**KOCAELİ ÜNİVERSİTESİ**

**MÜHENDİSLİK FAKÜLTESİ**

**BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ**

**NÖRON AĞLARINA GİRİŞ PROJE RAPORU**

**YAPAY SİNİR AĞLARI İLE İŞARET DİLİ ÇEVİRİCİ**

**MEHMET OĞUZ AKTAŞ**

**KOCAELİ 2020**

**İÇİNDEKİLER**

1. GİRİŞ..................................................... 3

2. TEMEL BİLGİLER 4

2.1. Yapay Nöron Ağları 4

2.2. Nöronların Matematiksel Gösterimi 7

2. KULLANILAN YÖNTEMLER 10

3. VERİ SETİ ............................................................................................................. 10

3.1. Veri Setinin Hazırlanması 10

3.2. Veri Setinin Ön İşlenmesi 11

4. KONVOLÜSYONEL NÖRON AĞLARI 11

4.1. Genel Bilgiler 11

4.2. Evrişim Katmanı ve Aktivasyon Fonksiyonu 12

4.3. Havuzlama Katmanı 12

4.4. Tam Bağlantı Katmanı 13

5. PROJE ADIMLARI 13

5.1. CNN Mimarisinin Oluşturulması 13

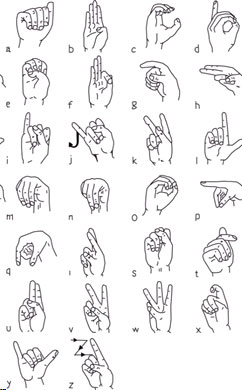
5.2. Eğitim ve Test Adımları 15

6. SONUÇLAR 16

KAYNAKLAR 18

**1. GİRİŞ**

İşaret dili işitme veya konuşma engellilerin kendi aralarında veya başkaları ile iletişim kurarken el hareketlerini, yüz mimiklerini ve beden dillerini kullanarak oluşturdukları görsel bir dildir. Kendine özel grameri ve cümle yapısı vardır. İşaret dili işitme engelli olmayanlar ile işitme engelli olanlar arasında kurulmuş bir köprüdür. El işaretleri ve yüz hareketlerinden oluşan birçok ifadeyi kullanarak iletişim kurulur. Birçok çeşidi bulunmakla birlikte bu çalışmada Amerikan işaret dili kullanılmıştır. Amerikan İşaret Dili, sağır ve dilsiz insanların kullandığı temel iletişim aracıdır. Aynı zamanda işitme zorluğu çeken insanlar tarafından da kullanılmaktadır. Bu dil, en az konuşulan diller kadar zengindir ve el işaretlerinin yanı sıra yüz hareketleri ve bedensel duruşlar da içerir.



Şekil 1.1. Amerikan işaret dili

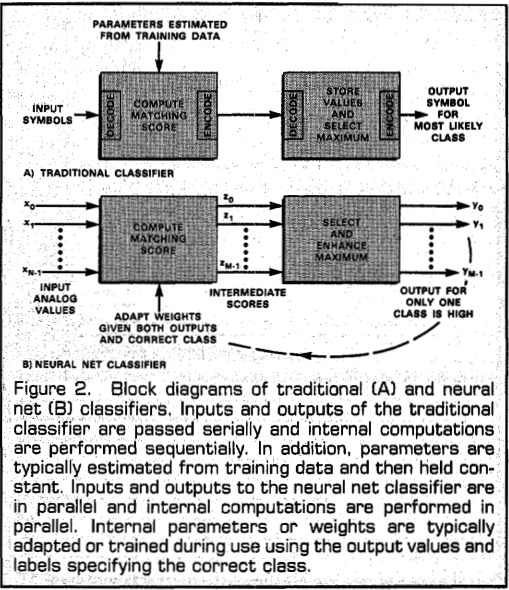
**2. TEMEL BİLGİLER**

**2.1. Yapay Nöron Ağları**

İşaret dilini tanıyıp konuşulan dillere çevirme bilgisayarla görü uygulamalarında son yıllarda birçok ilerleme sağlanmıştır. Bu teknoloji genellikle görüntü üzerindeki farklılıklara göre çıktılar üreten karmaşık çok katmanlı yapay sinir ağı modellerine dayanır. İşaret dilini normal konuşulan dillere çevirmenin ilk adımı olarak, cümleler yerine tek tek harfleri çevirerek problemi basitleştirebiliriz. Yapay sinir ağları bu problemin çözümü için iyi sonuçlar alınan bir yöntemdir.

Yapay sinir ağı modelleri ses ve görüntü tanıma alanlarında insana yakın bir performansa ulaşma umuduyla uzun yıllar boyunca incelenmiştir. Bu modeller paralel olarak çalışan ve biyolojik sinir ağlarını anımsatan desenler halinde düzenlenmiş birçok doğrusal olmayan hesaplama elemanından oluşur [1]. Hesaplamalı elemanlar veya düğümler performansı artırmak için kullanım sırasında tipik olarak uyarlanan ağırlıklar yoluyla bağlanır. Yeni ağ topolojileri ve algoritmalar, analog VLSI uygulama teknikleri ve yüksek performanslı ses ve görüntü tanıma için büyük miktarda paralel biçimde veri işlemenin mümkün olması nedeniyle yapay sinir ağları alanında son zamanlarda bir canlanma olmuştur. Bu makalede patern sınıflandırması için kullanılabilecek altı önemli sinir ağı modeli incelenerek yapay sinir ağları alanına bir giriş yapılmıştır. Bu ağlar, nöral ağ bileşenlerini ve tasarım prensiplerini gösteren oldukça paralel yapı taşlarıdır ve daha karmaşık sistemler oluşturmak için kullanılabilir. Bu ağları tarif etmenin yanı sıra mevcut bazı sınıflandırma ve kümeleme algoritmalarının basit nöron benzeri bileşenler kullanılarak nasıl gerçekleştirilebileceğini araştırmaya da büyük önem verilmektedir. Tek katmanlı ağlar, Gauss maksimum olasılık sınıflandırıcıları ve gürültü nedeniyle bozulan ikili modeller için optimum minimum hata sınıflandırıcıları tarafından gerekli algoritmaları uygulayabilir. Daha genel olarak, herhangi bir sınıflandırma algoritması için gerekli olan karar bölgeleri, üç katmanlı ileri besleme ağları tarafından basit bir şekilde üretilebilir. Yapay nöron ağları bağlantıcı sistemler, paralel dağılımlı işlem modeli, nöromorfik sistemler gibi birçok farklı şekilde isimlendirilebilir. Tüm yapay nöron ağları modelleri yoğun bağlantılı hesaplama elemanlarını kullanarak iyi performans almak için kullanılır. Yapay nöron ağları insandaki biyolojik sinir sisteminden esinlenmiştir. von Neumann bilgisayarında olduğu gibi program talimatlarını sırasıyla yerine getirmek yerine nöron ağ modelleri içinde değişken ağırlıkları bulunduran birbirine bağlı birçok paralel ağ yapısı ile farklı hipotezleri simultane biçimde yerine getirebilir [1]. Hesaplanabilir elemanlar (düğümler) lineer değildir ve genellikle analogdur. Modern dijital devrelere göre yavaş kalabilir. Nöron ağları ağ topolojisi, düğüm karakteristiği, eğitim ve öğrenme kuralları ile tanımlanır. Eğitim ve öğrenme kuralları bazı başlangıç ağırlığı kümeleri içerir ve ağırlıkların model performansını yükseltmek için nasıl değişmesi gerektiğini belirler. Tasarım prosedürleri ve eğitim kuralları güncel araştırma konularıdır. Nöron ağlarının en önemli yararı paralel yüksek hesaplanabilirlik sağlamasıdır.

Şekil 2.1’de geleneksel sınıflandırıcılar ile nöron ağları sınıflandırıcıları gösterilmiştir. İki sınıflandırıcıda da statik giriş paternini içeren N tane girdi elemanı ile M tane sınıf temsil edilir. Bir ses tanıyıcıda girişler, bir seferde örneklenen bir filtre bankası spektral analizöründen çıkış zarfı değerleri olabilir ve sınıflar farklı sesli harfleri temsil edebilir. Bir resim sınıflandırıcısında, girişler bir resim için her pikselin gri tonlama seviyesi olabilir ve sınıflar farklı nesneleri temsil edebilir. Şekil 2.1'in tepesindeki geleneksel sınıflandırıcı iki aşama içerir. Birincisi her sınıf için eşleşen puanları, ikincisi ise azami puanı olan sınıfı seçer [1].



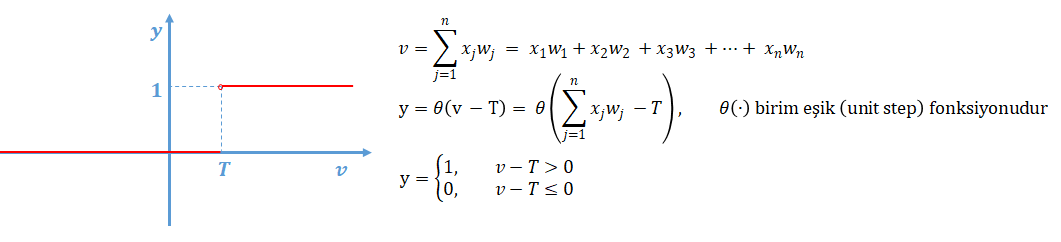
Yapay sinir ağları beyindeki biyolojik sinir ağlarının yapısını, öğrenme, hatırlama ve genelleme kabiliyetlerini taklit eder. Öğrenme işlemi örnekler kullanılarak gerçekleştirilir. Öğrenme sonrasında yapay sinir ağı örüntü tanıma ve sınıflandırma (ses, yazı, yüz, şekil, nesne gibi), kümeleme ve gruplama, optimizasyon, tahmin, kontrol, karar, vb. işleri için kullanılabilmektedir. Yapay sinir ağları, basit ve birbirlerine paralel ve dağıtık olarak bağlı ve her biri kendi belleğine sahip çok sayıda yapay nöron işlemcilerden oluşmaktadır. Yapay sinir ağları zaman zaman sinirsel hesaplama (neural computation), ağ hesaplama (network computation), bağlantılı ağlar (connectionist networks), paralel dağıtılmış ağlar (parallel distributed networks) ve nöromorfik sistemler (neuromorfic systems) ifadeleriyle de anılmaktadır [2].

Yapay sinir ağlarına herhangi bir örnek giriş verisinin tanımlanabilmesi ve bunun daha sonra kullanılabilmesi için verinin ağda nasıl temsil edildiğini, nerede saklandığını ve nasıl geri alındığının bilinmesi gerekmektedir. Klasik bilgisayarlarda bilgi 1 ve 0 serileri ile temsil edilirken sinir ağlarında matematiksel işlev ile temsil edilir. Sinir hücreleri (işlem elemanları) arasındaki bağlantının ağırlıkları bu işlevin değişkenleri olarak görev yaparlar. Burada ağırlıklar saklanan bilginin ne olduğuna karar verir fakat kendilerinin bir anlamı yoktur.

Yapay sinir ağlarındaki bilgi, ağ içindeki bağlantılarla ve bu bağlantılara ait ağırlıklar yoluyla dağıtılmaktadır. Klasik bilgisayar bilgiyi belleğinde belirli bir yerde saklar, yapay sinir ağları ise bilgiyi tüm ağ boyunca dağıtır. Bu durum dağıtılmış bellek olarak adlandırılır. Yapay sinir ağlarının en büyük avantajı öğrenme kabiliyeti olması ve farklı öğrenme algoritmalarının kullanılabilmesidir [2].

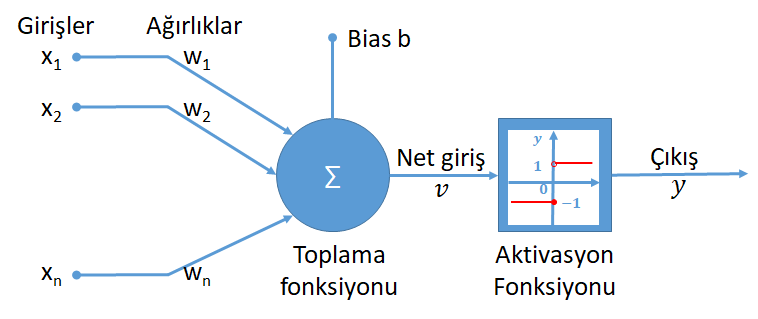
**2.2. Nöronların Matematiksel Modeli**

Nöron ağlarının matematiksel modellemesi ilk olarak McCulloch ve Pitts tarafından gerçekleştirilen McCulloch-Pitts nöronu veya Threshold Logic Unit olarak adlandırılan nöron ağları modeline dayanmaktadır. TLU nöron modelinde, n tane giriş değeri ile n tane ağırlık değeri ve bir adet eşik değeri ile bir adet çıkış değeri yer almaktadır.



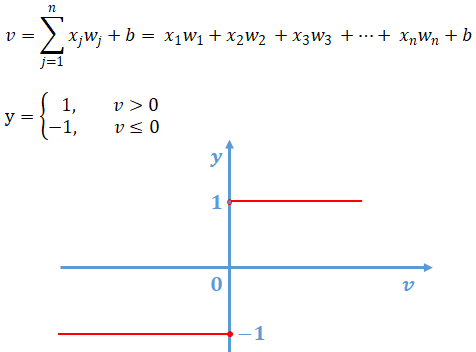
Şekil 2.2.1. TLU matematiksel formülleri

Daha sonra 1959’da Widrow ve Hoff tarafından Adaline (Adaptive Linear Neuron) modeli geliştirilmiştir. Adaline ünitesi yapı olarak tek katmanlı doğrusal sınıflandırıcı olan Perceptron ünitesine benzer fakat öğrenme kuralı farklıdır. Adaline modelinde giriş değerleri ağırlıklar ile çarpıldıktan sonra bias değerleri ile toplanıyor. Elde edilen toplam 0’dan büyükse çıkış değeri 1, 0’dan küçük veya eşitse çıkış değeri olarak -1 atanıyor.



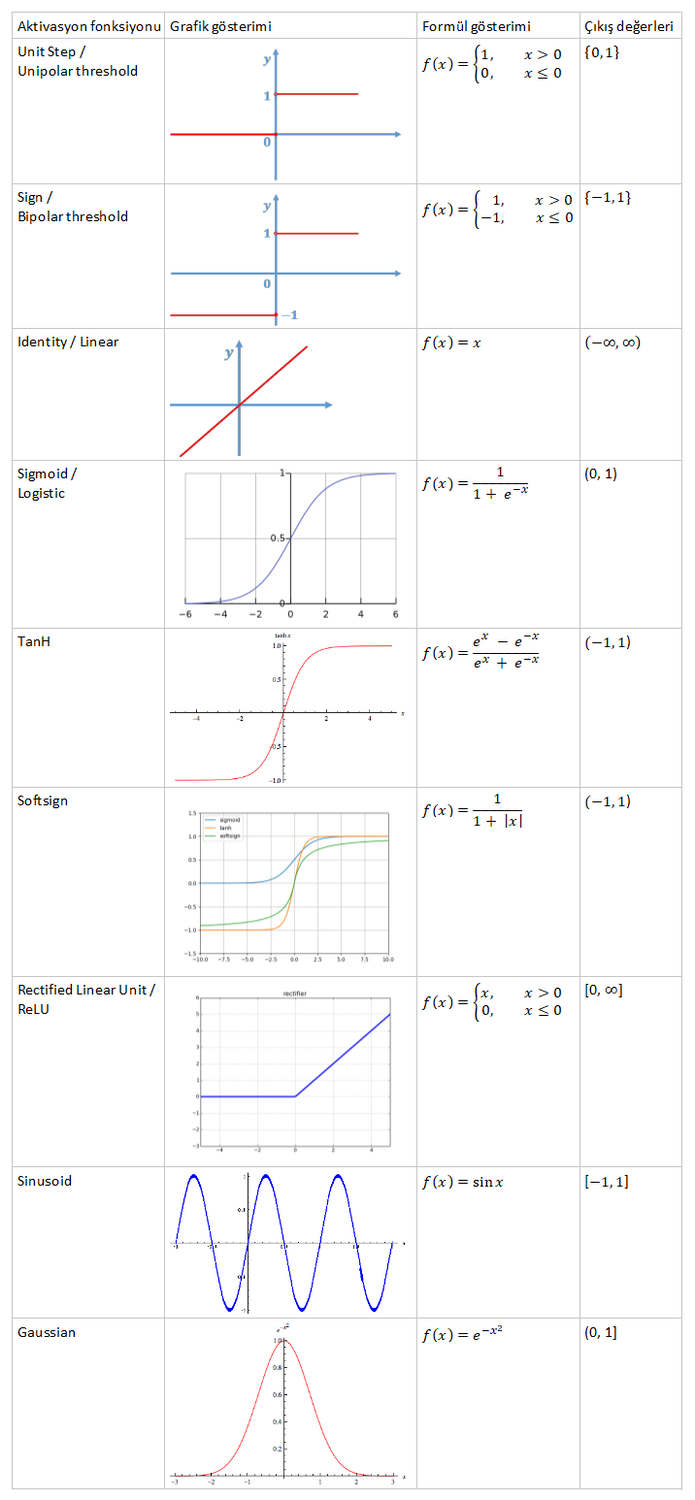
Şekil 2.2.2. Adaline modeli

Adaline nöron ağları modelinin matematiksel formülleri aşağıdaki şekilde gösterilmiştir.



Şekil 2.2.3. Adaline matematiksel formülleri

Bias değeri iki sınıfı birbirinden ayıran doğruyu (karar sınırını) başlangıç noktasından uzaklaştırmaya yarar. Günümüzdeki modellerde toplama ve aktivasyon (etkinlik) fonksiyonları genelleştirildi. Toplama fonksiyonu bir yapay nörona gelen net girdiyi hesaplar. Uygulanacak toplama fonksiyonu yapay sinir ağı modeline göre belirlenmektedir. Bir yapay sinir ağında bütün nöronların toplama fonksiyonu aynı olmayabilir. Aktivasyon fonksiyonu ise nörona gelen net girdiyi işleyerek nöronun bu girdiye karşılık üreteceği çıktıyı belirler. Toplama fonksiyonunda olduğu gibi yapay sinir ağındaki tüm nöronlar aynı aktivasyon fonksiyonunu kullanmayabilir [2].



**3. VERİ SETİ**

**3.1. Veri Setinin Hazırlanması**

Bu projede işaret dili karakterlerini temsil etmek için 29 sınıf bulunduran ASL Alphabet veri seti kullanılmıştır. Bu 29 sınıfın 26 tanesi A’dan Z’ye harflerden oluşuyor. Diğer sınıflar ise space (boşluk), delete (silme) ve nothing’dir. Bu sınıflar özellikle gerçek zamanlı sınıflandırma problemlerinde oldukça işe yaramaktadır. Veri setinde toplam 87000 resim bulunmaktadır ve bu resimler 200x200 piksel formatındadır. Bu resimlerden 78300 tanesi eğitim için geri kalanı da doğrulama seti için ayrılmıştır. Test dosyasında da her bir sınıf için 29 resim bulunmaktadır.



Şekil 3.1. İşaret dili harfleri

ASL Alphabet dışında incelenen bir diğer veri seti de Sign Language MNIST veri setidir [4]. Orjinal MNIST veri seti el yazısı rakamlarının bulunduğu ve resim tabanlı makine öğrenmesi metotlarında sıklıkla kullanılan bir veri setidir. Veri seti formatında eğitim ve test etiketleri 0-25 aralığında A'dan Z'ye her harfe karşılık gelen değerler içerir. (9=J ve 25=Z için bir değer bulunmuyor hareket içerdikleri için). Eğitim dosyasında 27455 kayıt ve test dosyasında 7172 kayıt bulunuyor. Yaklaşık olarak orjinal MNIST veri setinin yarısı boyutunda bu veri seti. Sonuç olarak ASL Alphabet veri seti kullanılmıştır.

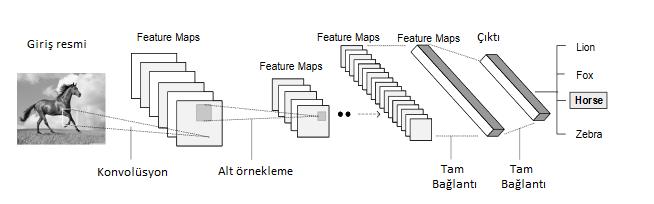
**3.2. Veri Setinin Ön İşlenmesi**

Train ve test ismindeki klasörlerde tutulan veriler load\_unique fonksiyonu ile yüklenmiştir. Daha sonra test verileri Matplotlib ile görselleştirilmiştir. 29 sınıf için bir dictionary tanımlandıktan sonra load\_data fonksiyonu ile resimler ve etiketler tanımlanmıştır. to\_categorical fonksiyonu ile veriler kategorik olarak kaydedilmiştir. Daha sonra scikit-learn train\_test\_split fonksiyonu ile eğitim ve test verileri 0.95, 0.05 oranında ayrılmıştır. Bu işlem sonucunda 82650 tanesi eğitim, 4350 tanesi de test verisi olarak ayrılmıştır

**4. KONVOLÜSYONEL SİNİR AĞLARI**

**4.1. Genel Bilgiler**

Konvolüsyonel sinir ağları, derin sinir ağları ile görüntü sınıflandırma probleminin çözümüne yönelik biyolojik sinir ağlarından esinlenerek oluşturulmuş yarı denetimli bir makine öğrenmesi türüdür [5]. Çok katmanlı ileri beslemeli bir sinir ağı olan bu metot ile nesne tespiti ve sınıflandırma problemleri için çok başarılı sonuçlar elde edilmiştir [6]. Konvolüsyonel sinir ağlarının en önemli özelliklerinden biri elle hazırlanan özniteliklere ve filtrelere ihtiyaç duyulmamasıdır. Öznitelikler otomatik olarak çıkarılabilir. Konvolüsyonel sinir ağları genel olarak; konvolüsyon operatörü (evrişim katmanı), aktivasyon fonksiyonu, havuzlama (alt örnekleme) ve tam bağlantı katmanı olmak üzere dört ana katmandan oluşur. Bu ana katmanlar yapay sinir ağları oluşturulurken birden fazla kez kullanılabilmektedir. Ayrıca kendine özgü görevleri olan ayrı katmanlar da bulunabilir.



Şekil 4.1. Konvolüsyonel sinir ağları genel mimarisi [9]

**4.2. Evrişim Katmanı ve Aktivasyon Fonksiyonu**

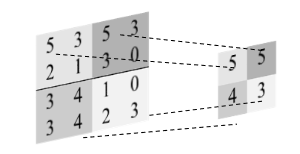
Konvolüsyon operatöründen oluşan evrişim katmanı, konvolüsyonel sinir ağlarının temel yapı taşıdır. Konvolüsyon ile giriş görüntüsünün özellikleri çıkarılır. İlk olarak görüntüdeki kenarlar gibi düşük seviyedeki bazı özellikleri çıkarmak için görüntüye bazı filtreler uygulanır. Bu filtreler genellikle çok boyuttan oluşur ve piksel değerleri içerir. Filtreler birçok kez farklı boyut ve değerlerde uygulanarak düşük seviyeden yüksek seviyeye görüntüdeki özellikler çıkarılır [7]. Matematiksel olarak evrişim katmanı aşağıdaki denklem ile ifade edilebilir.

ℎ𝑙+1=𝑓(𝑊𝑙∗ℎ𝑙+𝑏𝑙) (1.1)

Denklemde bulunan ℎ𝑙+1 katmandaki çıktı sonucunu, 𝑊𝑙, ℎ𝑙 ve 𝑏𝑙 ise bir önceki katmanda bulunan çekirdek ağırlıklarını, öznitelik haritasını (feature map) ve yanlılık diğer bir isimle bias değerlerini ifade eder. 𝑓 ise bir önceki katmandan elde edilen çıktı sonucuna doğrusal olmayanlık (non-linearity) işlemini uygulayan bir aktivasyon fonksiyonunu ifade eder [8]. Her bir konvolüsyon işleminden sonra, yapay sinir ağına doğrusal olmayan özellikleri tanıtmak için aktivasyon fonksiyonu kullanılmaktadır. Aktivasyon fonksiyonları kullanılmazsa, ağ basit bir şekilde veriyi doğrusal olarak modelleyen bir yapıda kalır. Aktivasyon fonksiyonu olarak nesne tespiti problemlerinde genellikle ReLu veya sigmoid kullanılır. Hesaplama yükünün hiperbolik tanjant ve sigmoid aktivasyon fonksiyonlarına göre daha az olması nedeniyse çok katmanlı yapay sinir ağlarında ReLu daha çok tercih edilmiştir. ReLu, görüntüdeki her bir piksel için tekrar uygulanır ve negatif değerler taşıyan pikselleri sıfıra eşitler. Evrişim katmanı ve aktivasyon fonksiyonuyla doğrusal olmayanlık sağlandıktan sonra genellikle havuzlama işlemi yapılarak parametre sayısı azaltılır.

**4.3. Havuzlama Katmanı**

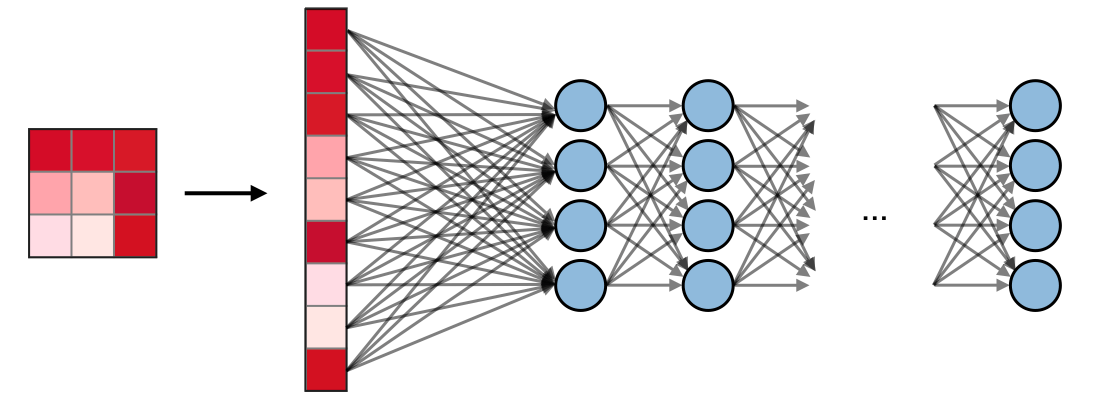
Literatürde alt örnekleme (ortaklama) olarak da kullanılan havuzlama katmanı, tipik olarak bir miktar uzamsal değişkenlik gösteren bir konvolüsyon katmanından uygulanan bir örnekleme işlemidir [7]. Bu katmanda girdi olarak alınan öznitelik haritaları ile elde edilen bilgilerin özeti belirtilerek, hem parametre sayısı azaltılarak verinin boyutu düşürülmüş olur, hem de modelin aşırı uyumunun (overfitting) önüne geçilmiş olunur.



Şekil 4.2. Havuzlama işleminin gerçekleştirilmesi

**4.4. Tam Bağlantı Katmanı**

Tam bağlantı katmanı, her girişin tüm nöronlara bağlı olduğu bir giriş üzerinde çalışır. Nöronlar bu katmanda birbirine tam bağlı biçimde yer alırlar. Eğer varsa tam bağlantı katmanı genellikle CNN mimarisinin sonuna doğru bulunur ve sınıf skorları gibi hedefleri optimize etmek için kullanılabilir [7]. Şekil 1.5’te bir konvolüsyonel sinir ağında tam bağlantı katmanı örneği sunulmuştur.



Şekil 4.3. Tam bağlantı katmanı

**5. PROJE ADIMLARI**

**5.1. CNN Mimarisinin Oluşturulması**

CNN modeli create\_model fonksiyonu ile oluşturulmuştur. İlk olarak sıralı model oluşturulduktan sonra 16 birim bulunan ReLu aktivasyon fonksiyonu kullanılarak Conv2D tipinde bir katman eklenmiştir. Ara katmanlarda sırasıyla 32, 32, 64, 128, 256 birim sayısında aktivasyon fonksiyonu olarak ReLu kullanılarak Conv2D katmanları eklenmiştir.

Daha sonra MaxPool2D katmanı eklenip BatchNormalization işlemi yapılmıştır. Flatten ve Dropout katmanları ile optimizasyon ve regülarizasyon işlemleri sağlanmıştır. Son olarak 512 birim bulunan Dense katmanı, ReLu aktivasyon fonksiyonu parametresiyle eklenmiştir. Çıkış katmanı olarak 29 birim bulunan son Dense katmanı da Softmax aktivasyon fonksiyonu ile eklenmiştir. Model derlenirken optimizer olarak Adam kullanılmıştır ve kayıp fonksiyonu olarak kategorik çapraz entropi seçilmiştir.

Model özeti şu şekildedir;

Model: "sequential\_1"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

============================================================conv2d\_1 (Conv2D) (None, 64, 64, 16) 448 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv2d\_2 (Conv2D) (None, 64, 64, 32) 4640 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

max\_pooling2d\_1 (MaxPooling2 (None, 21, 21, 32) 0 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv2d\_3 (Conv2D) (None, 21, 21, 32) 9248 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv2d\_4 (Conv2D) (None, 21, 21, 64) 18496 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

max\_pooling2d\_2 (MaxPooling2 (None, 7, 7, 64) 0 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv2d\_5 (Conv2D) (None, 7, 7, 128) 73856 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv2d\_6 (Conv2D) (None, 7, 7, 256) 295168 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

max\_pooling2d\_3 (MaxPooling2 (None, 2, 2, 256) 0 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

batch\_normalization\_1 (Batch (None, 2, 2, 256) 1024 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

flatten\_1 (Flatten) (None, 1024) 0 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dropout\_1 (Dropout) (None, 1024) 0 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_1 (Dense) (None, 512) 524800 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_2 (Dense) (None, 29) 14877 ============================================================Total params: 942,557

Trainable params: 942,045

Non-trainable params: 512

**5.2. Eğitim ve Test Adımları**

Oluşturulan model fit\_model fonksiyonu ile batch size ve epochs parametreleri sırasıyla 64 ve 5 seçilerek, 1’e 10 oranında doğrulama ve eğitim kümeleri ayrılarak eğitilmiştir. Eğitim işleminde her bir adımdaki kayıp, doğruluk, doğrulama kümesindeki doğruluk, doğrulama kümesindeki kayıp metrikleri aşağıda gösterilmiştir.

Train on 74385 samples, validate on 8265 samples

Epoch 1/5

74385/74385 [==============================] - 301s 4ms/step - loss: 0.8165 - accuracy: 0.8359 - val\_loss: 0.2613 - val\_accuracy: 0.9695

Epoch 2/5

74385/74385 [==============================] - 297s 4ms/step - loss: 0.1913 - accuracy: 0.9776 - val\_loss: 0.2633 - val\_accuracy: 0.9486

Epoch 3/5

74385/74385 [==============================] - 294s 4ms/step - loss: 0.1773 - accuracy: 0.9804 - val\_loss: 0.1520 - val\_accuracy: 0.9873

Epoch 4/5

74385/74385 [==============================] - 289s 4ms/step - loss: 0.1462 - accuracy: 0.9856 - val\_loss: 0.2436 - val\_accuracy: 0.9678

Epoch 5/5

74385/74385 [==============================] - 327s 4ms/step - loss: 0.1222 - accuracy: 0.9912 - val\_loss: 0.0910 - val\_accuracy: 0.9936

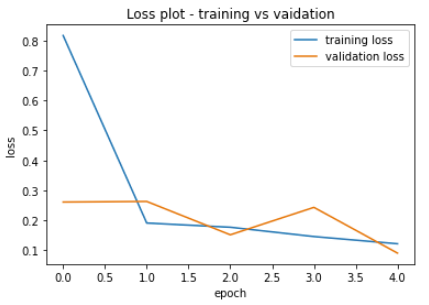
Daha sonra load\_test\_data fonksiyonu ile test kümesindeki resimler yüklenmiştir.

**6. SONUÇLAR**

Eğitim ve test kümelerindeki eğitim adımları boyunca her bir devirdeki başarı oranları ve kayıp oranları Matplotlib ile görselleştirilmiştir.



Şekil 6.1. Her devirdeki doğruluk değerleri değişimi



Şekil 6.2. Her devirdeki kayıp değerleri değişimi

evaluate fonksiyonu ile en son ortaya çıkan doğruluk ve kayıp değerleri sırasıyla %99.22 ve 0.092007 olarak elde edilmiştir. Test kümesindeki resimler ile sınıflandırma işlemi görselleştirildiğinde aşağıdaki sonuç elde edilmiştir.

****

**KAYNAKÇA**

[1] Lippmann, R.P. An introduction to computing with neural nets. IEEE ASSP Magazine, 4, 4-22, 1987.

[2] https://www.linkedin.com/pulse/yapay-sinir-a%C4%9Flar%C4%B1-ve-tek-katmanl%C4%B1-a%C4%9Flarda-%C3%B6%C4%9Frenme-tanju-do%C4%9Fan (Ziyaret tarihi: 4 Haziran 2020)

[3] LeCun, Y., Bengio, Y., Hinton, G. Deep learning. Nature, 436–444, 2015, doi:10.1038/nature14539.

[4] https://www.kaggle.com/datamunge/sign-language-mnist (Ziyaret tarihi: 4 Haziran 2020)

[5] Zhou, S., Chen, Q., Whang, X., Convolutional Deep Networks for Visual Data Classification, Neural Process Lett, 38:17-27, 2013.

[6] Uijlings, J., R., R., Selective Search for Object Recognition, International Journal of Computer Vision, 104.2, 2013.

[7] Amidi, S., https://stanford.edu/~shervine/l/tr/teaching/cs-230/cheatsheet-convolutional-neural-networks (Ziyaret tarihi: 28 Aralık 2019).

[8] Kurt, F., Evrişimli Sinir Ağlarında Hiper Parametrelerin Etkisinin İncelenmesi, Master's thesis, Eğitim Bilimleri Enstitüsü, 2018.

[9] Ajeet R. P., Manjusha P., Siddharth R., Application of Deep Learning for Object Detection, Procedia Computer Science, Volume 132, Pages 1706-1717, ISSN 1877-0509, 2018, https://doi.org/10.1016/j.procs.2018.05.144.

[10] https://www.kaggle.com/grassknoted/asl-alphabet (Ziyaret tarihi: 4 Haziran 2020)