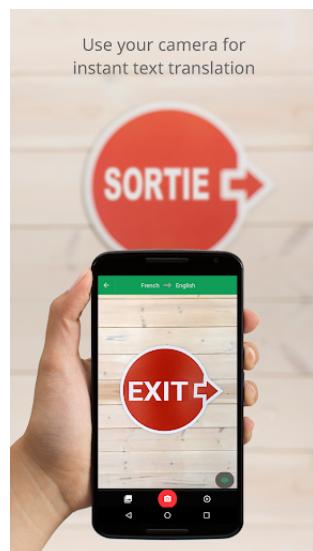
Elbette ki elimizdeki teknolojiler bu belgelerin fotoğrafını saklamak için son derece yeterlidir. Ancak sakladığımız her veri bize bir bütün olarak gerekmemektedir. Bu tip belgeleri metin belgesi olarak saklamak, içeriği bilinen bir belgenin daha kolay

bulunmasına, bu belgenin değiştirilebilmesine, belgenin içeriğinin farklı biçimlere dönüştürülmesine ve bu belgeden kolayca alıntı yapılmasına olanak sağlayacaktır. Üstelik bu teknoloji sadece belge saklamak için değil, günlük hayatı anlık çeviri, anlık yazı/ işaret algılama, yazıyı dikte etme ve algılanan yazıyı analiz etme gibi birçok farklı alanda da kullanılabilir ve kullanılmaktadır.

Bu alanda yapılan ilk çalışma elbette ki bu proje olmayacağıdır. Bu projenin yapılmasıındaki en önemli amaç, doğruluk oranı yüksek olan bir çalışma ortaya koyarak, bu gelişime açık uğraş alanına bir perspektif eklemek ve bu alanda ortaya koyacağım çalışmalar için iyi bir temel atmaktır.

2 ÖN İNCELEME

Bu bölümde bu projenin çalışma alanının geçmişi ve günümüzdeki durumu incelenecuk ve adım adım proje planlamasının üstünden geçilecektir. Görüntü işleme temelli çalışmalar her ne kadar hızla popülerleşiyorsa da bu teknolojileri kullanan ürünler hayatımıza uzun zaman önce girmiştir. Günümüzde kullandığımız görüntü işlemeli birçok program yüz, renk, nesne, yazı, sayı ve daha birçok öğeyi görüntü içerisinde seçip çıkarabilmektedir. Her ne kadar bir insanın kapasiteleri ile yarışma konusunda zorluk çekilse de, gelişime son derece açık ve ortaya koyulan çalışmaları heyecan verici olan bu alan gelecekte yapılacak çalışmalarla birlikte günlük yaşamda şüphesiz ki çok daha büyük bir yer tutacaktır.



Şekil 2.1 Google Translate, kamera destekli anında çeviri özelliği

Bilimsel çalışmalar aşamalarından uzun zaman önce çıkıştı ve günlük hayatı yerini çöktürdü. Çok sayıda uygulama yapabildikleri ile insanların dikkatini toplamış ve programlama dünyasının içindeki birçok meraklı kişiyi kendine çekmiştir. Bu uygulamalardan bazıları çoktan günlük yaşamımızın temel parçalarından biri olmuştur. Resim-2.1'de örnek verilen bu uygulamalardan biri kamera ile gösterilen bir alanda bulunan yazıları algılayıp bu yazıları anlık olarak çevirebiliyorken, başka

bir tanesi gösterdiğiniz el yazısını analiz ederek kişilik tahlili yapmakta[1], bir başkası da insan yüzünün bölümlerini ayırt edip o yüzdeki parçaların oranlarıyla oynaması şansı vermektedir[2]. Şüphesiz ki bu uygulamalar gelecekte bu alanda ortaya çıkacak daha kompleks teknolojilerin temel taşılarıdır.

Robotik alanı da son yıllarda popülerleşen alanlardan bir tanesidir[3]. Tahmin edilebileceği üzere görüntü işleme ile hiç de uzak değildir, teknolojinin insan hayatına gelecekte olabilecek etkisi en çok tartışılan çalışma alanları arasındadır[4]. Öyle ki dünyanın bir yerinde robotik alanında yapılan bir çalışmanın başarılı olan sonucu, günler içinde herkes tarafından bilinir hale gelmemektedir. Peki herkesi son derece heyecanlandıran bu çalışmaların ürünlerini, yerine getirebileceği fonksiyonları yerinde kullanmayı başaramazsa bu ürünlerin hayatımızda yeri ne olacaktır? Şüphesiz ki insansı veya insansı olmayan mekanik bir robotun fonksiyonel olması için ihtiyacı duyduğu ilk yeti, veri toplayabilmek ve onu iyi işleyebilmektir. Günümüzde halihazırda birçok cihaz, düğmesine bastığınız anda çalışmaya başlamaktadır. Ancak gelecek denince insanların ilk hayal ettiği şey düğmesine basılınca işini bugün yaptığından daha iyi yapacak makineler değil, kendi düğmesine ne zaman basacağını bilen cihazlardır.

Bu projede üstünde çalışılacak konu da az önce bahsedilen robotik gibi daha birçok heyecan verici uğraş alanının çok önemli bir parçasıdır. Bir makineye tipki bir insan gibi yazı okuma yeteneğini aşılama fikrinin verdiği heyecanla seçilmiş ve üzerine çalışmaya başlanmıştır. Bu projenin amaçlarından biri de, bugüne dek yapılan projelerin doğruluk oranlarının yükseltilmesinin mümkün olup olmadığı hakkında fikir sahibi olmaktadır.

Bu projede hedeflenen ürünün ortaya çıkması, farklı işlevlere sahip birden fazla parçanın üretilerek bir araya getirilmesiyle mümkün olacaktır. Öncelikle, programın algılayabileceği bir görsele ihtiyacı olacaktır. Bu aşamada bu resmin yolu sağlanan, önceden hazırlanmış bir resim dosyası olması planlanmıştır. Eş zamanlı resmi çekilen metni algılayan bir programın bilgisayar ortamında çalışması pek mümkün gözükmemektedir. Bu resmin içinde açık renkli bir kağıt üstünde koyu renkli bir kalemle yazılmış bir metin olacağı düşünülecektir. Aynı zamanda şu anki birikimimle ve elimizde olan zaman miktarı sebebiyle, insan beyni tarafından bile okunurken zorlanılan karmaşık el yazıları, bitişik el yazıları ve satır çizgilerine sadık kalınmayan metin yapıları hedef alınmayacağından emindi. Programın bir fotoğrafaktaki yazılı algılaması için kağıt ve yazının renk farkından faydallanması planlanmaktadır. Bu renk farkının maksimize edilmesi ile bu işlemin kolaylaştırılması hedeflenmektedir. Fotoğrafın tamamen siyah beyaz hale getirilmesi ile elde edilecek görselin daha kolay analiz edileceği düşünülmektedir. Bu aşama esnasında görseldeki gürültü programın kafasını

karıştıracağından gürültü azaltıcı filtrelere de ihtiyaç duyacaktır.

Bu aşamadan sonra izlenebilecek birkaç yol vardır. Program metindeki cümleleri, kelimeleri veya harfleri ayrı etmeye çalışabilir. Her ne kadar kelimeyi bütün olarak ayırt etmek daha az işlem adımı ortaya çıkaracak olsa da, harfleri ayırip onları tanımlamanın daha verimli olacağı düşüncesiyle bu yolun gerçekleştirilmesi amaçlanmaktadır. Bunların yanında, programın bütün dillere hizmet veremeyeceğini belirtmek gereklidir. İngilizce ve Türkçe dilleri kullanışlı olacaktır. Ancak internette Türkçe karakterler için hazırlanan dataset eksikliği sebebiyle, İngilizce dili için çalışan bir programın oluşturulması hedef olarak seçilmiştir.

Harfleri ayırt eden programın onları tanımlaması için bir çeşit düşünme mekanizmasına sahip olması gerekmektedir. Bu ihtiyaç yapay sinir ağları ile sağlanacaktır. LSTM biçimine sahip bu yapının bu çalışmada kullanışlı olacağı düşünülmüş ve bu yapının dijit/karakter algılama işlevine sahip çalışmalarda popüler olduğu görülmüştür. Bu projede de bu yapının kullanılması düşünülmektedir. Seçilecek datasetin büyüklüğü de hesaba katılarak en kullanışlı layer sayısı ve yapısı bulunacak ve tasarılanacaktır.

Sistemin bir başka ihtiyacı da veridir. Nasıl insan beyni bir insanın el yazısını daha önce görmeden, o insanın yazdığı bir yazının o insana ait olduğunu tespit edemeyecekse, bu programın da el yazısına aşina olması gereği aşikardır. Bu sebeple elle çizilmiş karakter ve dijit örneklerine sahip kullanışlı bir dataset bulunarak tasarlanan sinir ağı yapısı eğitilecek, test edilecek ve bu test sonucuna göre optimize edilecektir.

Ayrılmış karakterleri tespit eden program daha sonra bu karakterleri birleştirmek durumunda olacaktır. Bu birleştirme adımı için hangi boşluğun harf arası, hangi boşluğun kelimeler arası olduğunu tespit etmek durumunda da kalacaktır. İnsan el yazısının değişkenliği göz önüne alındığında bu adımın problem yaratabileceği öngörülmektedir.

Başka hedeflenen adımlardan biri de elde edilen kelimelerin doğruluğunun kontrolüdür. Eğer bir sözlük verilerinin kullanılabileceği ve programa dahil edilebileceği görülürse, bu adım uygulanacaktır. Bir İngilizce sözlükten faydalananarak kelimelerin gerçekçiliğinin kontrol edilmesi planlanmaktadır.

En sonunda elde edilen metnin ekranaya yazdırılması veya text olarak kaydedilmesi planlanmaktadır.

Belirtilen program adımlarında kullanılmak üzere faydalı ve yüksek performanslı kütüphanelere sahip olması, bu ve bu tip çalışmalarda sık kullanılan bir dil olması ve

bir sorun yaşadığı takdirde sorun çözümü için internet ortamında kolay bir şekilde yardım temin edilebilebilmesi dolayısıyla kullanılacak programlama dili olarak Python uygun görülmüştür.

3 FİZİBİLİTE

3.1 Teknik Fizibilite

3.1.1 Yazılım Fizibilitesi

3.1.1.1 Programlama Dilleri

Programlama dili seçiminde Python, Java ve R dillerinin avantajı ve dezavantajları karşılaştırılmıştır.

Python'un bu gibi çalışmalar için en popüler seçim olduğu ve barındırdığı kütüphaneler ve kolay bir syntax'ı olması sayesinde yeni başlayanlar için iyi bir seçim olduğu öğrenilmiştir. Ayrıca object-oriented tasarımlı da destekliyor olması onu daha da kullanışlı kılmaktadır. Öte yandan R dilinin Python'a kıyasla, veri görselleştirilmesi konusunda daha iyi olduğu görülmüştür. Bu, geliştirme sürecinde sık sık grafikler ve çizimler üzerine çalışacağımız düşünüldüğünde, önemli bir avantaj olarak göze çarpmaktadır. Java ise sunduğu kolay debugging süreci, yine veri görselleştirme konusunda iyi olması ve çok geniş bir kullanım alanına sahip olması sayesinde bu karşılaştırmaya giren diller arasında olmuştur.

Programın kapsamı çerçevesinde ihtiyaç duyulacak fonksiyonları yerine getirecek olan kütüphanelere (Tensorflow, Keras, Numpy, OpenCV) sahip olması sebebiyle ve diğerlerine kıyasla bu alanda çok daha popüler bir programlama dili olmasından dolayı karşılaşılan sorunlar için daha kolay çözüm bulabileceği varsayımlı ile Python dilinin kullanılması uygun görülmüştür.

3.1.1.2 Geliştirme Ortamı

Elde halihazırda bulunması ve kullanılacak program ve kütüphanelerin tümünün desteklenmesi sebebiyle Windows 10 işletim sistemi kullanılacaktır. Programın Python ile geliştirilecek olması ve projenin kapsamı göz önünde bulundurulduğunda uygun IDE'nin Spyder olacağını karar verilmiştir. Bu seçimde Spyder'in popülerliği, (diğerlerine kıyasla) sadece bulundurulmak istenen kütüphaneleri bulundurması göz önünde bulundurulmuştur.

3.1.2 Donanım Fizibilitesi

İşlemci: Intel(R) Core(TM) i7-6500U (2.5 GHz frekans)

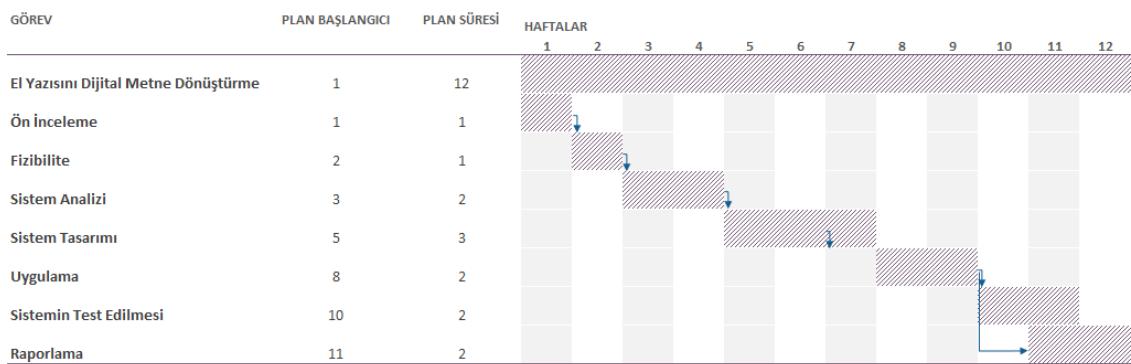
Bellek: 12 GB DDR3

Yukarıdaki özelliklere sahip bir adet dizüstü bilgisayar mevcuttur, proje için yeterli olacağına kanaat getirilmiştir.

3.2 İş Gücü Ve Zaman Fizibilitesi

Projenin bir üniversite eğitim dönemi sürmesi planlanmıştır.

El Yazısını Dijital Metne Dönüşümme



Şekil 3.1 Gantt Chart

3.3 Yasal Fizibilite

Projede kullanılan bütün yazılımsal ürünler ve kullanılan EMNIST dataset açık içeriklerdir. Projede örnek oluşturmak adına kullanılan el yazıları görüntülerinin kullanılması için yazan kişilerin rızası alınmıştır.

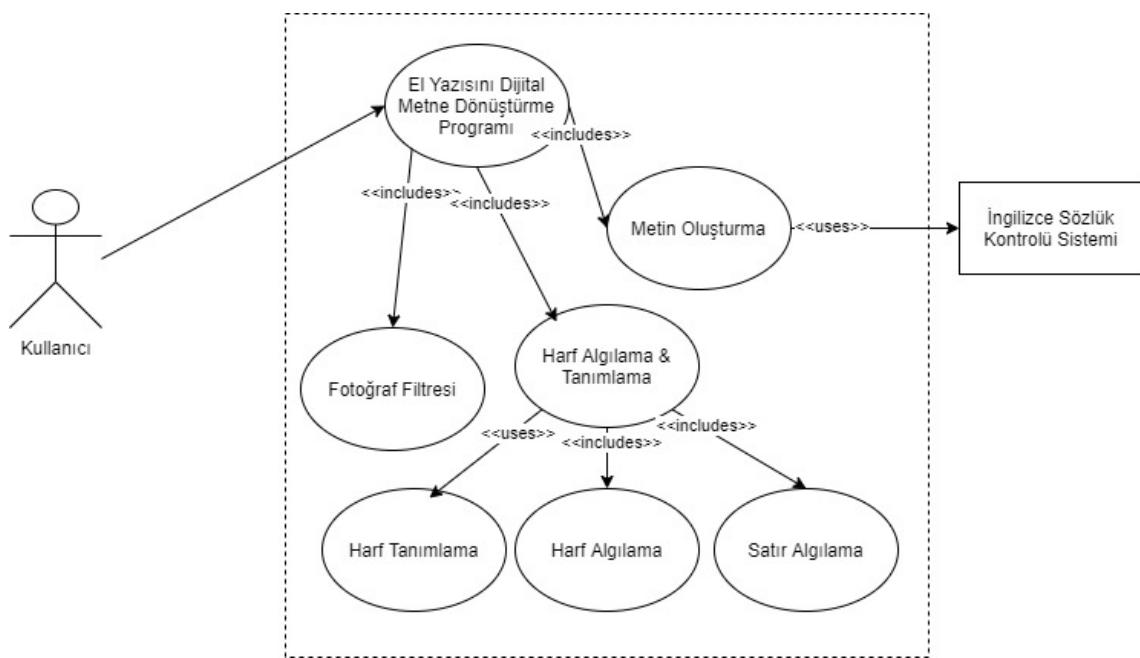
3.4 Ekonomik Fizibilite

Tablo 3.1 Maliyet Tablosu

Gereklik	Maliyet (Türk Lirası)
Gerekli özelliklere sahip bir bilgisayar	5000
Internet erişimi (Proje süreci boyunca)	180
Geliştirme ortamı yazılımı	0
Veri seti	0

Kullanılan ortamlar ve veri seti açık olduğundan herhangi bir ekonomik yük oluşturmayacaktır. Donanımsal gereklilikler ve internet erişimi ihtiyacı zaten karşılanmış olduğu için projenin yaratacağı bir maliyet yoktur.

4 SİSTEM ANALİZİ



Şekil 4.1 Proje için hazırlanan Use Case Diyagramı

El yazısını dijital metne çevirme çalışmaları, gerçek hayattan toplanan verileri dijitalleştirerek daha iyi bir saklama planı, daha kolay işleme imkanı ve bu verileri daha iyi analiz etme şansı yaratmayı amaçlayan çalışmaların bir tanesidir. Bu konuda yapılan çalışmalarla en büyük hedef, el yazısıyla oluşturulmuş belge, eser ve bunlara benzer dökümanları bir insanın bu yazılı dökümanları analiz etme becerisine erişmektir. Bu hedefe ulaşılması durumunda hem büyük bir iş yükü ortadan kalkacak, hem de elektronik cihazlarımızın fonksiyonelliği artacaktır.

Bugün kadar yapılan el yazısı tanımlama çalışmalarında yüksek doğruluk oranlarına erişilmiştir. Bu çalışmalar arasında dijit tanımlamaya yönelik çalışmaların sonuçlarının doğruluğunun daha yüksek olduğu gözlemlenmiştir. Bu projede de bu çalışmalara benzer bir yaklaşımla olabildiğince yüksek doğruluk oranına sahip bir model kullanarak, bir kağıt görselinden metin çıkarımı yapılması amaçlanmaktadır.

4.1 Görüntü İşlemleri

4.1.1 Görüntünün Ayristirmaya Uygun Hale Getirilmesi

Kullanıcı tarafından verilen görselin taranması ve ayristirılması için belli islemlerden geçirilmesi gerekmektedir. Kullanıcıdan beklenen gürültüsü olabildiğince az bir görsel sağlaması olsa da, gerçek hayatı bunu her zaman sıfıra indirmek mümkün olmamaktadır. Bu sebeple görüntünün gürültüsünün azaltılması ihtiyacı ortaya çıkmaktadır.

Bunun ardından fotoğraftaki yazı ve arka planın en doğru şekilde anlaşılması adına, fotoğrafın binerizasyon işleminden geçmesi gerekmektedir. Bu işlem sonucunda temiz, ayırt edilebilirliği yüksek, arka plandan tamamen aykırı renkte bir metin beklenmektedir.

Bu işlem için Python'un görüntü işleme kütüphaneleri son derece yeterlidir. Bu işlemler için PIL, numpy kütüphanelerinin kullanılmasına karar verilmiştir.

4.1.2 Harflerin Ayristirilmasi

Elimizdeki görselin taranması aşamasında, görselin bir numpy array formatına getirildikten sonra, matris üzerinde işlem yapma mantığıyla ek bir kütüphane yardımını gerektirmeden taranması düşünülmüştür. Bu ayristirma işleminde önce satırlar tespit edilecek, bu satırlar daha sonra harflere ve boşluklara bölünecektir.

4.1.3 Harflerin Tahmin Aşaması İçin Uygun Hale Getirilmesi

Onceki adımdan sonra elimizdeki harflerin de işlenmek için uygun hale gelmesi gerekmektedir. Bunun için harf görselinin kare hale getirilmesi, renklerinin sinir ağlarının eğitim setiyle uyumlu hale getirilmesi ve boyutlarının sinir ağları modelinin girdi boyutuna uyarılması gerekmektedir.

4.2 Çıktı Kontrolü

4.2.1 Metin İçerigindeki Boşlukların Kontrolü

Maalesef sinir ağları modelimizi sadece karakter ve rakamlar üzerine eğitme şansımız olduğu için, kelime ve harfler arasındaki boşlukların tespiti ve boşlukların sınıflandırılması için ayrı bir yapıya ihtiyaç duyulacaktır. Bunun için görsel tarama aşamasında kaydı tutulacak beyaz piksel sütunlarının sayılarının okunup bu değerlere göre sınıflandırma yapılması amaçlanmaktadır.

4.2.2 Tahmin Hatalarının Telafi Edilmesi

Her ne kadar bir zorluk olarak işlenen karakterlerin sayısı geleneksel karakter tanımlama çalışmalarının aksine birden çok olsa da, bu bize aynı zamanda bir avantaj sağlamaktadır. Şüphesiz ki proje sonunda elde edeceğimiz model tam doğrulukta olmayacağından emin olmak isteyen bir kullanıcının hedefi, bu hataları telafi etmek mümkündür.

Dijit sınıflandırma çalışmalarının en büyük avantajı, sadece 10 adet sınıflandırma opsiyonu olmasıdır. Ancak bu proje gibi çalışmalarında bu sayıya alfabetik karakterler de eklenmektedir. Bu durumda bazı çizimlerin aşırı biçimde başka karakterlere benzemesi ve model tarafından yanlış sınıflandırılması ihtimali yüksektir. En temel örnek olarak "O" harfi ve "0" rakamı benzerliği verilebilir. Bu tip hataları düzeltmek adına kelimenin context bilgisinin kullanılması amaçlanmaktadır.

Maalesef sınıflandırma hataları bununla sınırlı kalmayacaktır. Bu nedenle başka bir kontrol mekanizması olarak İngilizce sözlük kontrolünün kullanılması planlanmaktadır.

4.3 Sinir Ağları Modeli

Karakter, dijit, nesne, yüz, parmak izi, ses ve bunun gibi diğer verilerin tanımlanması/sınıflandırması çalışmalarında eğitilebilen ve bu öğrenciklerinden çıkarım yapabilen bir yapının varlığı gereklidir. Bu yüzden uygun niteliklerde bir sinir ağlarının tasalanması gerekecektir.

4.3.1 Veri İhtiyacı

Kuşkusuz her sinir ağı yapısının çıkarım yapması için öncelikle çıkarım yapacağı verilere benzer veriler görmüş ve bunların ne olduğunu öğrenmiş olması gerekmektedir. Aynı şekilde, programcının oluşturduğu yapının doğruluğunun gözlemlemesi için yine örnek verilere ihtiyacı olacaktır. Göz önünde bulundurulmalıdır ki, bahsedilen yapının tahmin verimliliği ve modelin doğru eğitilmesinin anahtarı da kaliteli bir veri setidir.

El yazısı karakter ve rakam veri seti seçiminde internette paylaşılmış ve bu projeye benzer amaçla oluşturulmuş projeler ve onların aldığı sonuçlar göz önünde bulundurulmuştur. Bu araştırmalar sonucunda uygun veri setinin EMNIST olduğunda karar kılınmıştır.

EMNIST, NIST dataseti temelli bir el yazısı dijit/karakter datasetidir. Sadece digitlerden oluşan ve oldukça zengin olan NIST datasetinin veri sınıflandırmadaki problemlerini çözme amacıyla ortaya çıkan, daha dar bir veri genişliğine sahip olan MNIST datasetinin genişletilmiş ve karakter çizimleri ile zenginleştirilmiş

versiyonudur.[5] İçerisindeki görselleri 28x28 piksel değerleri ile tutmaktadır. EMNIST dataseti her ne kadar geniş veri yelpazesine sahip bir dataset olsa da, sınıflandırma konusunda karmaşık bir datasetidir. 6 farklı bölüme ayrılmıştır ve her bölümün farklı özellikleri vardır. Bu bölümlerin çoğu bir çok ortak karakterler vardır ancak test sınıfı genişliği, training sınıfının genişliği, harf verileri genişliği ve sayı verileri genişliği gibi kriterlere göre farklılaşmışlardır. Biz bu sınıflar arasından en dengeli sayılara sahip olan ve 131,600 görsel sayısına sahip “Balanced” adlı sınıfı kullanacağız.

Name	Classes	No. Training	No. Testing	Validation	Total
By_Class	62	697,932	116,323	No	814,255
By_Merge	47	697,932	116,323	No	814,255
Balanced	47	112,800	18,800	Yes	131,600
Digits	10	240,000	40,000	Yes	280,000
Letters	37	88,800	14,800	Yes	103,600
MNIST	10	60,000	10,000	Yes	70,000

Şekil 4.2 EMNIST veri seti sınıflarının içeriği. [5]

4.3.1.1 Veri Setinin Bölünmesi

Her ne kadar veri setinin temin edilmesindeki temel amaç modelimize deneyim kazandırmak olsa da, yaptığımız eğitimin gidişatını takip etmek ve bu eğitimin testini yapmak için de veriye ihtiyacımız olacaktır. EMNIST Balanced Class bize training ve testing olmak üzere iki küme örnek sunuyor. Bunların yanında eğitim boyunca modelimizin gidişatını izlemek, her adımdan sonra onu test edip adım adım gözlem yapmak amacıyla bir grup "validation" verisine olan ihtiyacımızı karşılamak adına training kümelerinin bir kısmını ayırmamız gerekecektir. Eğitim boyunca hem training hem de validation adımlarının doğru ilerlemesi ve sonuçların güvenilir olması adına bu bölge işleminin oranı doğru seçilmelidir.

4.3.2 Sinir Ağları Mimarisi

Proje seçimi ve planlanması aşamasında, LSTM mimarisinde bir yapının kullanılması planlanmaktadır. LSTM yapısı, geri besleme bağlantıları olduğundan ve elimizde bir harf değil, bir metin olduğundan böyle bir yapının gelecek harfin ne olduğu konusunda tanımlama işlevinin yanı sıra bir tahminde bulunabileceği öngörüsünde bulunmuştur.

Ancak yapılan araştırmalar ve incelenen projeler doğrultusunda, görüntü tanımlama/sınıflandırma çalışmalarında CNN mimarisinin son derece kullanışlı olduğu çıkarımı yapılmıştır. Bu sebeple yola CNN mimarisi ile devam etme kararı alınmıştır.

4.3.3 Keras

Projenin zaman kısıtı ve konu üzerine yapılan çalışmaların kullandığı teknikler göz önünde bulundurularak, sinir ağları yapısını oluşturmak adına Keras kütüphanesinden faydalanan makta yarar grülmüştür.

Keras, Tensorflow adında açık ve makine öğrenme temalı bir kütüphanenin üzerine oluşturulmuş bir sinir ağı kütüphanesidir. Tensorflow ile çalışabildiği gibi Theano ile de çalışabilmektedir. Kullanımının kolay olması ve hakkında ipuçlarının internette kolayca bulunabilir olması nedeniyle tercih edilmiştir.

4.3.3.1 Hyperparameters

Sinir ağları mimarisi için en optimal kurgu sağlansa bile, bir modelin doğruluğunu en çok etkileyen faktörlerden biri hyperparameters diye adlandırılan değişkenlerdir. Nöronlar arası bağlantıların ağırlıkları, frame içi değerler gibi faktörleri eğitim süreci ilerlerken en optimal ayara adapte edebilirken, hyperparameters tamamen programcının seçtiği ve eğitim sürecinde değişmeyen parametrelerdir. Bu parametreler Learning Rate, Epoch sayısı, Hidden Layers, Hidden Units, Activations Functions, Batch Size diye örneklenirilebilir.

4.3.3.2 Underfitting ve Overfitting

Az önce bahsedilen hyperparameterların yanlış seçilmesinin doğurduğu en istenmeyen sonuçlardan ikisi Underfitting ve Overfitting'dir.

Şüphesiz ki seçtiğimiz mimarinin training boyunca doğruluk oranının yüksek olması hedeflediğimiz bir durumdur. Ancak training süreci iyi ilerlerken validation doğruluğunun düşük kalması ya da tam tersine, doğruluk düzeyinin hiç de iyi olmaması bize overfitting ve underfitting durumlarının sinyalini verecektir. Bu sebeple model seçimi ve hyperparameter seçiminde bu durumdan kaçınmak önceliğimiz olacaktır.

4.3.4 Başarı Değerlendirmesi

Modelimizi oluşturuktan sonra bu modeli test etmek için veri oluşturulacaktır ve her test verisinin kesin ve tek bir doğru sonucu olduğu için bu testlerin sonucunu gözlemlerek kolay olacaktır.

Ancak modelin doğru model olup olmadığını anlamak için baktamız gereken parametreler bunlardan daha detaylı olacaktır.

4.3.4.1 Accuracy

Bakmamız gereken ilk değerlerden biri accuracy olacaktır. Eğitim sırasında modelimizin validation verisini yüksek accuracy ile işlemesi veya test işleminden sonra yüksek accuracy değerleri vermesi istediğimiz bir durumdur.

$$Accuracy = \frac{\text{Number Of Correct Predictions}}{\text{The Number Of Predictions}} \quad (4.1)$$

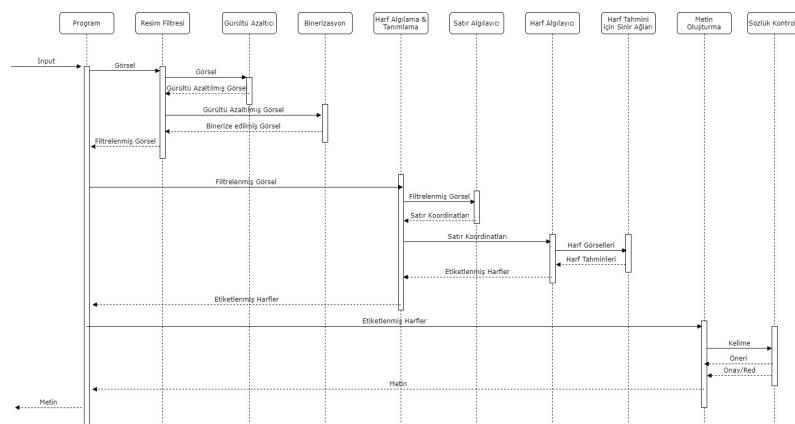
Ancak önceki alt başlıkta bahsedildiği üzere, accuracy değerinin doğru yorumlanması çok önemlidir. Overfitting durumunda bir modelin sadece training verileri için doğru sonuç oluşturacağı, bunun dışındaki kullanımlarda güvenilir olmayacağı, generalization yeteneğinin zayıf olduğunu unutmamak gereklidir.

4.3.4.2 Loss

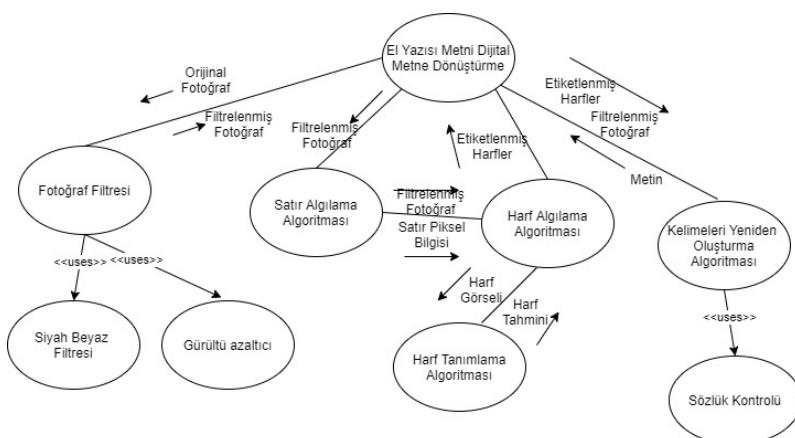
Dikkat edilmesi gereken başka bir parametre de loss çıktılarıdır. Loss basitçe, modelimizin bir input için yaptığı tahmin hatalarıdır. Loss değerinin her koşulda accuracy ile ters orantıda değişmeyeceği unutulmamalıdır.

Loss değeri Loss Functions denilen fonksiyonlar ile hesaplanır. Training loss değeri aynı zamanda modelimizin kendini bir sonraki adımda doğru yapıya adapte etmesini sağlar.

5 SİSTEM TASARIMI



Şekil 5.1 Sistem için hazırlanan Sequence Diyagramı



Şekil 5.2 Sistemin yapısını açıklamak için oluşturulan şema

5.1 Yazılım Tasarımı

Sistem her ne kadar birçok işlev sahip birimler bütünü olsa da bu birimlerin uyumlu olmasını sağlamak adına hiyerarşik yapıda kalmaları amaçlanmıştır. Bu birimlerin işlevleri hakkında temel bilgiler bu bölümde, teknik detayları Uygulama bölümünde verilmiştir. Sistemin genel çalışma prensibi hakkında fikir sahibi olmak adına Şekil-5.1 ve Şekil-5.2 incelenebilir.

5.1.1 Görüntü Üzerine İşlemler

Verilen görüntü üzerindeki metnin ayırt edilebilmesi adına, görselin arka planı ile görseldeki metnin renk farkının bariz olması beklenmektedir. Her ne kadar kullanıcı bu farklı sağlasa da yine de program içerisinde bu farkı maksimum seviyeye çıkarmak adına binarizasyon işlemi gereklidir. Bunun yanında kağıtta olabilecek "gürültü"leri azaltmak adına bir adet de denoising işlemi yapılmalıdır.

Binarizasyon işlemi için numpy array haline getirilmiş görselin piksel değerlerini 0 veya 255 değerlerine atamak istediğimiz işlemi yapacaktır. Gürültü azaltmak için ise cv2'nin denoising fonksiyonlarının kullanımı olduğu görülmüştür.

5.1.1.1 Fotoğraf Filtresi

Çeşitli görseller denenerek hangi filtre değerlerinin daha iyi sonuçlar vererek arka planı soyutlaştırip metni ön plana çıkarttığı sorgulanmıştır.

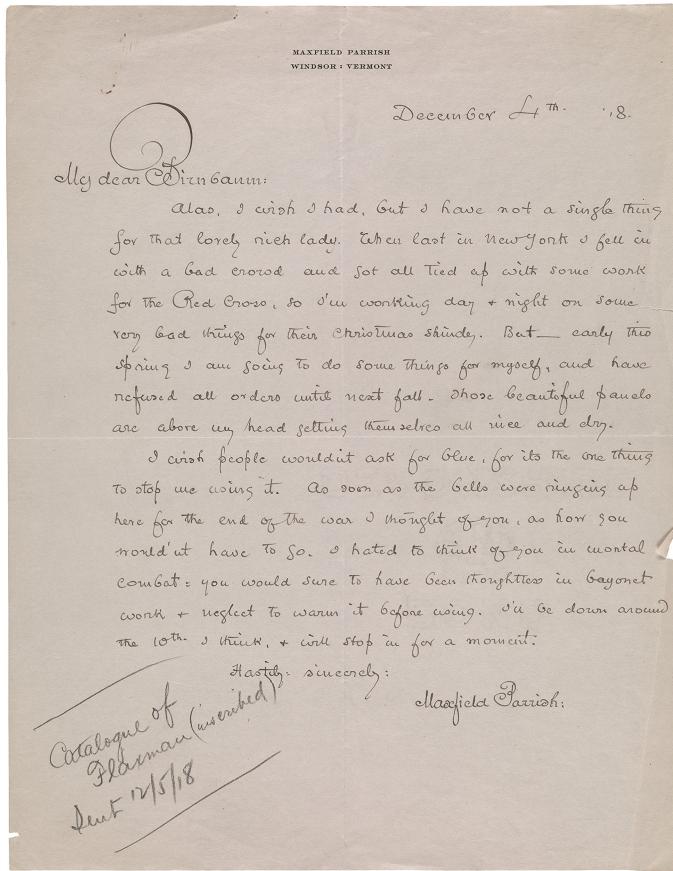
Bu filtre biriminde “Numpy”, “PIL” ve “OpenCV” kütüphaneleri import edilerek fonksiyonlarından faydalانılmıştır.

Gürültü azaltma işlevi için OpenCV'den fastNLMeansDenoising (veya sırasına göre “fastNLMeansDenoisingColored”) kullanılmıştır. Bu fonksiyonun hangi filtre değerine göre hangi sonuçlar verdiği verdiği gözlemlenerek her türlü fotoğrafta en iyi çalışan değerler saptanmıştır. Bu konu hakkındaki denemelerden örnekler Resim-5.4 ve Resim-5.5'de verilmiştir.

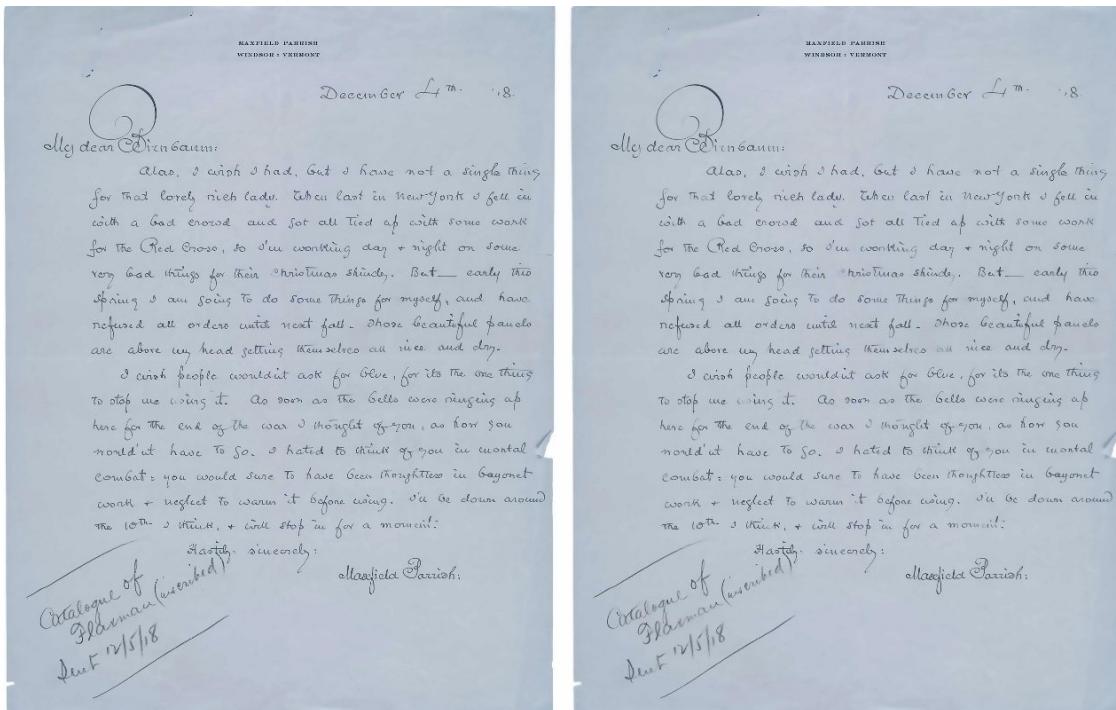
Resim-5.4 ve Resim-5.5'de farklı filtreleme değerlerine göre fonksiyonumuzun verdiği sonuçlar gösterilmiştir. 30 ve 40 değerleri harflerde belirli bozulmalara yol açarken, 10 ve 15 de ne kadar bütün gürültüleri yok edemese de tatmin edici sonuçlar ortaya koymuştur. Bu ikisi arasında seçim yapmak için bu değerlerle filtre edilen daha anlaşılır el yazısına sahip iki fotoğraf “Tesseract”的 metin algılama algoritmasına sokulmuş ve 15 filtresinin belirli kayıplara yol açtığı gözlemlenmiştir. Filtre seçimi konusunda 10 değerinde karar kılınmıştır.

Filtremizin ikinci işlevi olan siyah beyaz yapma işlevi için herhangi bir kütüphane kullanılmayacak, basitçe değeri 128'den az ya da çok olan pikseller 0 ve 255 olarak ayrılarak bu işlem gerçekleştirilecektir. Bu aşamada filtre hakkındaki kalan tek sorun ise hangi filtre sıralamasının daha iyi sonuç verdiğidir.

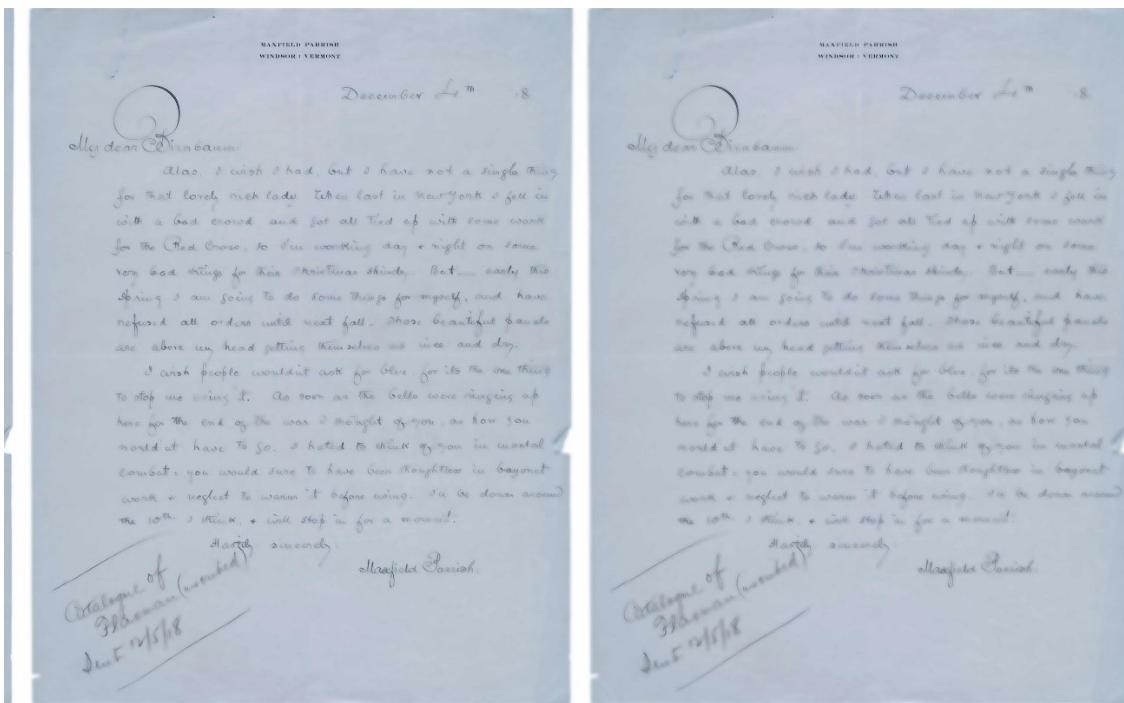
Resim-5.6 ve Resim-5.7'ye yapılan ilk karşılaştırmada harfler üzerinde gözle görülür



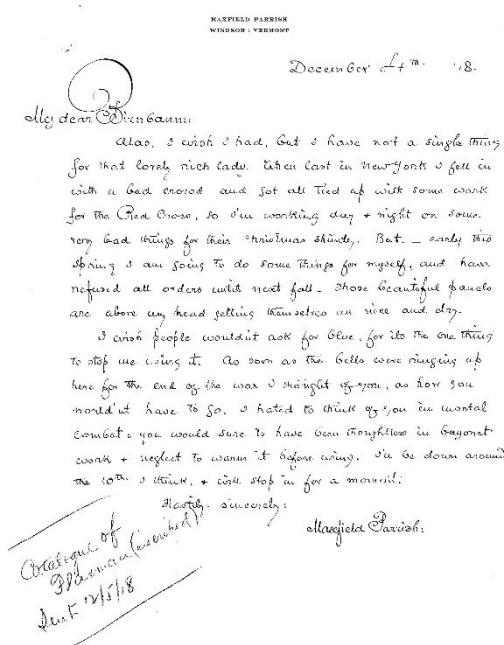
Şekil 5.3 Kullanılacak görselin orijinal hali



Şekil 5.4 Filtre değerleri: 10 ve 11

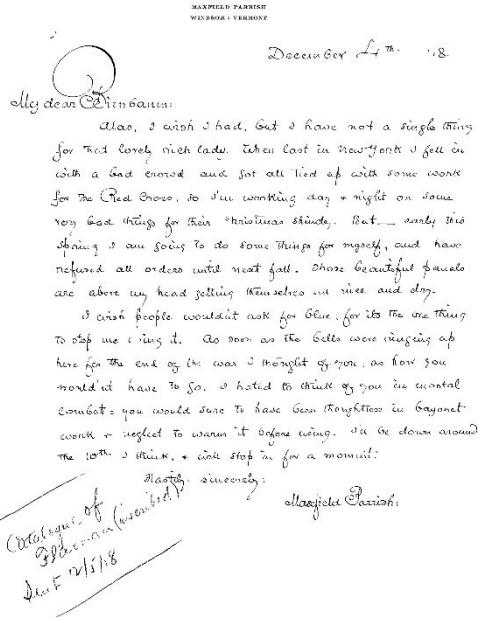


Şekil 5.5 Filtre değerleri: 30 ve 40



Şekil 5.6 Önce binerizasyon, daha sonra gürültü azaltma uygulanmış görsel.

bir kayıp fark edilmezken, Resim-5.7'de gürültünün azaldığı görülmüştür. Bu iki şekilde işlenen resimleri "Tesseract"ın metin algılama fonksiyonuna soktuğumuzda ikinci görsellerin hep daha iyi sonuç verdiği gözlemlenmiştir. Bu sebeple çalışmanın devamında önce 10 filtre değeriyle gürültü azaltma, daha sonra siyah beyaz yapma metodunun kullanılmasında karar kılınmıştır.



Şekil 5.7 Önce gürültü azaltma, sonra binerizasyon uygulanmış görsel.

5.1.2 Harf Ayırma

Görsel içerisindeki harfler ayırma için ise herhangi bir kütüphaneden faydalananmadan, yalnızca numpy array haline getirilmiş görselin, matris tarar gibi taranması yaklaşımı uygulanacaktır. Bu yaklaşım göre görsel üst kenardan başlayarak aşağıya doğru taranacak, bu işlem esnasında satır başları ve sonlarının koordinatları tespit edilecek ve bu satırlar sol kenardan itibaren taranarak harfler ayrılacaktır.

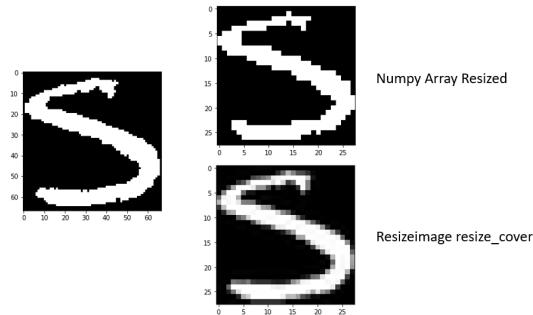
Bu yöntemle elde ettiğimiz harfler maalesef istediğimiz şekil ve boyutlarda olmayacağı için onları ortalayıp kare bir resme oturtmak gerekecek. Ayrıca boyutları da seçilen EMNIST veri setinin verilerinin boyutu olan 28x28'e küçültülecektir.

Harf ayırma süreci boyunca, çıktı metninin formatını ayarlayabilmek adına bir satırda harfler arası gezilen boş sütunların sayısının kaydı tutulacaktır.

5.1.2.1 Harf Görseli Yeniden Boyutlandırma Metotları

Kare haline getirilmiş ancak boyutları model tarafından işlenmeye henüz müsait olmayan bir harf görselini istenen 28x28 boyutlarına indirmek için kullanılabilecek iki farklı metot bulunmuş, bu metotların sonuçları incelenmiştir. Bu sonuçlar Şekil-5.8'da gösterilmiştir.

Bu adımda çıkacak sonucun Şekil-5.16 ile benzer durumda olması hedeflediğimiz durumdur. Harfin küçültme işlemi sonrasında detay kaybının daha az olduğu görüldüğü için resizeimage kütüphanesi tercih edilecektir.



Şekil 5.8 Aynı görselin iki farklı metotla küçültülmesi.

5.1.3 Model Çıktısında Hata Düzeltme ve Kelime Tespiti

Tahmin edilen harflerin bir string formatında bir araya getirilmesinin ardından bu harflerin taranması esnasında tutulmuş boş piksel kayıtlarının yorumlanarak kelimelerin ayırtılması gerekmektedir. Ayrıca, kullanacağımız CNN modelinde yüzde yüz doğruluğa ulaşamayacağımız için hataların olmasını bekliyor olacağız. Ancak bu hataları elimizdeki context bilgisi ile azaltmaya çalışmak mümkün.

5.1.3.1 Kelimeleri Ayırt Etme Yöntemi

Daha önceki adımlarda harflerin arasındaki boşluk miktarının kaydı tutulduğu için, bu adımda bir satırdaki harfler arası boş piksel sayılarının yorumlanmasıyla kelimelerin doğru bir şekilde ayırt edilmesi hedeflenmektedir.

Bu işlemi yerine getirmek için uygun yöntem arayışi boyunca yapılan denemelerde, bir satırdaki harfler arası en büyük boşluk ve en küçük boşlukların farkının doğru bir eşik değer sağladığı görülmüştür. Buna göre bu eşik değerinin altındaki boşluklar harfler arası, eşik değerinin üstündeki boşluklar ise kelimeler arası boşluk olarak değerlendirilecektir.

5.1.3.2 Rakam-Harf Benzerliği Problemi

Her ne kadar bazı sayılar ve rakamlar başka herhangi bir şekilde benzerlik göstermese de, bazı sayılar ve şekiller insan gözü için bile karışıklık yaratabilir. Bu nedenle başarılı bir okuma sürecinde bile "O" harfinin "0" ile, "S" harfinin "5" ile karıştırılması muhtemel olacaktır. Bu hatalarla başa çıkmak bir noktaya kadar mümkün değildir. Programın bu kısmında, elde edilen metindeki kelimeler içerisinde harfler arasında kalmış rakamların benzerleri ile değiştirilmesi, alakasız bir rakam bulunma durumunda ise sonraki modüle kolaylık olması amacıyla rastgele bir karakter ile değiştirilmesi planlanmaktadır.

5.1.3.3 Sözlük Kontrolü

Önceki adımların sonunda, elimizdeki kelimelerin doğruluğunu test etmek için takip edeceğimiz son bir yöntem de sözlük kontrolü olacaktır.

İngilizce sözlük için "pyenchant" çok geniş bir kelime haznesi sunuyor. Bir kelimenin İngilizce olup olmadığına tespitini yapabilen ve eğer değilse sözcük önerisinde bulunabilen fonksiyonlara sahip olan bu kütüphanenin programa dahil edilmesinin faydalı olacağı düşünülmüştür. Ancak sözlükte olmayan kelimelerin değiştirilmesi işlemi için kelimelerin uzunluklarının kıyaslanması, kelimelerin harf pozisyonlarının mukayese edilmesi ve en uygun kelimenin seçilmesi gerekecektir.

```
>>> import enchant
>>> d = enchant.Dict("en_US")
>>> d.check("Hello")
True
>>> d.check("HeLo")
False
>>> d.suggest("HeLo")
['He lo', 'He-lo', 'Hello', 'Helot', 'Help', 'Halo', 'Hell', 'Held', 'Helm', 'Hero', "He'll"]
>>>
```

Şekil 5.9 pyenchant.github.io/pyenchant linkinde bulunan Enchant kullanım snippeti

5.1.4 Yapay Sinir Ağı Mimarisi

Yapay sinir ağı mimarisini Keras ile oluşturulmuş bir CNN modelinin kullanılacağı önceki bölümde belirtildi. İyi sonuçlar veren, veri seti ile uyumlu ve stabil çalışacak bir model oluşturma işlemi için yapılmış olan projelerde kullanılan mimarilerin incelenmesi ve karşılaştırılması, sonuçta bulunan mimarinin geliştirilmesi yolu tercih edilecektir.

Bu amaçla başta bu tip projeler için çok zengin bir bilgi kaynağı olan "kaggle.com" olmak üzere internetteki çalışmalar incelenmiş ve göze çarpan iki adet model için inceleme yapılmıştır.

Bahsedilen modellerden ilki, Yufeng Guo'nun Kaggle üzerinden paylaştığı ve EMNIST üzerinde çalıştığı "emnist GPU keras to TF" isimli projesinden, ikincisi ise Chris Deotte'in yine Kaggle üzerinden paylaştığı ve MNIST üzerinde çalıştığı dijit tanımlama projesine aittir. Bu projelerin incelenmek üzere seçilmesinin farklı sebepleri vardır. Yufeng Guo'nun projesinin EMNIST veri seti için son derece eğitici olması ve bu veri seti ile güzel bir "loss" grafiği yakalamış olması nedeniyle hem bu alanda ilk defa bir proje ortaya koyacak şahsım için faydalı olacağı düşünülmüş, hem de projenin geliştirildiği takdirde iyi sonuçlar verebileceği öngörülmüştür. Chris Deotte'nin projesinin doğrultusu ve kullandığı veri setinin farklımasına rağmen, yakalamış olduğu 0.99757 doğruluk oranının etkileyici olması ve dijit tanımlama projesinin

karakter tanımlama projesine uyarlanabilir olmasının düşünülmüştür. Ancak Chris Deotte'nin bu modeli 15 defa kullanarak, eğitilen bu 15 modelin sonuçları ile bu başarı oranını yakaladığı belirtilmelidir. İki model Şekil-5.10'de gösterilmiştir.

```

model = Sequential()
model.add(Conv2D(32, kernel_size = 3, activation='relu', input_shape = (img_size,img_size,1)))
model.add(BatchNormalization())
model.add(Conv2D(32, kernel_size = 3, activation='relu'))
model.add(BatchNormalization())
model.add(Conv2D(32, kernel_size = 5, strides=2, padding='same', activation='relu'))
model.add(BatchNormalization())
model.add(Dropout(0.4))

model.add(Conv2D(64, kernel_size = 3, activation='relu'))
model.add(BatchNormalization())
model.add(Conv2D(64, kernel_size = 3, activation='relu'))
model.add(BatchNormalization())
model.add(Conv2D(64, kernel_size = 5, strides=2, padding='same', activation='relu'))
model.add(BatchNormalization())
model.add(Dropout(0.4))

model.add(Conv2D(128, kernel_size = 4, activation='relu'))
model.add(BatchNormalization())
model.add(Flatten())
model.add(Dropout(0.4))
model.add(Dense(units=num_classes, activation='softmax'))

model.compile(loss='categorical_crossentropy', optimizer='Adam', metrics=['accuracy'])
model.summary()

```

Chris Deotte, 25 Million Images! [0.99757] MNIST
Kaggle.com

Yufeng Guo, emnist GPU keras to TF
Kaggle.com

Şekil 5.10 İncelenen CNN Modelleri

İki modelin sonuçları da tatmin edici olsa da, projenin zaman kısıtı ve bir modelde en optimal sonucu almanın zaman alabileceği düşünüldüğü için, iki modelden birinin seçilmesi ve bu modelin iyileştirilmesi yolu seçilmiştir. Yufeng Guo'nun modeli tipki onun tavsiye ettiği gibi 500 adımlı 5 ve 10 arasındaki epoch değerleri ile denenmiştir. Bunun yanında Chris Deotte'nin training yöntemi çok uzun zaman aldığı için, onun modeli de fikir vermesi açısından 5 ile 18 değerleri arasında epoch sayıları ile ölçülmüştür. Yufeng Guo'nun modeli validation loss parametresi için çok daha iyi sonuçlar verse de, accuracy olarak diğer modelden yaklaşık 0.13 aşağıda kalmıştır. Bu sebeple zaten benzer bir projede çok yüksek doğruluk oranı ile başarılı sonuçlar vermiş olması, hem de deneyler sonucunda tatmin edici accuracy değerleri vermesi sebebiyle tercih ikinci modelden yana kullanılmıştır.

Bu modelin eğitimi üzerine denenen yollar ve alınan sonuçlar bu bölümde verilecektir ancak öncelikle bu modelin çalışma mantığını kavramak için kullanılan Keras katmanlarının ne işe yaradığı anlaşılmalıdır. Bu yapılar hakkında temel bilgilendirmeler aşağıda verilmiştir.

5.1.4.1 İki Boyutlu Convolution Layer

Seçilen modelde tam 7 tane bulunan bu katman, adından da anlaşılabileceği üzere CNN modellerinin olmazsa olmaz bir parçasıdır. Bir, iki veya üç boyutlu olabilir. Bu katmanın ne işe yaradığını özetlemek gerekirse, özellik saptama için kullanıldığı söylenebilir. Bunu yapmak için bu katmanlar boyutlarını bizim belirlediğimiz filtreleri kullanır. Bu filtreler, bu proje kapsamında iki boyutlu olacak şekilde kullanılmıştır. Bu filtrelerin her biri, verilen görselde (veya input formatı neyse) farklı özelliklerin kontrolünü yapar. Bu ayrimı da filtrenin içerisindeki değerlerden anlayabiliriz.

Filtrenin içerisindeki değerler o filtrenin hedeflediği özelliği ortaya çıkarmak üzere ayarlanır ve filtre resmi dolaştıkça resmin o bölgesinde aranan o özelliğin olma durumunu kontrol eder. Filtreler resmi dolaştıktan sonra Feature Map'ler üretilir ve bu yapı başka layerlarda kullanılmak üzere aktarılır.

Bufiltrelerin her birinin bir özelliği kontrol etmek için var olduğunu söylemişik, peki bir filtrenin parçaları için en optimal değerlerin ne olduğunu programcı nasıl bilmektedir? Bu yapı için programcının filtre değerleri girmez, CNN yapısının training süreci ilerledikçe, aldığı loss değerlerine göre bufiltreleri şekillenir ve filtre değerleri optimal hallerini alır.

Modelin içerisinde aktivasyon fonksiyonu adlı bir paramtere için ReLu'nun seçildğini görmektesiniz. Relu temel olarak bir rampa fonksiyonudur. Elde edilen sonuçtaki negatif değerleri sıfıra eşitlemeye yarar. ReLu ve diğer aktivasyon fonksiyonları, modelimize bir değerin kullanışlı veya kullanıssız olması ayrimini yapmaya yardımcı olur. ReLu'nun matematiksel gösterimi aşağıda verilmiştir.

$$y = \max(0, x) \quad (5.1)$$

5.1.4.2 Batch Normalization

Modelimizin training süresini kısaltmak, gürültü yaratarak overfitting'den kaçınmak için Batch Normalization kullanmak faydalı olacaktır. Batch Normalization, adından da anlaşılabileceği üzere, önceki katmanın aktivasyon fonksiyonundan çıkan verileri normalize etmek amacıyla kullanılır. Böylece katmanlar arası aktarılan değerler arasından büyük uçurumlar olmaz, değerler normalize edilmiştir ve belli bir aralık arasına sıkıştırılmıştır.

5.1.4.3 Dropout

Dropout, bir ağın doğruluğunu elde edilen bilgilerden bazılarını yok ederek iyileştiren enteresan bir yaklaşımdır. Bu teknigue göre, katmanlar arası veri aktaran nöronlar, programcı tarafından verilen bir oranda deaktive edilir. Bunu yaparak aslında modelin training seti ezberlemesinin önüne geçilmiş olur. Böylece model, bir verinin farklı hallerini de tanıma yetisine erişmiş olur.

5.1.4.4 Dense

Dense Layer'da yapılan işlemi bir oylamaya benzemek mümkündür. Bu katman önceki katmandaki nöronların taşıdığı değerlere göre sınıflandırma yapılan katmandır. Ancak bahsedilen bu oylama her zaman demokratik olmayacağı. Bazı nöronların verdiği

oylar, taşıdığı değerle de bağlantılı olarak, daha büyük önem taşır ve bu katmandan elde edilen sonuç aynı zamanda seçtiğimiz modelin de sonucu olmuştur.

5.1.4.5 Batch Size ve Epoch

Önceki bölümde hyperparameters arasında saydığımız bu parametreler üzerine çokça denemeler yapacağımız için bunları anlamak da çok önemlidir. Sıklıkla karıştırıldığı gözlenen bu parametreler aslında çok farklı anlamlar taşır. Batch size, bir seferde gelecek veri miktarının seçimi iken, epoch bütün training verisinin ağdan kaç defa geçeceğini belirlemesi amacıyla kullanılmaktadır.

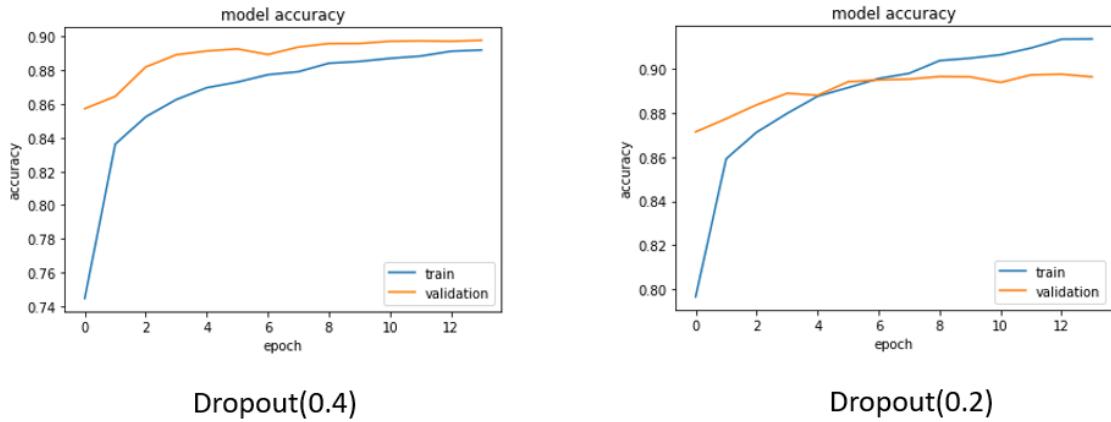
5.1.4.6 Sinir Ağ Modeli Üzerinde Yapılan Denemeler

Her ne kadar modelin veri setimizle halihazırda sunmuş olduğu validation accuracy değeri (değişik denemelerde yaklaşık 0.9) tatmin edici olsa da, farklı bir veri seti için oluşturulmuş ve orijinal projesinde tamamen farklı bir biçimde kullanılan bu modelin farklı training yaklaşımlarına karşı verdiği sonuçlar analiz edilerek, en doğru ve en iyi sonuçlar veren yöntemin arayışına girilmiştir. Model ve training üzerine yapılan her değişiklik için genel yönelik kontrol edilmiştir. Her deneme de elde edilen sonuçlar not alınmış ve en son karar kılacağımız yaklaşım için fikir alınması sağlanmıştır.

A) Farklı Dropout Rate Denemeleri

Projemizde kullandığımız veri setinin, sinir ağ mimarisinin alındığı projede kullanılan MNIST veri setine oranla çok daha geniş bir veri seti olduğu göz önüne alınarak, ilk sorulan soru verilen 0.4'lük Dropout değerinin modelin öğrenmesi için kötü bir etki yaratıp yaratmadığı olmuştur. Modelin Dropout değerinin 0.4'ten 0.2'ye düşürülmesi karşısında verdiği tepkiyi ölçmek adına model iki farklı dropout değeri ile aynı şekilde eğitilmiştir. Bu eğitimlerde amaç genel tavrı gözlemlemek olduğundan, düşük bir epoch sayısı seçilmiş ve batch size default değeri olan 32'de bırakılmıştır. İki eğitim sonucunda alınan accuracy grafikleri Şekil-5.11'de gösterilmiştir.

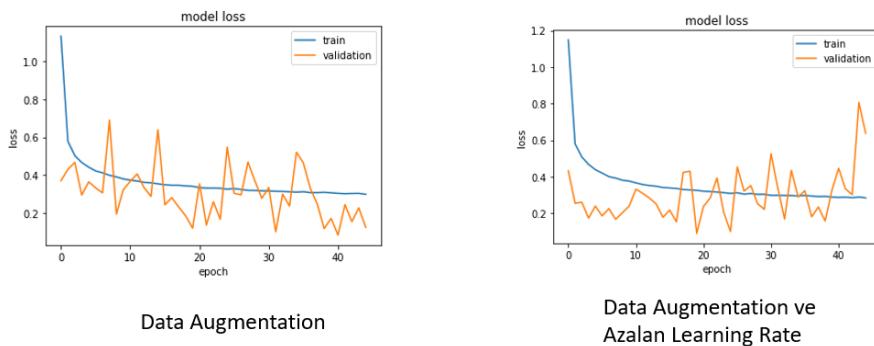
Anlaşılacağı üzere, modelimiz Dropout değerinin değiştirilmesine olumsuz bir sonuç vermiştir. Bu kadar düşük epoch sayısı kullanmamıza karşın, accuracy ve validation accuracy egrilerinin yöneliklerinden de görülebileceği üzere, 0.2'lük bir Dropout Overfitting durumunun önüne geçmekte zorlanmaktadır. Bu deneme sonucunda 0.4'lük Dropout değerinin korunmasında karar kılınmıştır.



Şekil 5.11 Farklı Dropout değerleri için alınan accuracy değerleri

B) Her Adımda Azalan Learning Rate

Orijinal projede 64'lük batch size ve 45 epoch hyperparameter değerleri kullanarak, data augmentation ile training verilerinin zenginleştirildiği ve LearningRateScheduler kullanılarak learning rate'in git gide azaldığı görülmüştür. Bu adımda bu tarz bir training yaklaşımının bu veri seti için de olumlu yansıyıp yansımayacağı merak edilmiştir. Bunun için model önce tipki orijinal projede olduğu gibi eğitilmiş, daha sonra LearningRateScheduler kullanımı olmadan eğitilme durumunda sonuçların ne olacağı kontrol edilmiştir. Bu kontroller esnasında çok benzer bir validation accuracy elde edilmiş ve accuracy grafikleri de neredeyse aynı olduğu görülmüştür. Ancak farklılık yaratan loss grafikleri Şekil-5.12'de gösterilmiştir.

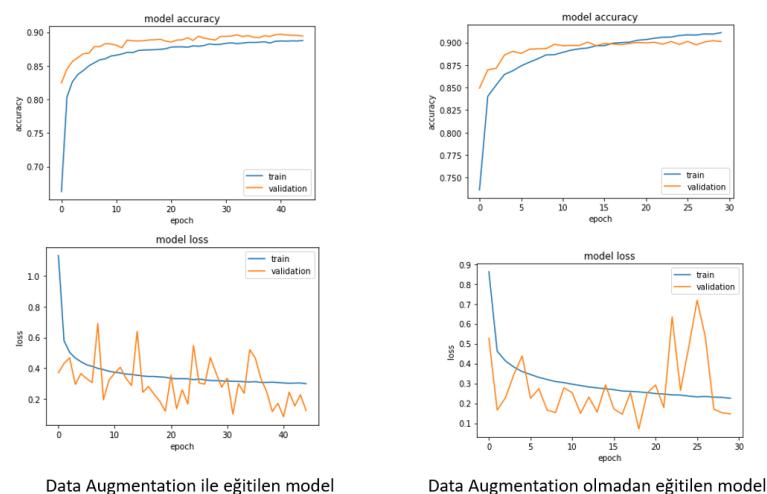


Şekil 5.12 Learning Rate Scheduler kullanımının loss grafiklerinde yarattığı değişim

Gördüğü üzere her iki grafikte de validation loss için dengesiz bir eğri yakalanmıştır. Ancak sadece data augmentation yapılan modelin validation loss değerinin eğrisinin yüksekliğinin git gide küçüldüğü, öte yandan learning rate'in git gide küçülmesi karşısında validation loss'un çok daha kötü bir hale gelmiş olduğu gözlemlenmiştir.

C) Data Augmentation

Data augmentation, elimizdeki training verilerini her adımda belirlenen limitler çerçevesinde değiştirerek, sinir ağı yapısına daha az veriyle daha fazla örnek öğretme amacıyla uygulanan bir yaklaşımdır. Önceki adımda azalan learning rate ve data augmentation ile eğitilen bir modelin, sadece data augmentation olan bir model karşısında başarısız olduğunu görmüştük. Bu adımda, yaklaşık 110.000 veri içeren bir veri setiyle eğitilen bir modelin data augmentation'a gerçekten ihtiyacı olup olmadığı sorusu sorulacaktır. Bu amaçla az önceki adımda kullanılan data augmentation ile eğitilmiş model, aynı batch size'a sahip ama data augmentation olmadan eğitilmiş versiyonu ile karşılaştırılmıştır. Augmentation olmayan modelin zamanla verileri ezberleyip overfitting durumuna gelmesi daha muhtemel olduğu için, augmentation olan metodun epoch sayısı sabit bırakılırken, diğer modelin epoch sayısı (başarılı sonuç alındığı takdirde sonra arttırılarak yeniden denenmek üzere) 30 olarak belirlenmiştir. İki senaryoda modelin oluşturduğu accuracy ve loss grafikleri Şekil-5.13 gibi olmuştur.



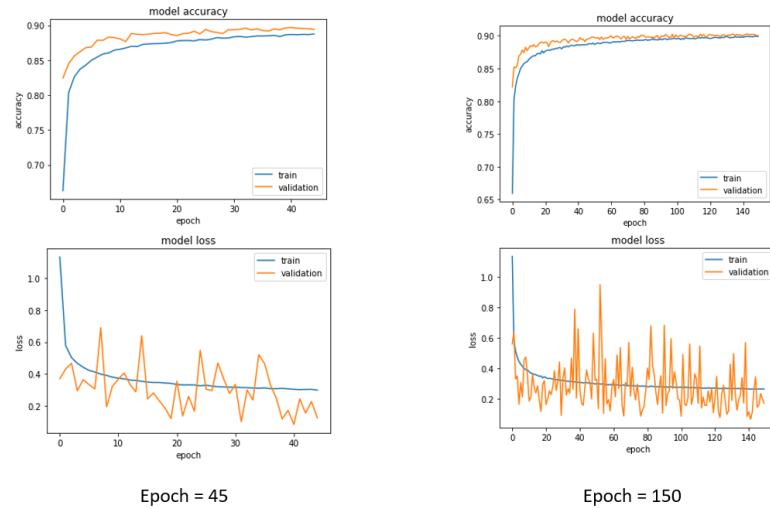
Şekil 5.13 Data Augmentation kullanımının accuracy ve loss grafiklerinde yarattığı değişim

Gördüğü üzere, epoch sayısı az olmasına rağmen augmentation kullanılmayan senaryoda accuracy ve validation accuracy korkulan senaryonun gerçekleştiğini göstermiştir. Bu loss grafiğinden de görülebilmektedir. Bu adımda belirtilen şartlarda data augmentation yaklaşımının olumlu sonuç verdiği anlaşılmıştır.

D) Epoch Sayısının Arttırılması

Bundan önceki örneklerde her ne kadar git gide loss değerinin altına düşen bir validation loss değeri yakalasak da, bir türlü tutarlı bir şekilde düşen validation

loss çizimi ile karşılaşamamıştık. Bu adımda, bunun sebebinin yetersiz sayıda epoch yapılması mıdır sorusu sorulacaktır. Bunun için aynı model, aynı training parametreleriyle, sadece epoch sayısı 150 ile değiştirilerek tam 26 saatlik bir eğitime sokulmuştur. Oluşacak yeni validation loss eğrisinde herhangi bir zamanda önceki elde ettiğimiz gafiye göre daha iyi bir sonuç alınıp alınamayacağı kontrol edilecektir. Bu deneme sonucunda oluşan grafikler Şekil-5.14'ta verilmiştir.



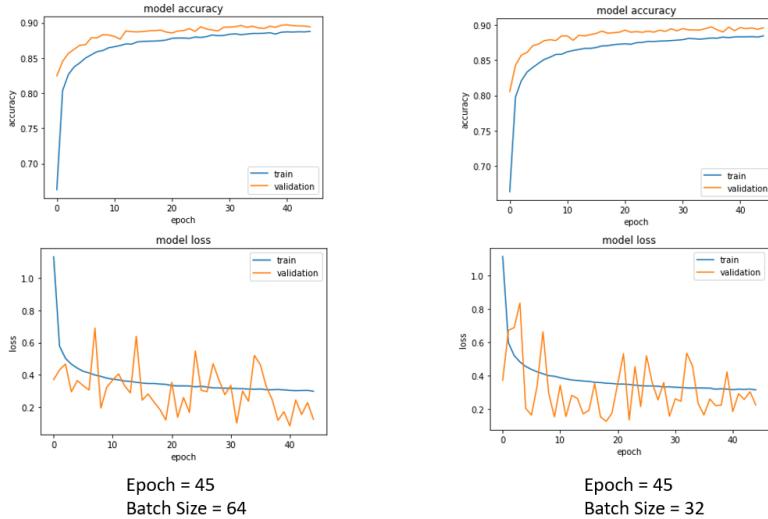
Şekil 5.14 Farklı epoch sayılarıyla yapılan training sonucunda oluşan accuracy ve loss grafikleri.

Gördüğü üzere 150 tane epoch sonucunda grafiğin hiçbir yerinde tutarlı bir şekilde azalan bir validation loss görülmemiştir. Aksine tutarsızlığın arttığı görülmüştür. Validation accuracy değerleri de 150 epoch sonunda 0.9 ve diğer senaryoda 0.8940 olarak bulunmuştur. Oluşturulan örnek verilerle (Bu örnekler ileriki bölümlerde son modelin denenmesinde gösterilecektir) yapılan denemelerde ise 45 epoch sayısıyla eğittimiz modelin daha doğru sonuçlar verdiği gözlemlenmiştir.

E) Batch Size'in Etkisi

Yapılan bu son denemedede ise, elimizdeki 64 batch size ile eğitilmiş modelin, batch size'ı başka bir popüler batch size değeri olan 32'ye düşürülürse, sonuçlarımızın nasıl değişeceği sorgulanmaktadır. Bu deneme için tamamiyle aynı training yaklaşımı kullanılmış, sadece bir modelin eğitiminde 64, diğerinde 32 batch size kullanılmıştır. Oluşan grafikler Şekil-5.15'de gösterilmiştir.

Bu iki farklı training sonucunda, 0.8940 ve 0.8959 olmak üzere yakın validation accuracy değerlerinin ortaya çıktığını gözlemlenmiştir. İki modelde de (yne) inişli



Şekil 5.15 Farklı batch size değerleriyle yapılan training sonucunda oluşan accuracy ve loss grafikleri.

çıkışlı ancak git gide belli bir zaman aralığındaki ortalama baz alındığında azalma eğilimi gösteren validation loss değerleri elde edilmiştir.

5.1.4.7 Kullanılacak Model ve Training Yönteminin Seçilmesi

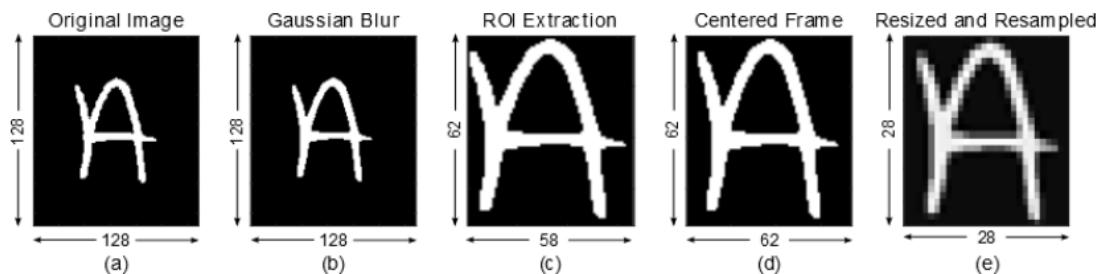
Yapılan denemeler boyunca validation accuracy değerini 0.89 bandının aşağısına düşürmemekle beraber başta elde edilen 0.9 değerinin üstüne çıkabilmek; bununla beraber her deneme karşımıza çıkan inişli çıkışlı validation loss grafiğini daha kararlı bir şekilde azalan bir yapıya dönüştürmek hedeflenmiştir.

Gelinen son noktada, gösterilen (ve kayda değer olmadığı için burada gösterilmeyen) bütün denemeler sonucunda, 0.8940 validation accuracy değerine sahip, inişli çıkışlı olsa da epoch sayısı arttıkça belli aralıktaki ortalaması baz alındığında azalma eğilimi gösteren, hemen hemen her deneme adımında kıyaslandığı modele göre daha iyi değerler sunduğu gözlenen, orijinal projedeki sinir ağının yapısının 64 batch size, 45 epoch, 0.2 split oranında training/validation verisi ve data augmentation teknigi ile eğitilmiş versiyonunda karar kılınmıştır. Bu karar sadece bu grafiklerdeki sonuçlara göre verilmemiş, aynı zamanda oluşturulan gerçek deneme verilerine karşı çıktığı sonuçlar da baz alınmıştır. Bu sonuçlar ilerleyen bölümlerde gösterilecektir.

5.2 Veri Seti Kullanımı

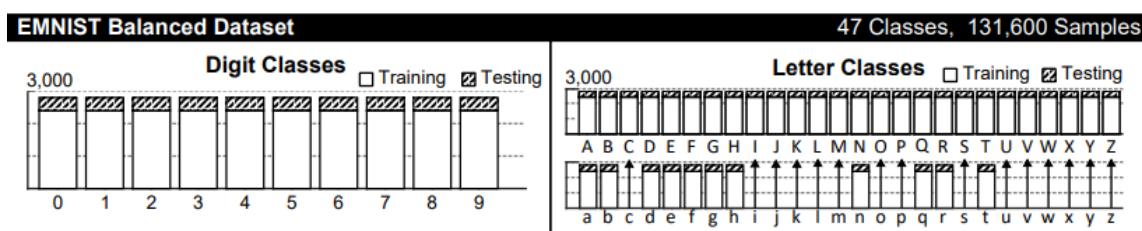
Önceki bölümde projenin veri seti olarak kullanılmak üzere EMNIST adlı veri seti hakkında kısa bir tanıtım yapılmıştı ve bu veri setinin içerisinde sınıflar olduğundan söz edilmiştir. EMNIST veri setinin seçilmesinin ardından motivasyonu anlamak ve bu veri setinin hangi biçimde kullanılacağına karar vermek için önce biraz bu veri setini daha iyi anlamak gerekiyor.

EMNIST veri seti tipki belirtildiği üzere NIST veri setindeki görsellerin değiştirilmesi



Şekil 5.16 EMNIST verilerinin oluşturulma şekli[5]

ve yeni verilerle zenginleştirilmesi yoluyla oluşturulmuştur. Bu projenin kapsamına benzer doğrultuda yapılmış diğer projelerin genelinde tercih edilen ve standart haline gelmiş bir veri setidir. Farklı kullanımlar için farklı sınıflara bölünmüştür. Bu sınıflar arasından Balanced veri seti ise en "uygulanabilir" sınıf olarak nitelendirilmektedir.[5] Balanced sınıfı 47 farklı sınıfta veri bulundurur ve her türden aynı miktarda örnek bulundurur. Bunun yanında Balanced Training verilerinin bir kısmı validation verileri için ayrılmıştır. Bu validation verilerinin içerisindeki örnek dağılımı da dengelidir ve toplam veri sayısı testing veri sayısı ile aynıdır. Bu koşullar göz

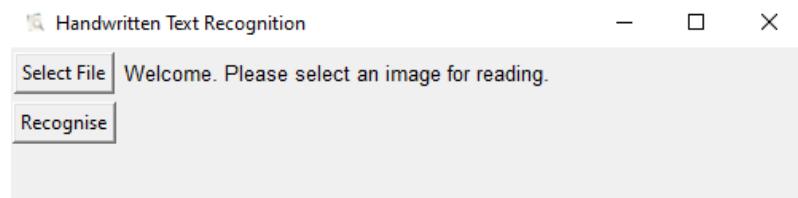


Şekil 5.17 Balanced sınıfı içerisindeki veri örneği dağılımı[5]

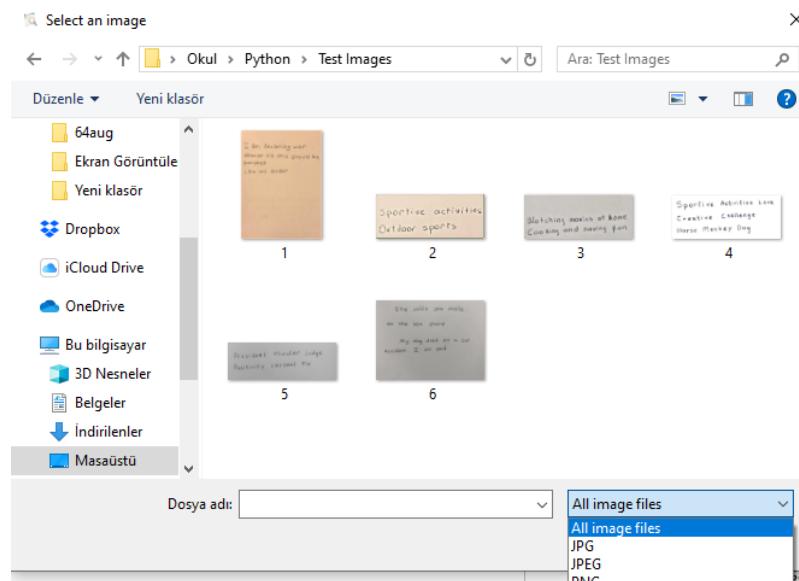
önüne alındığında, EMNIST veri setinin training verilerinin bir kısmının validation olarak kullanılması, kalanının training için kullanılması ve testing verilerinin test için kullanılması kararlaştırılmıştır.

5.3 Girdi ve Çıktı Tasarımı

Programın kullanıcıdan talep ettiği tek girdi bir adet görseldir. Bu görselin formatı png, jpg, jpeg, tif veya tiff seçeneklerinden biri olabilir. Daha sonra kullanıcının seçtiği bu görsel işlendikten sonra bulunan metin kullanıcıya sunulacaktır. Kullanıcının yapacakları ve programın sunacağı verilerin çok karmaşık olmamasından dolayı basit bir tkinter yapısının bunun için yeterli olacağı düşünülmüştür. Bu tkinter içerisinde dosya seçme opsyonu seçildiğinde bir adet dosya yolu seçme penceresi açılacak ve bu pencere kullanıcının sadece resim dosyası seçmesine izin verecektir. Daha sonra okunan metin programla aynı konumda bir .txt dosyasına kaydedilecektir.



Şekil 5.18 Kullanıcı için tasarlanmış tkinter yapısı.



Şekil 5.19 Kullanıcının görsel seçmesi için sağlanan yol.

6 UYGULAMA

6.1 Fotoğraf Filtresi

Sistemin ilk adımı olan fotoğraf滤resi birimi için yapılan tasarım ve denemeler sonucunda varılan son滤re yapısına göre alınan sonuç Şekil-6.1'de verilmiştir.



Şekil 6.1 Örnek bir fotoğraf ve滤relemeden sonraki hali.

6.2 Satır ve Harf Algılama

Bu bölümde tasarım kısmında anlatılan harf algılama metodu takip edilmiştir. Bu kısmında bulunan harfin görseli düzenlenerek sinir ağları yapısına gönderilerek tahmin sürecine sokulacaktır. Bu bölümde takip edilen yolun işlevselliği, her harfi başka renk tonuna boyama yöntemi ile Şekil-6.2'de gösterilmiştir.

Resim-6.2'de de görüldüğü üzere algoritma harfleri çakışmayan metinlerin fotoğraflarını başarılı bir şekilde harflerine ayırmaktadır.

Metinlerde elde edilen performanslara bir örnek de Resim-6.3'de görülmektedir. Harflerin bitişik olmaması durumunda ayrı etme işlemi başarılıken,滤renin ayıramadığı gürültüler de karakter olarak işaretlenmektedir. Bu sorunla metini yeniden oluşturma adımımda başa çıkılacaktır. Bu ana kadar anlatılan bölümler, elimizdeki görseldeki her bir karakteri ayırt ederek kırpmaya yeteneğine sahiptir.

You think
I'm crazy?

You should see me with my best friend

Şekil 6.2 Harf algılama algoritmasının görsellerdeki performansına bir örnek.

BİR DİKDÖRTGENLER PRİZMASI

Şekil 6.3 Harf algılama algoritmasının görsellerdeki performansına bir örnek.

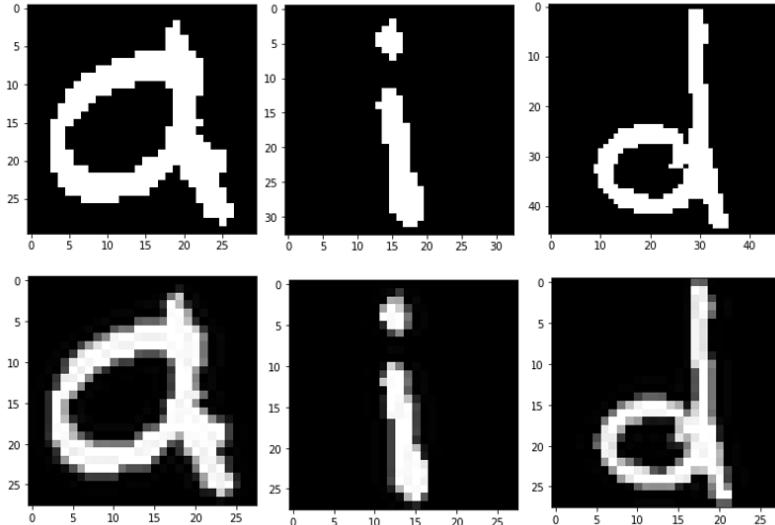
6.3 Model Tarafından Tahmin Yürüttülecek Görsellerin Input Formatına Uyarlanması

Daha önceki bölümden de anlaşılmak üzere, elde ettiğimiz harf görsellerinin şekli, boyutu ve renkleri henüz sınır ağı modeline aktarılmaya hazır değildir.

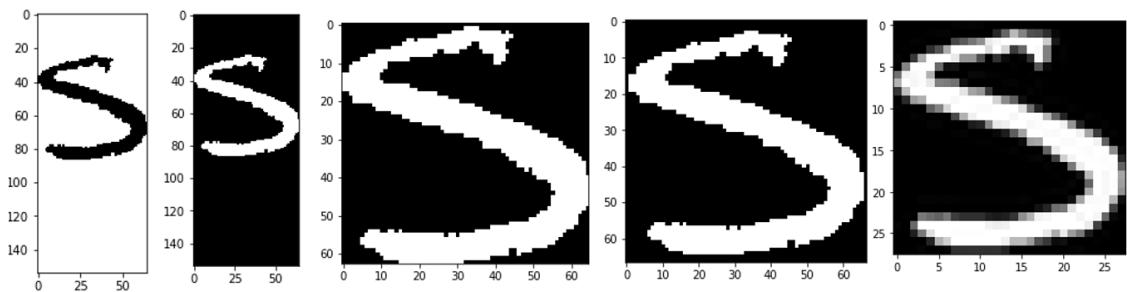
İlk yapılacak işlem, elimizdeki beyaz arka plandan alınmış koyu renkli harf çiziminin tipki EMNIST verilerinde olduğu gibi siyah arka plan üzerine beyaz yazı olarak değiştirilmesidir. Bu işlemin ardından harf görselini (yne tipki EMNIST verileri gibi) resimde ortalamak gerekecektir. Bunun için öncelikle harfin etrafındaki boşlukları kırpmak, daha sonra bunu uygun boyutlu siyah arka plan bir görselin ortasına yapıştırmak uygun ve işlevsel bir yol olmuştur.

Bu adımlardan sonra elde edilen görselin boyutunun 28x28 formatına getirilmesi gerekmektedir. Bunu yapmak için kullanılabilen yaklaşımın analizi "Sistem Tasarımı" bölümünde gösterilmiştir. Seçilen resize metodunun verdiği diğer sonuçlar Şekil-6.4'de gösterilmiştir.

Buna göre bir harfin görselde tespit edilmesinden tahmin yürütümü için modele aktarılmasına kadar geçirdiği değişim Şekil-6.5'de verilmiştir.



Şekil 6.4 Resize edilmiş bazı harf görselleri



Şekil 6.5 Algılanan harfin bir görsel halinde kırıplamasından sonra istenen input formuna dönüştürme aşamaları

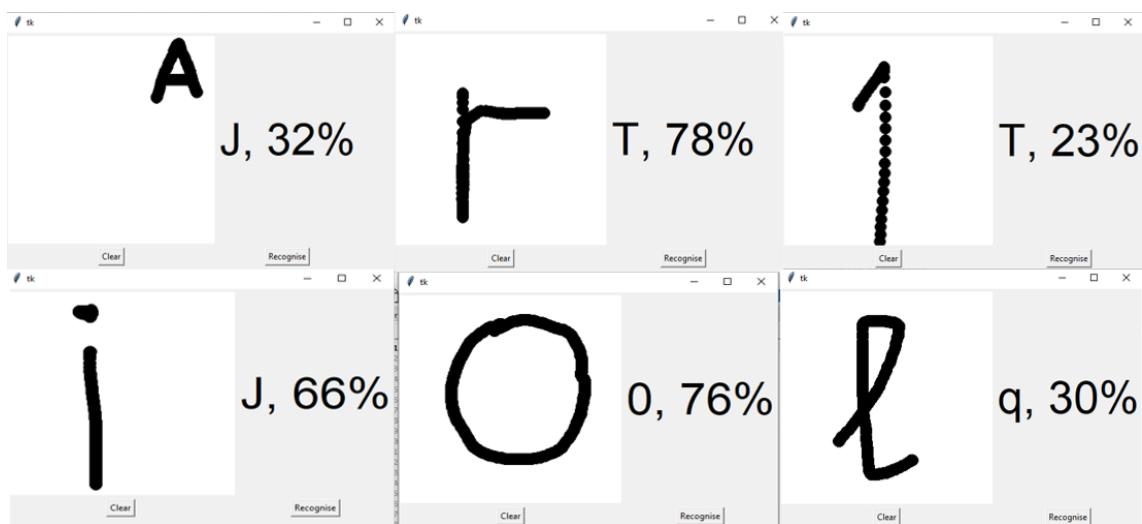
6.4 Harf Tanımlama ve Metin Oluşturma

Bu bölümde, seçilen sinir ağı mimarisi ve denemelerle en uygun olduğu görülen training yaklaşımının, tek bir harf/rakam çizimine karşı verdiği sonuçlar gösterilecektir. Bunu yapmak için basit bir tkinter'da beyaz bir zemin üzerine çizimler yapılmış, yapılan çizimler tipki tasarlanan gerçek programdaki harf görsellerinin işlenmesi gibi işlenmiş ve model tarafından sınıflandırılmıştır. Eldeki örnek metin fotoğraflarına karşı alınan sonuçlar bir sonraki bölümde verilecektir. Bu bölümde yapılan bu testin sonra da denenebilmesi için, kullanılan bu basit tkinter yapısının kodları dosyaya dahil edilecektir. Yapılan denemelerin sonuçları Şekil-6.6 ve Şekil-6.7'de gösterilmiştir.

Gösterilen bu sonuçlar yapılan denemeler sonucu elde edilen pozitif sonuçlar ve negatif sonuçların kayda değer olanlarını içermektedir. Yapılan bu denemede gerçek hayatı programın karşılaşabileceği çizimler taklit edilmeye çalışılmış ve yaklaşık



Şekil 6.6 Tek karakter çizimleriyle modelden alınan bazı pozitif sonuçlar



Şekil 6.7 Tek karakter çizimleriyle modelden alınan negatif sonuçlar

her 15 denemede 1 defa hatalı sonuç elde edildiği görülmüştür. Ancak bu projede asıl amaç metin görselinden harf tanımlaması yapmaktadır. Bu doğrultuda yapılan denemeler "Deneysel Sonuçlar" bölümünde verilecektir.

6.5 Çıktı Metninde Hata Azaltma Çalışmaları

Tasarım bölümünde anlatıldığı üzere, CNN modelimizde tamamen hatasız bir model elde edemeyeceğimiz için, model tahminlerinde hataların olması son derece muhtemeldir. Anlatılan satır içi boşluklar hesabı, rakam-harf karışıklığının giderilmesi ve son metindeki kelimeler için pyenchant üzerinden sözlük kontrolü aşamaları Şekil-6.8'de gösterilmiştir.

```
- Original: P$11$r$25$e$27$$27$I$28$d$14$e$20$A$27$t$122$M
$20$I$20$n$11$J$19$S$13$t$23$e$18$r$76$J$27$U$20$d$16$g$17$e
P$9$0$11$Z$13$I$19$t$15$J$20$V$18$I$25$t$26$Y$122$C$9$a$21$r
$21$a$16$M$14$e$20$L$94$P$15$I$22$e

Modified: PreSIdeAt MInJSter JUdge
POZItJVItY CaraMeL PIE

Modified: presideat minjster judge
positivity caramel pie

['president', 'preside at', 'preside-at', 'preside',
'presidia', 'desiderate', 'desiderata']
['minster', 'minister', 'Mister', 'minter']
['positivity']

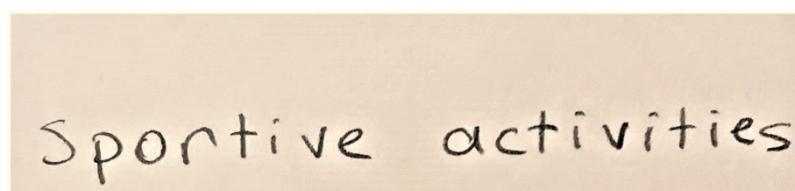
Text: president minister judge
positivity caramel pie
```

Şekil 6.8 Son çıktıda hata azaltma çalışmaları

7

DENEYSEL SONUÇLAR

Bu bölümde, programın farklı kişilerce oluşturulan el yazısı metin fotoğraflarıyla yaptığı okuma işlemleri sonucunda verdiği çıktılar gösterilecektir. Daha sonra bu çıktılardaki (varsı) hataların sebepleri sorgulanacaktır.



Text: sportive activities

Şekil 7.1 Örnek input ile programın yaptığı okuma işleminin sonucu.

Şekil-7.1'de iki kelimeli bir metin için programın yaptığı okuma işlemi gösterilmiştir.

I am declaring war Whoever did this should be punished Law and order	Text: i am declaring war whoever did this should be punk sh e d law end order
---	--

Şekil 7.2 Örnek input ile programın yaptığı okuma işleminin sonucu.

Şekil-7.2'de görüldüğü üzere, çok kelimeli bir metinde bir satır içerisinde sadece tek bir kelime olduğunda, program o kelimedeki harflerin arasındaki boşlukları analiz edip bu kelimeyi parçalama eğilimindedir. Böylece belki de doğru bir şekilde okumuş olduğu bir kelimeyi parçalara ayırıp, ayrılmış parçalarda doğruluk kontrolü yapmakadır ve kelimedede büyük bozulmalar meydana gelebilmektedir.

Watching movies at home
Cooking and having fun

Text: witching movies at home
cooking end having fun

Şekil 7.3 Örnek input ile programın yaptığı okuma işleminin sonucu.

Şekil-7.3'deki okumada, 8 kelimelik bir metinde iki harfin son metinde değişmiş olduğu görülmektedir.

Sportive Activities Love
Creative Challenge
Horse Monkey Dog

Text: sportive actjvjtjes love
creative challenge
horse monkey dog

Şekil 7.4 Örnek input ile programın yaptığı okuma işleminin sonucu.

Şekil-7.4'de görüldüğü üzere, üç satırlık bir metindeki ikinci kelimenin "i" harfleri program tarafından "j" ile karıştırılmıştır. Bu hata yazılı yazan kişinin yazı stiline bağlı olarak ortaya çıkmaktadır. Bu iki harfin karışması hatasının bu örnekte başka kelimelerde de ortaya çıkmış olma ihtimali vardır ancak böyle bir durum varsa bile programın çıktısını kontrol eden mekanizma bu hataları düzeltmeyi başarmıştır.

President Minister Judge
Positivity Caramel Pie

Text: president minister judge
positivity caramel pie

Şekil 7.5 Örnek input ile programın yaptığı okuma işleminin sonucu.

Şekil-7.5'deki örnekte kelimelerin doğru okunduğu, "positivity" kelimesinin orijinal metindeki yazım hatasının programın yaptığı sözlük kontrolü ile düzeltildiği görülmektedir.

She sells sea shells
on the sea shore

Text: she sells sea shells
pa the sea shore

Şekil 7.6 Örnek input ile programın yaptığı okuma işleminin sonucu.

Şekil-7.6'deki okumada programın iki harflik bir kelime dışında doğru okuma yaptığı görülmektedir.

He should stop
It makes me sad

Text: he should sip
ht makes me sad

Şekil 7.7 Örnek input ile programın yaptığı okuma işleminin sonucu.

Şekil-7.7'deki örnekte inputtaki "stop" kelimesinin ortadaki iki harfi yatay eksende çakışmış olduğu için, program bu iki harfi ayırmamış ve bu şekilde tahmin yapmaya çalışmıştır. Ayrıca bu el yazısı stilindeki "I" harfinin program tarafından tanınmadığı görülmektedir.

I am in love
Text: j am kn love
that ks true
That is true

Şekil 7.8 Örnek input ile programın yaptığı okuma işleminin sonucu.

Şekil-7.8'deki örnek sol elimle yazarak oluşturduğum bir metindir. Görüldüğü üzere harf hatları düzgün çizilmemiş olmasına karşın program "i" harfleri dışında başarılı bir okuma yapmıştır.

I go everywhere

Text: i go everywhere

Şekil 7.9 Örnek input ile programın yaptığı okuma işleminin sonucu.

Şekil-7.9'deki kısa metinde doğru bir okuma yapılmıştır.

8 PERFORMANS ANALİZİ

Her ne kadar proje kapsamında offline tabanlı bir program tasarlanmış olsa da, bu tip çalışmaların online tabanlı versiyonları da mevcuttur ve kullanıcı sayısı tek seferde çok fazla olabilmektedir. Ayrıca hem bahsedilen senaryoda, hem de bu proje kapsamında, kullanıcının bir metni kendi eliyle bilgisayara yazmayıp bu tip programlara başvurmasının ardından en büyük motivasyonun bu programların sağladığı hız ve kolaylık olduğu unutulmamalıdır. Bu sebeple her ne kadar bu geliştirme sürecinde en büyük odak noktası doğruluk olmuş olsa da, programın performanısının da çok önemli olduğu unutulmamalıdır.

Bu bölümde kullanıcının bir görsel seçip, "recognize" butonuna basması ile okunan metnin .txt olarak kaydedilmesi arasında geçen süre incelenecaktır. Input olarak önceki bölümde kullanılan görseller kullanılmıştır.

Tablo 8.1 Input Özelliklerine Göre Okuma Süresi

Görsel Boyutu	Görseldeki Çizim Sayısı	Geçen Zaman (ns)	Geçen Zaman (s)
1200x1600	56	7509525300	7.5095253
1578x371	18	2413629300	2.4136293
1578x654	39	4262053200	4.2620532
1600x628	52	4447947900	4.4479479
1600x548	42	3900986300	3.9009863
1011x306	30	1819317300	1.8193173
1024x206	24	1276476600	1.2764766
507x233	19	1056634400	1.0566344
997x165	13	1239851100	1.2398511

Tablodan hızı etkileyen en büyük faktörün input olarak verilen görselin piksel sayısı olduğu anlaşılmaktadır. Buna göre programın en çok zaman alan kısmının fotoğraf piksellerini tarayarak harfleri ayıran kısım olduğu çıkarımı yapılmaktadır.

9 SONUÇ

Bu proje kapsamında, kağıda el yazısıyla yazılmış bir metnin bilgisayar yazısına çevrilmesi işlevinde bir program oluşturulması amaçlanmıştı. Bu amaç doğrultusunda bu ve bunun gibi amaçlarla oluşturulmuş projeler araştırılmış, yaklaşımıları incelenmiş ve dersler çıkarılmıştır. Sıfırdan başlanmış olan bu yapıyı kurarken oluşan büyük veya küçük problemlerin çözümü adına yapılan her araştırmada bu alana dair yeni bakış açıları kazanılmıştır.

Yazılım kısmında öncelikle kullanıcıdan alınan görselin okuma işlemi için uygun hale getirilmesi sağlanmıştır. Daha sonra bu görseldeki çizimlerin birer birer ayrılması ve istenen formata sokulması yöntemi planlanmıştır. Bu esnada projenin amacına uygun ve en iyi sonuçları vaadeden sinir ağları modelinin ve bu sinir ağlarını en iyi şekilde eğitecek veri setinin arayışına girişilmiştir. Bu arayışın her adımından dersler çıkarılmış ve en doğru bulunan kombinasyon seçilmiştir. Daha sonra fotoğraftan alınan verilerin sinir ağlarına doğru bir şekilde aktarılması, sonra da sonuçların birleştirilerek bir metin oluşturulması sağlanmıştır. Çalışan bu sistemde daha da iyi sonuçlar almak adına her bir modül geliştirilmiş, bu geliştirme çalışmasının her adımı kayıt altında tutulmuştur. Projenin bu noktasında elde edilen ürün projenin en başında konulan hedefe ulaşmış ve istenen görevi yerine getirebilir bir program haline gelmiştir.

Elimizdeki program (şartları sağlayan ve metin içeren) bir görseli başarıyla çizimlerine ayırbılır ve bu çizimleri yaklaşık 90% başarı oranına sahip bir sinir ağı mimarisile sınıflandırılabilir durumdadır. Programımız ayrıca kullanıcıya çıktı vermeden önce, kendi içinde doğruluk kontrolü yapabilecek şekilde tasarlanmıştır. Yapılan denemelerde, bu kontrol mekanızması sayesinde programın yapmış olduğu hataların yarısından fazlasının önüne geçilebilediği gözlemlenmiştir.

Programımız her ne kadar gereklilikleri karşılıyor durumda olsa da, geliştirilebilir yönleri olduğu şüphesizdir. Örnek olarak (performans kaybı yaratmamak şartıyla) daha efektif bir harf ayırma algoritması; dijit, karakter ve boşluk algılama işlevlerinin yanı sıra noktalama işaretlerini de okuyabilmek adına noktalama işaretleri de içeren bir veri setiyle yeniden oluşturulacak bir training yaklaşımı bunlara örnek verilebilir.

Ayrıca "Sistem Tasarımı" bölümünde detaylı bir şekilde anlatılan "daha tutarlı bir validation loss eğrisi" arayışında çok daha fazla mesai harcanıp elimizdekine göre daha tercih edilebilir bir durum elde edilebilir. Bunların yanı sıra, programda kullanılan sinir ağı mimarisi ve bu sinir ağının training parametreleri gibi öğeler için, en iyisi diye bir limit olmaksızın, daha iyiye ulaşmak amacıyla her zaman daha çok deneme ve geliştirme yapılabilir.

Bu proje üzerine çalışırken yaşadıklarım sayesinde, bu tip projelerde yol katedebilmek için öncelikle projenin modüllerini iyi tanımlanması ve her modülde kullanılacak metodun arayışında rastlanan önerilerin mantıklarının iyi kavranması gerektiği dersini çıkardım. Python gibi programcılar arasında son derece popüler olan bir dil veya yapay sinir ağları tasarıımı gibi gündemde olan bir çalışma alanı söz konusu olduğunda, her zaman düşündüğünüzden daha iyisinin önerilebilir olduğunu öğrendim. Bu sebeple bir projede en iyi sonuca ulaşmanın her zaman yeni fikirlere açık olmaktan ve araştırma yapmaktan sıkılmamaktan geçtiğini gördüm.

Referanslar

- [1] M. He, S. Zhang, H. Mao, and L. Jin, “Recognition confidence analysis of handwritten chinese character with cnn,” in *2015 13th International Conference on Document Analysis and Recognition (ICDAR)*, IEEE, 2015, pp. 61–65.
- [2] K. Horii, *Facial image processing method and facial image processing apparatus*, US Patent 5,850,463, Dec. 1998.
- [3] A. Sander and M. Wolfgang, “The rise of robotics,” *Bcg perspectives*, 2014.
- [4] C. Clifford, “Elon musk: Robots will take your jobs, government will have to pay your wage,” *CNBC. com*, vol. 4, 2016.
- [5] G. Cohen, S. Afshar, J. Tapson, and A. Van Schaik, “Emnist: Extending mnist to handwritten letters,” in *2017 International Joint Conference on Neural Networks (IJCNN)*, IEEE, 2017, pp. 2921–2926.

Özgeçmiş

BİRİNCİ ÜYE

İsim-Soyisim: Ahmet Onur AKMAN
Doğum Tarihi ve Yeri: 12.11.1997, Kayseri
E-mail: ahmetonurakman@gmail.com
Telefon: 05072805753
Staj Tecrübeleri: Bilişim Destek Hizmetleri

Proje Sistem Bilgileri

Sistem ve Yazılım: Windows İşletim Sistemi, Anaconda Spyder IDE, Python
Gerekli RAM: 2 GB
Gerekli Disk: 6 GB