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

DERİN ÖĞRENME ALGORİTMALARI İLE TRAFİK İŞARET VE LEVHALARININ TANIMLANMASI

YÜKSEK LİSANS TEZİ

AHMET YAVUZ

DENİZLİ, ŞUBAT - 2021

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

DERİN ÖĞRENME ALGORİTMALARI İLE TRAFİK İŞARET VE LEVHALARININ TANIMLANMASI

YÜKSEK LİSANS TEZİ

AHMET YAVUZ

DENİZLİ, ŞUBAT - 2021

Bu tezin tasarımı, hazırlanması, yürütülmesi, araştırmalarının yapılması ve bulgularının analizlerinde bilimsel etiğe ve akademik kurallara özenle riayet edildiğini; bu çalışmanın doğrudan birincil ürünü olmayan bulguların, verilerin ve materyallerin bilimsel etiğe uygun olarak kaynak gösterildiğini ve alıntı yapılan çalışmalara atfedildiğine beyan ederim.

Ahmet YAVUZ

ÖZET

DERİN ÖĞRENME ALGORİTMALARI İLE TRAFİK İŞARET VE LEVHALARININ TANIMLANMASI

YÜKSEK LİSANS TEZİ

AHMET YAVUZ

PAMUKKALE ÜNİVERSİTESİ FEN BİLİMLERİ ENSTİTÜSÜ
ELEKTRİK-ELEKTRONİK MÜHENDİSLİĞİ ANABİLİM DALI
(TEZ DANIŞMANI:PROF. DR. SERDAR İPLİKÇİ)

DENİZLİ, ŞUBAT - 2021

Günümüz teknoloji koşullarında araçlarda kullanılan sürücü destek sistemleri oldukça yaygınlaşmıştır. Araç sayısındaki artış nedeni ile oluşacak kazalar, araçlarda kullanılan sürücü destek sistemleri ile azaltılması amaçlanmaktadır.

Sürücü destek sistemlerinin başında da yollardaki trafik işaretlerini tanıyıp sürücüye, işaretlere göre yol koşullarını bildiren otomatik trafik işareti tanıma sistemleri gelmektedir.

Bu tezin temel amacı, görüntü işleme tekniklerini ve yapay zekanın temellerini kullanarak, trafik işaretlerini otomatik olarak tanımak için derin öğrenmeye dayalı modeller oluşturup bu modellerin başarısını karşılaştırmaktır.

Bu çalışmada 21249 veriden ve 91 sınıftan oluşan "Traffic Sign Images From Turkey" veri seti kullanılmıştır. Kullanılan bu veri seti İleri Beslemeli Sinir Ağları (Feed Forward Neural Network - FFNN), Evrişimli Sinir Ağları (Convolutional Neural Network - CNN), Yinelemeli Sinir Ağları (Recurrent Neural network - RNN), Uzun Kısa Süreli Bellek (Long-Short Term Memory - LSTM) ve Kapı Yinelemeli Birimler (Gated Recurrent Units - GRU) derin ağ modellerinde eğitilmiş ve çıkan sonuçlar karşılaştırılmıştır.

Kullanılan modellerde CNN %99.78 tahmin başarısıyla en iyi tahmin başarısını elde eden model olmuştur. GRU modeli %96.7, LSTM modeli %94.725, RNN modeli %89.67, FFNN modeli ise %84.395 test başarısı sağlayarak başarılı sonuçlar vermiştir.

ANAHTAR KELİMELER: Derin öğrenme, Makine öğrenmesi, Trafik işareti tanıma, Evrişimli sinir ağları, Yinelemeli sinir ağları, Sinir ağları

ABSTRACT

TRAFFIC SIGN RECOGNITION WITH DEEP LEARNING ALGORITHMS

MSC THESIS

AHMET YAVUZ

PAMUKKALE UNIVERSITY INSTITUTE OF SCIENCE ELECTRİCAL AND ELECTRONİCS ENGİNEERİNG

(SUPERVISOR:PROF. DR. SERDAR İPLİKÇİ)

DENİZLİ, FEBRUARY 2021

Driver support systems used in vehicles with today's technology are widely used. With these systems, it is planned to prevent increasing traffic accidents.

At the top of the driver support systems, there is automatic traffic sign recognition systems that recognize the traffic signs on the roads and inform the driver of the road conditions according to the signs.

The main purpose of this thesis is to create models based on deep learning to automatically recognize traffic signs using image processing techniques and the fundamentals of artificial intelligence, and to compare the success of these models.

The "Traffic Sign Images From Turkey" data set consisting of 21249 data and 91 classes was used in the thesis. This data set used includes Feed Forward Neural Network (FFNN), Convolutional Neural Network (CNN), Recurrent Neural network (RNN), Long-Short Term Memory (LSTM) and Gated Recurrent Units (GRU) were trained in deep network models and the results were compared.

In the models used, CNN was the model that achieved the best prediction success with 99.78% prediction success. GRU model 96.7%, LSTM model 94.725%, RNN model 89.67%, FFNN model yielded very good results with 84.395% test success.

KEYWORDS: Deep learning, Machine learning, Traffic sign recognition, Convolutional neural network, Recurrent neural network, Neural network

İÇİNDEKİLER

Sayfa

ŞEKİL LİSTESİ

Sayfa

TABLO LİSTESİ

Sayfa

KISALTMALAR

GTRSB	**:**	German Traffic Sign Benchmarks
BTSC	**:**	Belgium Traffic Sign Classification
ELM	**:**	Extreme Learning Machine
GPU	**:**	Graphic Processing Unit
CPU	**:**	Central Processing Unit
GHT	**:**	Generalized Hough Transform
PCA	**:**	Principal Component Analysis
MLP	**:**	Multi Layer Perceptron
CNN	**:**	Convolution Neural Network
RNN	**:**	Recurrent Neural Network
GRU	**:**	Gated Recurrent Unit
LSTM	**:**	Long-Short Term Memory

ÖNSÖZ

Bu tez çalışmasında trafik işaretlerinin makine öğrenimi metoduyla tanınabilmesi için beş adet farklı sinir ağı modellenmiş ve kullanılan veri setinde özellikle CNN modelinde oldukça yüksek test başarısı sağlanmıştır.

Yüksek lisans eğitimim boyunca bilgi birikimini benimle paylaşan, çalışmalarıma yön veren, bana güvenen ve destekleyen değerli hocam Prof. Dr. Serdar İPLİKÇİ'ye ve her daim yanımda olan ve benden yardımlarını esirgemeyen arkadaşlarım Batuhan BİLGİ'ye ve Ali MENEMEN'e teşekkürlerimi sunarım.

Hayatım boyunca bende maddi ve manevi desteklerini esirgemeyen, her zaman ve her koşulda yanımda olan aileme çok teşekkür ederim.

1. GİRİŞ

Otomobil teknolojisi bilgisayarların gelişmesiyle mekanik sistemlerden daha çok elektronik sistemler üzerine yoğunlaşmaya başlamıştır. Elektronik sistemler ilk olarak araç yol tutuşunu artıran, fren mesafesini azaltan ve araç içerisinde konforu sağlayan sistemler için kullanılmışlardır. Elektronik sistemler geliştikçe araçlar da daha efektif, otonom sürüş teknolojinin temeli olan, paralel park etme, adaptif hız ve far kontrolü, sürücü takip sistemi, trafik işareti tanıma sistemi, şerit takip sistemi, otomatik frenleme ile çarpışma önleyici sistem gibi birçok sürücüyü asiste eden özellikler kazanmıştır. Bu yardımcı özellikler sürüş konforu, can ve mal güvenliği ve ekonomik açıdan oldukça önem arz etmektedirler.

Trafik işaretleri sürüş güvenliği için oldukça önem arz etmektedir. Sürüş sırasında güvenliği sağlamak için trafik işaretleri dikkatli bir şekilde takip edilip işaretlere göre gerekli reaksiyonların verilmesi gerekmektedir. Örneğin "Sola Tehlikeli Viraj", azami hız sırı işaretleri gibi işaretler sürücü tarafından fark edilip can güvenliği için araç hızı güvenli bir hıza düşürülmelidir. Fakat çevre etkenleri, dikkat dağınıklığı gibi sebeplerle trafik işaretlerinin önceden fark edilmesi pek mümkün olmamaktadır. Bu gibi olumsuz durumları ortadan kaldırmak için sürücüyü trafik işaretleri için önceden uyaran sistemlere günümüz araç teknolojilerinde oldukça önem verilmektedir.

Araçlar üzerlerinde bulundurdukları kameralar yardımıyla yol taraması yapıp trafik işaretlerini otomatik olarak tanıyabilir duruma gelmiştir. Düşük seviye otonom sistemlerde az sayıda trafik işareti tanıması yapılmaktadır. Örneğin azami hız limiti tabelaları tanınıp sürücünün yolda izin verilen azami hızı aşması durumunda araçlar sesli uyarı sistemleriyle sürücüyü uyarabilmektedirler. Ayrıca sürücünün izin vermesi durumunda aracın azami hız limitini geçmesi de engellenebilmektedir.

Daha yüksek seviye otonom sistemlerde ise azami hız limiti gibi belli başlı trafik işaretlerinin yanında, aracın verdiği kararların daha güvenli hale getirilmesi için daha fazla trafik işareti, yol çizgileri ve çevresel faktörler taranmak zorundadır. Örneğin aracın şerit değiştirme kararı verebilmesi için trafik işaretleri ve yol

çizgilerini tanıyıp geçeceği şeritteki trafik durumuna bakarak karar vermesi gerekmektedir.

Bu tez çalışmasında, otonom sürüş seviyelerinin temelini oluşturan trafik işaretlerinin makine öğrenmesinin alt dalı olan derin öğrenme metoduyla tanınması amaçlanmıştır.

2. LİTERATÜR TARAMASI

Trafik işareti tanıma konusu trafikte can ve mal güvenliği açısından önemli bir konu olduğu için bu konuda gerek lisans üstü gerek lisans seviyesinde birçok önemli çalışma yapılmıştır. Bu çalışmalar trafik işareti tanıma konusunda farklı yaklaşımlarda bulunmuşlardır.

Miura ve diğ. (2000) tarafından yapılan çalışmada gerçek zamanlı trafik işareti tanıma için aktif bir görüş sistemi sunulmuştur. Sunulan sistem, biri geniş açılı diğeri telefoto lensli iki adet kameradan alınan görüntüler üzerinde görüntü işleme kartlı bir PC'de renk ve yoğunluğa göre ayırıp kenar tarama yöntemleri ile işleyerek trafik işaretlerini tespit etmeye çalışmaktadır.

Huang ve diğ. (2017) tarafından yapılan çalışmada trafik işaretlerini tanıma için GTSRB, BTSC ve revize edilmiş MASTIF veri setleri üzerinde çalışarak, işaretli ve işaretsiz gradyanlara sahip değişken bir HOG tanımlayıcısı sunmuşlardır ve sınıflandırma için ELM algoritması kullanarak tek bir gizli katmanlı sinir ağı modeli geliştirmişleridir. Yapılan bu çalışmada diğer görüntü tanıma metodlarına göre veri işleme zamanından önemli ölçüde tasarruf sağlanmış ve %98,26'lık bir başarı oranı elde edilmiştir.

Shustanov ve Yakimov (2017) tarafından yapılan çalışmada evrişimli sinir ağları tabanlı ve CUDA çekirdekli bir taşınabilir GPU üzerinde çalışan gerçek zamanlı bir trafik işareti tanıma sistemi tasarlanmıştır. Yapılan bu çalışmada alınan görüntüler üzerinde GHT (Generalized Hough Transform) algoritması kullanılarak trafik işaretlerinin koordinatları belirlenmeye çalışılmış ve bu belirlenen alanlardaki işaretlerin sınıfı evrişimli sinir ağlarıyla tespit edilmeye çalışılmıştır. GTSRB ve GTSDB veri setlerini ve tensorflow kütüphanesini kullanarak tasarlamış oldukları modeli eğitmişlerdir ve %99,94 gibi bir yüksek sınıflandırma oranı yakalamayı başarmışlardır.

Hannan ve diğ. (2014) tarafından yapılan çalışmada trafik işaretlerini sınıflandırmak için farklı ışık koşullarında bile çalışabilecek görüntüleri ön işlemlerden geçirerek öznitelik çıkarımı yapan çok katmanlı algılayıcı modeli

geliştirmişlerdir. Yapılan araştırmada önerilen yöntem farklı ışık koşullarında trafik işaretlerini tanımanın üstesinden gelebilmesi dışında 0.134 saniye gibi bir hesaplama süresi sunmuştur. Ancak önerilen bu yöntemin doğruluk oranı düşüktür. 300 görüntüden oluşan bir test verisinde %84.4 başarı oranı vermiştir.

Thanh (2014) tarafından yapılan çalışmada Ana Bileşen Analizi (PCA) ve Çok Katmanlı Algılayıcı ağ (MLP) kullanarak trafik işareti tanıma için yeni bir yöntem önerilmiştir. Önerilen bu yöntemde sinyaller, biri renk olmak üzere iki bileşenden ayrı ayrı algılanır ve daha sonra şekle göre üç sınıfa ayrılır: daire, kare ve üçgen. Bu sinyallerin PCA tabanlı karakteristikleri, eğitim aşamasında MLP için girdi olarak verilmiş ve eğitim trafik işareti sınıflarına göre gerçekleştirilmiştir. Önerilen bu yöntemle ile 500'den fazla test verisi üzerinde %96'lık bir başarım sağlanmıştır.

Bueno ve diğ. (2007) tarafından yapılan çalışmada farklı görüntü ön işleme tekniklerinin bir kombinasyonunu kullanarak İspanya'dan 9 tür trafik işaretini sınıflandırmışlardır. 2 katmanlı bir algılayıcı ağ ile birlikte, test için 78 görüntüden oluşan bir sette %98.72 doğruluk başarısı yakalamışlardır. Görüntü ön işleme tekniklerinin uygulama sırası, bu araştırmanın en önemli özelliğidir ve görüntüyü yumuşatmak ve histogramları eşitlemek için medyan filtresinin kullanılmasını önermişlerdir.

Görüldüğü üzere literatürde trafik işaretleri tanıma ile ilgili birçok farklı yaklaşım bulunmaktadır. Yapılan bu çalışmalar genellikle trafik işaretlerini sınıflandırma ve tespit etme olarak iki aşamada gerçekleştirilmiştir. Literatürde trafik işaretlerinin sınıflandırılması için tek bir sınıflandırma yöntemi üzerinde durulmuştur. Yapılan bu tez çalışmasında, beş farklı derin öğrenme algoritması kullanılarak trafik işaretleri sınıflandırılmaya çalışılmıştır. Bu tez çalışması da bu yönüyle literatürdeki diğer çalışmalardan ayrılmaktadır.

3. YAPAY SİNİR AĞLARI

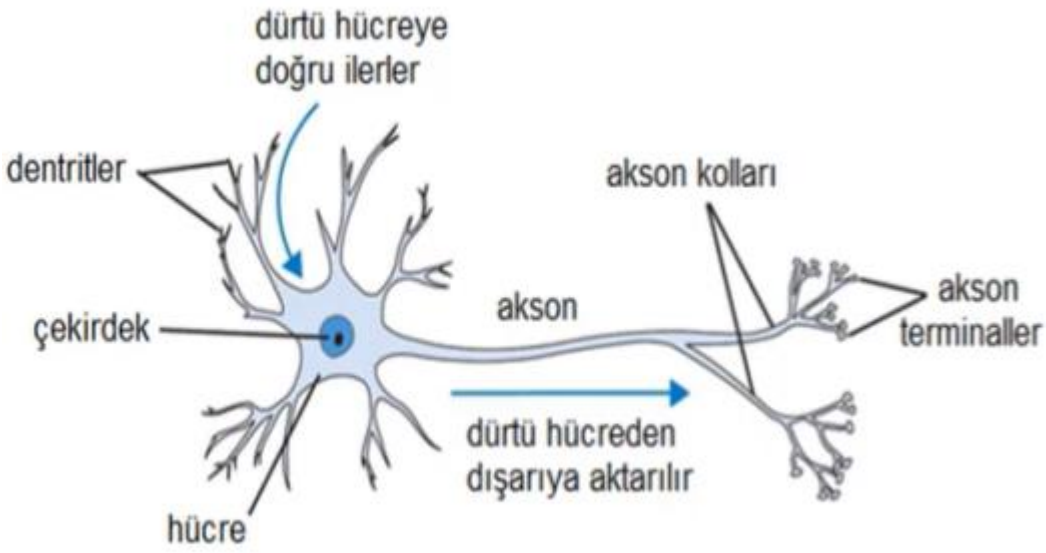
3.1 Yapay Sinir Ağlarının Kısa Tarihi

Günümüzde yapay sinir ağları ve yapay zeka terimlerinden sıkça bahsedilse de aslında yapay sinir ağları gibi konular 1950'li yıllarda ortaya atılmış ve çalışılmaya başlanmıştır. 1950 yılında yayınlanan "Computing Machinery and Intelligence" adlı makalede "Makineler düşünebilir mi?" sorusu bu konunun öncüsü olmuştur (Turing 1950). Bu sorudan sonra 1955 yılında John McCharthy ilk kez yapay zeka terimini kullanmıştır. 1957 yılında ise Frank Rosenblatt bugün tüm yapay sinir ağı modellerinin temelini oluşturan perseptronun yani algılayıcıların tanımını yapmıştır (Rosenblatt 1958). 1969 yılında Minsky XOR probleminin tek katmanlı bir ağ yapısıyla çözülmesinin mümkün olmadığını ortaya koyup, çok katmanlı yapay sinir ağı mimarilerinin temelini atmıştır (Minsky 1969). 1987 yılında yapay sinir ağlarının edindiği bilgiyle yeni bilgileri ve kararları güncelleyebildiği, yani tahminleme ve öğrenme başarısını yüksek ölçüde artıran geriye yayılım algoritması ortaya atılmıştır (Plaut ve Hinton 1987). 1998 yılında ise bugün derin öğrenmenin temeli olan evrişimli sinir ağları modeli ile rakamları sınıflandıran yüksek oranda başarılı bir model geliştirilmiştir. 2009 yılında ise 167 ülkenin iş birliğiyle "IMAGENET" veri setinin ücretsiz bir şekilde herkese açılmasıyla, CPU ve GPU teknolojilerinin de gelişip matris çarpımlarının daha kolay ve hızlı bir şekilde yapılabilir hale gelmesiyle derin öğrenmenin önü açılmıştır.

Bu gelişmelerle birlikte birçok teknoloji şirketi yapay zeka alanında önemli yatırımlar yapmıştır ve bu sayede o yıllardan günümüze kadar bir çok derin öğrenme modeli geliştirilmiştir.

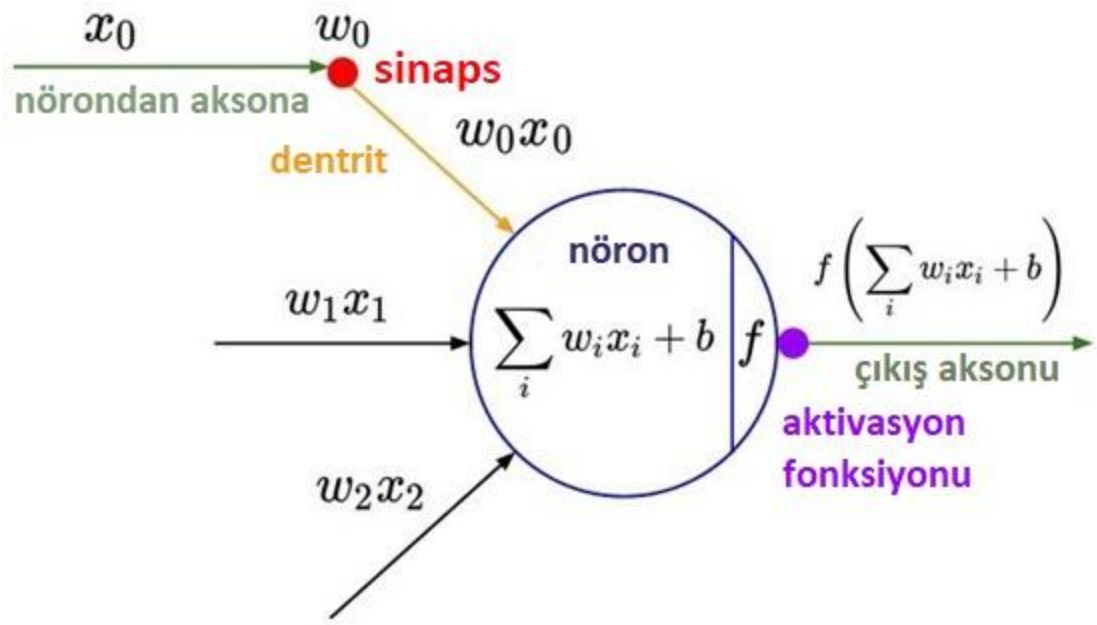
3.2 Yapay Sinir Ağlarının Biyolojik Temelleri

Şekil 3.1'de gösterilen biyolojik canlılarda bulunan sinir hücreleri, kabaca dentrit, çekirdek ve aksonlardan meydana gelmektedir. Dentritlerden gelen bilgiler çekirdekte işlenerek akson boyunca diğer sinir bağlanarak aktarılmaktadır.

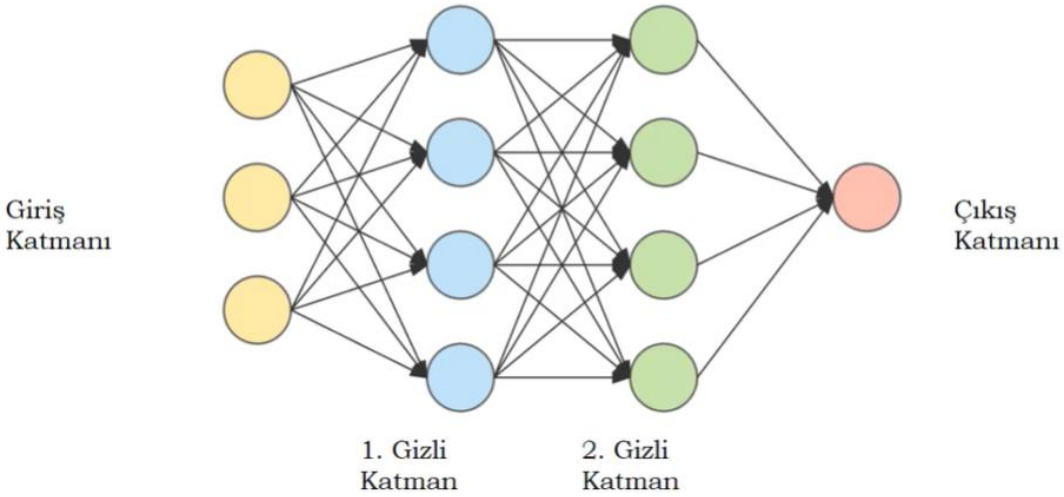


Şekil 3.1: Biyolojik sinir hücresi.

Şekil 3.1' de gösterilen sinir hücresi Şekil 3.2'de gösterildiği şekilde matematiksel olarak modellenmektedir (Kızrak 2018). Bu matematiksel modelde girişe gelen x_i değeri (bir önceki nörondan da gelmiş olabilir) dentrit bölgesinde w_i ağırlık değeriyle çarpılır ve sinir hücresine iletilir. Sinir hücresinde dentritten gelen bütün ağırlık ve giriş çarpımları toplanır ve üzerine bir b değeri eklenir. Çıkan sonuç bir aktivasyon fonksiyonundan geçirilerek sonuç çıkışa aktarılır. Bu sonuç nihai çıkış olabildiği gibi başka bir nöronun girişi de olabilmektedir. Her bir nöron bu şekilde hesaplandıktan sonra birbirlerine seri ve paralel olarak bağlanmaktadır. Bu durum Şekil 3.3'te gösterilmiştir (Chandupatla 2019). Bu bağlantılar giriş katmanı, gizli katman ve çıkış katmanı olarak adlandırılmaktadır. Birer adet giriş ve çıkış bulunan modeller tek katmanlı, birden çok gizli katman bulunan modeller ise çok katmanlı model olarak adlandırılırlar.



Şekil 3.2: Sinir ağlarının matematiksel modeli.

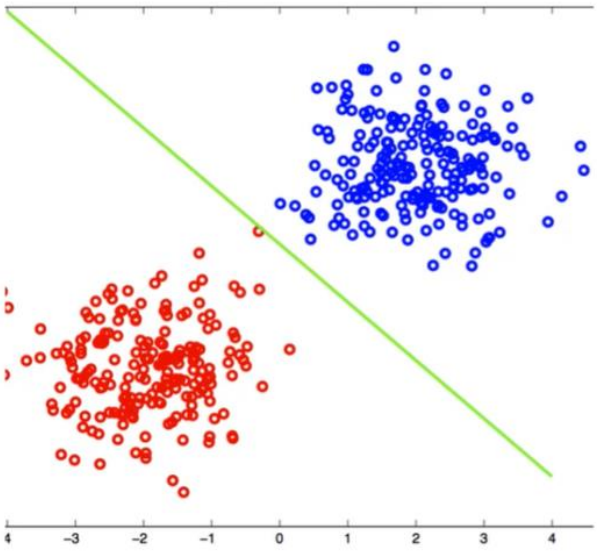


Şekil 3.3: Genel yapay sinir ağı modeli.

3.3 Tek Katmanlı Sinir Ağı Modelleri

Tek katmanlı sinir ağları en basit sinir ağı modelidir. Kısaca mantık devrelerinde kullanılan kapılar gibi düşünülebilir. Girdileri doğrudan çıkış katmanına veya bir sonraki katmana eklenebilir. Genellikle iki girdi ve bir çıkıştan oluşurlar.

Tek katmanlı sinir ağları sınırların birbirinden kolayca ayrılabildiği genellikle iki sınıf içeren basit sınıflandırma problemlerini çözmek için kullanılmaktadırlar. Bu modeller sınıfların dış çevrelerini kestirmeye çalışmak yerine Şekil 3.4'te gösterildiği gibi bu iki sınıfı birbirinden ayıran karar doğrusunu (diskriminant) çizmeye çalışmaktadırlar. İkiden fazla sınıf olduğu durumlarda ise birden çok tek katmanlı sinir ağları birleştirilerek ayrı ayrı doğrular çizdirilerek ayırma işlemi gerçekleştirilmektedir.



Şekil 3.4: Basit bir sınıflandırma grafiği.

Diskriminantların x girdisine bağlı olduğu varsayılırsa, diskriminantlar Denklem (3.1)'deki ifade edilebilirler.

$$g(x|w_i, w_{i0}) = w_i^T + w_{i0} = \sum_{j=1}^{d} w_{ij}x_j + w_{i0} \tag{3.1}$$

Burada w ağırlıkları, x ise girişleri simgelemektedir.

Denklem (3.1)'in çıktısı x_j girdilerinin ağırlıkları toplamıdır. w_j ağırlığının büyüklüğü ise x_j girdisinin önemini simgelemektedir. Diğer bir deyişle ağırlığı büyük olan girdi girişten gelen önemli özellikleri taşıyor demektir. Yani çıktı; farklı özniteliklerin toplamı olarak yazılmıştır.

Denklem (3.1) genelleştirilirse;

$$g(x|W_i, w_i, w_{i0}) = x^T W_i x + w_i x + w_{i0} \tag{3.2}$$

şeklinde ifade edilebilir.

1. dereceden diskriminantlar kapsamlı modeller oluşturmak için yeterince esnek değillerdir. Bu tür kapsamlı modeller için yüksek dereceli diskriminant denklemlerine ihtiyaç duyulur. Yüksek dereceli diskriminant denklemlerinin oluşturulabilmesi için büyük veri kümelerine ihtiyaç duyulmaktadır. Küçük veri kümeleriyle yüksek dereceli bir diskriminant denklemi kurulacak olursa aşırı öğrenme riskiyle karşılaşılabilir.

Diskriminantı $\emptyset_{ij}(x)$ ile tanımlanırsa genel formül

$$g_i(x) = \sum_{j=1}^{k} w_j \emptyset_{ij}(x) \tag{3.3}$$

şeklinde ifade edilebilir.

x_1 ve x_2 girdileri z'ler şeklinde tanımlanıp;

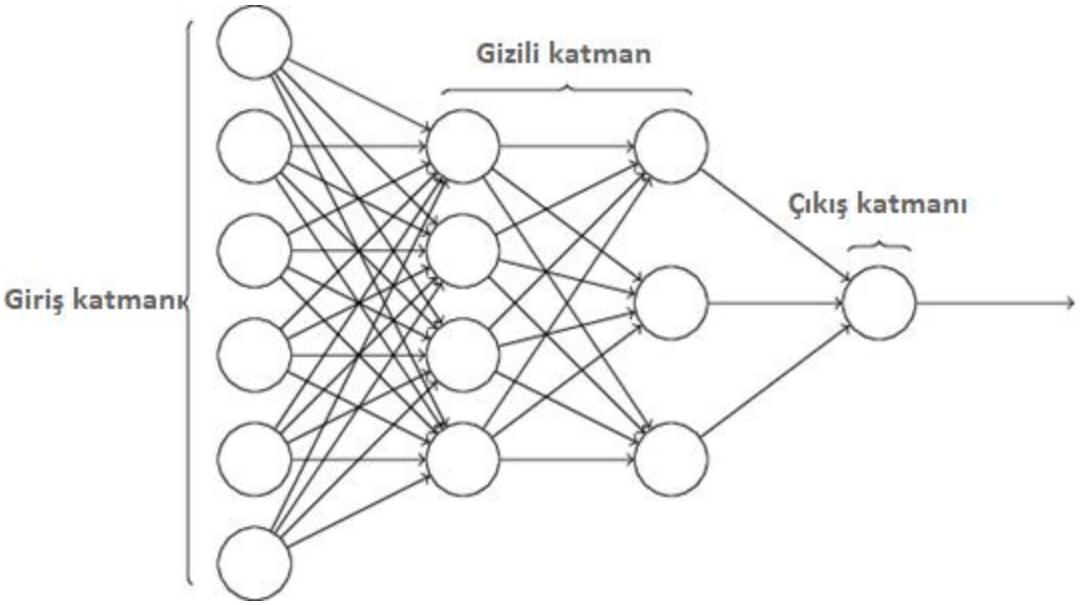
$$z_1 = x_1 \tag{3.4}$$
$$z_2 = x_2 \tag{3.5}$$
$$z_3 = x_1^2 \tag{3.6}$$
$$z_4 = x_2^2 \tag{3.7}$$
$$z_5 = x_1 x_2 \tag{3.8}$$
$$z = [z_1, z_2, z_3, z_4, z_5] \tag{3.9}$$

Denklem (3.9)'daki z vektörü oluşturulduğunda, 5 boyutlu z uzayında yazılan diskriminant, 2 boyutlu x uzayında doğrusal olmayan bir diskriminant olarak

tanımlanabilir. Bu sayede ilk uzayda doğrusal olmayan bir diskriminant fonksiyonunu öğrenmek yerine, doğrusal olmayan bir dönüşümle yeni bir uzaya geçip orada doğrusal bir fonksiyonla tanımlanabilir. Bu sayede doğrusal olmayan bir fonksiyonu doğrusal bir fonksiyonla ayırmak mümkün olmaktadır.

3.4 Çok Katmanlı Sinir Ağı Modelleri

Çok katmanlı algılayıcılar doğrusal bir şekilde ayrılamayan sınıflandırma ve regresyon problemlerinde kullanılan, Şekil 3.5'te gösterildiği gibi birden fazla algılayıcı katmanlarından oluşurlar (Güzel 2018). Çok katmanlı sinir ağı mimarisinde ilk çalışmayı 1960 yılında Widrow ve Hoff yapmışlardır (Widrow ve Hoff 1960).



Şekil 3.5: Çok katmanlı sinir ağı modeli.

Çok katmanlı sinir ağlarını giriş katmanı, gizli katman ve çıkış katmanı olmak üzere üç katmanla tanımlanırlar. Burada yine, modeli eğitmek için bir dizi girdi kullanılır, ancak tek bir algılayıcı beslemek yerine, ilk katmandaki tüm algılayıcılar (nöronlar) beslenirler. Daha sonra, bu katmandan elde edilen çıktılar, bir sonraki katmandaki algılayıcılar için girdi olarak kullanılır ve bir sonucun çıkarılmasından sorumlu olan son katmana ulaşılana kadar böyle devam eder.

Kısaca çok katmanlı bir algılayıcının ilk katmanının girdileri ağırlıklandırarak basit bir karar sürecini yönetir. Sonraki katmanlar ise önceki katmanların çıktısına bağlı olarak daha karmaşık karar süreçlerini yönetirler.

3.5 İleri Yönde Yayılım (Forward Propagation)

Yapay sinir ağlarında ileri yönde yayılım, doğru sonuçlarla karşılaştırılabilecek bir tahmine ulaşmak için giriş verileri kullanılarak ağın mimarisi boyunca soldan sağa doğru yapılan hesaplama işlemidir. Bu işlem, her nöronun giriş verilerini, kendisiyle ilişkilendirdiği ağırlık ve bias değerlerini kullanarak son katmanda bir tahmin çıktısı üreteceği anlamına gelmektedir.

Yapay sinir ağlarında bias değerleri, eğitim sürecini etkileyebilecek sıfır değerlerinden kaçınmak için her bir nöronun aktivasyon fonksiyonunu değiştirmeye yardımcı olan sayısal değerlerdir.

Her bir nörondaki hesaplamalar, giriş verilerini bir ağırlık değeriyle çarpan ve bir bias değeriyle toplayıp, aktivasyon fonksiyonundan geçiren doğrusal bir fonksiyon içerir. Aktivasyon fonksiyonlarının amaçları ise modelin doğrusallığını kırmaktır. Gerçek hayattaki sınıflandırma problemlerinin çoğunun doğrusal olmamasından dolayı modeldeki doğrusallığı kırmak için aktivasyon fonksiyonlarının kullanımı oldukça önemlidir.

Her bir nöronun hesaplama işlevi aşağıdaki denklemlerle ifade edilebilir.

$$Z = X * W + b \tag{3.10}$$

$$\hat{Y} = \sigma(Z) \tag{3.11}$$

Burada X giriş verilerini, W ağırlıkları, b bias değerini ve σ fonksiyonu ise aktivasyon fonksiyonunu ifade eder.

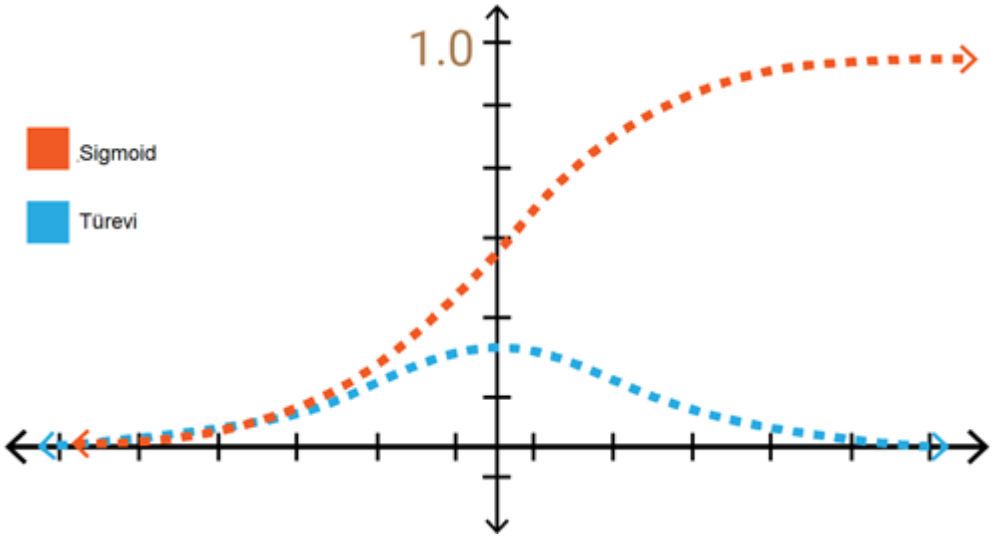
3.6 Aktivasyon Fonksiyonları

Aktivasyon fonksiyonları temelde yapay sinir ağı modellerinde doğrusal girdileri, doğrusal olmayan çıktılara dönüştürmek için kullanılırlar bu da daha derin ağların daha yüksek dereceli polinomları öğrenmesine yardımcı olur.

Doğrusal olmayan aktivasyon fonksiyonlarının bir diğer özelliği de türevlenebilir olmalıdır. Eğer seçilen aktivasyon fonksiyonları türevlenebilir olarak seçilmezlerse geri yayılım sırasında çalışmaları mümkün olmayacaktır. Yapay sinir ağı modellerinde sıklıkla kullanılan ve türevlenebilir olan aktivasyon fonksiyonları aşağıda belirtilmiştir:

1- **Sigmoid:** S şeklinde bir fonksiyondur ve girdiyi 0-1 arasında basit olasılıksal değere dönüştürür. Sigmoid fonksiyonun grafiği ve denklemi Denklem (3.12)'de ve Şekil 3.6'da belirtildiği gibidir (Kızrak 2019).

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{3.12}$$



Şekil 3.6: Sigmoid fonksiyonu ve türevi.

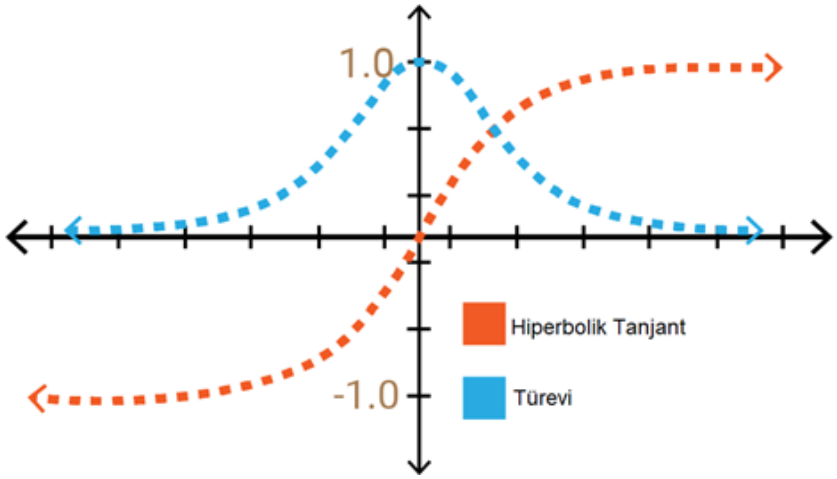
2- **Softmax:** Sigmoid fonksiyonuna benzer şekilde bir olayın n tane olay üzerindeki olasılıksal dağılımını hesaplar. Basit ifadeyle, bu fonksiyon çıktının diğer sınıflara kıyasla, hedef sınıflardan biri olma olasılığını hesaplar. Bu yüzden bu fonksiyon genellikle sınıflandırma ağlarının çıktı

katmanında bulunurlar. Softmax fonksiyonu Denklem (3.13)'te belirtilmiştir:

$$\sigma(x)_i = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}} \tag{3.13}$$

3- **Tanh:** Bu fonksiyon, hiperbolik sinüs ile hiperbolik kosinüs arasındaki ilişkiyi temsil eder ve sonuç -1 ile 1 arasındadır. Bu aktivasyon fonksiyonunun ana avantajı, negatif değerlerin daha kolay ele alınabilmesidir. Fonksiyonun denklemi ve grafiği Denklem (3.14)'te ve Şekil 3.7'de belirtilmiştir (Kızrak 2019).

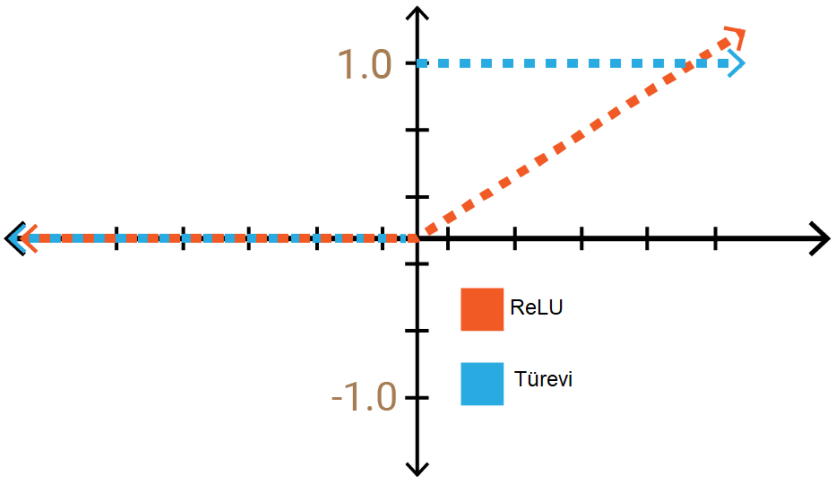
$$\sigma(x) = \frac{\sinh(x)}{\cosh(x)} \tag{3.14}$$



Şekil 3.7: Tanh fonksiyonunun ve türevinin grafiği.

4- **Rectified Linear Unit (ReLU):** Doğrusal işlevin çıktısının 0'ın üzerinde olması koşuluyla, temelde bir düğümü etkinleştirir; aksi takdirde çıktısı 0 olacaktır. Doğrusal işlevin çıktısı 0'ın üzerindeyse, bu etkinleştirme işlevinin sonucu girdi olarak aldığı ham sayı olacaktır. Genel olarak ReLU fonksiyonu gizli katmanlar için kullanılırlar. Denklem (3.15) ve Şekil 3.8'de fonksiyonun denklemi ve grafiği verilmiştir (Kızrak 2019).

$$\sigma(x) = \max(x, 0) \tag{3.15}$$



Şekil 3.8: ReLU aktivasyon fonksiyonunun grafiği ve türevi.

3.7 Loss (Kayıp, Yitim) Fonksiyonunun Hesaplanması

İleri yayılma tamamlandıktan sonra, eğitim sürecindeki bir sonraki adım, modelin tahmininin gerçek değerlere göre ne kadar iyi veya kötü olduğunu diğer bir deyişle modelin hatasını görmek için bir kayıp fonksiyonu hesaplamaktır. Bu göz önüne alındığında ulaşılması gereken ideal değer 0'dır, bu da gerçek ve tahmin değeri arasında hiçbir sapma olmadığı anlamına gelir.

Eğitim sürecindeki ağırlık ve bias değerlerinin güncelleme işlemleri bu kayıp fonksiyonunu minimize etmek için gerçekleştirilir. Aşağıda en sık kullanılan kayıp fonksiyonları maddeler halinde açıklanarak belirtilmiştir (Saleh 2020).

1- **Ortalama karesel hata (MSE):** Regresyon modellerinin performansını ölçmek için yaygın olarak kullanılan MSE fonksiyonu, gerçek değer ile tahmin değeri arasındaki farkların karelerinin toplamını hesaplar. Bu fonksiyonun denklemi Denklem (3.16)'da verilmiştir.

$$loss = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{3.16}$$

Burada n; örneklerin sayısını, y_i; gerçek değerleri, $\hat{y}_i$ modelin tahmin ettiği değerleri simgeler.

2- **Çapraz entropi (Cross-entropy):** Bu fonksiyon geleneksel olarak ikili veya çok sınıflı sınıflandırma modelleri için kullanılır ve iki olasılık dağılımı arasındaki farklılığı ölçer. Bu fonksiyonun denklemi Denklem (3.17)'de verilmiştir.

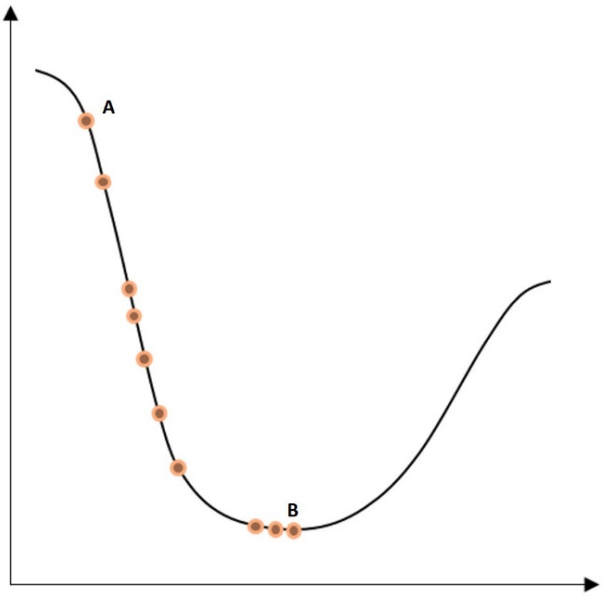
$$loss = -\frac{1}{n}\sum_{i=1}^{n} y_i * \log(\hat{y}_i) + (1 - y_i) * \log(1 - \hat{y}_i) \quad (3.17)$$

Burada yine n; örneklerin sayısını, y_i; gerçek değerleri, $\hat{y}_i$ modelin tahmin ettiği değerleri simgeler.

3.8 Geri Yönde Yayılım (Backward Propagation)

Eğitim sürecindeki son adım, ağırlık ve bias değerlerine göre kayıp fonksiyonunun kısmi türevlerini (gradyan) hesaplayarak her bir katman için ağırlık ve bias parametrelerini güncellemek için ağ yapısında sağdan sola doğru gitme işlemidir. Böylece ağ çıkışında hesaplanan kayıp fonksiyonun değeri minimize edilmiş olur.

Yerel minimum noktası, fonksiyonun alanının bir bölümündeki en küçük değeri ifade ederken, global minimum noktası fonksiyonun tüm alanının minimum değerini ifade eder. Optimizasyon algoritmasının amacı, aşağıdaki grafikte gösterildiği gibi, kayıp fonksiyonunun mümkün olan en az değere ulaştığı global minimumu bulmaktır (Saleh 2020).



Şekil 3.9: İki boyutlu uzayda yineleme adımlarıyla kayıp fonksiyonu optimizasyonu.

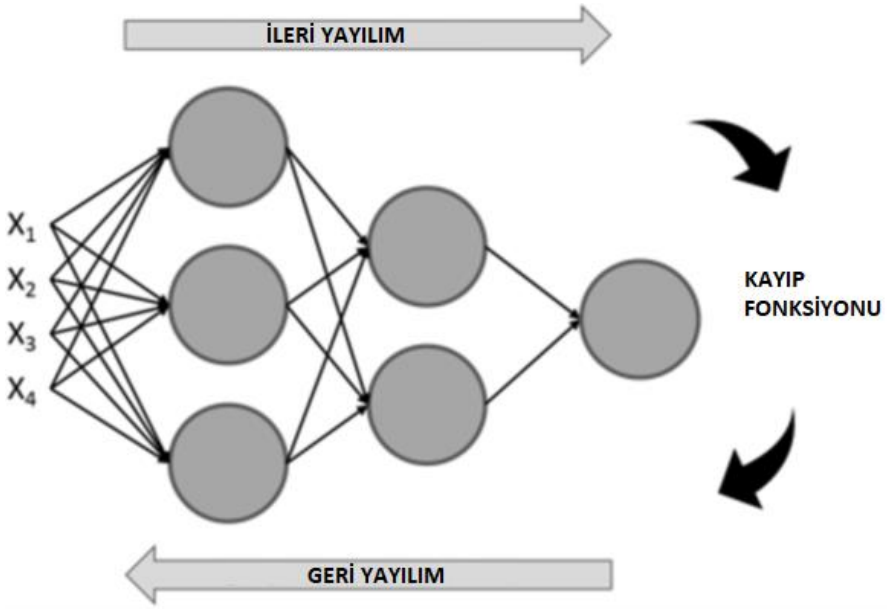
Şekil 3.9'da; en soldaki nokta olan A herhangi bir optimizasyondan önceki kayıp fonksiyonunun başlangıç değeridir (Saleh 2020). Eğrinin en altındaki B noktası ise; fonksiyonun değerinin en aza indirildiği birkaç yineleme adımından sonra kayıp fonksiyonunun minimum noktasıdır (Saleh 2020).

3.9 Yapay Sinir Ağlarında Öğrenme Süreci

Genel anlamda, bir sinir ağı birden fazla nörondan oluşur. Burada her nöron, bazı girdilere dayalı bir çıktıya ulaşmak için bir aktivasyon fonksiyonu ile doğrusal bir fonksiyonu hesaplar. Hesaplamada kullanılan aktivasyon fonksiyonu doğrusallığı bozmak için tasarlanmıştır. Hesaplanan çıktı, önem seviyesini temsil eden bir ağırlığa bağlıdır ve sonraki katmanda hesaplamalar için kullanılır.

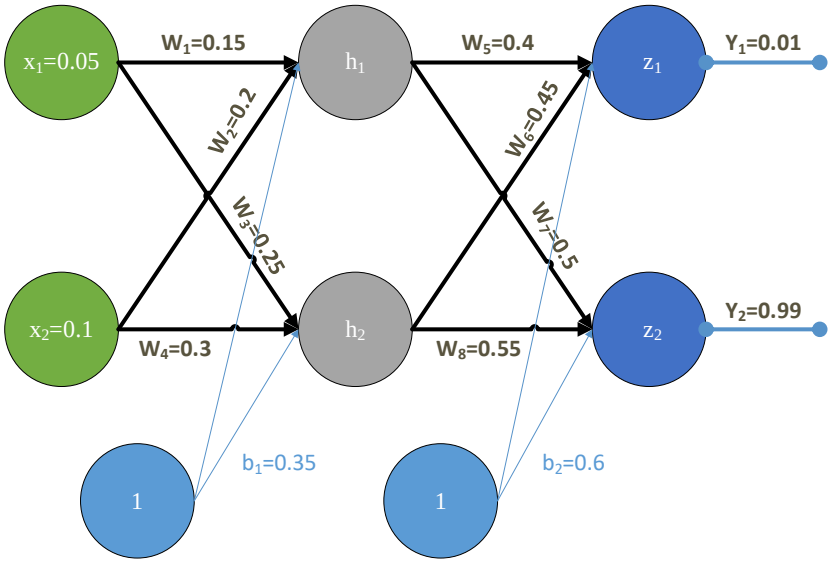
Bu hesaplamalar, nihai bir çıktıya ulaşılana kadar ağın tüm mimarisi boyunca gerçekleştirilir. Hesaplanan çıktılar, daha sonra hesaplama sürecini yeniden başlatmak ve ağın parametrelerini güncellemek için geri yayılımla kullanılırlar.

Bir sinir ağının eğitim süreci, Şekil 3.10'da gösterildiği gibi optimum sonuca ulaşmak için ağın katmanları boyunca ileri ve geri yönde işleyen yinelemeli bir süreç olarak tanımlanabilir (Saleh 2020).



Şekil 3.10: Yapay sinir ağlarının öğrenme diyagramı.

Yapay sinir ağlarında ileri ve geri yayılım sürecini Şekil 3.11'de verilen sırasıyla giriş, gizli ve çıkış katmanından oluşan ağ yapısı üzerinden açıklayacak olursak öğrenme işlemi aşağıdaki adımlarla gerçekleşir.



Şekil 3.11: Örnek iki katmanlı yapay sinir ağı yapısı.

Burada; $x = [0.05, 0.1]$ giriş verileri, $y = [0.01, 0.99]$ çıkış verileri, $w = [0.15, 0.2, 0.25, 0.3, 0.35, 0.4, 0.45, 0.5, 0.55]$ ağırlıklar ve $b = [0.35, 0.6]$ bias değerleridir.

Yapay sinir ağlarında amaç verilen giriş verileri ile yapay sinir ağı çıkışını orijinal çıkışa oldukça yaklaştırmaktır.

- **Adım 1: İleri Yayılma**
  - ***Adım 1.1:*** Giriş Katmanı → Gizli Katman

    h_1 nöronunun girdi ağırlıkları toplamı aşağıdaki hesaplanır:

$$h_1 = w_1 * x_1 + w_2 * x_2 + b_1 * 1 \tag{3.18}$$

$$h_1 = 0.15 * 0.05 + 0.2 * 0.1 + 0.35 * 1 = 0.3775 \tag{3.19}$$

    h_1 nöronunun çıkışı sigmoid aktivasyon fonksiyonu ile şu şekilde hesaplanır:

$$out_{h_1} = \sigma(h_1) = \frac{1}{1 + e^{-h_1}} = \frac{1}{1 + e^{-0.3775}} = 0.59326999 \tag{3.20}$$

    Aynı şekilde h_2 nöronunun çıkışı da hesaplanabilir:

$$out_{h_2} = \sigma(h_2) = 0.59688438 \tag{3.21}$$

  - ***Adım 1.2:*** Gizli Katman → Çıkış Katmanı

    Çıkış katmanındaki z_1 ve z_2 nöronlarının değerleri de Denklem (3.22), (3.23), (3.24) ve (3.25)'teki gibi hesaplanır:

$$z_1 = w_5 * \sigma(h_1) + w_6 * \sigma(h_2) + b_2 * 1 \tag{3.22}$$

$$z_1 = 0.4 * 0.59326999 + 0.45 * 0.59688438 + 0.6 * 1 = 1.10590596 \tag{3.23}$$

$$\hat{y}_1 = \sigma(z_1) = \frac{1}{1 + e^{-z_1}} = \frac{1}{1 + e^{-1.10590596}} = 0.75136507 \tag{3.24}$$

$$\hat{y}_2 = \sigma(z_2) = 0.772928465 \tag{3.25}$$

Bu şekilde yapay sinir ağlarında ileri doğru yayılım süreci sona ermiş olur. İleri yayılım süreci sonucunda $\hat{y} = [0.75136507, 0.772928465]$ şeklinde çıktı değerini elde ederiz fakat bu değerler hala gerçek çıktı değerleri olan $y = [0.01, 0.99]$ değerlerinden oldukça uzaktır. Çıktı değerlerini gerçek değerlere yaklaştırmak için hata geriye doğru yayılır ve ağırlıklar güncellenir.

- Adım 2: Geri Yayılım
  - Adım 2.1: Toplam Hatanın Hesaplanması

$$loss = E_{total} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{3.26}$$

Örnek olarak verilen sinir ağı yapısında iki adet çıktı olduğu için çıkış hataları ayrı ayrı hesaplanır ve toplam hata ikisinin toplamı olarak hesaplanır.

$$E_{\hat{y}_1} = \frac{1}{2}(y_1 - \hat{y}_1)^2 = \frac{1}{2}(0.01 - 0.75136507)^2 = 0.274811083 \quad (3.27)$$

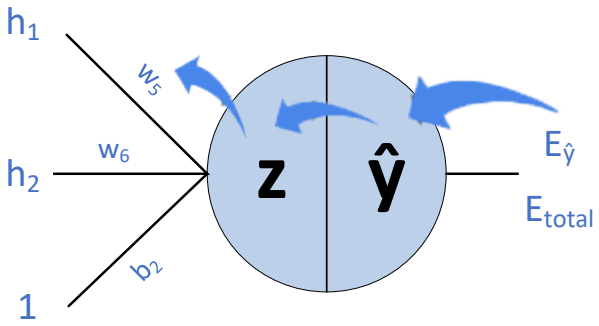
$$E_{\hat{y}_2} = 0.023560026 \quad (3.28)$$

$$E_{total} = E_{\hat{y}_1} + E_{\hat{y}_2} = 0.298371109 \quad (3.29)$$

- Adım 2.2: Gizli Katman → Çıktı Katmanının Ağırlık Güncellemesi

  Örnek olarak, ağırlık parametresi w_5'i ele aldığımızda w_5'in genel hata üzerinde ne kadar etkisi olduğunu görmek için genel hatanın w_5'e göre Şekil 3.12'de gösterildiği doğrultuda zincir kuralı ile kısmi türevi alınır.

$$\frac{\partial E_{total}}{\partial w_5} = \frac{\partial E_{total}}{\partial \hat{y}_1} * \frac{\partial \hat{y}_1}{\partial z_1} * \frac{\partial z_1}{\partial w_5} \quad (3.30)$$



Şekil 3.12: Hatanın geri yayılımı.

Denklem (3.30)'daki her bir formülün değerini ayrı ayrı hesaplarsak;

$$E_{total} = \frac{1}{2}(y_1 - \hat{y}_1)^2 + \frac{1}{2}(y_2 - \hat{y}_2)^2 \quad (3.31)$$

$$\frac{\partial E_{total}}{\partial \hat{y}_1} = 2 * (y_1 - \hat{y}_1)^{2-1} * (-1) + 0 \quad (3.32)$$

$$\frac{\partial E_{total}}{\partial \hat{y}_1} = -(y_1 - \hat{y}_1) = -(0.01 - 0.75136507) = 0.74136507 \quad (3.33)$$

$$\hat{y}_1 = \frac{1}{1 + e^{-z_1}} \quad (3.34)$$

$$\frac{\partial \hat{y}_1}{\partial z_1} = \hat{y}_1(1-\hat{y}_1) = 0.75136507(1-0.75136507) = 0.186815602 \quad (3.35)$$

$$z_1 = w_5 * \sigma(h_1) + w_6 * \sigma(h_2) + b_2 * 1 \quad (3.36)$$

$$\frac{\partial z_1}{\partial w_5} = 1 * \ \sigma(h_1) * {w_5}^{1-1} + 0 + 0 = \sigma(h_1) = 0.59326999 \quad (3.37)$$

İşlemlerden sonra Denklem (3.33), (3.35), (3.37)'deki değerler çarpılarak Denklem (3.30)'un sonucu bulunur.

$$\frac{\partial E_{total}}{\partial w_5} = 0.74136507*0.186815602*0.59326999=0.082167041 \quad (3.38)$$

Son olarak w_5'in değerini güncellersek;

$$w_5^+ = w_5 - \eta * \frac{\partial E_{total}}{\partial w_5} = 0.4 - 0.5 *0.082167041=0.35891648 \quad (3.39)$$

Denklem (3.39)'daki η değeri öğrenme oranıdır ve 0.5 olarak alınmıştır. Benzer şekilde w_6, w_7 ve w_8 güncellendiğinde Denklem (3.40), (3.41) ve (3.42)'deki sonuçlar elde edilir:

$$w_6^+ = 0.408666186 \quad (3.40)$$

$$w_7^+ = 0.511301270 \quad (3.41)$$

$$w_8^+ = 0.561370121 \quad (3.42)$$

- Adım 2.3: Gizli Katman → Gizli Katmanının Ağırlık Güncellemesi
  Yöntem aslında yukarıda bahsedilen yöntem ile aynıdır fakat güncelleme yapılırken $E_{\hat{y}_1}$ ve $E_{\hat{y}_2}$ 'den gelen hata da hesaba katılmalıdır.

$$\frac{\partial E_{total}}{\partial w_1} = \frac{\partial E_{total}}{\partial out_{h_1}} * \frac{\partial out_{h_1}}{\partial h_1} * \frac{\partial h_1}{\partial w_1} \quad (3.43)$$

$$\frac{\partial E_{total}}{\partial out_{h_1}} = \frac{\partial E_{\hat{y}_1}}{\partial out_{h_1}} + \frac{\partial E_{\hat{y}_2}}{\partial out_{h_1}} \quad (3.44)$$

$$\frac{\partial E_{\hat{y}_1}}{\partial out_{h_1}} = \frac{\partial E_{\hat{y}_1}}{\partial z_1} * \frac{\partial z_1}{\partial out_{h_1}} \quad (3.45)$$

$$\frac{\partial E_{\hat{y}_1}}{\partial z_1} = \frac{\partial E_{\hat{y}_1}}{\partial \hat{y}_1} * \frac{\partial \hat{y}_1}{\partial z_1} = 0.74136507 * 0.186815602 = 0.138498562 \quad (3.46)$$

$$z_1 = w_5 * out_{h_1} + w_6 * out_{h_2} + b_2 * 1 \tag{3.47}$$

$$\frac{\partial z_1}{\partial out_{h_1}} = w_5 = 0.4 \tag{3.48}$$

$$\frac{\partial E_{\hat{y}_1}}{\partial out_{h_1}} = 0.138498562 * 0.4 = 0.055399425 \tag{3.49}$$

Benzer şekilde Denklem (3.50), (3.51), (3.52), (3.53), (3.54), (3.55) hesaplanabilir;

$$\frac{\partial E_{\hat{y}_2}}{\partial out_{h_1}} = -0.019049119 \tag{3.50}$$

$$\frac{\partial E_{total}}{\partial out_{h_1}} = 0.055399425 - 0.019049119 = 0.036350306 \tag{3.51}$$

$$out_{h_1} = \frac{1}{1 + e^{-h_1}} \tag{3.52}$$

$$\frac{\partial out_{h_1}}{\partial h_1} = out_{h_1}(1 - out_{h_1}) = 0.241300709 \tag{3.53}$$

$$h_1 = w_1 * x_1 + w_2 * x_2 + b_1 * 1 \tag{3.54}$$

$$\frac{\partial h_1}{\partial w_1} = x_1 = 0.05 \tag{3.55}$$

Son olarak Denklem (3.51), (3.53) ve (3.55) de bulunan değerler çarpılır.

$$\frac{\partial E_{total}}{\partial w_1} = 0.036350306 * 0.241300709 * 0.05 = 0.000438568 \tag{3.56}$$

w_1'in ağırlığını güncellediğimizde de;

$$w_1^+ = w_1 - \eta * \frac{\partial E_{total}}{\partial w_1} = 0.15 - 0.5 * 0.000438568 = 0.149780716 \tag{3.57}$$

Benzer şekilde Denklem (3.58), (3.59), (3.60) hesaplanabilir;

$$w_2^+ = 0.19956143 \tag{3.58}$$

$$w_3^+ = 0.24975114 \tag{3.59}$$

$$w_4^+ = 0.29950229 \tag{3.60}$$

Güncellenmiş ağırlık değerleri bulunur. Bu şekilde hatanın geri yayılımı tamamlanmış olur. Bu işlemlerden sonra güncellenmiş ağırlıklarla tekrar ileri yayılım gerçekleştirilir ve bu işlem sürekli olarak yinelenir. Bu örnekte ilk yinelemeden sonra toplam hata 0,298371109'dan 0,291027924'e düşer. 10000 yinelemeden sonra ise toplam hata 0.000035085'e düşerek, ağın çıkışları olan $\hat{y}_1$ ve $\hat{y}_2$ değerleri $[0.015912196, 0.984065734]$ olarak bulunur. Bulunan bu değerler ise gerçek değerler olan $y = [0.01, 0.99]$ değerlerine oldukça yakındır.

4. DERİN ÖĞRENME

Yapay zekanın ilk günlerinde, insanlar için entelektüel olarak zor olan, ancak bilgisayarlar için nispeten basit olan, biçimsel ve matematiksel kurallar listesiyle tanımlanabilecek problemleri hızla çözmüştür. Yapay zeka için asıl zorluğun, insanların gerçekleştirmesi kolay ancak matematiksel bir şekilde tanımlaması zor olan görevleri, sezgisel olarak çözdüğümüz, resimlerdeki kelimeleri, yüzleri veya nesneleri tanıma gibi insanların duyusal olarak günlük hayatta sürekli çözdüğü problemler olduğu ortaya çıkmıştır (Goodfellow ve diğ. 2018).

Makine öğrenimi, genellikle yapay zekanın bir alt disiplini olarak tanımlanırken, en iyi disiplindeki en iyi teknik olarak görülür. Başka bir deyişle, bugün endüstrinin ve toplumun değişimi etkilemek için kullanabileceği araçlar sağlayarak en büyük vaatleri gösteren yapay zeka alanıdır (Goodfellow ve diğ. 2018). Bu anlamda, makine öğrenimi, yapay zekanın temel fikirlerinden bazılarını alır ve bunları, kendi karar verme sürecimizi taklit etmek için tasarlanmış sinir ağları ile gerçek dünyadaki sorunları çözmeye odaklanır. Bununla birlikte, derin veya gelişmiş makine öğrenimi (derin öğrenme), makine öğrenimi araçlarının ve tekniklerinin bir alt kümesine daha dar bir şekilde odaklanır ve düşünme gerektiren hemen hemen her sorunu çözmeye çalışır (Goodfellow ve diğ. 2018).

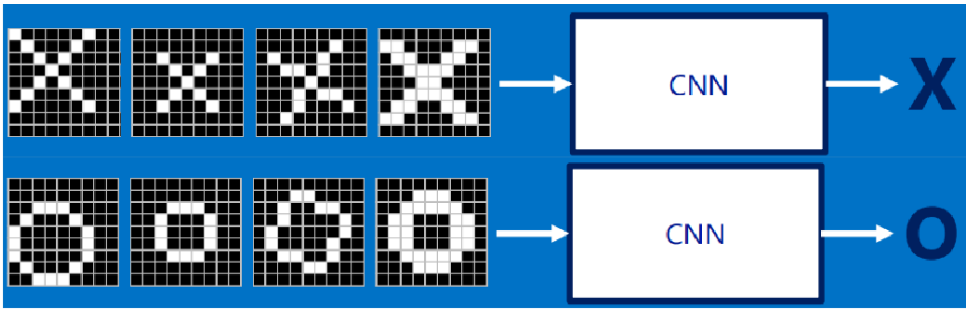
Derin öğrenme, oldukça umut verici kararlar almak için sinir ağları oluşturmak ve eğitmek için kullanılan özel bir yaklaşımdır. Girdi verileri, çıktı verilmeden önce bir dizi doğrusal olmayan katmanlı dönüşümden geçirilirse, bir algoritma derin olarak kabul edilir. Bu yaklaşım, bilgisayarların deneyimlerden öğrenmesine ve dünyayı kavramlar hiyerarşisi açısından anlamasına olanak tanır ve her bir kavram, daha basit kavramlarla ilişkisi aracılığıyla tanımlanır. Deneyimden elde edilen bilgileri bir araya getiren bu yaklaşım, insan operatörlerin bilgisayarın ihtiyaç duyduğu tüm bilgileri resmi olarak belirtme ihtiyacını ortadan kaldırır. Kavram hiyerarşisi, bilgisayarın daha basit kavramlar oluşturarak karmaşık kavramları öğrenmesini sağlar (Goodfellow ve diğ. 2018).

4.1 Evrişimli Sinir Ağları

Derin öğrenmenin yeni bir teknoloji engelini aştığını duyduğunuzda, onda dokuzunda evrişimli sinir ağları devreye girer. CNN (Convolutional Neural Networks) veya ConvNets olarak da adlandırılan bu yapılar, derin sinir ağları alanının favorileridir. Bazı durumlarda görüntüleri insanlardan daha iyi sınıflandırmayı öğrenebilirler (Rohrer 2016).

Evrişimli ağ modelleri her zaman, girdilerin, tasarlanacak modelin özelliklerinin veya özelliklerinin ilk bölümünü yapılandırmamıza izin veren görüntüler biçiminde eşlenmesi gerektiğini varsayar. Bu özellik nedeniyle, CNN'in sınırlaması, yalnızca verilerdeki yerel uzamsal kalıpları yakalamalarıdır. Yani, verilerin bir görüntü gibi görünmesi sağlanamıyorsa, CNN pratikte çok kullanışsızdır (Rohrer 2016).

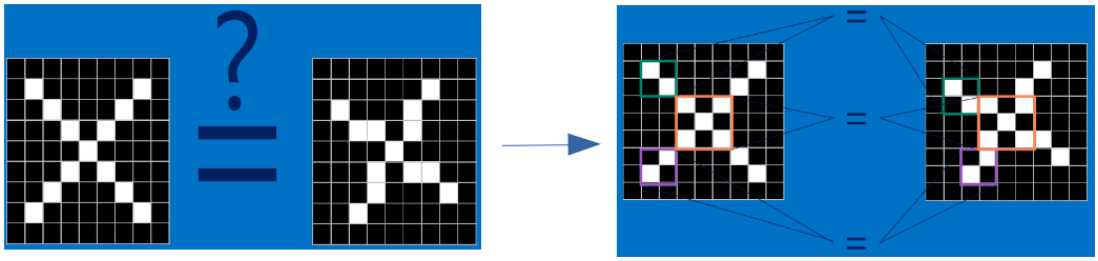
Evrişimli sinir ağları, sıradan sinir ağlarına çok benzer, eğitimleri sırasında bilgi öğrenebilen ağırlıkları ve bias değerleri olan nöronlardan oluşurlar. Her nöron bazı girdiler alır ve matematiksel işlemler gerçekleştirir. Tüm ağ, tek bir farklılaştırılabilir puanlama fonksiyonunu ifade eder: bir uçtaki (giriş) görüntü piksellerinin içeriğinden diğer uçtaki (çıktı) karşılık gelen sınıfı veya sonucu tanımlayan puanlayan bir ağ modelidir. CNN'in basit bir işlevi Şekil 4.1'de gösterilmiştir (Rohrer 2016).



Şekil 4.1: CNN girişi ve çıkışı.

CNN'lerin sonucu, bir özelliğin bir görüntüde olup olmadığını, tam olarak nerede olduğu konusunda endişelenmeden öğrenebilmeleridir. Bu, görüntüleri hiper-literal bir şekilde karşılaştırırken, yani piksel piksel eşleştirerek eşit görüntü olmaları

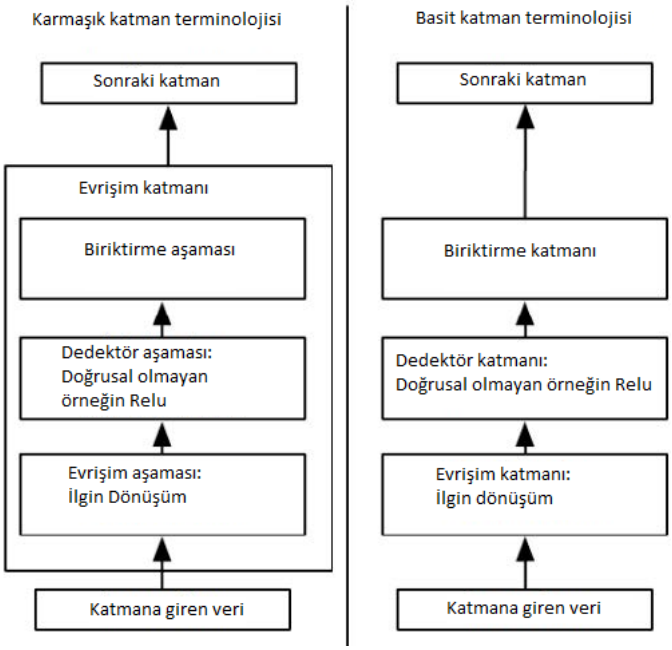
için bilgisayarların sorununu çözmeye yardımcı olur. CNN'in bu sorunları çözerken yaptığı analiz Şekil 4.2'de verilmiştir (Rohrer 2016).



Şekil 4.2: Bir CNN'in analizi.

Bir CNN, bu katmanların her birinin fonksiyonlar yoluyla bir aktivasyon hacmini yenisine dönüştürdüğü bir katman dizisine sahip olmasıyla karakterize edilir ve bu katmanlardan birkaçı söz konusu katmanlar içindeki ve arasındaki yapılandırma parametrelerini değiştirerek kullanıldığında derin öğrenme gerçekleşir.

Şekil 4.3'te gösterildiği gibi bu katmanları tanımlayan iki grup terminoloji vardır (Goodfellow ve diğ. 2018). Birincisi, evrişimli ağın çok sayıda basit katman olarak görülmesi ve her işlem adımının kendi başına bir katman olarak görülmesidir. Diğer bir terminoloji, evrişimli ağın, her katmanın birden çok aşamaya sahip olduğu az sayıda nispeten karmaşık katman olarak görülmesidir. Bu terminolojide, aktivasyon hacimleri ve ağ katmanları arasında doğrudan bir eşleştirme vardır.



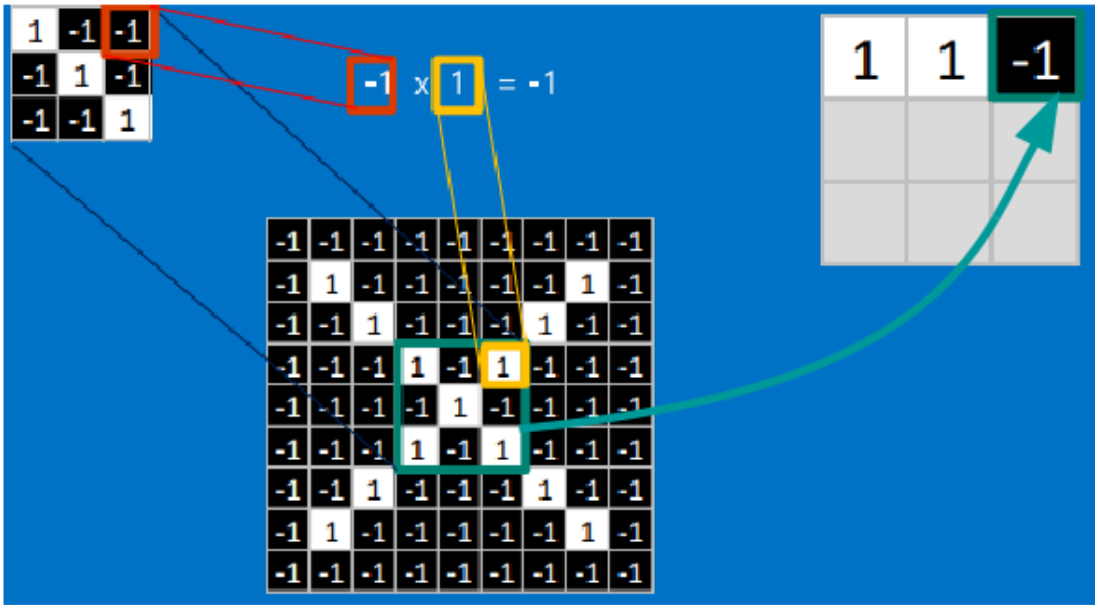
Şekil 4.3: a) Karmaşık katmanlar ve b) basit katmanlar terminolojisi.

4.1.1 Evrişim Katmanı

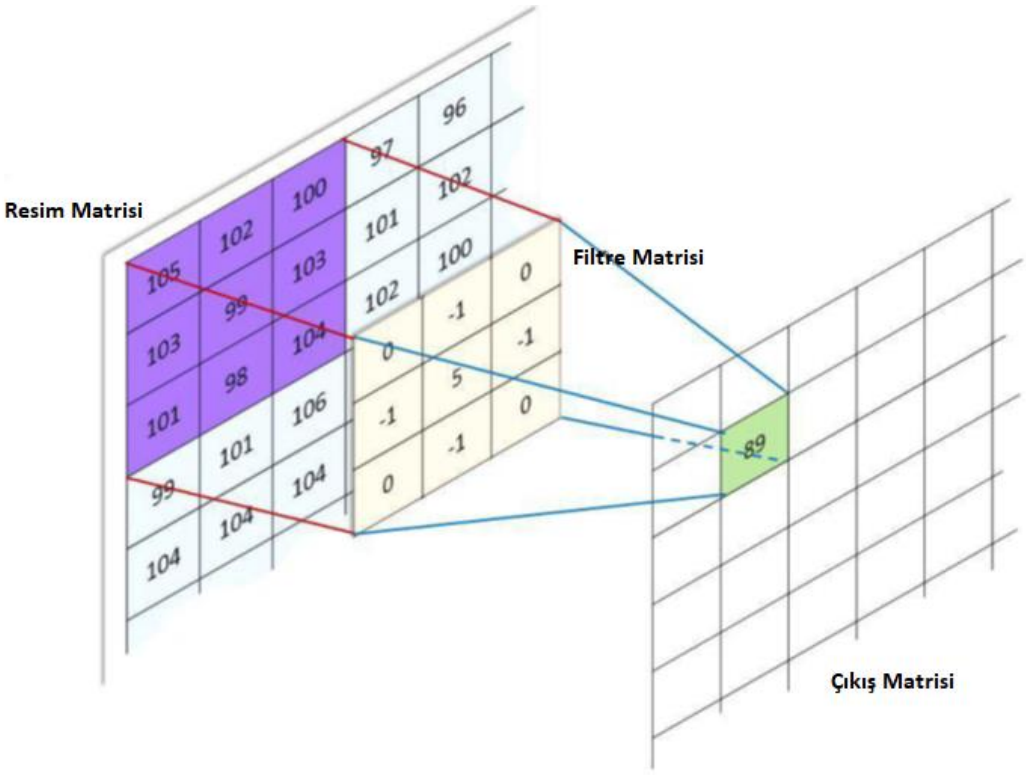
Evrişimli katmanın parametreleri temelde iki veri parçasından oluşur. Girdi ve değerleri öğrenilen bir dizi filtreyle (çekirdekler olarak da adlandırılır) ilgili her şey, yani rastgele verilerle başlar ve eğitim ilerledikçe değiştirilir.

4.1.1.1 Evrişim Aşaması

Evrişimli süreç için her filtre boyut olarak küçüktür (genişlik ve yükseklik olarak), hatta giriş boyutunun (görüntü) toplam derinliği boyunca uzanır. Örneğin, bir ConvNet'in ilk katmanındaki tipik bir filtre 5x5x3 boyutunda olabilir (yani, 5 piksel genişliğinde ve yüksekliğinde ve renk kanalları nedeniyle 3 katman derinliğinde - RGB). İleri yayılım sırasında, her bir filtre, herhangi bir pozisyondaki filtre girişleri ile giriş arasındaki noktaları hesaplamak için giriş hacminin genişliği ve yüksekliği (eşit derinlik) boyunca kaydırılır. Evrişim işleminin temsili Şekil 4.4'te gösterilmiştir (Rohrer 2016). Filtre matrisinin konumlandırılması ise Şekil 4.5'te gösterilmiştir (Nigam 2018).

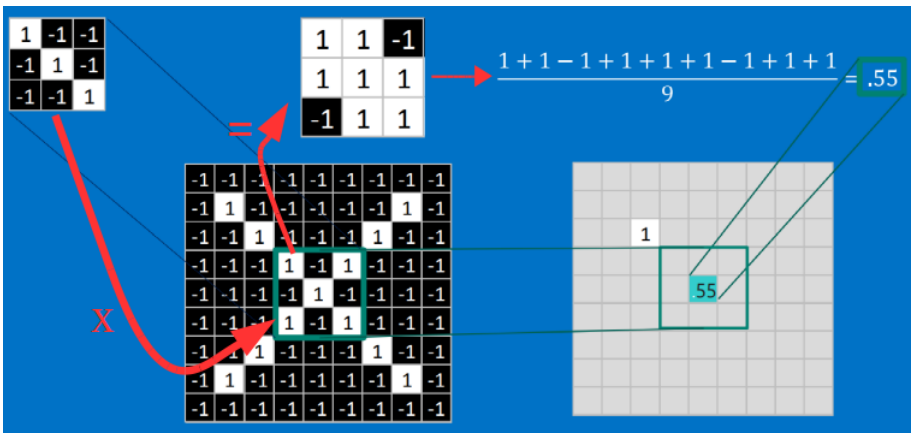


Şekil 4.4: Filtre ve görüntünün parçası arasında evrişim.

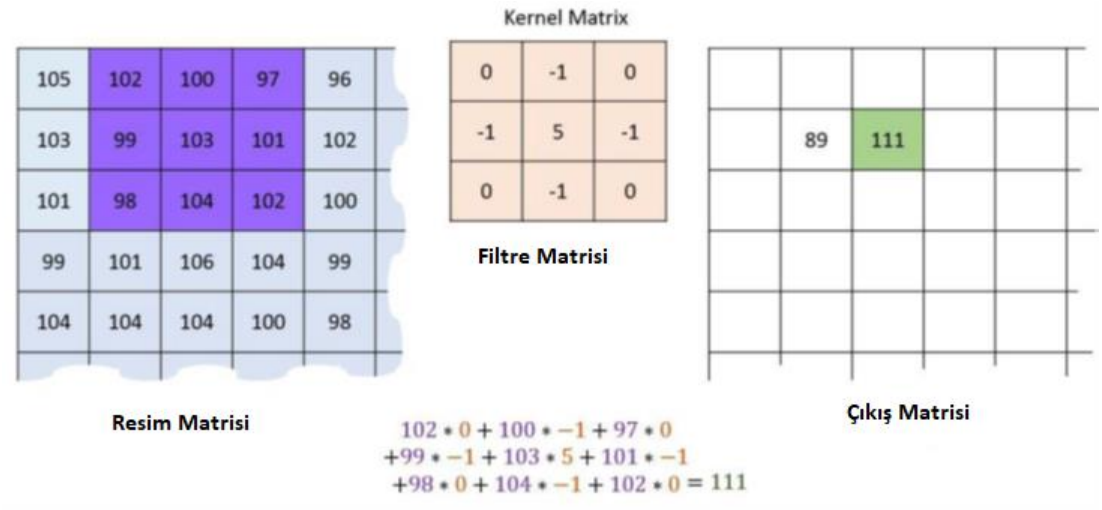


Şekil 4.5: Piksel başına kernel / filtre konumlandırma.

Filtreyi giriş boyutunun genişliği ve yüksekliği boyunca kaydırıldığında, her bir uzaysal konumda o filtrenin yanıtlarını sağlayan iki boyutlu bir aktivasyon haritası oluşturulur. Diğer bir deyişle, bu katmandaki süreç, bir filtrenin görüntünün bir kısmı ile çakışmasının hesaplanmasından ibarettir ve bunu başarmak için filtredeki her bir piksel basitçe görüntüdeki pikselin değeri ile çarpılır. Ardından çıkan değerler toplanır ve filtredeki toplam piksel sayısına bölünür. Bu işlem Şekil 4.6'da gösterilmiştir (Rohrer 2016).

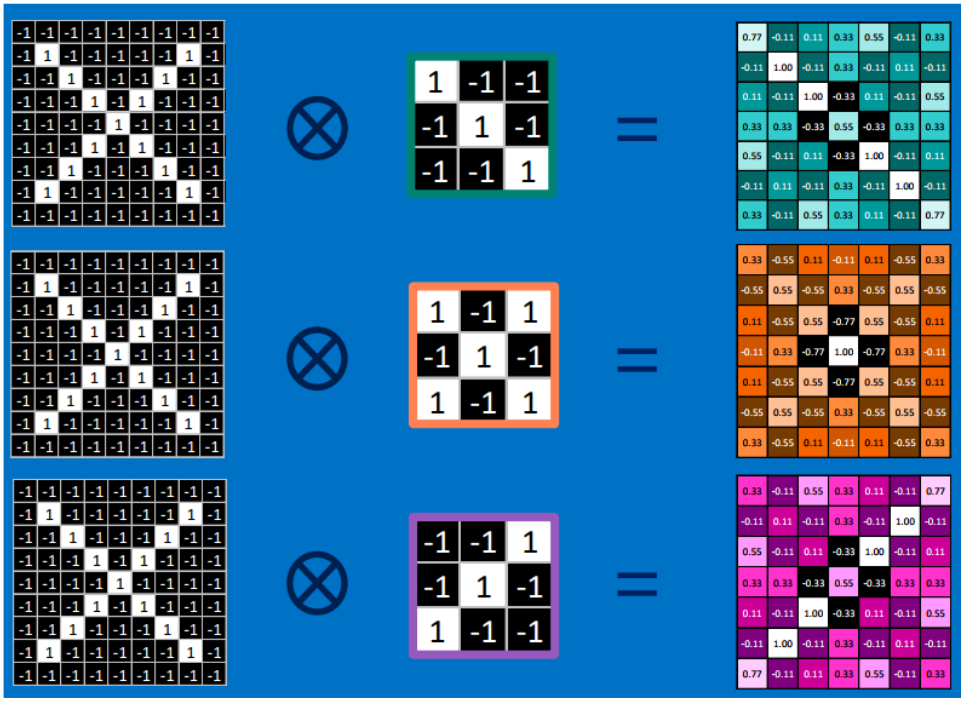


Şekil 4.6: Evrişim işlemindeki matematiksel süreç.



Şekil 4.7: Evrişim işleminin hesaplanması.

Her filtre, değeri Şekil 4.7'de gösterildiği gibi ağırlıkların toplamı kullanılarak bulunan bir çıkış nöronu olarak temsil edilebilir (Ippolito 2020). Ağ, sezgisel olarak, belirli bir yönelimde bir sınır veya ana hat gibi bir tür görsel özellik gördüklerinde etkinleştirilen filtreleri öğrenecektir. Şekil 4.8'de 3 özellik için 3 filtreden oluşan bir evrişim işlemi gösterilmiştir (Rohrer 2016).



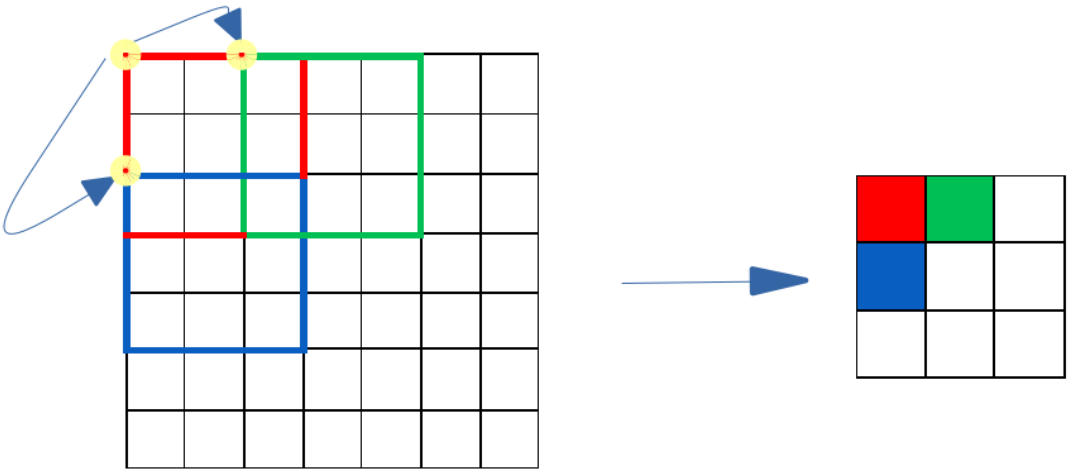
Şekil 4.8: Evrişim işleminin sonucu.

Kısacası, her evrişimli katmanda (katmanın derinliğini belirleyen) eksiksiz bir "n" filtre seti bulunur ve bu filtrelerin her biri ayrı bir iki boyutlu aktivasyon haritası oluşturur. Bu etkinleştirme haritalarını derinlik boyutu boyunca biriktirilir ve çıktı hacmini üretilir, yani sonuç, Şekil 4.8'de görülebileceği gibi, orijinal görüntünün önemli özelliklerini veya modellerini vurgulayan filtrelenmiş bir versiyonunu gösteren bir dizi görüntüdür.

Bir filtre veya çekirdeğin inşası için, üç önemli noktayı göz önünde bulundurmak gerekir: uzaysal kapsam, adım (stride) ve doldurulacak sıfır miktarı (zero padding).

1. Uzaysal kapsam, filtrenin boyutudur, genellikle hem uzunluk hem de genişlik açısından boyut olarak farklıdır.

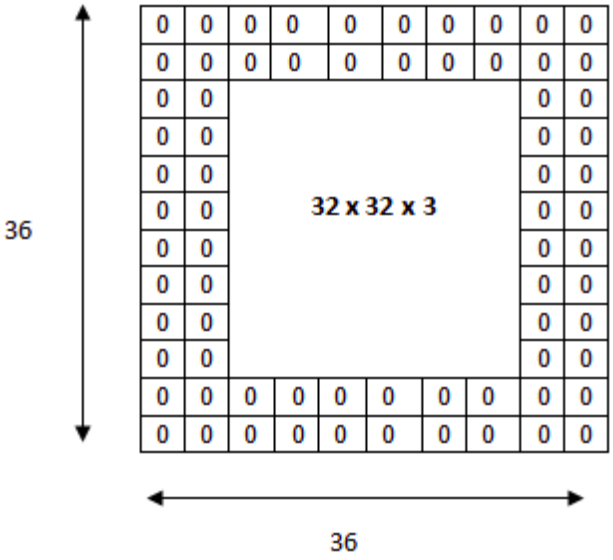
2. Adım, evrişimli filtrelerin temel yapı taşının başka bir parçasıdır. Bu, her adımda bir görüntüde bir filtrenin ne kadar kaydırılması gerektiğini gösteren evrişim işlemindeki 'adımı' temsil eder. Filtre görüntünün üzerinde kayar, her atlama uzunluğunda durur ve o adımda gerekli işlemleri gerçekleştirir. 2 adımlık filtre uygulamasının temsili Şekil 4.9'da gösterilmiştir (Deshpande 2016).



Şekil 4.9: Hem uzunluk hemde genişlik için 2 adımlık filtre uygulaması.

3. Sıfır dolgu Şekil 4.10'daki, görüntünün kenarlarının çevresine sıfırlar ekler (Deshpande 2016). Sıfır dolgunun başlıca faydaları şunlardır:

    a. Hacimlerin yüksekliğini ve genişliğini zorunlu olarak azaltmadan bir evrişim katmanı kullanmanıza izin verir. Bu, daha derin ağlar oluşturmak için önemlidir, aksi takdirde daha derin katmanlara geçerken yükseklik / genişlik azalır.
    b. Bir görüntünün kenarında daha fazla bilgi tutmamıza yardımcı olur. Dolgu olmadan, bir sonraki katmandaki çok az değer, bir görüntünün kenarları gibi piksellerden etkilenir.



Şekil 4.10: 2 boyutlu sıfır dolgu örneği.

Her bir evrişim katmanı girdi olarak şu parametreleri alır: W_0, H_0, D_0

Aşağıdaki formüllerle bir çıktı verisi üretir: W_1, H_1, D_1

Bu çıkışlar, filtrelerin yapılandırılma şekillerinden etkilenirler.

$$W_1 = \frac{W_0 - F + 2P}{S} + 1 \tag{4.1}$$

$$H_1 = \frac{H_0 - F + 2P}{S} + 1 \tag{4.2}$$

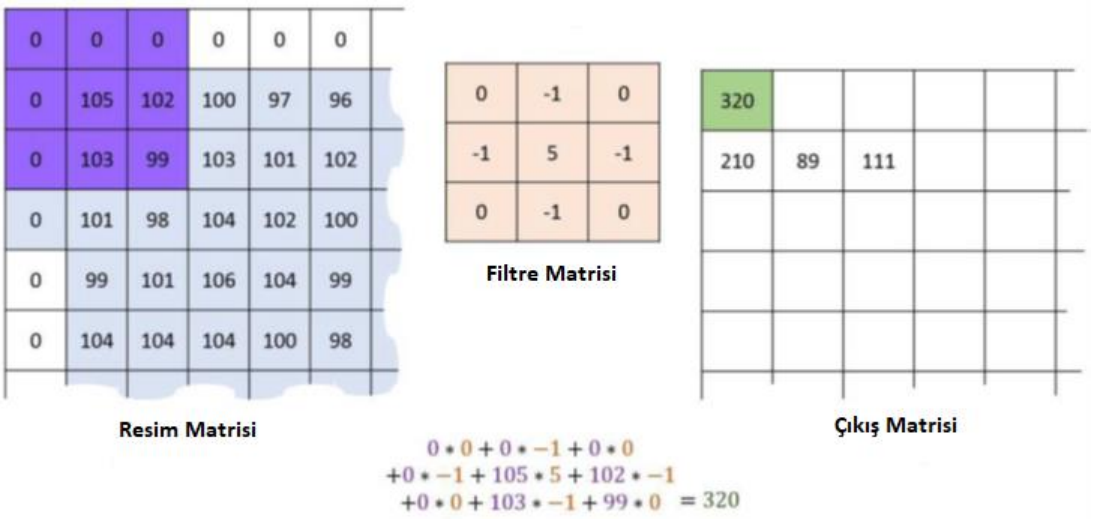
$$D_1 = K \tag{4.3}$$

Denklem (4.1), (4.2) ve (4.3)'de W görüntünün genişliğini, H görüntünün yüksekliğini, D görüntünün derinliğini, F filtrenin boyutunu, S filtrenin adımını, P filtrenin dolgu miktarını ve K filtre sayısını temsil eder.

Genelleştirilmiş bir şekilde evrişim işleminin denklemi aşağıdaki şekilde ifade edilebilir.

$$conv_j^n = \sum_{k=1}^{k} x_k^n * w_{kj}^n + b_n \tag{4.4}$$

Denklem (4.4)'te; x, w, b sırasıyla girdi değerlerini, ağırlıkları ve bias değerlerini temsil eder. n katman numarası, j çıkış filtresinin numarası ve k $(n-1)$. veya n. katmandaki filtre sayısıdır. Evrişim işlemi Şekil 4.11'de gösterilmiştir (Kava 2018).



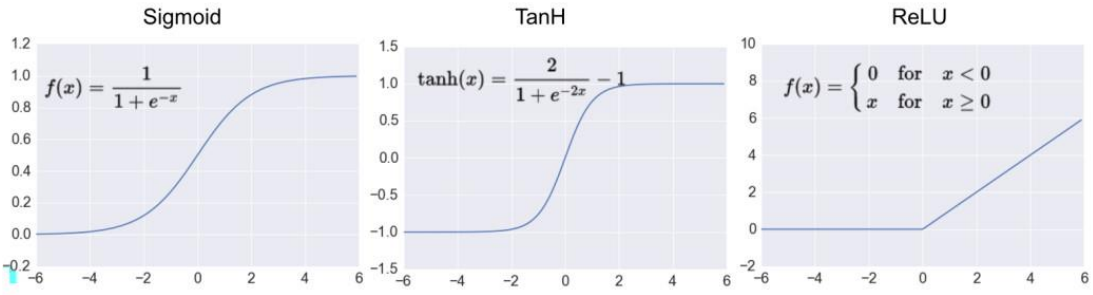
Şekil 4.11: 3x3'lük filtre ile evrişim işlemi.

4.1.1.2 Aktivasyon Katmanı

Bir sinir ağındaki tüm katmanların doğrusal olmaması nedeniyle, sinir ağındaki nöronların her birinin değerlerini hesapladıktan sonra, bu değerler bir aktivasyon fonksiyonundan geçirilir. Yapay bir sinir ağı temelde matrislerin çarpılması ve eklenmesinden oluşur. Sadece bu doğrusal hesaplamaları kullanıyor olsaydık, onları birbirinin üzerine istifleyebilirdik ve bu çok derin bir ağ olmazdı. Bu nedenle, doğrusal olmayan aktivasyon fonksiyonları genellikle ağın her katmanında kullanılır. Doğrusal ve doğrusal olmayan fonksiyonların katmanlarını üst üste yığarak, herhangi bir problemi teorik olarak modelleyebiliriz.

Derin ağ modellerinde en popüler üç aktivasyon fonksiyonu aşağıdaki fonksiyonlardır ve grafikleri Şekil 4.12'de verilmiştir.

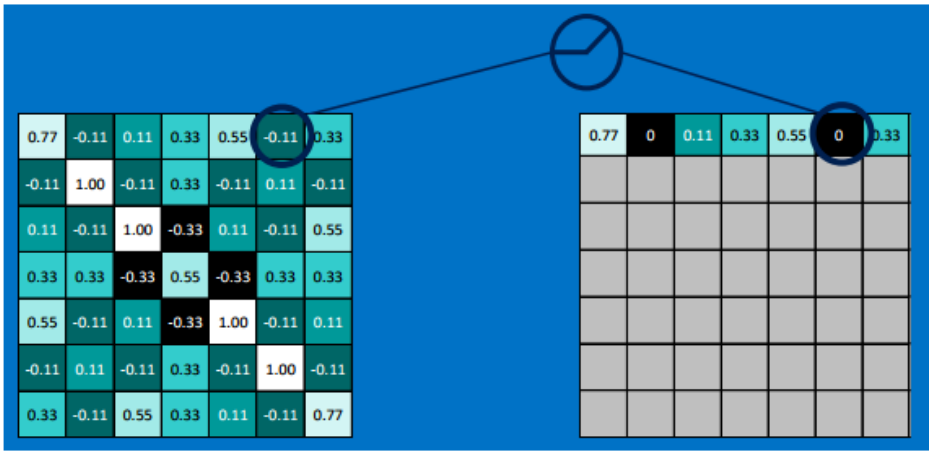
1- Sigmoid: 0 ile 1 arasındaki değerleri analiz eder.
2- Tanh: -1 ile 1 arasındaki değerleri analiz eder.
3- Relu: Değer negatifse 0 olur, aksi halde aynı kalır.



Şekil 4.12: En popüler üç aktivasyon fonksiyonunun grafikleri.

Şu anda, ReLU en çok kullanılan doğrusal olmayan aktivasyon fonksiyonudur. Bunun ana nedeni, ağın doğrulukta önemli bir fark yaratmadan çok daha hızlı eğitilebilmesidir (hesaplama verimliliği nedeniyle). Aynı zamanda, optimizasyon gradyanı katmanlar boyunca katlanarak azaldığı için, ağın alt katmanlarının çok yavaş çalıştığı, sorun olan sızıntı gradyan problemini hafifletmeye yardımcı olur. ReLU katmanı, $f(x) = \max(0, x)$ fonksiyonunu ağ girişindeki tüm değerlere uygular. Temel olarak, bu katman basitçe tüm negatif ve sıfır olan tetikleyicileri 0'la değiştirir. Bu katman, ağın doğrusal olmayan özelliklerini artırır. Sıfırdan küçük veya sıfıra eşit değerlerin aktivasyon fonksiyonundaki girişlerin sıfıra

eşitlenmesi, yukarıda tartışıldığı gibi, daha değerli kabul edilen dağınık temsiller üreten gizli birimlerde dağılmaya neden olur (Nair ve Hinton, 2010). ReLU aktivasyon fonksiyonun prosedürü Şekil 4.13'te gösterilmiştir (Rohrer 2016).



Şekil 4.13: ReLU aktivasyon fonksiyonu prosedürü.

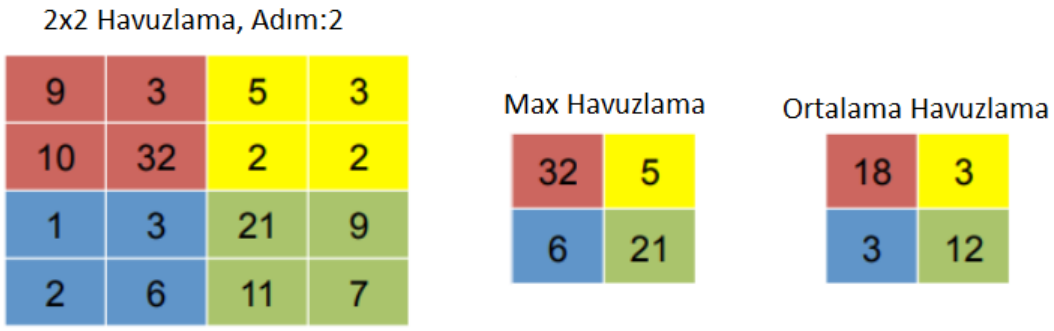
Bu nedenle evrişim katmanı ile havuz katmanı arasında, sigmoidal aktivasyon fonksiyonundan türeyen Rectified Linear Unit Function adı verilen bir aktivasyon fonksiyonuna sahip, ancak bundan daha büyük avantajları olan ReLU katmanı bulunabilir. ReLU aktivasyonları sigmoidlerden daha kolay işlenebilir, bu onları dropout tekniğiyle kombinasyon halinde kullanılmak üzere hazırlar.

Dropout tekniği kısaca aşırı öğrenmeyi engellemek için bazı nöronları unutmak olarak açıklanabilir. Genellikle büyük ağlarda ve uzun süren eğitimlerde kullanılır. Fakat özellikle evrişimli sinir ağlarının özellik çıkarımı kısmında bu işlem uygulandığında, özellikleri tutan nöronlardan bazıları görmezden gelindiğinden ağın test başarısı düşmektedir. Bu sebeple bu yöntem eğitim sırasında pek fazla tercih edilmemektedir.

4.1.1.3 Havuzlama (Pooling) Katmanı

Bir özellik haritasının boyutunu azaltan çeşitli hesaplamalar, havuzlama olarak bilinir. Her kanala ayrı ayrı uygulanan havuzlama işlemi, ağın sağlam ve küçük değişikliklere ve bozulmalara karşı değişmez olmasını sağlar.

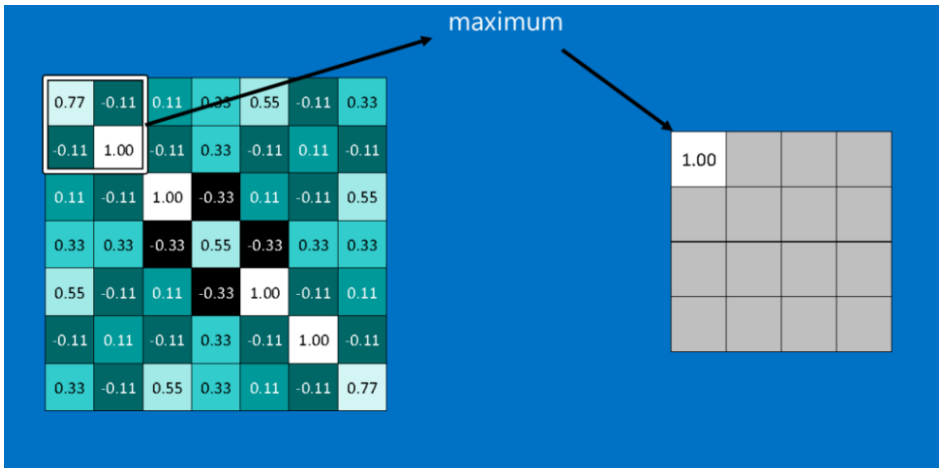
Havuzlama katmanı (çözünürlük azaltma katmanı olarak da bilinir), alıcı alanında daha az değer üreten bir dizi değeri birleştirir veya gruplandırır. Alıcı alanınızın boyutuna (örn. 2 x 2) ve Şekil 4.14'te gösterildiği gibi gruplama işlemine göre yapılandırılabilir. Tipik olarak havuzlama işlemi (max-pooling veya average-pooling), üst üste binmeyen belirli adımlarla kaydırılan bloklarda meydana gelir (Jia ve diğ 2014).



Şekil 4.14: Max ve Average Pooling işlemi.

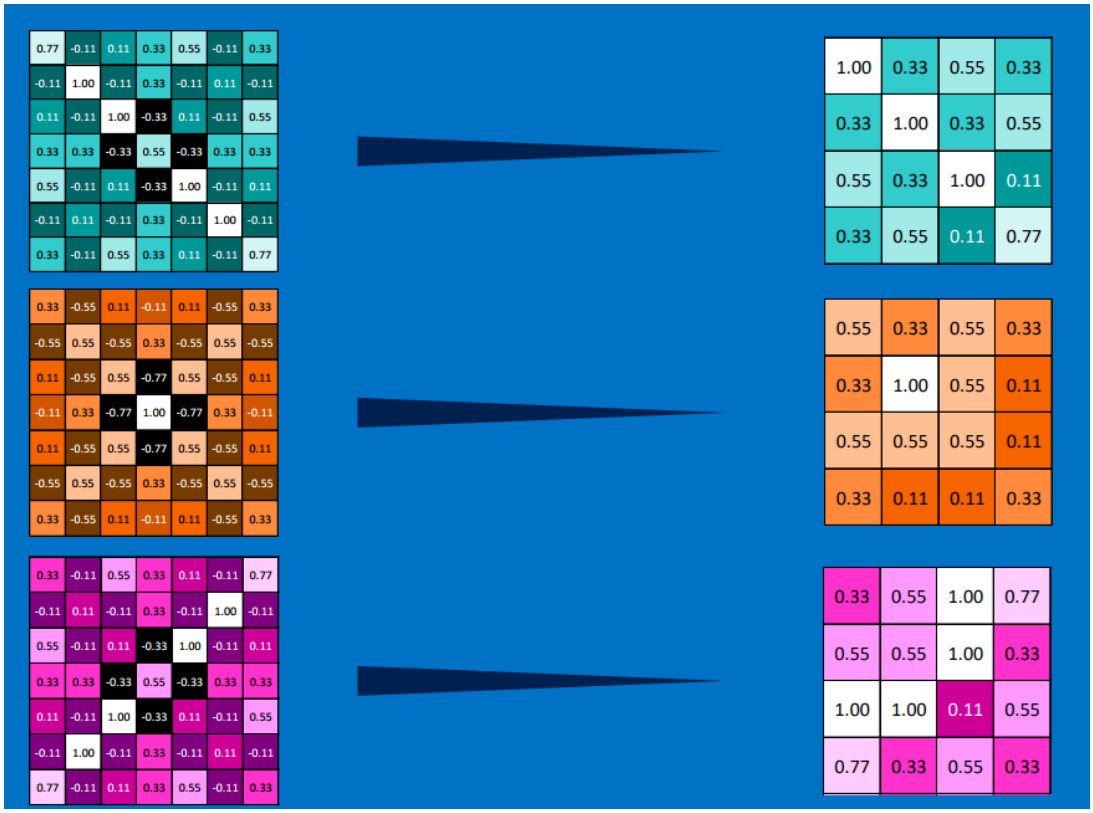
Birkaç ReLU katından sonra, bir gruplama katının uygulanması yaygındır. Bu kategoride, MAX işlemi (Max-pooling olarak bilinir) en popüler olanıdır, içindeki en önemli bilgileri korurken bir görüntü temsilinin (özellik haritası) uzaysal boyutunu aşamalı olarak küçültür. Bu işlem iki amaç için gerçekleştirilir. Birincisi, parametre sayısını azaltmak ve ağdaki hesaplamayı azaltarak işlem süresinde tasarruf sağlamaktır. İkincisi ise ağın öğreniminde büyük önem taşıyan aşırı öğrenmeyi engellemektir (Rohrer 2016).

Havuzlama katmanı, girdinin her derinlik bölümünde bağımsız olarak çalışır ve onu uzamsal olarak yeniden boyutlandırır (çözünürlüğünü azaltır). Matematiksel süreç, küçük bir pencereyi (kernel) bir görüntüden geçirip her adımda pencerenin maksimum değerini almaktan oluşur. Bu işlem Şekil 4.15'te gösterilmiştir (Rohrer 2016).



Şekil 4.15: Havuzlama katmanındaki Max Havuzlama işlemi.

Bu işlem, her aktivasyon haritası için uygulanır (evrişim katmanının çıkışı). Evrişim katmanına benzer şekilde, bu katmandaki sonuç, girdi görüntülerinin gruplanmış bir versiyonunu gösteren bir dizi görüntüdür. Havuzlama işleminin sonucu Şekil 4.16'da gösterilmiştir (Rohrer 2016).



Şekil 4.16: Havuzlama işleminin sonucu.

Bir havuzlama katmanı, girdi olarak şu parametreleri alır: W_0, H_0, D_0

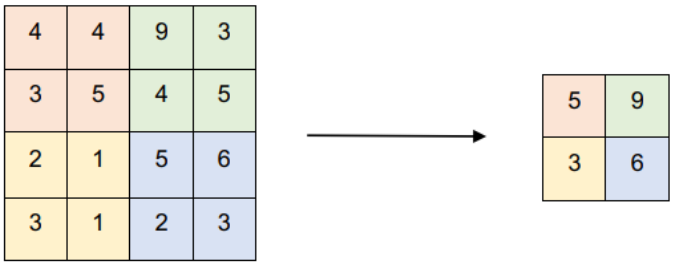
Aşağıdaki formüller ile de bir çıktı verisi üretir: W_1, H_1, D_1

$$W_1 = \frac{W_0 - F}{S} + 1 \tag{4.5}$$

$$H_1 = \frac{H_0 - F}{S} + 1 \tag{4.6}$$

$$D_1 = D_0 \tag{4.7}$$

Denklem (4.5), (4.6) ve (4.7)'de; W görüntünün genişliğini, H görüntünün yüksekliğini, D görüntünün derinliğini, F kernelin boyutunu, S ise adım boyutunu temsil eder. Max-Pooling işleminin sonucu Şekil 4.17'de gösterilmiştir.



Şekil 4.17: 2 adımlı Max-Pooling işlemi.

4.1.2 Tam Bağlı (Fully-Connected) Katman

Veriler evrişim katmanlarından geçtikten sonra, sonunda yeterince küçük bir özellik haritası oluşur ve içerik tek boyutlu bir vektöre sıkıştırılır. Bu noktaya kadar, veriler sınıf tanımada belirli bir kesinlik derecesi için zaten yeterlidir. Bununla birlikte, karmaşıklık ve hassasiyet açısından daha iyi bir sonuç elde etmek için, modelin sonunda, çıkış nöronunun tüm nöronlara bağlandığı çok katmanlı sinir ağlarına benzer şekilde, tamamen bağlı bir nöral katman kullanılacaktır. Bağlantıların girdisi ve ağırlığı, geri yayılım yöntemi kullanılarak güncellenir (SuperDataScience 2018).

Evrişim katmanlarından elde edilen sonuç vektörü, sınıflandırıcının tahmin olasılığını temsil eden belirlenmiş bir çıktı elde etmenin mümkün olacağı, tamamen

bağlı çok katmanlı sinir ağının girdisi olacaktır. Bu, evrişim sırasında kullanılandan farklı bir aktivasyon fonksiyonu kullanılarak elde edilecektir (ReLU fonksiyonu). Bu yeni fonksiyon, modelin bu son katmanında yaygın olarak kullanılan softmax fonksiyonu olarak bilinir (Krizhevsky ve diğ. 2012).

4.1.3 Eğitim ve Doğrulama

Ağ, gizli katmanlardaki ağırlıkları, biasları ve fonksiyonları kullanarak eğitim verilerindeki kayıtları işler, ardından elde edilen çıktıları istenen çıktılarla karşılaştırır. Hatalar sisteme yayılır ve sistemin bir sonraki eğitim yinelemesinde işlenecek ağırlıkları ve biasları ayarlamasına neden olur. Bu süreç, ağırlıklar sürekli olarak ayarlandığından bir veri setinde tekrar tekrar gerçekleşir. Bunu yapmak için, Gradient Descent adlı çok popüler bir optimizasyon yöntemi vardır.

4.1.3.1 Gradyan İniş

Bir gradyan, girdiler biraz değiştirilirse bir fonksiyonun çıktısının ne kadar değiştiğini ölçer.

Bir sinir ağının eğitilmesi durumunda, gradyan, hatadaki değişime göre tüm ağırlıklardaki değişikliği basitçe ölçer. Gradyan, bir fonksiyonun eğimi olarak temsil edilebilir, eğim ne kadar yüksekse, eğim o kadar diktir ve bir model o kadar hızlı öğrenebilir. Ancak eğim sıfırsa, model öğrenmeyi durdurur. Matematiksel olarak ifade edildiğinde gradyan, girdilerine göre kısmi bir türevdir (Dong ve Zhou 2008).

Gradyan inişi, bir dışbükey işleve dayalı bir makine öğrenimi modelini eğitirken kullanılan ve kayıp fonksiyonunu (hata) yerel minimuma indirmek için parametrelerini yinelemeli olarak ayarlayan yinelemeli bir optimizasyon algoritmasıdır (Dong ve Zhou 2008).

Gradyan inişinin arkasındaki fikir, ağırlıkları ve biasları ayarlayarak çıktı hatasını kademeli olarak ancak istikrarlı bir şekilde azaltmaktır. Sezgisel olarak, ağırlıktaki bir değişiklik hatayı artıracak veya azaltacaksa, o ağırlığı azaltmak veya

artırmak istediğimiz bilinmektedir. Matematiksel olarak, gradyan inişi sinir ağlarında toplam hatanın ağırlıklara göre kısmi türevidir ve Denklem (4.8)'deki gibi ifade edilir.

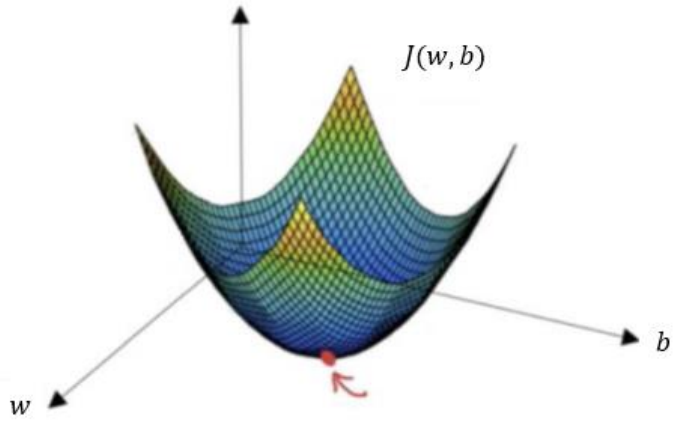
$$\frac{\partial E}{\partial w_{ij}} \tag{4.8}$$

Bu türevin sonucu bulunduğunda ağırlıklar aşağıdaki formülle güncellenir:

$$\Delta \boldsymbol{w}_{ij} = -\eta \frac{\partial E}{\partial \boldsymbol{w}_{ij}} \tag{4.9}$$

Denklem (4.9)'da η öğrenme oranını temsil etmektedir.

Öğrenme oranı, eğitim aşamasının adımlarında kademeli olarak düşme eğilimindedir. Aynı formülü kullanarak tüm ağırlıkları güncellersek, bu hata yüzeyi boyunca en dik iniş yönünde hareket etmeye eşdeğerdir. Gradyan inişin grafiği Şekil 4.18'de verilmiştir (Donges 2018).



Şekil 4.18: Ağırlık ve bias değerlerini güncellemede gradyan inişi.

Farklı gradyan iniş türleri vardır, bunlar:

4.1.3.2 Toplu Gradyan İnişi (Batch Gradient Descent–BGD)

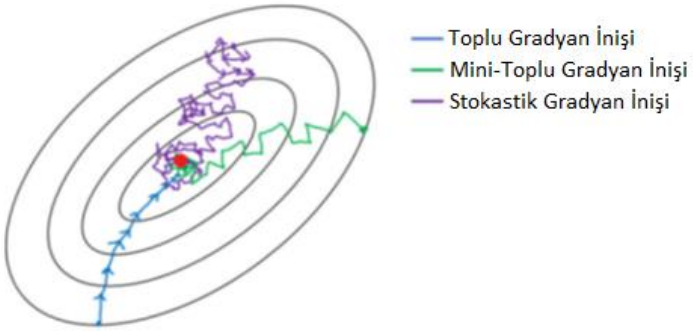
Eğitim veri kümesindeki her örnek için hatayı hesaplar, ancak model yalnızca tüm eğitim örnekleri değerlendirildikten sonra güncellenir. Tüm bu süreç bir döngü gibidir ve buna eğitim sezonu denir.

4.1.3.3 Stokastik Gradyan İnişi (Stochastic Gradient Descent–SGD)

Veri kümesindeki her eğitim örneği için bunu yapar. Bu, her eğitim örneği için parametreleri birer birer güncellediğiniz anlamına gelir. Buna bağlı olarak SGD'yi BGD'den daha hızlı hale getirebilir. Bir başka avantajı da sık güncellemelerin oldukça ayrıntılı bir iyileştirme oranına sahip olmamızı sağlamasıdır. Gerçek şu ki, sık güncellemeler hesaplama açısından BGD yaklaşımından daha pahalıdır ve bu güncellemelerin sıklığı da gürültülü gradyanlar oluşturabilir, bu da hata oranının yavaş yavaş düşmek yerine artmasına neden olabilir.

4.1.3.4 Mini-Toplu Gradyan İnişi (Mini-batch Gradient Descent)

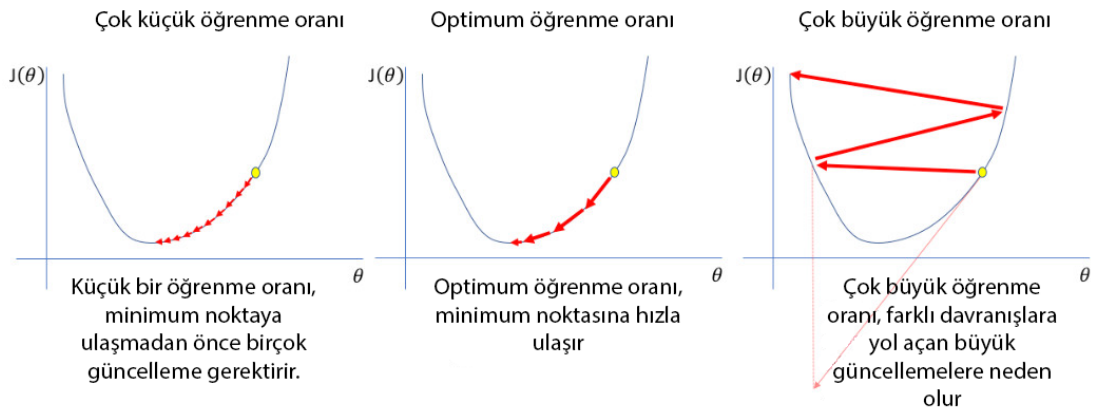
SGD ve BGD kavramlarının bir kombinasyonu olduğu için en çok kullanılan gradyan yöntemidir. Eğitim veri kümesini küçük gruplara ayırmak ve bu grupların her biri için bir güncelleme yapmak yeterlidir. Bu nedenle, daha önce bahsedilen yöntemlerden daha hızlıdır ve öğrenme sürecine gürültü ekleyerek genel hatayı iyileştirmeye yardımcı olur. Ek olarak, küçük serileri değerlendirirken, öğrenmeye fazla katkı sağlamayan çok benzer örnekleri işlemekten kaçınarak birkaç örnek rastgele seçilir. Gradyan inişin çeşitleri Şekil 4.19'da gösterilmiştir (Dabbura 2017).



Şekil 4.19: Gradyan inişin çeşitleri ve minimum hataya doğru yönü.

4.1.4 Öğrenme Oranı

Sinir ağı eğitimi sırasında en önemli hiperparametrelerden biri gradyan iniş için öğrenme hızı / oranıdır. Bu parametre, ağın kayıp fonksiyonunu en aza indirmek için optimizasyondaki adımın boyutu olarak anlaşılabilir, yani bir güncelleme adımının ağırlıkların mevcut değerini ne kadar etkilediğini belirleyen bir parametredir (Jordan 2018).



Şekil 4.20: Öğrenme oranının belirlenmesi.

Öğrenme oranı çok düşük olduğunda, birçok öğrenme tekrarı gerekir; ancak öğrenme oranı çok büyük olduğunda, sonuç uç değerlerden her iki yönde de ileri geri hareket edecektir (salınım) ve Şekil 4.20'de görülebileceği gibi optimal çözüme ulaşılamaz (Jordan, 2018).

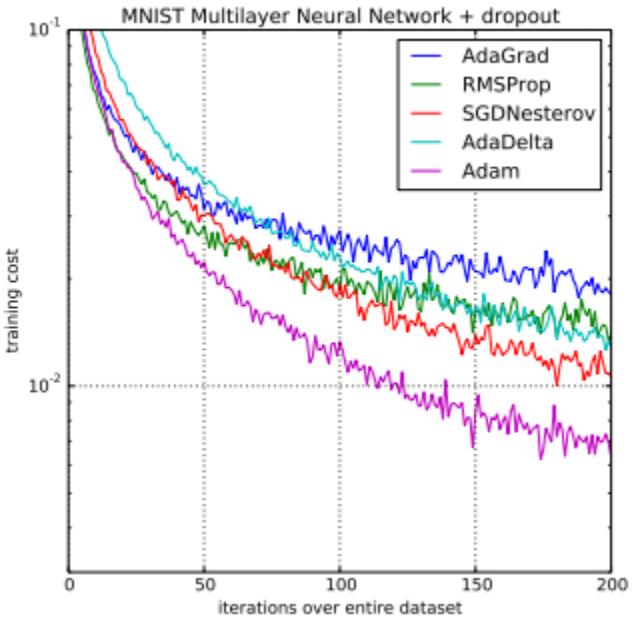
Bu nedenle, derin sinir ağlarını eğitirken, eğitim ilerledikçe öğrenme oranını düşürmek ve onu sabit tutmak sapmayı önlemek için genellikle yararlıdır ve zaman içinde öğrenme oranını yavaşça azaltmak, öğrenmeyi hızlandırmaya yardımcı olur (Jordan 2018).

Bununla birlikte, öğrenme bozulma oranına sahip Gradient Descent optimize edicisi (herhangi bir türünde), önceden tanımlanması gereken ve büyük ölçüde bağımlı olan hiperparametreleri belirlemek zor olduğundan, derin sinir ağlarını eğitmek için model ve probleme bağlı olarak çokça kullanılmaz. Ayrıca aynı öğrenme hızı tüm parametre güncellemeleri için geçerli olduğundan daha hızlı yakınsama oranına sahip, çeşitli durumlara uyarlanabilen ve ikinci dereceden gradyan iniş optimize ediciler olarak bilinen daha gelişmiş Adagrad, Nadam, RMSprop, Adam gibi optimize edici yöntemler kullanılır (Ser ve Bati 2019).

4.1.5 Adam Optimizasyonu

Adam optimize edici (Adaptive Moment Estimation) derin sinir ağı eğitimi için özel olarak tasarlanmış uyarlanabilir bir öğrenme hızı optimizasyon algoritmasıdır (Kingma ve Ba, 2015).

Öğrenme hızının parametrelere göre uyarlanmasını sağlar ve seyrek parametreler için daha büyük güncellemeler ve sık olanlar için daha küçük güncellemeler gerçekleştirir. Bu nedenle, doğal dil işleme veya görüntü tanıma gibi seyrek veriler için uygundur. Diğer bir avantajı, temelde öğrenme oranını ayarlama ihtiyacını basitleştirmesidir, çünkü her parametrenin kendi öğrenme oranı vardır. Optimizasyon algoritmalarının karşılaştırılması Şekil 4.21'de verilmiştir (Kingma ve Ba, 2015).



Şekil 4.21: Optimize edicilerin çok katmanlı bir algılayıcı eğitirken karşılaştırılmaları.

4.1.6 Çapraz Doğrulama (Cross-Validation)

Makine öğrenmesi modellerinde veriler eğitim ve doğrulama için iki alt kümeye ayrılırlar. Bu ayırma işlemi rastgele örneklemeler olması açısından veri seti üzerinde rastgele gerçekleştirilir.

Doğrulama alt kümesi, farklı mimarilerle oluşturulmuş modellerin performansını ölçmek ve bunlardan birini seçmek için kullanılır. Başka bir deyişle bu doğrulama seti hatanın belirli bir aralıkta olup olmadığını kontrol etmek ve bu değere göre eğitimin durma aşamasını belirlemek için kullanılır.

Çapraz doğrulama, istatistiksel bir analizin sonuçlarını değerlendirmek ve bunların eğitim ile doğrulama verileri arasındaki bölümden bağımsız olmasını sağlamak için kullanılan bir tekniktir. Sınıflandırma problemlerinde veriler sonlu sınıflara bölündüğünden, belirli bir sonucun hangi kategoriye ait olduğu olasılıksal olarak hesaplanır. Olasılıkta, çapraz entropi iki olasılık dağılımı arasındaki mesafedir ve genellikle kayıp fonksiyonu olarak kullanılır. Çapraz doğrulama sonucunun veri setinin entropi kaybını temsil etmesinin nedeni de budur. Entropi bilgi kuramında bir kümenin bilgi içeriğini ölçmektedir. Bu bağlamda da derin öğrenmede olasılıksal

olarak çıkan sonuçların birbirinden ne kadar farklı olduğunu ölçer ve x ayrık değişkeni üzerinde iki dağılımı içeren çapraz entropi formülü aşağıdaki şekilde ifade edilir:

$$H(p,q) = -\sum_{\forall x} p(x)\log(q(x)) \tag{4.10}$$

Burada $q(x)$ sınıflandırmanın gerçek değerlerini, $p(x)$ ise tahmin değerlerini ifade etmektedir.

Bir sinir ağı modeli için, modelin tahmin etmesi gereken gerçek değeri y ve modelin tahmin değerlerini $\hat{y}$ şeklinde tanımlarsak bir örnek bazında Denklem (4.10) aşağıdaki şekilde ifade edilebilir.

$$L = -y.\log\hat{y} \tag{4.11}$$

Genellikle çok sınıflı sınıflandırıcı modellerinde Denklem (4.11)'in tüm örnekler üzerinde maliyet fonksiyonu olarak ortalaması alınır. Optimizasyon problemlerinde kayıp fonksiyonu daha düşük seviyededir ve tek bir örneğin veya bileşenin hata değerini belirlemede kullanılır, maliyet fonksiyonu ise daha yüksek bir seviyededir ve bir sistemin nasıl değerlendirildiğini açıklar. N boyutlu bir veri kümesi için çok sınıflı bir sınıflandırma kaybı maliyeti aşağıdaki denklem ile hesaplanır.

$$J = -\frac{1}{N}\sum_{i=1}^{N} y_i.\log(\hat{y}_i) \tag{4.12}$$

Öğrenme modellerinde çapraz doğrulama ile verilen maliyet fonksiyonu, bir ağın eğitiminin ne zaman bitirilmesi gerektiğine karar vermeye yardımcı olur. Ortalama karesel hata genellikle bir regresyon probleminin performansını ölçmek için kayıp fonksiyonu olarak kullanılırken, çok sınıflı sınıflandırma problemlerinde çapraz entropi hesabı tercih edilir çünkü çapraz entropi çıkış nöronlarının doygunluğuna neden olmayarak daha hızlı bir öğrenme gerçekleştirir. Öngörülen olasılık gerçek sınıf değerlerinden uzaklaştıkça çapraz entropi kaybı da artacaktır. Bu nedenle neredeyse mükemmel bir sınıflandırıcı sıfıra yakın bir entropi kaybına sahip olacaktır (Golik ve diğ. 2013).

4.1.7 Softmax Fonksiyonu

Bu fonksiyon bir sinir ağının son katmanında kullanılan bir aktivasyon fonksiyonudur. Bu fonksiyon tam bağlı bir sinir ağının çıktısının olasılık dağılımını vermez ancak bu fonksiyonu kullanmak sinir ağının çıkışının olasılık dağılımını hesaplamaya imkan tanır. Bu fonksiyon, her bir nöronun değerinin [0,1] arasına çekilmesini sağlayarak, nöronların sonuçlarında bir normalizasyon gerçekleştirir ve girdi görüntüsünün hangi sınıfa ait olma olasılığının belirlenmesine izin verir (Bishop 2006).

$$P(y = j|x) = \frac{e^{x^T w_j}}{\sum_{k=1}^{K} e^{x^T w_k}} \tag{4.13}$$

Burada; $x^T w$, x giriş ve w ağırlık değerlerinin iç çarpımını göstermektedir. Bu fonksiyon, tam bağlı katmanın çıktılarını normalize ettiğinden, çapraz doğrulamanın uygulanmasına ve entropi kaybının hesaplanmasına olanak tanır.

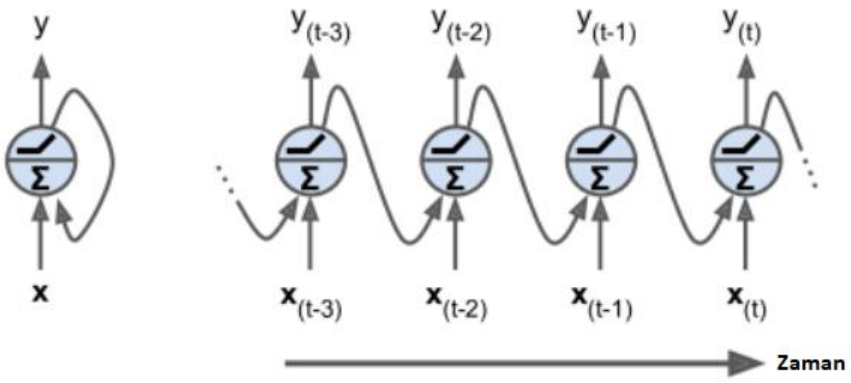
4.2 Yinelemeli Sinir Ağları

Yinelemeli sinir ağları (Recurrent neural network - RNN) sıralı verileri işlemek için özelleşmiş bir yapay sinir ağı ailesidir. Evrişimli sinir ağları uzamsal ilişkisi olan verileri işlemek için özelleşmiştir. Evrişimli sinir ağları büyük genişliğe ve yüksekliğe sahip verilerde ölçeklenebilir ya da değişken boyutlardaki verileri işleyebilir. Bunun gibi yinelemeli sinir ağları da sıra tabanlı veriler için özelleşmiş olmayan ağların işleyebileceğinden daha büyük sıralı dizilere ölçeklenebilir ve bu dizileri işleyebilirler (Goodfellow ve diğ. 2018).

Yinelemeli ağlarda parametrelerin, modelin farklı bölümleri arasında paylaşılması fikrinden yararlanılır. Parametre paylaşımı, modeli farklı formların örneklerine genişletip uygulayabilir ve modeli genelleştirebilir (Goodfellow ve diğ. 2018).

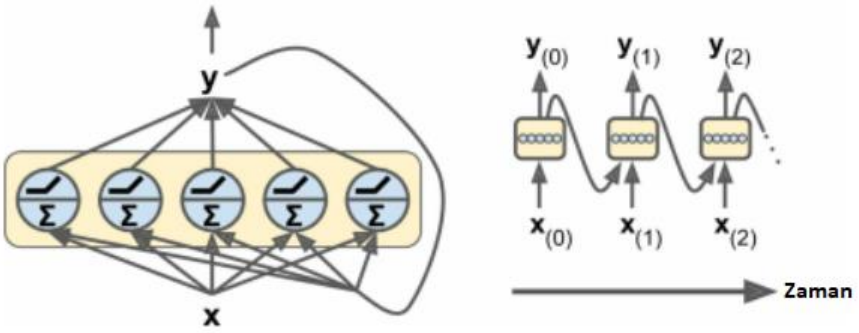
Standart sinir ağları ile yinelemeli sinir ağları arasındaki fark önemsiz görünse de yinelemeli sinir ağlarının sıralı öğreniminin çıkarımları geniş

kapsamlıdır. Bir yapay sinir ağı modeli yalnızca giriş vektörlerinden çıktı vektörlerine eşleşme yapabilir, yinelemeli sinir ağları ise önceki girdilerin tüm geçmişinden her çıktıya eşleyebilir. Gerçekte, standart sinir ağları için evrensel yaklaşım teorisinin eşdeğer sonucu, yeterli sayıda gizli birim içeren bir yinelemeli sinir ağının, herhangi bir ölçülebilir sekans-sekans eşlemesini doğruluğa yaklaştırabilmesidir (Hammer 2000). Buradaki kilit nokta, tekrarlayan bağlantıların, önceki girdilerin bir "belleğinin" ağın dahili durumunda kalmasına ve dolayısıyla ağ çıkışını etkilemesine izin vermesidir. Yani yinelenen sinir ağları dinamik bir yapıya sahiptir ve geriye nöronların geriye dönük bağlantıları vardır. Ağın çıkışı, ağın giriş değerlerine, önceki giriş değerlerine veya önceki çıkış değerleri gibi parametrelere bağlıdır (Geron 2017).



Şekil 4.22: a) Tekrarlayan nöron, b) zamanla gelişim.

Şekil 4.22'de girdileri alan ve bir çıktı üreten basit bir yinelemeli sinir ağı mimarisi gösterilmiştir (Geron 2017). Şekilden de görüldüğü üzere her zaman adımı t'de (zaman çerçevesi de denir), bu yinelenen nöron hem $x_{(t)}$ girdisini hem de önceki adımın y_{t-1} çıktısını alır. Bu süreci bir zaman ekseninde gösterilirse Şekil 4.22.b elde edilir. Yinelenen nöronların birleştirilmesiyle Şekil 4.23'teki gibi katmanlar oluşturulur ve her t anında tüm nöronlar giriş vektörü $x_{(t)}$'yi ve önceki adımın çıktı vektörü olan y_{t-1}'i alır (Geron 2017).



Şekil 4.23: a) Yinelenen nöron katmanı, b) zaman içinde gelişme.

Her bir yinelenen nöronun iki farklı ağırlık parametresi vardır. Bu ağırlıkların biri $x_{(t)}$ girişleri, diğeri ise önceki zaman adımı olan y_{t-1} içindir. Bu ağırlıkları w_x ve w_y olarak adlandırırsak $y_{(t)}$ çıkışı aşağıdaki gibi hesaplanabilir.

$$y_{(t)} = \sigma(x_{(t)}^T . w_x + y_{(t-1)}^T . w_y + b) \tag{4.14}$$

Burada; b bias değeridir ve σ aktivasyon fonksiyonunu temsil etmektedir. Yukarıdaki ifade tüm girişlere bağlı olarak yinelenen nöron katmanının çıktısı olarak aşağıdaki şekilde ifade edilebilir.

$$\boldsymbol{Y}_{(t)} = \sigma\left(\boldsymbol{X}_{(t)} . \boldsymbol{W}_x + \boldsymbol{Y}_{(t-1)} . \boldsymbol{W}_y + b\right) = \sigma([\boldsymbol{X}_{(t)} \quad \boldsymbol{Y}_{(t-1)}] . [\boldsymbol{W}_x \quad \boldsymbol{W}_y]^T + b) \tag{4.15}$$

Burada;

- $\boldsymbol{Y}_{(t)}$, alt örneklemedeki her örnek için t adımındaki katmanın çıktılarını ifade eden bir $m \; x \; n_{nöron}$'luk bir matristir. Burada m alt örneklemedeki örnek sayısıdır ve $n_{nöron}$ ise katmandaki nöron sayısıdır.
- $\boldsymbol{X}_{(t)}$, tüm örnekler için girdileri içeren bir $m \; x \; n_{giriş}$'lik bir matristir. Burada $n_{giriş}$ giriş sayısını ifade etmektedir.
- $\boldsymbol{W}_x$, mevcut zamanın girişleri için bağlantı ağırlıklarını içeren bir $n_{giriş} \; x \; n_{nöron}$'luk bir matristir.
- $\boldsymbol{W}_y$, önceki zaman adımının çıktıları için bağlantı ağırlıklarını içeren $n_{nöron} \; x \; n_{nöron}$'luk bir matristir.
- Ağırlık matrisleri $\boldsymbol{W}_x$ ve $\boldsymbol{W}_y$ genellikle $\left(n_{giriş} + n_{nöron}\right) x \; n_{nöron}$ 'luk bir matris olarak W şeklinde birleştirilir.
- b ise her bir nöron için bias terimi barındıran $n_{nöron}$ boyutunda bir vektördür.

Genellikle bias vektörünün başlangıç değerleri sıfır olarak ayarlanır. Fakat yapılan çalışmalar sonucunda yinelemeli sinir ağlarının kararlılığının ve performansının sıfır olmayan başlangıç değerleri kullanılarak geliştirilebileceği bulunmuştur.

Kısaca $\boldsymbol{Y}_{(t)}$, $\boldsymbol{X}_{(t)}$ ve $\boldsymbol{Y}_{(t-1)}$'in bir fonksiyonu olarak tanımlanabilir. Benzer şekilde $\boldsymbol{X}_{(t-1)}$ ve $\boldsymbol{Y}_{(t-2)}$ de $\boldsymbol{X}_{(t-2)}$ ve $\boldsymbol{Y}_{(t-3)}$ 'ü içeren fonksiyonlar olarak tanımlanabilir. Bu tanımlamalar $\boldsymbol{Y}_{(t)}$'yi $t = 0$ anından itibaren tüm girdilerin bir fonksiyonu yapar. İlk zaman adımında yani $t = 0$ iken önceki çıktılar yoktur bu nedenle tipik olarak hepsinin sıfır olduğu varsayılır.

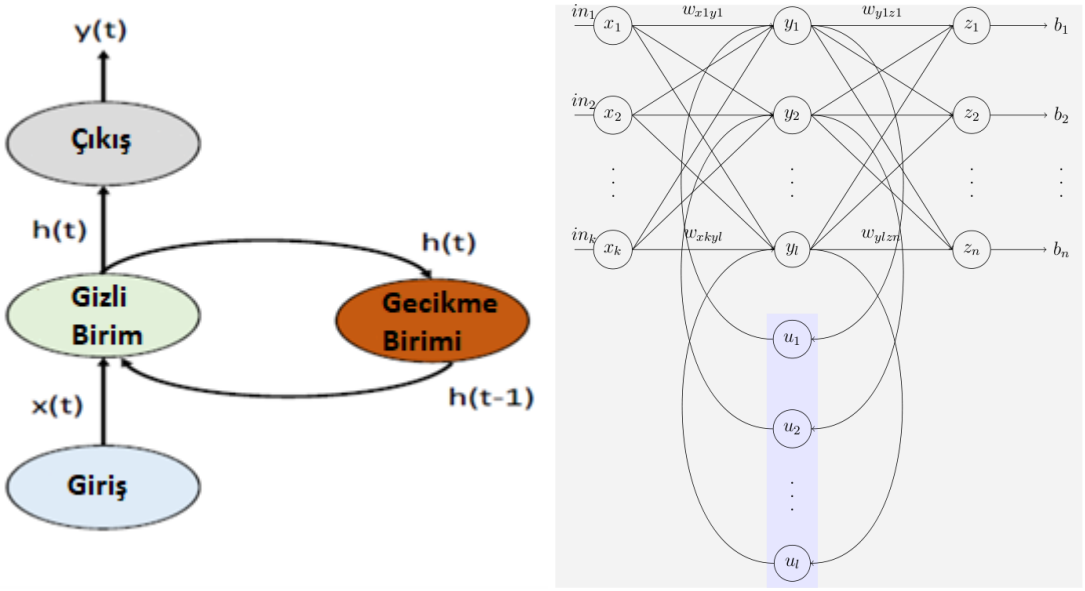
Yinelemeli sinir ağlarında da birçok tür önerilmiştir. Bunlardan en çok bileneni Elman Ağları (Elman 1990) ve Jordan ağlarıdır (Jordan 1990).

4.2.1 Elman Ağları

Bir Elman sinir ağı, 'bağlam birimleri' katmanı (veya gecikme birimleri olarak da adlandırılır) ile bir katman eklenen üç katmana (giriş, gizli ve çıktı katmanları) sahip bir ağdır. Her katman, ayarlanmış olan öğrenme kuralıyla bilgileri bir katmandan diğerine ileten bir veya daha fazla nöron içerir (Geron 2017).

Elman ağlarında bağlam katmanındaki nöron sayısı, gizli katmandaki nöronların sayısına eşit olmalıdır. Ayrıca bağlam katmanındaki tüm nöronlar, gizli katmandaki bir nörona bağlıdır.

Hafıza süreci, gizli katmanın nöronları tarafından güçlendirilen gecikme birimleri aracılığıyla gerçekleşir. Gizli katman ile gecikme birimleri arasındaki bağlantıların ağırlıkları sabit ve 1'e eşittir. Bu sayede gecikme birimlerinde her zaman gizli katmanın nöronlarının bir önceki adımdaki çıkış değerlerinin bir kopyası saklanır.

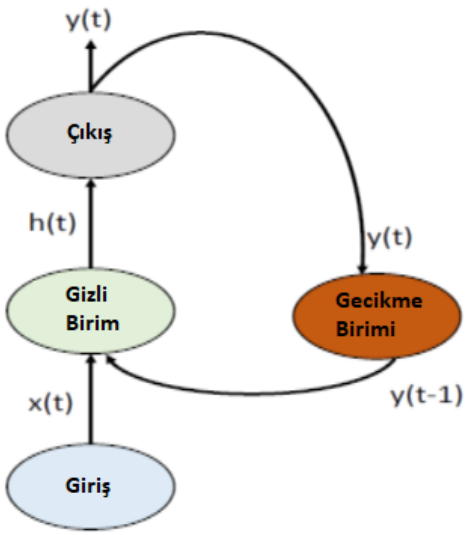


Şekil 4.24: a) Genelleştirilmiş Elman ağ yapısı, b) Katmanlı ağ yapısı.

Şekil 4.24.a'da en genelleştirilmiş Elman ağ şeması verilmiştir. Gizli katmanın çıktısı gecikme katmanında saklanır ve saklanan bu değerler bir sonraki adımda gizli katman için giriş değerleri olarak kullanılırlar. Şekil 4.24.b'de ise nöron çıktısının y_i değerinin saklandığı ve bir sonraki adımda girdi olarak verildiği gecikme birimlerini u_i gösterilmiştir (Lewis 2016).

4.2.2 Jordan Ağları

Jordan sinir ağları da Elman'a benzer bir mimariye sahiptir. Yani giriş katmanı, gizli bir katman, çıktı katmanı ve bağlam katmanından oluşur. Aradaki tek fark, gizli katmandaki bilgiler yerine çıktı katmanından gelen bilgilerin kaydedilmesidir. Bu nedenle bağlam katmanı, Şekil 4.25'te gördüğü gibi gizli katmana ve çıktı katmanına bağlanır. Bu nedenle, bu ağ gizli katman çıktısı yerine önceki adımın son çıktısını hatırlar. Jordan'ın sinir ağının yapısı Şekil 4.25'teki gibidir (Lewis 2016):



Şekil 4.25: Jordan sinir ağı yapısı.

Her iki süreci de tanımlayan denklemler Denklem (4.16), (4.17), (4.18) ve (4.19)'da verilmiştir.

- Elman denklemleri:

$$h_t = \sigma_h(\boldsymbol{W}_h x_t + \boldsymbol{U}_h h_{t-1} + b_h) \tag{4.16}$$

$$y_t = \sigma_y(\boldsymbol{W}_y h_t + b_y) \tag{4.17}$$

- Jordan denklemleri:

$$h_t = \sigma_h(\boldsymbol{W}_h x_t + \boldsymbol{U}_h y_{t-1} + b_h) \tag{4.18}$$

$$y_t = \sigma_y(\boldsymbol{W}_y h_t + b_y) \tag{4.19}$$

Burada; x_t giriş değişkenlerini içeren bir vektör, h_t gizli katman vektörü ve y_t ise çıkış katmanının vektörüdür. $\boldsymbol{W} = [\boldsymbol{W}_x \quad \boldsymbol{W}_y]^T$ ve $\boldsymbol{U}_h$ ağırlık matrislerini ifade etmektedir. $b = [b_x \quad b_y]^T$ bias vektörleridir ve σ_y ve σ_h ilgili katmanın aktivasyon fonksiyonlarıdır.

4.2.3 Yinelemeli Sinir Ağlarında Geri Yayılım

Standart sinir ağlarında da olduğu gibi yinelemeli sinir ağlarında da ileri yayılımdan sonra kayıp fonksiyonuna göre hatanın kısmi türev yardımıyla geriye yayılıp ağırlıkların ve bias değerlerinin güncellenmesi gerekir. Ağırlıkları gradyan iniş yöntemiyle güncellemek için yinelemeli sinir ağlarında "Real Time Recurrent Learning (RTRL)" (Robinson ve Fallside, 1987), "Backpropagation Through Time (BPTT)" (Werbos, 1990) gibi yöntemler araştırmacılar tarafından önerilmiştir. BPTT algoritması hesaplama zamanında önemli ölçüde tasarruf sağladığı için en çok kullanılan yöntemler arasındadır.

Standart geri yayılım gibi, BPTT de zincir kuralının tekrarlanan bir uygulamasından oluşur. Buradaki fark, yinelemeli ağlar için, kayıp işlevinin yalnızca çıktı katmanındaki etkisi aracılığıyla değil, aynı zamanda bir sonraki zaman adımında gizli katmandaki etkisi aracılığıyla da gizli katmanın etkinleştirilmesine bağlı olmasıdır. Bu nedenle;

$$\delta_h^t = \sigma'(\boldsymbol{Y}_{(t)})(\sum_{k=1}^{K} \delta_k^t \boldsymbol{w}_{hk} + \sum_{h'=1}^{H} \delta_{h'}^{t+1} \boldsymbol{w}_{hh'}) \tag{4.20}$$

şeklinde hesaplanır. Burada;

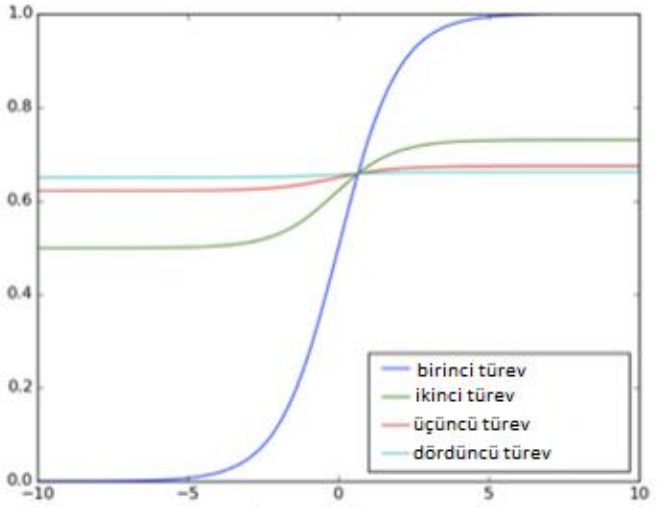
$$\delta_j^t = \frac{\partial L}{\partial y_j^t} \tag{4.21}$$

δ terimlerinin tam dizisi $t = T$'den başlayarak yinelemeli olarak Denklem (4.20) uygulanarak her adımda t'yi azaltarak hesaplanabilir. Son olarak her zaman adımında aynı ağırlıkların yeniden kullanıldığını akılda tutarak, ağ ağırlıklarına göre türevleri elde etmek için tüm diziyi Denklem (4.22)'deki gibi toplarız.

$$\frac{\partial L}{\partial w_{ij}} = \sum_{t=1}^{T} \frac{\partial L}{\partial y_j^t} \frac{\partial y_j^t}{\partial w_{ij}} = \sum_{t=1}^{T} \delta_j^t b_i^t \tag{4.22}$$

4.2.3.1 Gradyan Zayıflaması Problemi

Gradyan ağırlıkları güncellemek için kullanılan bir değerdir. Bir birlerine uzun bağlamlar içeren ağlarda hatanın etkisi oldukça düşer ve bütün katmanlar zamana bağlı adımlarla birbirine çarpımla bağlı oldukları için türevleri zayıflamaya yani gradyan sıfıra yaklaşmaya başlayabilir. Bu durum sigmoid fonksiyonu üzerinden Şekil 4.26'da grafiksel olarak gösterilmiştir (Ergüder 2018).

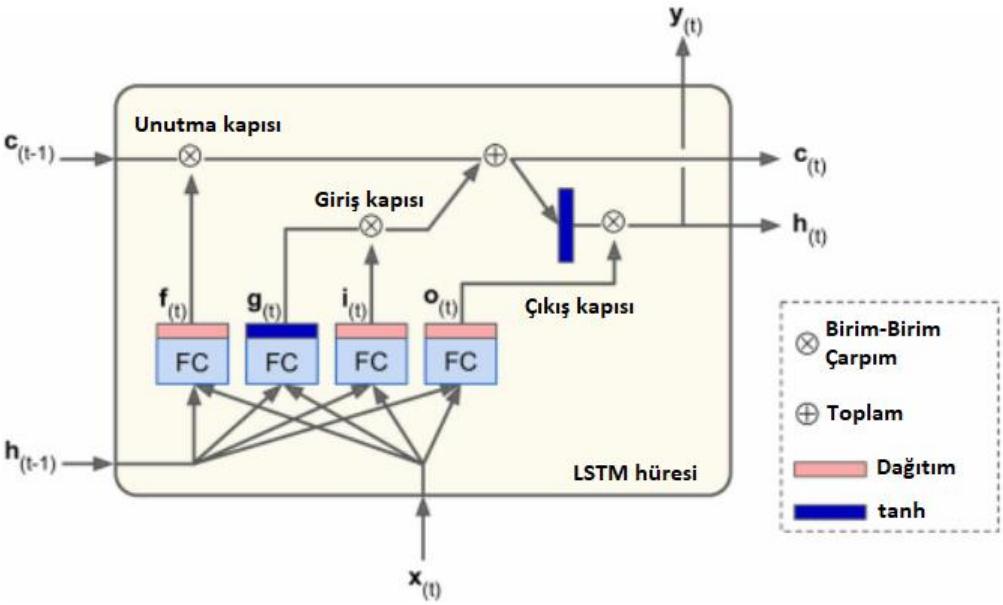


Şekil 4.26: Sigmoid aktivasyon fonksiyonunun türeve bağlı değişimi.

Gradyan değerinin karşılaştığı bu tehlike ağın doğru sonuçlar üretmesini engelleyecektir. Bu sorunu çözmek için birkaç çözüm önerilmiştir. Ağırlık değerleri için ağa uygun değerler seçmek yok olma etkisini azaltacaktır fakat bu yöntem oldukça zor ve deneme yanılma gerektirmektedir. Diğer bir yöntem ise ReLU aktivasyon fonksiyonunu kullanmaktır çünkü bu fonksiyonun türevi 0 veya 1'dir. Diğer ve en etkili yöntem ise bu problemi çözmek için önerilmiş olan "Uzun-Kısa Vadeli Bellek" (Long-Short Term Memory - LSTM) ve "Kapılı Yinelemeli Hücre" (Gated Recurrent Unit - GRU) yöntemleridir.

4.2.4 Uzun-Kısa Vadeli Bellek Modeli

Uzun Kısa Süreli Bellek tekrarlayan sinir ağı, Elman ve Jordan ağlarına benzer, yalnızca nöronları ve gizli birimleri bir bellek bloğu ile değiştirir (Hochreiter ve Schmidhuber 1997). Değiştirilen bu blok LSTM hücresi olarak adlandırılır. Şekil 4.27'de bir LSTM hücresi gösterilmiştir (Geron 2017).



Şekil 4.27: Bir LSTM hücresi.

Bir LSTM hücresinin çıkışı yukarıdaki şekilde de görüldüğü gibi uzun vadeli durumu temsil eden $c_{(t)}$ ve kısa vadeli durumu ifade eden $h_{(t)}$ vektörlerinden oluşur. LSTM hücresinin ana fikri, ağın uzun vadede neyi depolayacağını, neyi atacağını ve onlardan ne okuyacağını öğrenebilmesidir. Uzun vadeli terime bakarsak, $c_{(t-1)}$ ağı soldan sağa dolaşır, önce bir unutma kapısından geçer, depolanan bazı verilerden kurtulur, ardından kapıdan gelen yeni bilgileri ekler. Giriş geçidi ve sonuç $c_{(t)}$ doğrudan hücrenin dışına gönderilir ve olduğu gibi bırakılır ve bir sonraki dönem için girdi olarak işlev görür. Bu nedenle, her adımda yeni hatıralar eklenirken bazı anılar silinir. Ek olarak, yeni bellekler eklendikten sonra, uzun vadeli durumun bir kopyası alınır ve daha sonra çıkış geçidinden filtrelenmek üzere bir tanjant hiperbolik aktivasyon fonksiyonundan geçer. Bu adım, kısa vadeli durum $h_{(t)}$'yi (bu zaman adımı t için hücre çıktısı y_t'ye eşittir) üretir.

Öncelikle mevcut giriş vektörü $x_{(t)}$ ve önceki kısa vadeli durum $h_{(t-1)}$ birbirine bağlı dört farklı katmana girer ve hepsinin farklı bir amacı vardır.

- Ana katman $g_{(t)}$'yi veren katmandır. Mevcut giriş $x_{(t)}$ ve önceki kısa vadeli durum olan $h_{(t-1)}$'yi analiz etme gibi bir role sahiptir. Normal bir hücrede sadece bu katman bulunur ve çıktısı doğrudan $y_{(t)}$ ve $h_{(t)}$'ye girer. LSTM hücresinde bu katmanın çıktısı doğrudan dışarı çıkmaz, bunun yerine uzun vadeli durumda depolanır.
- Diğer üç katman kapı denetleyicileridir. Lojistik bir aktivasyon fonksiyonu kullandıklarından çıktıları 0 ve 1 arasında değişir. Çıktılar eleman bazında çarpma işlemlerine beslenir yani eğer çıkışlar 0 ise kapıyı kapatırlar eğer çıkışlar 1 ise kapıyı açık bırakırlar. Buda öğrenilen verilerin tutulmasını veya atılmasını sağlar.
- Unutma kapısı ($f_{(t)}$ tarafından kontrol edilir) uzun vadeli durumun hangi bölümlerinin silinmesi gerektiğini kontrol eder.
- Giriş kapısı ($i_{(t)}$ tarafından kontrol edilir), $g_{(t)}$'nin hangi kısımlarının uzun vadeli duruma eklenmesi gerektiğini kontrol eder.
- Son olarak çıkış kapısı ($o_{(t)}$ tarafından kontrol edilir) geçerli zaman adımında uzun vadeli durumun hangi bölümlerinin okunması ve çıktı alınması gerektiğini kontrol eder.

Kısaca bir LSTM hücresi önemli giriş geçidinin bir rolü olarak bir girdiyi tanımayı öğrenebilir, onu uzun vadeli durumda saklayabilir ve gerektiği kadar korumayı öğrenebilir ve gerektiğinde bu bilgiyi unutmayı öğrenebilir.

LSTM hücrelerinde uzun ve kısa vadeli durumlar bir zaman adımı içerisinde aşağıdaki denklemlerle hesaplanabilir.

$$i_{(t)} = \sigma(\boldsymbol{W}_{xi}^{T}.x_{(t)} + \boldsymbol{W}_{hi}^{T}.h_{(t-1)} + b_i) \quad (4.23)$$

$$f_{(t)} = \sigma(\boldsymbol{W}_{xf}^{T}.x_{(t)} + \boldsymbol{W}_{hf}^{T}.h_{(t-1)} + b_f) \quad (4.24)$$

$$o_{(t)} = \sigma(\boldsymbol{W}_{xo}^{T}.x_{(t)} + \boldsymbol{W}_{ho}^{T}.h_{(t-1)} + b_o) \quad (4.25)$$

$$g_{(t)} = tanh(\boldsymbol{W}_{xg}^{T}.x_{(t)} + \boldsymbol{W}_{hg}^{T}.h_{(t-1)} + b_g) \quad (4.26)$$

$$c_{(t)} = f_{(t)}.c_{(t-1)} + i_{(t)}.g_{(t)} \quad (4.27)$$

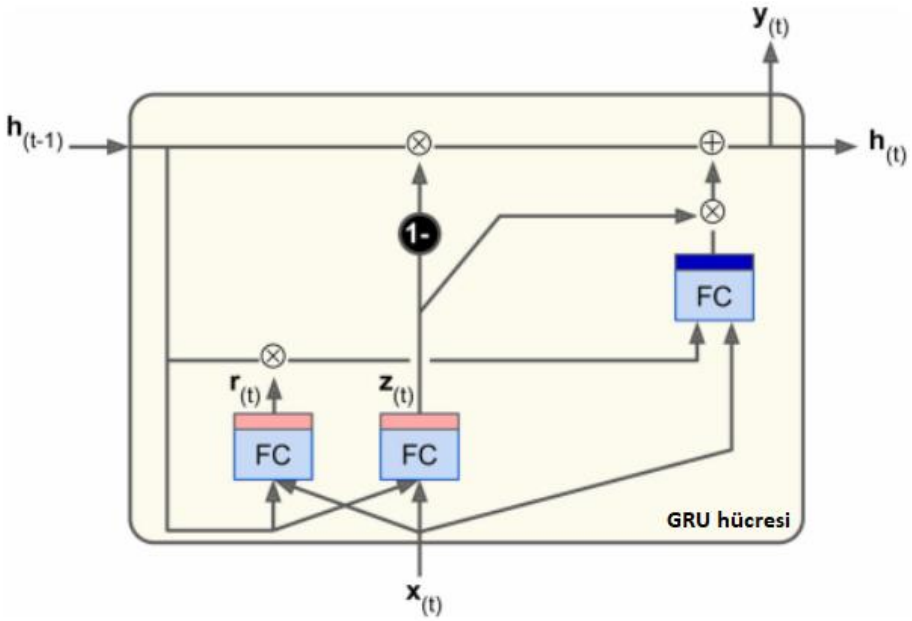
$$y_{(t)} = h_t = o_{(t)}.\tanh(c_{(t)}) \quad (4.28)$$

Burada;

- $\boldsymbol{W}_{xi}, \boldsymbol{W}_{xf}, \boldsymbol{W}_{xo}, \boldsymbol{W}_{xg}$ giriş vektörü $x_{(t)}$ ile bağlantıları için dört katmanın her birinin ağırlık matrisidir.
- $\boldsymbol{W}_{hi}, \boldsymbol{W}_{hf}, \boldsymbol{W}_{ho}, \boldsymbol{W}_{hg}$ önceki kısa vadeli durum $h_{(t-1)}$ ile bağlantıları için dört katmanın her birinin ağırlık matrisidir.
- b_i, b_f, b_o, b_g dört katmanın her biri için bias değerleridir.

4.2.5 Kapılı Yinelemeli Birim Modeli

Kapılı yinelemeli birim (GRU) hücresi Cho ve diğ. (2014) tarafından 2014 yılında sunulmuştur. GRU hücresi basitleştirilmiş bir LSTM hücresidir ve aynı derecede iyi bir öğrenme performansı gösterir. Bu basitleştirmeden dolayı işlem yükü önemli derecede azaltmış ve yapıyı popüler hale getirmiştir. Bu yapı Şekil 4.28'deki gibi temsil edilmiştir (Geron 2017).



Şekil 4.28: GRU hücresi.

Yapılan ana basitleştirmeler aşağıda listelenmiştir.

- Her iki durum vektörü tek bir $h_{(t)}$ vektöründe birleştirilir.
- Tek bir geçit denetleyicisi hem unutma kapısını hem de giriş kapısını kontrol eder. Geçit denetleyicisi 1 çıktısı verirse, giriş kapısı açıktır ve unutma geçidi kapalıdır. 0 çıkarsa, bunun tersi olur. Diğer bir deyişle, bir hafızanın saklanması gerektiğinde, önce depolanacağı yer silinir.
- Çıkış kapısı yoktur; tam durum vektörü her zaman adımında çıkar. Ancak, önceki durumun hangi bölümünün ana katmana gireceğini kontrol eden yeni bir geçit denetleyicisi vardır.

Aşağıda verilen denklemler tek bir örnek için her adımda hücrenin durumunun nasıl hesaplanacağını göstermektedir.

$$z_{(t)} = \sigma(\boldsymbol{W}_{xz}^{T}.x_{(t)} + \boldsymbol{W}_{hz}^{T}.h_{(t-1)}) \quad (4.29)$$

$$r_{(t)} = \sigma(\boldsymbol{W}_{xr}^{T}.x_{(t)} + \boldsymbol{W}_{hr}^{T}.h_{(t-1)}) \quad (4.30)$$

$$g_{(t)} = tanh(\boldsymbol{W}_{xg}^{T}.x_{(t)} + \boldsymbol{W}_{hg}^{T}.(r_{(t)}.h_{(t-1)})) \quad (4.31)$$

$$h_{(t)} = \left(1 - z_{(t)}\right).\tanh\left(\boldsymbol{W}_{xg}^{T}.h_{(t-1)} + z_{(t)}.g_{(t)}\right) \quad (4.32)$$

5. MATERYAL VE YÖNTEM

5.1 Trafik İşaretlerinin Tarihçesi

Trafik işaretleri karayollarında bilgi verme, yasak ve tehlike belirtme olmak üzere üç durum için kullanılırlar. Araç teknolojisinin de gelişmesiyle trafik işaretlerine ve çeşitliliğine olan ihtiyaç artmıştır.

Trafik işaretlerinin her ne kadar global çapta aynı işaretlerin aynı anlamları ifade etmesi için kullanılması amaçlansada bu pek mümkün olamamıştır. Bu sorunu ortadan kaldırmak için 1931 yılında "Geneva Convention Concerning the Unification of Road Signals" konferansı yapılmıştır. 1949 yılında ise 1931 yılında yapılan konferansı baz alan "Geneva Protocol on Road Signs and Signals" konferansı yapılmıştır. Son olarak 1949 yılında yayınlanan konferansı baz alan, 1968 yılında yayınlanan "Vienna Convention on Road Signs and Signals" bildirisi ile trafik işaretlerinde küreselleşme yolunda önemli bir yol kat edilmiştir.

5.2 Veri Seti

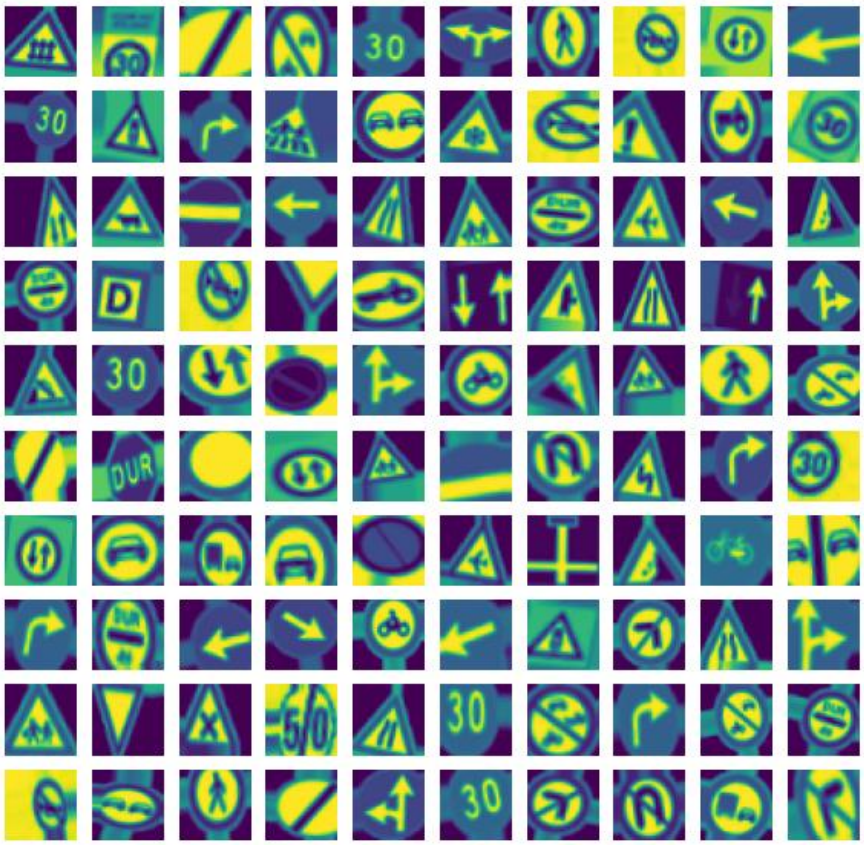
Trafik işaretlerinin makine öğrenmesi modelleriyle tanınmasını amaçlayan çalışmalarda kullanılmak üzere internet ortamında açık erişime sahip birçok veri seti bulunmaktadır. Bu tez çalışmasında da Kaggle'da bulunan "Traffic Sign Images From Turkey" veri seti kullanılmıştır (Yücesan 2020).

Bu veri seti Türkiye'de kullanılan trafik işaretlerinden çekilmiş görüntülerle oluşturulmuştur. Oluşturulan bu veri seti 21249 adet giriş görüntüsünden ve 91 adet sınıftan oluşmaktadır. Veri setindeki tüm resimler 32x32 piksellik boyutta ve png formatında sunulmuştur.

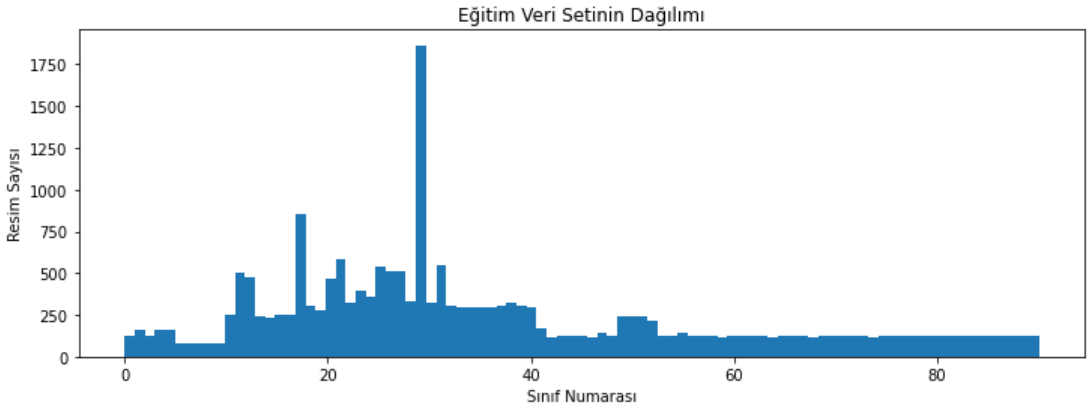
Yapılan bu tez çalışmasında tüm modelleri eğitmek ve başarımlarının karşılaştırmasını yapmak için veri setinde bulunan resimlerin 12467 tanesi eğitim

için, 8318 tanesi doğrulama için ve her bir sınıftan 5'er örnek alınarak 455 tanesi de modelin başarısını test etmek için kullanılmıştır.

Veri setinden bazı örnekler Şekil 5.1'de, veri setinin sınıflara göre dağılım oranları ise Şekil 5.2'de gösterilmiştir.



Şekil 5.1: Veri setinden bazı örnekler.



Şekil 5.2: Veri setinin sınıflara göre dağılım oranları.

5.3 Eğitimde Kullanılan Donanımlar ve Yazılımlar

Makine öğrenmesi modellerini oluşturmak için birçok yazılım dili vardır ve bunların en popüleri çok fazla makine öğrenimi kütüphanesi barındırması ve açık kaynak kodlu olması sebebiyle Python'dur.

Yapılan bu tez çalışmasında geliştirme için kullanılan Python dağıtımı Anaconda'dır. Anaconda içerisinde veri bilimi için özelleşmiş olan bir çok kütüphaneyi ve aracı ön yüklü olarak bulundurur. Geliştirilen yazılımlar Anaconda içerisinde yine ön yüklü olarak gelen Spyder derleyicisi üzerinde geliştirilmiştir. Spyder derleyicisi veri bilimi, veri gösterimi ve makine öğrenimi algoritmaları için üzerinde araçlar bulundurur. Bu sayede verileri görselleştirmek, verilerin grafiğini çizdirmek oldukça kolaylaşır ve zamandan tasarruf sağlanır.

Eğitimde kullanılan kişisel bilgisayar üzerinde Intel i7-9750H işlemci bulunduran ve 16GB RAM'e sahip bir bilgisayardır. Spyder üzerinde oluşturulan kodlar bu bilgisayar üzerinde çalıştırılmıştır.

Makine öğrenmesi alanında Python için hazırlanmış olan birçok kütüphane bulunmaktadır. Bu kütüphanelerden bazıları şunlardır: TensorFlow, Theano, Caffe, Keras. Yapılan bu tez çalışmasında da modeller TensorFlow, Theano gibi kütüphanelerin üzerinde çalışan üst düzey bir kütüphane olan Keras tercih edilmiştir.

5.4 Yöntemler

Bu tez çalışmasında veri setinin eğitilmesi için beş farklı yapay sinir ağı mimarisi kullanılmış ve bunların başarısı karşılaştırılmıştır.

Eğitim sonrası test verilerinin doğruluk oranlarının doğru bir şekilde karşılaştırılması için veri seti tüm modeller için aynı ön işlemlerden geçirilmiş ve model için gerekli ortak parametreler eşit değerlerde belirlenmiştir.

- Eğitimin hızlı olması ve oluşturulan tüm modellere uyumlu olması açısından veriler öncelikle 32x32x3 boyutundan 32x32x1 boyutlu verilere dönüştürülerek üç kanallı yapıdan tek kanallı yapıya indirilmiştir.
- Tüm veriler boyut dönüşümü sonrası normalizasyon işlemine tabi tutulmuştur.
- Tüm modeller için eğitim döngüsü sayısı (epoch) 100 olarak belirlenmiş fakat modeller ardışık 10 eğitim döngüsünden sonra doğrulama değerinde iyileştirme göstermedikleri takdirde eğitim işlemi durdurulmuştur.
- Tüm modeller için doğrulama verisi oranı 0.4 olarak belirlenmiştir.
- Tek seferde modele verilecek olan örnek sayısı (batch size) 128 olarak belirlenmiştir.
- Tüm modeller için aynı eğitim, doğrulama ve test verileri kullanılmıştır.
- Tüm modeller için optimizasyon yöntemi olarak Adam optimizasyon metodu kullanılmıştır.
- Modellerin maliyetini ölçmek amacıyla kayıp fonksiyonu hesaplamasında çapraz doğrulama fonksiyonu kullanılmıştır.

Modellerde kullanılan ortak parametreler Tablo 5.1'de özetlenmiştir.

Tablo 5.1: Modellerde kullanılan ortak parametreler

Model	**Max. Epoch**	**Doğrulama Verisi Oranı**	**Batch Size**	**Optimizatör**	**Kayıp Fonksiyonu**
FFNN	100	%40	128	Adam	Çapraz doğrulama
CNN	100	%40	128	Adam	Çapraz doğrulama
RNN	100	%40	128	Adam	Çapraz doğrulama
LSTM	100	%40	128	Adam	Çapraz doğrulama
GRU	100	%40	128	Adam	Çapraz doğrulama

5.4.1 İleri Beslemeli Sinir Ağları ile Eğitim

Veri setini eğitmek amacıyla 4 katmanlı standart bir sinir ağı mimarisi oluşturulmuştur. 32x32'lik görüntü verileri olduğu için verinin yapay sinir ağına giriş boyutu 1024 olarak ayarlanmıştır.

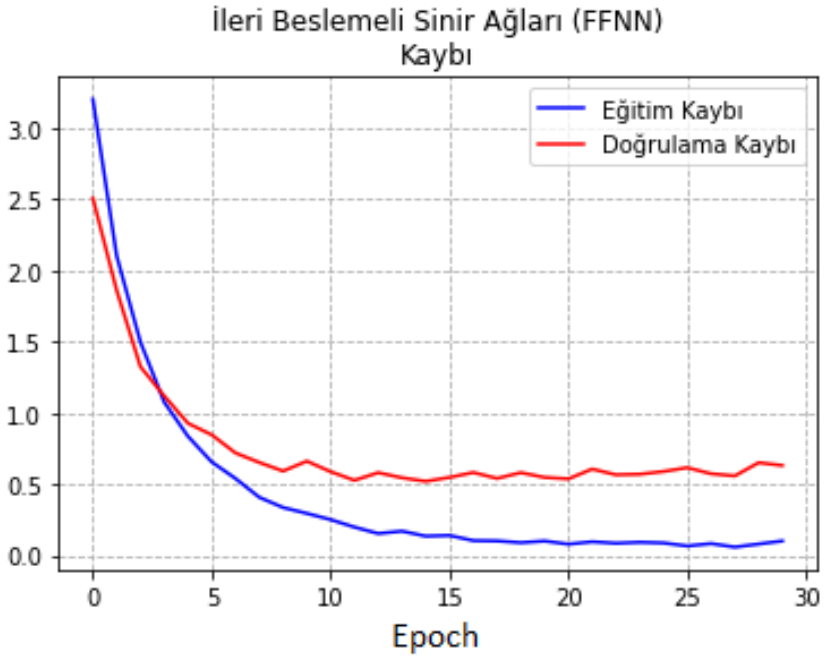
İlk gizli katmanda 384 nöron kullanılmış ve aktivasyon fonksiyonu ReLU olarak belirlenmiştir. İkinci ve üçüncü gizli katmanlarda da 384 nöron ve ReLU aktivasyon fonksiyonu kullanılmıştır. Çıkış katmanında ise 91 adet nöron kullanılmış ve aktivasyon fonksiyonu olarak softmax kullanılmıştır.

Toplamda bir adet giriş katmanı, üç adet gizli katman ve bir adet çıkış katmanından oluşan bu yapay sinir ağı modelinde, ilk gizli katmanında 393.600, ikinci ve üçüncü katmanında 147.840, çıkış katmanında 35.035 olmak üzere toplamda 724.315 adet parametre eğitilmiştir.

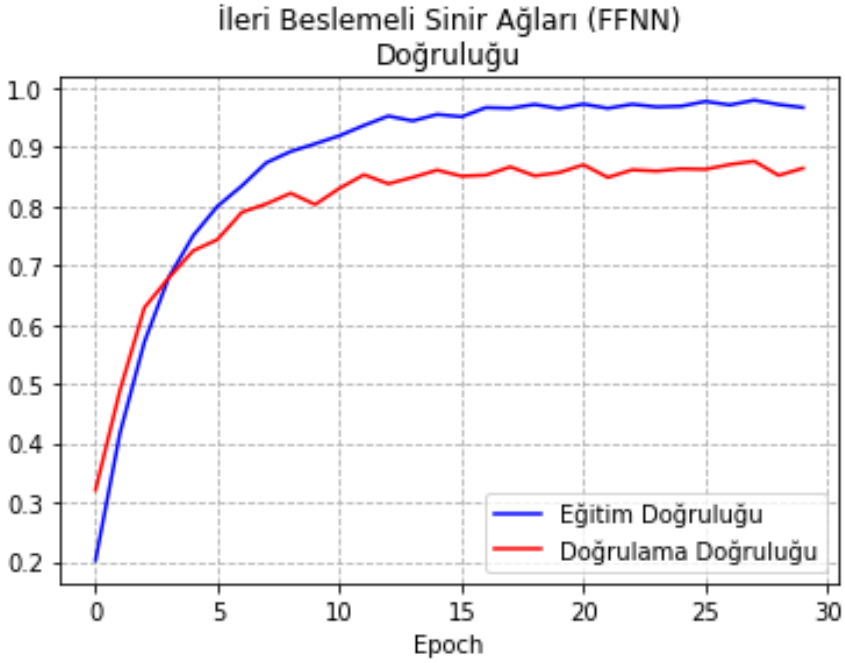
Model oluşturulurken eğitim adımı sayısı 100 olarak belirlenmiştir ancak ardışık 10 eğitimden sonra daha iyi bir doğrulama değeri bulamadığı için 30. eğitim adımında erken durdurma işlemiyle modelin eğitimi durdurulmuştur.

Oluşturulan model 455 test verisi üzerinde test edilmiştir. Yapılan test sonucunda model 455 veriden 384 tanesini doğru tahmin ederek %84,395 tahmin başarısı sağlamıştır.

Modelin eğitim sırasındaki kayıp fonksiyonun ve doğrulama değerinin değişimi Şekil 5.3 ve Şekil 5.4'te gösterilmiştir.



Şekil 5.3: İleri beslemeli sinir ağı modelinin kayıp fonksiyonu.



Şekil 5.4: İleri beslemeli sinir ağı modelinin eğitim ve doğrulama başarısı.

Test verilerinden bazıları ve modelin doğru tahmin sonuçları Tablo 5.1'de gösterilmiştir.

Tablo 5.2: İleri beslemeli derin sinir ağı modelinin bazı tahmin sonuçları.

Sınıf	Resim	Test Verisi Sayısı	Doğru Tahmin Sayısı
Hız Sınırı (20 km/s)		5	2
Öndeki Taşıtı Geçmek Yasaktır		5	4
Dur		5	4
Karşıdan Gelene Yol Ver		5	5
Girişi Olmayan Yol		5	5
Taşıt Trafiğine Kapalı Yol		5	5
Motorlu Taşıt Trafiğine Kapalı Yol (Motosiklet Hariç)		5	4
Motosiklet Giremez		5	4
Bisiklet Giremez		5	3
Mecburi Asgari Hız		5	4
Kamyon Giremez		5	5
Hız Sınırı (30 km/s)		5	4
Yaya Giremez		5	5
Okul Geçidi		5	5
Traktör Giremez		5	5
Motorlu Taşıt Giremez		5	4
Taşıt Giremez		5	5
Sağa Dönülmez		5	5
Sola Dönülmez		5	5
U Dönüşü Yapılamaz		5	4

5.4.2 Evrişimli Sinir Ağları ile Eğitim

Oluşturulan evrişimli sinir ağı modelinde veriler ilk evrişim katmanında 64 adet 5x5'li filtreden geçirilmiş ve aktivasyon fonksiyonu olarak ReLU kullanılmıştır.

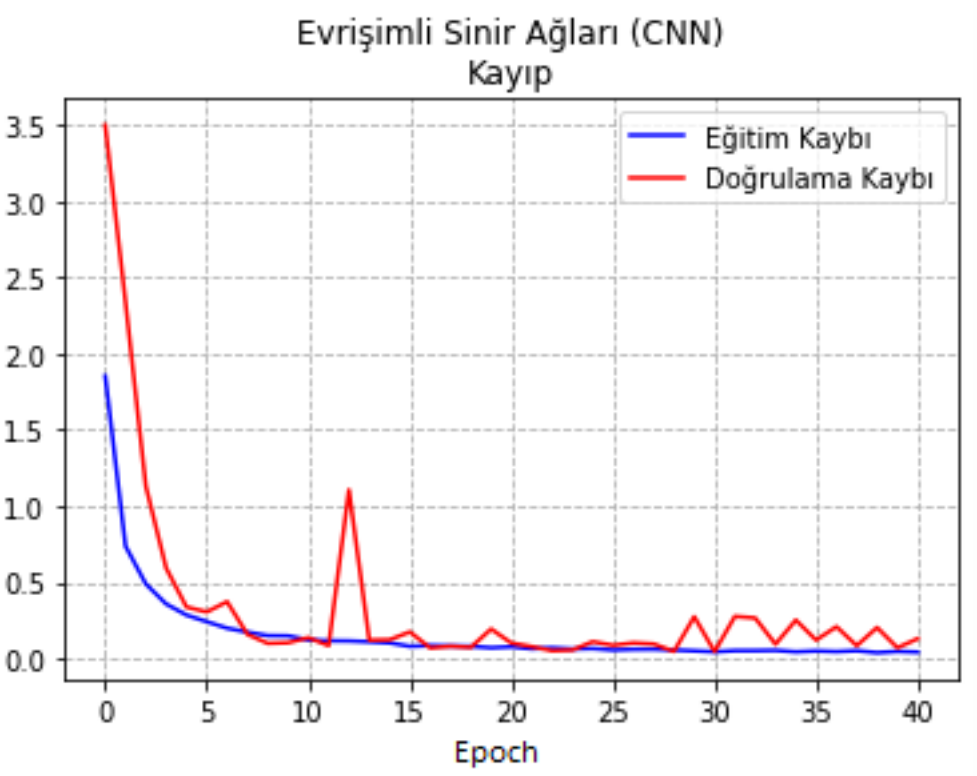
Birinci katmanın çıkışında elde edilen 28x28x32 boyuntundaki veriler ikinci evrişim katmanında, 32 adet 3x3'lük filtreden daha sonra ReLU aktivasyon fonksiyonundan geçirilmiştir. İkinci evrişim katmanından çıkan 26x26x32 boyutundaki veriler max havuzlama işlemine tabi tutulmuştur. Havuzlama katmanından çıkan 8x8x32 boyutundaki verilerin tam bağlı katmana uygun hale getirilmesi için düzleştirme katmanı kullanılmıştır.

Tam bağlı katmanda ise 512 nöronlu bir sinir ağı kullanılmış ve aktivasyon fonksiyonu ReLU olarak seçilmiştir. Çıkış katmanı olarak da 91 nöronlu ve aktivasyon fonksiyonu softmax olan bir sinir ağı kullanılmıştır.

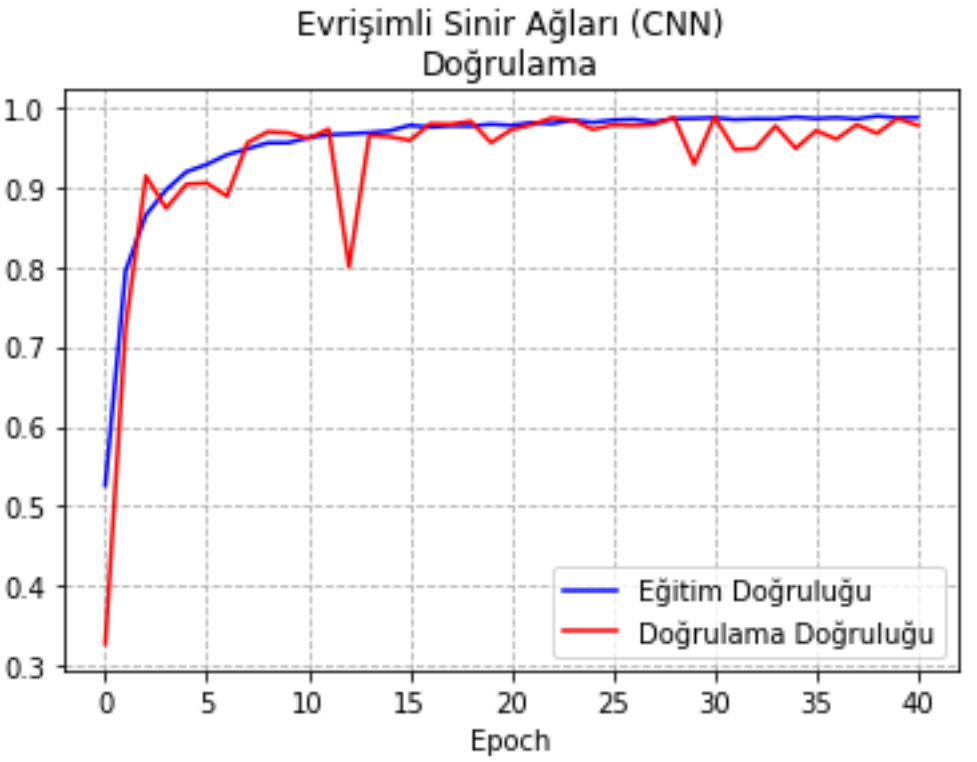
Eğtim adımı diğer modellerde olduğu gibi 100 adım olarak belirlenmiştir ancak ardışık 10 eğitim adımı sonrasında doğrulama değeri iyileşme göstermediği için 41 eğitim adımı sonrasında modelin eğitimi durdurulmuştur.

Oluşturulan bu model doğrultusunda modelde toplam 1.116.027 parametre eğitilmiştir. Eğitilen bu model 455 adet test verisinde test edilmiş ve 455 veriden 454 tanesini doğru tahmin ederek %99,78 tahmin başarısı sağlamıştır. Model sadece test verileri içinde bulanan "Sağdan Gidiniz" işaretini "Sağa Mecburi Yön" olarak tahmin ederek tek bir hata yapmıştır.

Modelin eğitim sırasındaki kayıp fonksiyonun ve doğrulama değerinin değişimi Şekil 5.5 ve Şekil 5.6'da gösterilmiştir.



Şekil 5.5: Evrişimli sinir ağı modelinin kayıp fonksiyonu.



Şekil 5.6: Evrişimli sinir ağı modelinin eğitim ve doğrulama başarısı.

Test verilerinden bazıları veri ve modelin doğru tahmin sonuçları Tablo 5.2'de gösterilmiştir.

Tablo 5.3: Evrişimli sinir ağı modelinin bazı tahmin sonuçları.

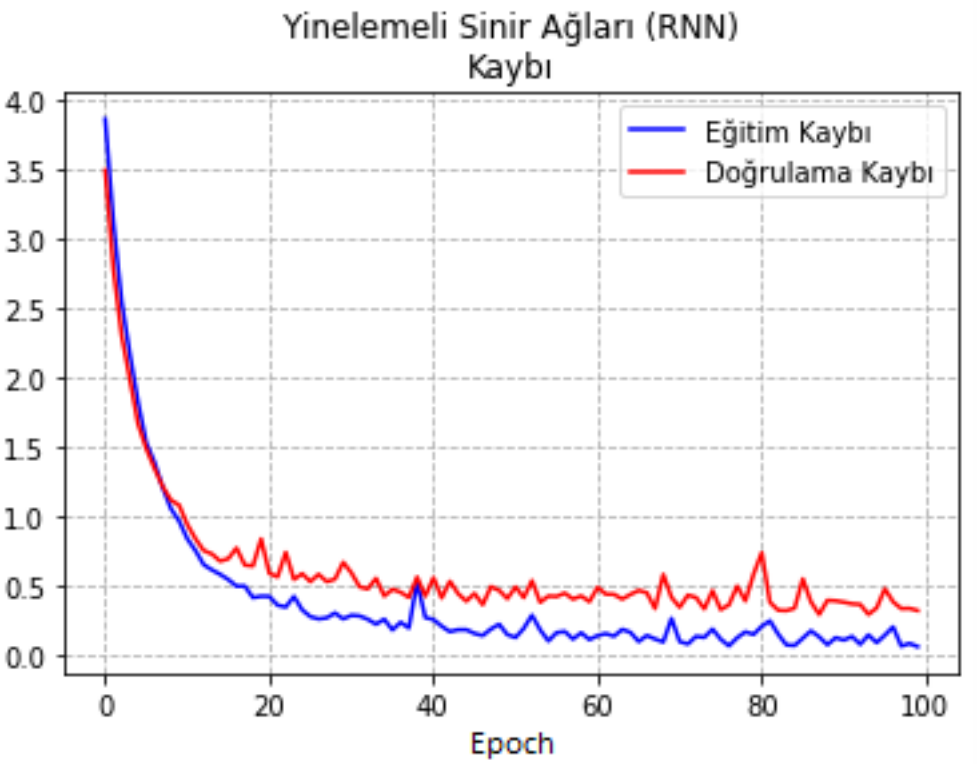
Sınıf	Resim	Test Verisi Sayısı	Doğru Tahmin Sayısı
Sağdan Gidiniz		5	4
Öndeki Taşıtı Geçmek Yasaktır		5	5
Dur		5	5
Karşıdan Gelene Yol Ver		5	5
Girişi Olmayan Yol		5	5
Taşıt Trafiğine Kapalı Yol		5	5
Motorlu Taşıt Trafiğine Kapalı Yol (Motosiklet Hariç)		5	5
Motosiklet Giremez		5	5
Bisiklet Giremez		5	5
Mecburi Asgari Hız		5	5
Kamyon Giremez		5	5
Hız Sınırı (30 km/s)		5	5
Yaya Giremez		5	5
Okul Geçidi		5	5
Traktör Giremez		5	5
Motorlu Taşıt Giremez		5	5
Taşıt Giremez		5	5
Sağa Dönülmez		5	5
Sola Dönülmez		5	5
U Dönüşü Yapılamaz		5	5

5.4.3 Yinelemeli Sinir Ağları ile Eğitim

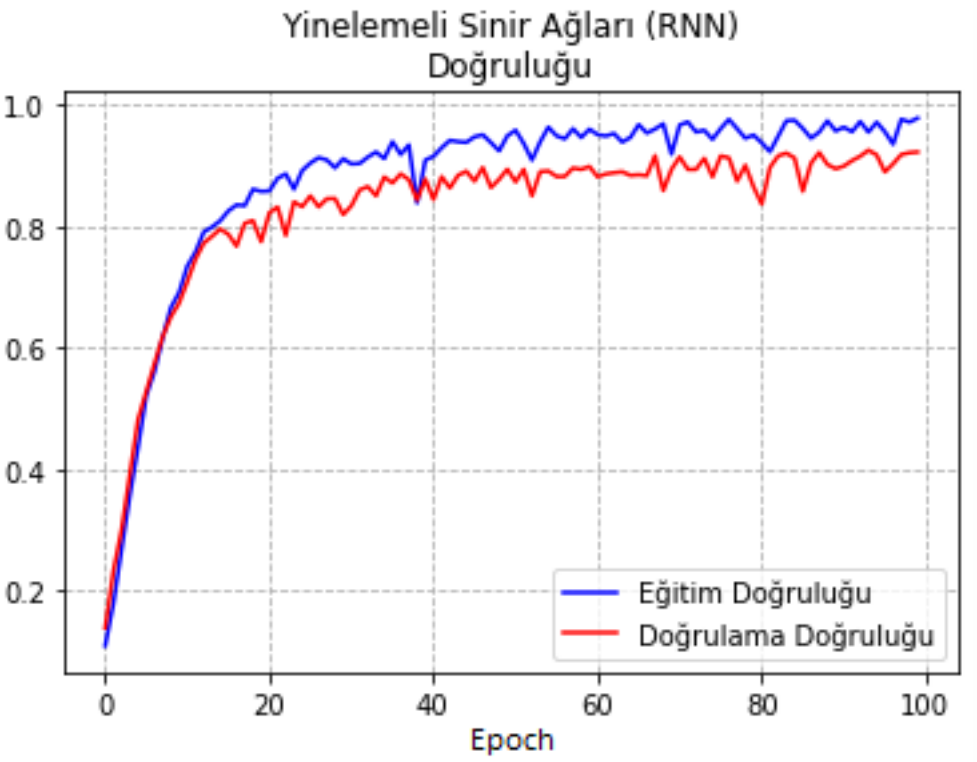
Oluşturulan bu modelde 128 yinelemeli nöron içeren ve aktivasyon fonksiyonu ReLU olan 3 adet yinelemeli katman kullanılmıştır. Bu katmanların çıkısı sınıflandırma için 91 nöronlu softmax aktivasyon fonksiyonlu standart bir sinir ağına giriş olarak bağlanmıştır.

Eğitim sırasında toplam 98,139 parametre eğitilmiştir. Eğitim karşılaştırmanın eşit olması için belirlenen 100 öğrenme adımının tamamını tamamlamış ve 455 test verisinin 408 tanesini doğru tahmin ederek %89.67'lik bir test başarısı elde etmiştir.

Modelin eğitim sırasındaki kayıp fonksiyonun ve doğrulama değerinin değişimi Şekil 5.7 ve Şekil 5.8'de gösterilmiştir.



Şekil 5.7: Yinelemeli sinir ağı modelinin kayıp fonksiyonu.



Şekil 5.8: Yinelemeli sinir ağı modelinin eğitim ve doğrulama başarısı.

Test verilerinden bazıları ve modelin doğru tahmin sonuçları Tablo 5.3'te gösterilmiştir.

Tablo 5.4: Yinelemeli sinir ağı modelinin bazı tahmin sonuçları.

Sınıf	Resim	Test Verisi Sayısı	Doğru Tahmin Sayısı
Hız Sınırı (20 km/s)		5	3
Öndeki Taşıtı Geçmek Yasaktır		5	4
Dur		5	5
Karşıdan Gelene Yol Ver		5	4
Girişi Olmayan Yol		5	5
Taşıt Trafiğine Kapalı Yol		5	5
Motorlu Taşıt Trafiğine Kapalı Yol (Motosiklet Hariç)		5	4
Motosiklet Giremez		5	5
Bisiklet Giremez		5	2

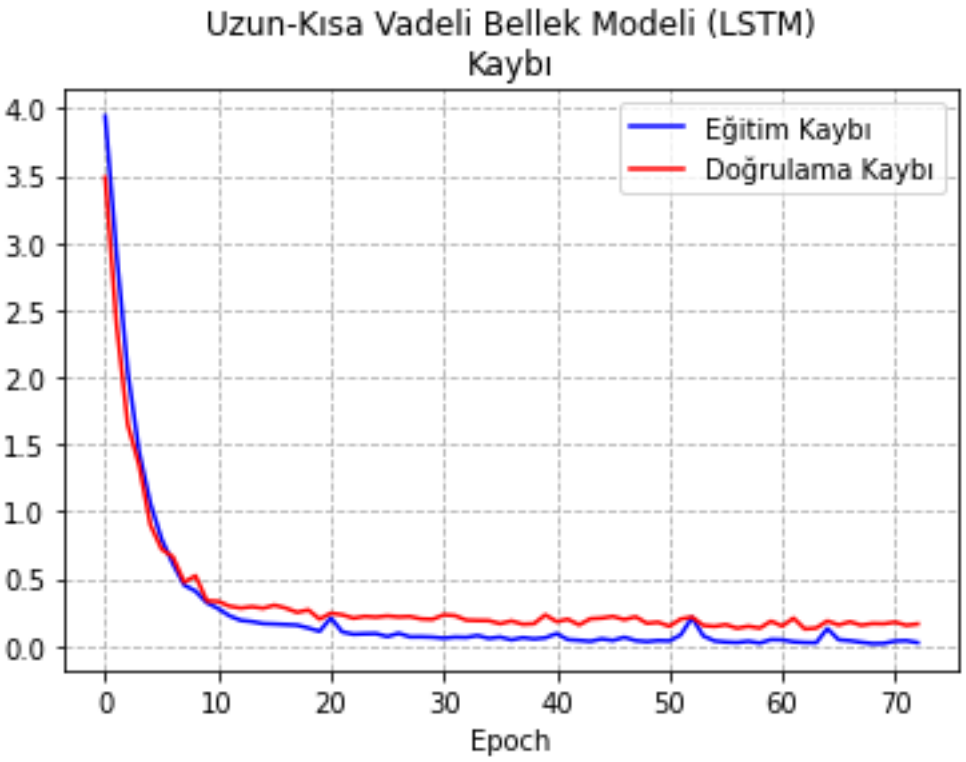
Mecburi Asgari Hız		5	4
Kamyon Giremez		5	5
Hız Sınırı (30 km/s)		5	3
Yaya Giremez		5	5
Okul Geçidi		5	5
Traktör Giremez		5	5
Motorlu Taşıt Giremez		5	4
Taşıt Giremez		5	5
Sağa Dönülmez		5	5
Sola Dönülmez		5	5
U Dönüşü Yapılamaz		5	5

5.4.4 Uzun-Kısa Vadeli Bellek Modeli ile Eğitim

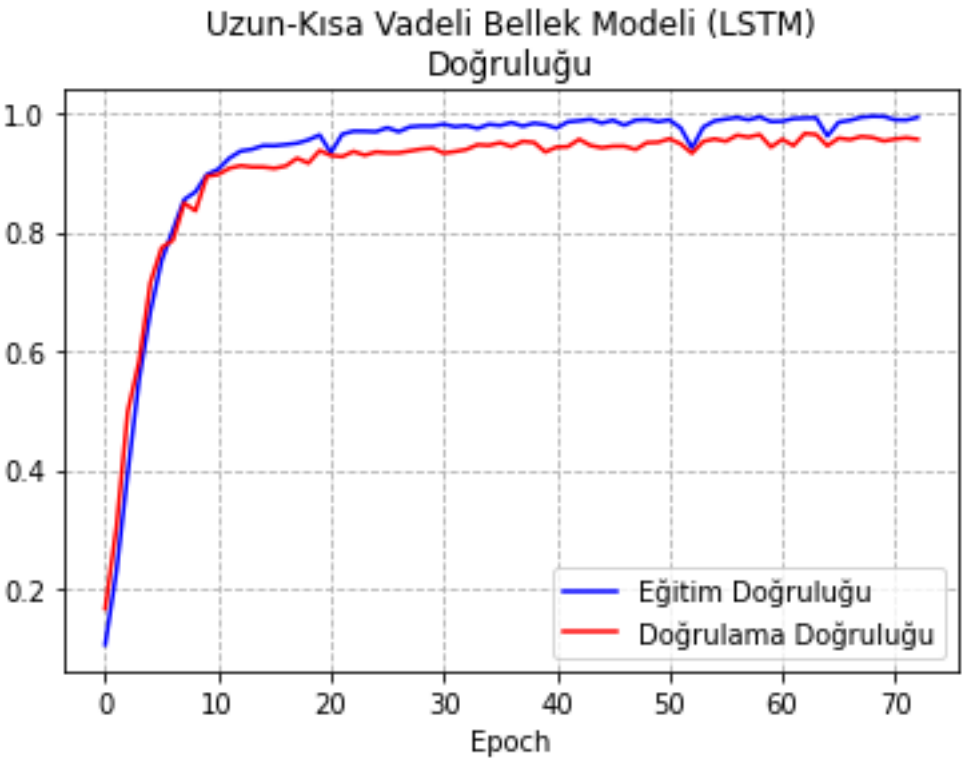
Oluşturulan bu model yinelemeli sinir ağlarında oluşturulan mimariye benzer şekilde oluşturulmuştur. 128 adet uzun-kısa vadeli bellek nöronu içeren ve aktivasyon fonksiyonu ReLU olan 2 adet uzun-kısa vadeli bellek katmanı kullanılmıştır. Bu katmanların sonuna 91 nöronlu ve softmax aktivasyon fonksiyonlu tanımlı standart sinir ağı katmanı eklenmiştir.

Eğitim sırasında toplam 225.755 parametre eğitilmiştir. Eğitim için belirlenen 100 öğrenme adımının doğrulama sonuçlarında iyileşme görülmediği için eğitim 72. öğrenme adımında otomatik olarak durdurulmuştur ve 455 test verisinin 431 tanesini doğru tahmin ederek %94,725'lik bir test başarısı elde etmiştir.

Modelin eğitim sırasındaki kayıp fonksiyonun ve doğrulama değerinin değişimi Şekil 5.9 ve Şekil 5.10'da gösterilmiştir.



Şekil 5.9: Uzun-kısa vadeli modelinin kayıp fonksiyonu.



Şekil 5.10: Uzun-kısa vadeli bellek modelinin eğitim ve doğrulama başarısı.

Test verilerinden bazıları ve modelin doğru tahmin sonuçları Tablo 5.4'te gösterilmiştir.

Tablo 5.5: Uzun-kısa vadeli bellek modelinin bazı tahmin sonuçları.

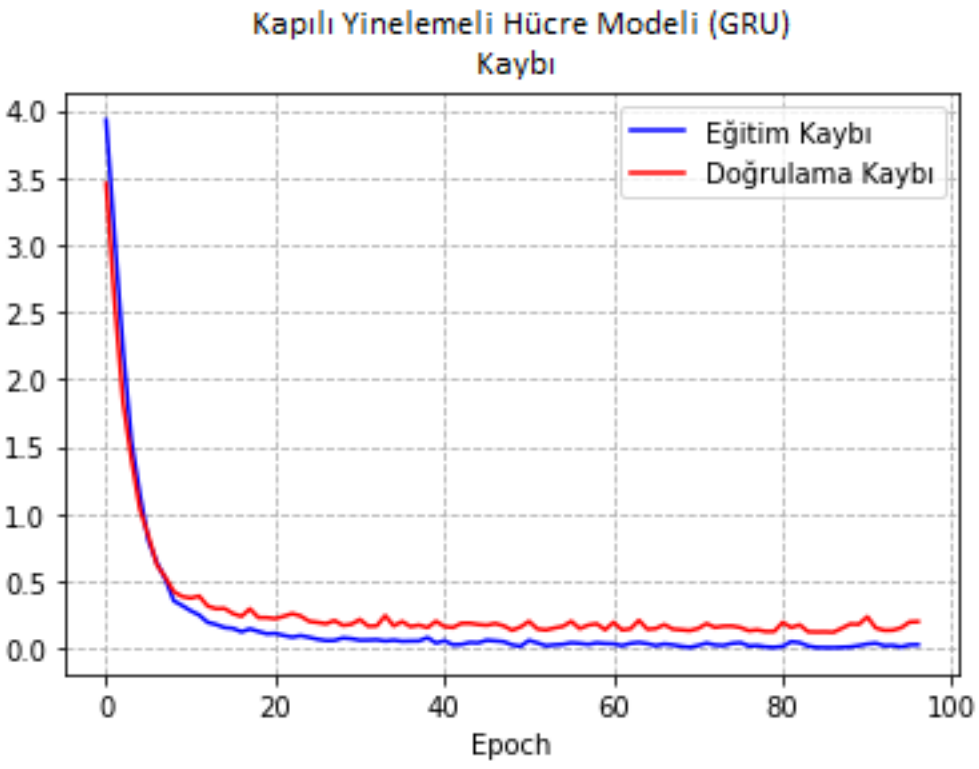
Sınıf	Resim	Test Verisi Sayısı	Doğru Tahmin Sayısı
Hız Sınırı (20 km/s)		5	3
Hız Sınırı (50 km/s)		5	2
Dur		5	5
Karşıdan Gelene Yol Ver		5	4
Girişi Olmayan Yol		5	5
Sola Tehlikeli Devamlı Virajlar		5	4
Motorlu Taşıt Trafiğine Kapalı Yol (Motosiklet Hariç)		5	5
Duraklamak ve Park Etmek Yasaktır		5	3
Bisiklet Giremez		5	5
Mecburi Asgari Hız		5	5
Kamyon Giremez		5	5
Hız Sınırı (30 km/s)		5	2
Yaya Giremez		5	4
Okul Geçidi		5	5
Traktör Giremez		5	5
Motorlu Taşıt Giremez		5	4
Taşıt Giremez		5	5
Sağa Dönülmez		5	5
Hız Sınırı (40 km/s)		5	4
U Dönüşü Yapılamaz		5	5

5.4.5 Kapılı Yinelemeli Birim Modeli ile Eğitim

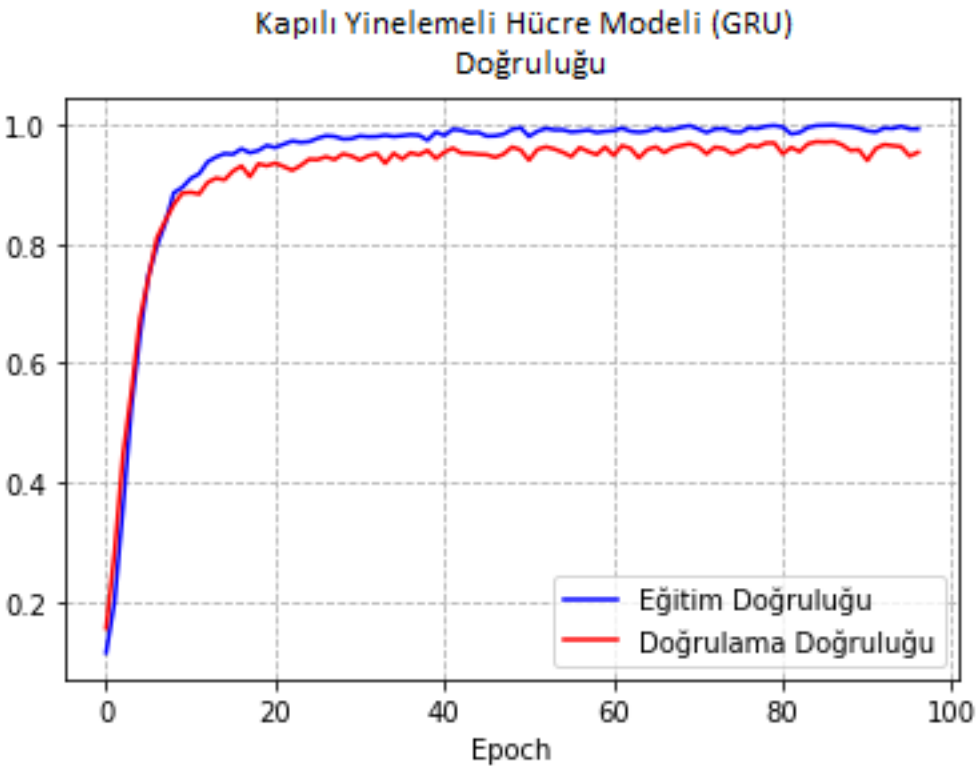
Tasarlanan kapılı yinelemeli hücre modeli uzun-kısa vadeli bellek modeline benzer şekilde tasarlanmıştır. İlk iki katman 128 adet kapılı yinelemeli hücre içeren ve aktivasyon fonksiyonu olarak ReLU kullanan katmandan oluşur. Bu katmanların çıkışı yine 91 adet nöron içeren ve aktivasyon fonksiyonu softmax olan sinir ağı katmanına bağlanmıştır.

Eğitim sırasında toplam 173.019 parametre eğitilmiştir. Eğitim sırasında doğrulama değerlerinde daha fazla iyileşme görülmediği için eğitim 96. Adımda otomatik olarak durdurulmuştur. Oluşturulan bu model 455 test verisinin 440 tanesini doğru tahmin ederek %94,725'lik bir test başarısı elde etmiştir.

Modelin eğitim sırasındaki kayıp fonksiyonun ve doğrulama değerinin değişimi Şekil 5.11 ve Şekil 5.12'de gösterilmiştir.



Şekil 5.11: Kapılı yinelemeli birim modelinin kayıp fonksiyonu.



Şekil 5.12: Kapılı yinelemeli birim modelinin eğitim ve doğrulama başarısı.

Test verilerinden bazıları ve modelin doğru tahmin sonuçları Tablo 5.5'te gösterilmiştir.

Tablo 5.6: Kapılı yinelemeli birim modelinin bazı tahmin sonuçları.

Sınıf	Resim	Test Verisi Sayısı	Doğru Tahmin Sayısı
Hız Sınırı (20 km/s)		5	3
Sağa Tehlikeli Devamlı Virajlar		5	4
Hız Sınırı (50 km/s)		5	2
Sola Tehlikeli Devamlı Virajlar		5	3
Girişi Olmayan Yol		5	5
Taşıt Trafiğine Kapalı Yol		5	5
Motorlu Taşıt Trafiğine Kapalı Yol (Motosiklet Hariç)		5	5
Motosiklet Giremez		5	5

Bisiklet Giremez		5	5
Mecburi Asgari Hız		5	5
Kamyon Giremez		5	5
Hız Sınırı (30 km/s)		5	4
Yaya Giremez		5	5
Sola Tehlikeli Viraj		5	4
Traktör Giremez		5	5
Motorlu Taşıt Giremez		5	5
Taşıt Giremez		5	5
Sağa Dönülmez		5	5
Sola Dönülmez		5	5
U Dönüşü Yapılamaz		5	5

6. SONUÇLAR VE ÖNERİLER

Bu tez çalışmasında 5 farklı yapay sinir ağı modeli kullanılmış ve oluşturulan bu modellerin 455 adet test verisi üzerindeki tahmin başarısı değerlendirilmiştir.

Modellerin tahmin başarısını değerlendirmek için yapay sinir ağı modellerinde ve derin öğrenme yöntemlerinde kullanılan ortak parametreler ve optimizayon yöntemleri gibi yöntemler birbirine eşit olarak seçilmiştir.

Yapılan deneyler sonucunda en iyi tahmin başarısını %99,78 oranla evrişimli sinir ağları göstermiştir. Bu sayede evrişimli sinir ağları görüntülerin makine öğrenimi metoduyla öğrenilmesi için en iyi model olarak belirlenmiştir. Diğer modellerin ve evrişimli sinir ağı modelinin tahmin başarısı Tablo 6.1'de özetlenmiştir.

Tablo 6.1: Oluşturulan modellerin tahmin başarısı.

Model	Tahmin Başarısı
Evrişimli Sinir Ağları Modeli	%99,78
Kapılı Yinelemeli Hücre Modeli	%96,703
Uzun-Kısa Vadeli Bellek Modeli	%94,725
Yinelemeli Sinir Ağları Modeli	%89,67
İleri Beslemeli Derin Sinir Ağı Modeli	%84,395

Oluşturulan modeller için tahmin başarısı iyileştirmek model parametreleri en uygun sonuca ulaşıncaya kadar tekrar ayarlanabilir veya oluşturulan derin ağ mimarilerinin katman sayıları artırılabilir. Fakat bu işlemler eğitim zamanı maliyetinin artmasına neden olacaktır. Modellerin eğitim süreleri Tablo 6.2'de verilmiştir.

Tablo 6.2: Modellerin eğitim süreleri

Model	Eğitim Süresi
Evrişimli Sinir Ağları Modeli	23 dakika
Kapılı Yinelemeli Hücre Modeli	11 dakika
Uzun-Kısa Vadeli Bellek Modeli	17 dakika
Yinelemeli Sinir Ağları Modeli	8 dakika
İleri Beslemeli Derin Sinir Ağı Modeli	38 saniye

Test başarısı sonuçları ve eğitim süreleri göz önünde bulundurulduğunda en iyi başarıyı evrişimli sinir ağı modeli göstersede eğitilecek olan parametre fazlalığından en uzun eğitim süresine sahip olan model olarak belirlenmiştir. Diğer modellere göre en kötü test sonucunu veren ileri beslemeli sinir ağları modeli ise diğer modellere göre çok çok daha düşük sürelerde eğitimi tamamlamıştır.

Bu çıkan sonuçlarla birlikte düşük bilgisayar sistemlerinde sistem kaynaklarının az olmasından dolayı eğitimin daha hızlı yapılabilmesi için ileri beslemeli sinir ağları kullanılabilir fakat bu durum özellikle resim sınıflandırma problemlerinde modelin test başarısını olumsuz yönde etkileyecektir. Diğer yandan da düşük sistemlerde evrişimli sinir ağlarının eğitim süresi diğer modellere göre sistem kaynaklarının azlığından dolayı çok daha fazla sürse de özellikle resim sınıflandırma problemlerinde test sonuçlarından da görüldüğü gibi çok iyi sonuçlar elde edilecektir.

İlerleyen dönemlerde yapılacak olan çalışmalarda bu tez çalışmasında oluşturulan evrişimli sinir ağı modeli kullanılarak bir sürücü destek sistemi oluşturulması planlanmaktadır. Bu sistemin araç içerisine yerleştirilen araç içi kameradan alınan görüntüleri bu tezde önerilen ve en iyi tahmin sonucunu veren evrişimli sinir ağı modelini kullanarak görüntü taraması yapıp yol bilgilerini sürücüye yansıtması planlanmaktadır.

7. KAYNAKLAR

Bishop, C. M., *Pattern Recognition and Machine Learning*, Springer, (2006).

Bueno, R. V., González, A.G., Gordo, E. T., Pita, R. G. and Zurera, M. R., "Traffic Sign Classification by Image Preprocessing and Neural Networks", *International Work-Conference on Artificial Neural Networks*, Berlin-Heidelberg, 741-748, (2007).

Chandupatla, A., Software 2.0- Playing with Neural Networks (Part 1)[online], (11 Ocak 2021), https://towardsdatascience.com/software-2-0-playing-with-neural-networks-part-1-5e9d50fa5422, (2019).

Cho, K., Merrienboer, B. V., Gülçehre, Ç., Bahdanau, D., Bougares, F., Schwenk, H. and Bengio, Y., "Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation", *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Doha-Qatar, 1724-1734, (2014).

Dabbura, I., Gradient descent algorithm and its variants[online], (11 Ocak 2021), https://towardsdatascience.com/gradient-descent-algorithm-and-its-variants-10f652806a3, (2017).

Deshpande, A., A Beginner's Guide To Understanding Convolutional Neural Networks Part 2[online], (11 Ocak 2021), https://adeshpande3.github.io/A-Beginner's-Guide-To-Understanding-Convolutional-Neural-Networks-Part-2, (2016).

Dong, X. and Zhou, D. X., "Learning gradients by a gradient descent algorithm", *Journal of Mathematical Analysis and Applications*, 341 (2), 1018-1027, (2008).

Donges, N., Gradient descent in a nutshell[online], (11 Ocak 2021), https://towardsdatascience.com/gradientdescent-in-a-nutshell-eaf8c18212f0, (2018.)

Elman, J. L., "Finding structure in time", *Cognitive Science*, 14 (2), 179-211, (1990).

Ergüder, H., Recurrent Neural Network nedir?[online], (11 Ocak 2021), https://medium.com/@hamzaerguder/recurrent-neural-network-nedir-bdd3d0839120, (2018).

Geron, A., *Hands-On Machine Learning with Scikit-Learn and TensorFlow*, O'Reilly Media, (2017).

Golik, P., Doetsch, P. and Ney, H., "Cross-Entropy vs. Squared Error Training: a Theoretical and Experimental Comparison", *Interspeech*, (2013).

Goodfellow, I. Bengio, Y. and Courville, A., *Derin Öğrenme*, (Çev: F. Y. Vural, R. G. Cinbiş, S. Kalkan), Buzdağı Yayınevi, (2018).

Güzel, K., Geri Yayılımlı Çok Katmanlı Sinir Ağları-1[online], (11 Ocak 2021), https://kadirguzel.medium.com/geri-yay%C4%B1l%C4%B1ml%C4%B1-%C3%A7ok-katmanl%C4%B1-yapay-sinir-a%C4%9Flar%C4%B1-1-47daa3856247, (2018).

Hammer, B., "On the approximation capability of recurrent neural networks", *Neurocomputing*, 31 (1-4), 107-123, (2000).

Hannan, M. A., Wali S.B., Pin, T.J., Hussain, A. and Samad S. A., "Traffic sign classification based on neural network for advance driver assistance system", *Przeglad Elektrotechniczny*, (2014).

Hochreiter, S. and Schmidhuber, J., "Long Short-Term Memory", *Neural Computation*, 9 (8), 1735-1780, (1997).

Huang, Z., Yu, Y., Gu, J. and Liu, H., "An Efficient Method for Traffic Sign Recognition Based on Extreme Learning Machine", 47 (2), 920-933, (2017).

Ippolito, P. P., Roadmap to Computer Vision[online], (11 Ocak 2021), https://www.kdnuggets.com/2020/10/roadmap-computer-vision.html, (2020).

Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S. and Darrell, T., "Caffe: Convolutional architecture for fast feature embedding", *In Proceedings of the 22Nd ACM International Conference on Multimedia, MM '14*, New York-USA, 675-678, (2014).

Jordan, J., Setting the learning rate of your neural network[online], (11 Ocak 2021), https://www.jeremyjordan.me/nn-learning-rate, (2018).

Jordan, M. I. "Attractor dynamics and parallelism in a connectionist sequential machine", *IEEE Press*, 112-127, (1990).

Kava, A., That's Not Enough, We Have to Go Deeper[online], (11 Ocak 2021), https://www.linkedin.com/pulse/thats-enough-we-have-go-deeper-arjun-kava, (2018).

Kingma, D. P. and Ba, J. L., “Adam: A method for stochastic optimization”, *ICLR*, (2015).

Kızrak, M. A., Derin Öğrenme İçin Aktivasyon Fonksiyonlarının Karşılaştırılması[online], (11 Ocak 2021), https://ayyucekizrak.medium.com/derin-%C3%B6%C4%9Frenme-i%C3%A7in-aktivasyon-fonksiyonlar%C4%B1n%C4%B1n-kar%C5%9F%C4%B1la%C5%9Ft%C4%B1r%C4%B1lmas%C4%B1-cee17fd1d9cd, (2019).

Kızrak, M. A., Şu Kara Kutuyu Açalım: Yapay Sinir Ağları[online], (11 Ocak 2021), https://ayyucekizrak.medium.com/%C5%9Fu-kara-kutuyu-a%C3%A7alim-yapay-sinir-a%C4%9Flar%C4%B1-7b65c6a5264a, (2018).

Krizhevsky, A., Sutskever, I. and Hinton, G. E., ImageNet Classification with Deep Convolutional Neural Networks, *Neural Information Processing Systems*, (2012).

Lewis, N. D., *Deep Time Series Forecasting with Python*, (2016).

Minsky, M. L. and Papert, S. A., “Review of 'Perceptrons: An Introduction to Computational Geometry'”, *IEEE Press*, 15 (6), 738-739, (1969).

Miura, J., Kanda, T. and Shirai, Y., “An active vision system for real-time traffic sign recognition”, *IEEE Intelligent Transportation Systems*, Dearborn, MI, USA, (2000).

Nair, V. and Hinton, G. E., “Rectified linear units improve restricted boltzmann machines”, *Proceedings of the 27th International Conference on International Conference on Machine Learning*, USA, 807-814, (2010).

Nigam, V., Understanding Neural Networks. From Neuron to RNN, CNN, and Deep Learning[online], (11 Ocak 2021), https://medium.com/analytics-vidhya/understanding-neural-networks-from-neuron-to-rnn-cnn-and-deep-learning-cd88e90e0a90, (2018).

Plaut, C. P. and Hinton, G. E., “Learning sets of filters using back-propagation”, *Computer Speech & Language*, 2 (1), 35-61, (1987).

Robinson, A. J. and Fallside, F., “The Utility Driven Dynamic Error Propagation Network”, *Cambridge University Engineering Department*, (1987).

Rohrer. B., How convolutional neural networks work[online], (11 Ocak 2021), https://e2eml.school/how_convolutional_neural_networks_work.html#:~:text=Each%20image%20the%20CNN%20processes%20results%20in%20a%20vote.&text=After%20doing%20this%20for%20every,the%20set%20of%20labeled%20images., (2016).

Rosenblatt, F., "The perceptron: A probabilistic model for information storage and organization in the brain", *Psychological Review*, 65 (6), 386-408, (1958).

Saleh, H., *The Deep Learning With Pytorch Workshop*, Packt Publishing, (2020).

Ser, G. and Bati C. T., "Derin Sinir Ağları ile En İyi Modelin Belirlenmesi: Mantar Verileri Üzerine Keras Uygulaması", *Yüzüncü Yıl Üniversitesi Tarım Bilimleri Dergisi*, 29 (3), 406-417, (2019).

Shustanov, A. and Yakimov P., "CNN Design for Real-Time Traffic Sign Recognition", *Information Technology and Nanotechnology*, Samara-Russia, 718-725, (2017).

SuperDataScience Team, Convolutional Neural Networks (CNN): Step 4 - Full Connection[online], (11 Ocak 2021), https://www.superdatascience.com/blogs/convolutional-neural-networks-cnn-step-4-full-connection, (2018).

Thanh, H. N., "Morphological classification for traffic sign recognition", *Faculty of Electrical and Electronics Engineering, HCMC University of Technical Education*, (2014).

Turing, A. M., "Computing Machinery and Intelligence", *Mind*, 433-460, (1950).

Werbos, P. J., "Backpropagation through time: what it does and how to do it", *IEEE Press*, 78 (10), 1550-1560, (1990).

Widrow B. and Hoff, M., "Adaptive Switching Circuits", *Ire Wescon Convention Record*, 96-104, (1960).

Yücesan, E., Traffic Sign Images From Turkey[online], (11 Ocak 2021), https://www.kaggle.com/erdicem/traffic-sign-images-from-turkey, (2020).

8. ÖZGEÇMİŞ

Adı Soyadı	: Ahmet YAVUZ
Doğum Yeri ve Tarihi	: Tire/1994
Lisans Üniversite	: Pamukkale Üniversitesi/Denizli
Elektronik posta	: ahmetyavuz1994@gmail.com
İletişim Adresi	: Zeytinköy Mah. Yunus Emre Cad. NO:56 Pamukkale/Denizli