10 Şubat 2021

İSTANBUL TEKNİK ÜNİVERSİTESİ
Elektrik Elektronik Fakültesi

YAPAY SİNİR AĞLARI (EHB 420)
2020-2021 Güz Öğretim Dönemi
DÖNEM PROJESİ

Öğretim Üyesi: Prof. Dr. Neslihan Serap ŞENGÖR
Dersin Yardımcısı: Araş. Gör. Halil DURMUŞ

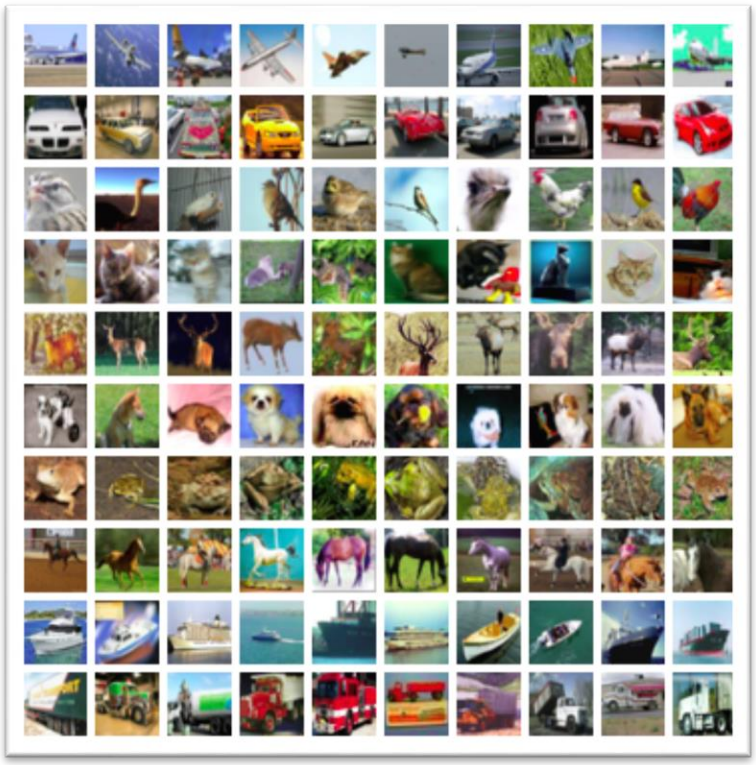
Recep Salih YAZAR
040170066

Ramazan Umut AKTAŞ
040190762

CIFAR10 VERİ KÜMESİ

Kullandığımız veri seti makine öğrenmesi, yapay sinir ağları, bilgisayar görüsü algoritmalarını eğitmek ve test etmek için kullanılan resimlerden oluşmaktadır. On sınıftan oluşan bu veri kümesinde her sınıfta altı bin tane resim bulunmaktadır. Resimler RGB formatındadır ve 32x32 boyutundadır. Sınıflar uçak, otomobil, kuş, kedi, geyik, köpek, kurbağa, at, gemi ve kamyon resimlerinden oluşmaktadır. Aşağıdaki figürde veri kümesine ait resim örnekleri gösterilmektedir.

FİGÜR-1



Veri kümesindeki resimlerin elli bin tanesi eğitim kümesinde, on bin tanesi ise test kümesinde kullanılmaktadır. Test için her sınıftan eşit sayıda örnek alınmıştır ve örnekler rastgele seçilmiştir.

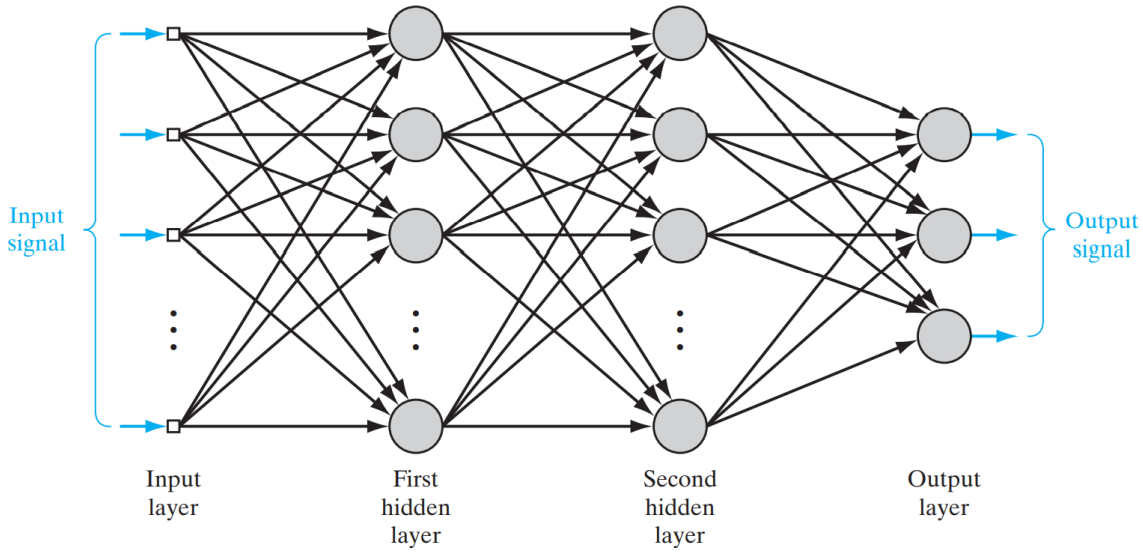
PROJENİN KAPSAMI

Projede CIFAR10 veri kümesini kullanarak çok katmanlı algılayıcı (ÇKA) eğitilmiştir. Çok katmanlı algılayıcıyı makine öğrenmesi uygulamaları için geliştirilen TensorFlow platformu ve Keras kütüphanesi kullanılmıştır. Ara fonksiyonlar için Scikit-Learn kütüphanesi kullanılmıştır. Bu şekilde oluşturulan çok katmanlı algılayıcının performansı, çeşitli parametrelerin, algoritmaların öğrenme sürecine etkisi tartışılmıştır. Makine öğrenmesi uygulamaları için yazılmış kütüphane ve platformları kullanmadan oluşturduğumuz çok katmanlı algılayıcı da CIFAR10 veri kümesi ile eğitilmiştir ve hazır fonksiyonlar kullanarak oluşturduğumuz ağ ile performansı karşılaştırılmıştır. Ardından görüntü sınıflandırma uygulamaları için sıkça kullanılan evrişimli sinir ağı (ESA) modeli aynı veri kümesi ile eğitilmiş ve test edilmiştir. Çok katmanlı algılayıcı ile evrişimli sinir ağının başarımları karşılaştırılmıştır ve sonuçlar yorumlanmıştır.

ÇOK KATMANLI ALGILAYICI

Çok katmanlı algılayıcı giriş katmanı, gizli katmanlar ve çıkış katmanından oluşan bir yapay sinir ağı yapısıdır. Ağın yapısı aşağıdaki figürde (figür-2) gösterilmiştir.

FİGÜR-2



Ağda bulunan nöronlar nonlineer ve en az bir kez türevlenebilir aktivasyon fonksiyonlarına sahiptirler. Çok katmanlı algılayıcı sık sinaptik bağlantılara

sahiptir ve oluşturduğumuz modellerde yoğun katmanlar olarak adlandırılan tüm nöronların diğer nöronlarla bağlantılı olduğu katmanlar kullanılmıştır. Ağın çalışması iki aşamadan oluşmaktadır. İlk aşamada ağa verilen girdiler sinaptik ağırlıklarla çarpılır ve ilgili nörona iletilir. Nöron bu girdiği aktivasyon fonksiyonundan geçirir nöronun çıktısı bir sonraki katmanın girişi olacak şekilde işlem çıkış katmanına kadar devam eder. Çıkış katmanında ağ çıktısı oluşturur. Bu aşamaya kadar olan kısım ileri yayılımdır. Ağın ağırlıklarının güncellenerek eğitilmesi ise geriye yayılım kısmında gerçekleşir. Ağdan beklenilen çıkış işaretleri ile ağın verdiği çıkış işaretleri arasındaki fark alınarak hata işareti oluşturulur. Bu hata işaretine göre ilk olarak çıkış katmanı ağırlıkları güncellenir. Ardından çıkış katmanından giriş katmanına kadar sırasıyla katmanların ağırlıkları güncellenir. Güncelleme işleminde farklı algoritmalar (Gradyan Azalması Algoritması, Stokastik Gradyan Azalması Algoritması, Adaptif Gradyan Algoritması vb.) kullanılabilir. Kullanılan algoritmaya ve ağın gerçekleştirmesi beklenilen işe göre performans değişebilir. Bu değişimleri gözlemlemek için makine öğrenmesi kütüphanelerini kullandığımız ağda farklı algoritmaların performansını test ettik. Kütüphane kullanmadığımız ağda ise Gradyan Azalması Algoritmasını kullandık. İlgili kısımlarda bu algoritmaların çalışma prensipleri açıklanmıştır. Sınıf bilgilerini 10x1 boyutundaki vektörler ile temsil ettik. Dolayısıyla oluşturduğumuz tüm ağların çıkışında on nöron vardır. Uçak, otomobil, kuş, kedi, geyik, köpek, kurbağa, at, gemi ve kamyon sınıflarını temsil eden vektörler sırasıyla aşağıda gösterilmiştir.

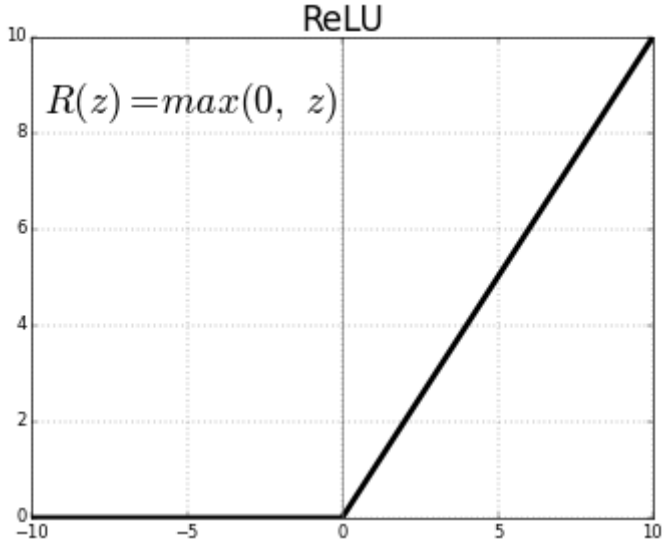
$$\begin{bmatrix}1\\0\\0\\0\\0\\0\\0\\0\\0\\0\end{bmatrix}\ \begin{bmatrix}0\\1\\0\\0\\0\\0\\0\\0\\0\\0\end{bmatrix}\ \begin{bmatrix}0\\0\\1\\0\\0\\0\\0\\0\\0\\0\end{bmatrix}\ \begin{bmatrix}0\\0\\0\\1\\0\\0\\0\\0\\0\\0\end{bmatrix}\ \begin{bmatrix}0\\0\\0\\0\\1\\0\\0\\0\\0\\0\end{bmatrix}\ \begin{bmatrix}0\\0\\0\\0\\0\\1\\0\\0\\0\\0\end{bmatrix}\ \begin{bmatrix}0\\0\\0\\0\\0\\0\\1\\0\\0\\0\end{bmatrix}\ \begin{bmatrix}0\\0\\0\\0\\0\\0\\0\\1\\0\\0\end{bmatrix}\ \begin{bmatrix}0\\0\\0\\0\\0\\0\\0\\0\\1\\0\end{bmatrix}\ \begin{bmatrix}0\\0\\0\\0\\0\\0\\0\\0\\0\\1\end{bmatrix}$$

Ağımızın girişi ise 32×32×3 boyutundaki resimlerin vektörize edilmesi ile oluşturulmuştur. RGB formatındaki renkler sıfır ile 255 arasında değerler ile temsil edildiği için bu vektörler 255'e bölünerek normalize edilmiştir. Dolayısıyla ağın girişi 3072×1 boyutundaki vektörlerden oluşmaktadır.

1) Makine Öğrenmesi Kütüphaneleri ile Oluşturulan Çok Katmanlı Algılayıcı

Oluşturduğumuz çok katmanlı algılayıcı altı katmandan oluşmaktadır. Bu katmanlardaki nöronların hepsi sonraki katmanlardaki nöronların hepsi ile bağlantı yapmaktadır, yani ağın katmanları yoğun (dense) olacak şekilde oluşturulmuştur. Sigmoid fonksiyonu yüksek değerlerde 1, düşük değerlerde ise 0 civarında kalmaktadır. Hiperbolik tanjant fonksiyonu ise yüksek değerlerde 1 düşük değerlerde ise -1 civarında doygunluk (satürasyon) göstermektedir. Ayrıca bu fonksiyonlar sadece orta noktalarındaki değişimlere karşı hassastırlar. Kısıtlı hassasiyet ve satürasyon eğitim sırasında ağırlıkların yeteri kadar değişmesinin önüne geçebilir ve bu durum ağ performansını sınırlar. Ayrıca hatanın geri yayılımı sırasında sigmoid ve hiperbolik tanjant fonksiyonları kullanılan çok katmanlı ağlarda katman sayısının artması ile gradyan giriş katmanına yakınlaştıkça önemli ölçüde azalmaktadır. Böylece girişe yakın katmanlarda hatanın etkisi küçülmektedir. Bu probleme kaybolan gradyan problemi denir. Kaybolan gradyan probleminin önüne geçmek ve hassaslığı arttırmak için ara katmanlarda doğrultulmuş lineer birim fonksiyonu (ReLU) kullanılmıştır. Aşağıdaki figürde (figür-3) ReLU fonksiyonunun grafiği gösterilmiştir.

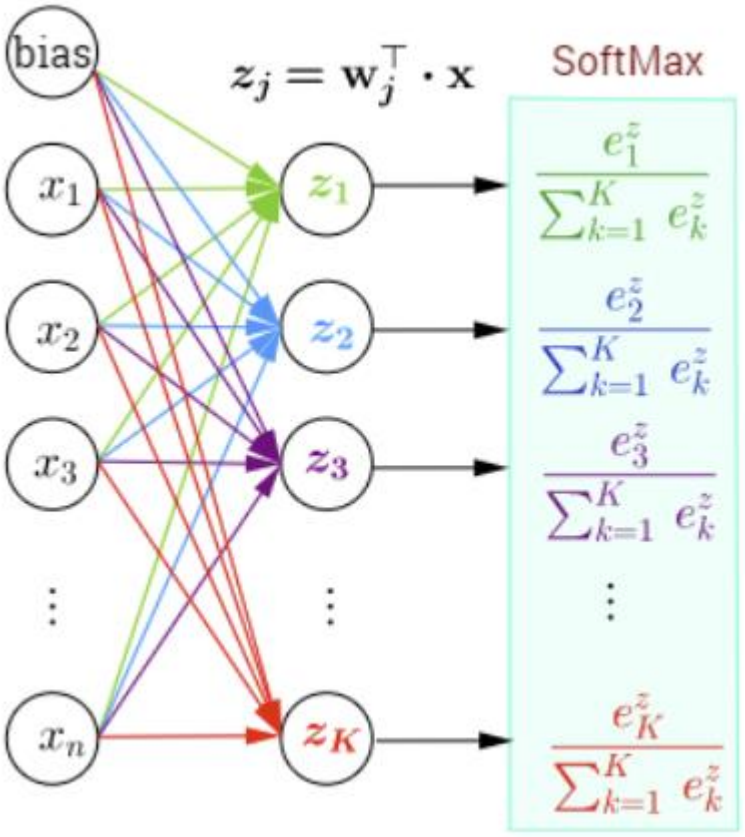
FİGÜR-3



ReLU fonksiyonu sıfır ve sıfırdan küçük değerler için sıfır verirken pozitif değerler için aldığı fonksiyon argümanını çıktı olarak verir. Bu aktivasyon fonksiyonunun kullanılması ile sadece gradyan kaybolması probleminin **önüne geçilmez, ayrıca kompleksliği düşürerek ağı hızlandırır. Sınıfları**

temsil ederken kullandığımız vektörlerin tüm elemanlarının pozitif olması ve ağın girdilerini oluşturan vektörlerin elemanlarının tamamının da pozitif olması dolayısıyla ReLU fonksiyonunun negatif değerler almaması bir problem yaratmayacaktır. Çıkış katmanında ise aktivasyon fonksiyonu olarak SoftMax fonksiyonu kullanıldı. Bu fonksiyon sınıflandırma problemlerinde sıkça kullanılan bir aktivasyon fonksiyonudur. Olasılık teorisi perspektifinden SoftMax fonksiyonunun çıktısı k adet sınıfı temsil edecek çıktılar ile ilişkili bir olasılık dağılımı olarak görülebilir. Aslında SoftMax model parametreleri için optimal en çok olabilirlik tahmin edicisidir. SoftMax aktivasyon fonksiyonu kullanılan bir katmanın çıkışı aşağıdaki figürde (figür-4) gösterilmiştir.

FİGÜR-4



Yukarıda görüldüğü gibi fonksiyonun çıktısı j. nöronun çıktısını tüm nöronların çıktısına bölerek hesaplanır. Bu fonksiyonun çıktısı sıfır ile bir arasında değerler alır.

Ağırlıkların güncellenmesi esnasında kullanılan kayıp (loss) fonksiyonunu şu ana kadar ortalama karesel hata kullanarak hesapladık. Ancak çıkış

katmanı aktivasyon fonksiyonunun SoftMax olarak belirlendiği durumlarda kategorik çapraz entropi kullanılmasının daha iyi sonuçlar verdiği gözlemlenmiştir. Aslında bu kayıp fonksiyonu ayrık olasılık dağılımlarının birbirlerinden ayırt edilebilirliğinin bir ölçüsüdür. Kategorik çapraz entropi (KÇE) aşağıdaki gibi tanımlanır.

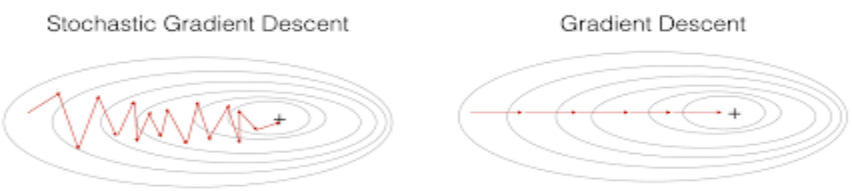
$$KÇE = -\sum_{i=1}^{Ks} y_{i,d} \log(Y_i)$$

Ks, kategori sayısını
$y_{i,d}$, ağın i. çıktısında karşılık gelen istenen sınıf bilgisini
Y_i, ağ çıktısının i. elemanını,

temsil etmektedir.

Farklı algoritmaların etkisini test etmek için çok katmanlı algılayıcı çeşitli algoritmalar ile eğitilmiştir. Ancak algoritmaların etkisini incelemediğimiz kısımda ağda stokastik gradyan azalması (SGD) algoritması kullanılıştır. Stokastik gradyan azalması algoritması girdilerin oluşturduğu çok boyutlu bir uzayda iteratif olarak gradyanı minimize ederek yerel bir minimumun arandığı gradyan azalması algoritmasının büyük verilerde getirdiği işlem fazlalığını aşmak için oluşturulmuştur. Gradyan azalması (GD) algoritmasında tüm veriler üzerinden hesaplama yapılırken SGD kullanıldığında bu veriler içerisinden rastgele veriler seçilerek aynı işlem gerçekleşir. Bu durum öğrenme sürecinin gürültülü olmasına yol açar ama yeterince iterasyon yapıldığında lokal minimuma yaklaşılırsa bu gürültünün etkisi azalır. Gradyan azalması çok boyutlu bir yüzey üzerinde bir başlangıç noktasından gradyanların en hızlı azaldığı doğrultu hesaplanarak (1.mertebe veya daha yüksek mertebe yaklaşıklar ile) lokal minimuma doğru iteratif olarak yaklaşılma algoritması olarak görülebilirken SGD, bu yüzey üzerinde rastgele azalış olan doğrultularda rastgele bir nokta seçilerek iteratif olarak yerel minimuma yaklaşmayı sağlayan algoritmadır. Düzey eğrileri (level set) verilen bir yüzeyde SGD ve GD algoritmaları ile yerel minimuma yakınsamanın farkı aşağıdaki figürde (figür-5) gösterilmiştir.

FİGÜR-5



Çok katmanlı algılayıcının katmanlarında ayrıca dropout özelliğini kullandık. Katman sayısı fazla olan, derin ağlar girdi olarak verilen veriye fazla uyum (overfit) sağlayabilir. Fazla uyum halinde ağ eğitim için kullanılan veriler haricindeki verileri sınıflandırma başarısı düşer. Bu durumun önüne ağa bir miktar gürültü verilebilir ancak ağ bu gürültüye de fazla uyum sağlaması durumu gerçekleşirse istenilenden daha kötü sonuçlar da alınabilir. Bu durumun önüne geçmenin bir yolu da dropout olarak adlandırılan bir düzenleme uygulamasına başvurmaktadır. Ağın aşırı uyum sağlamasını engellemek için eğitim esnasında katmanlardaki belirli sayıda rastgele seçilen nöronun göz ardı edilmesi ile sağlanabilir. Bu nöronun ağırlıklarının o iterasyonda değiştirilmemesi ile sağlanır. Bu uygulamaya dropout denir. Dropout etkisinin araştırılmadığı tüm uygulamalarda gizli katmanların tamamında dropout kullanılmıştır.

- **KATMAN SAYISININ ÖĞRENME SÜRECİNE ETKİSİ**

Katman sayısının etkisi gözlenirken nöron sayısı 768 olarak sabit tutulmuştur. Katman sayısının değişimi ile doğru sınıflandırılan veri sayısı ve eğitim süresi incelenmiştir. Aşağıdaki tabloda (tablo-1) bu deneye ilişkin veriler yer almaktadır. Katmanlardaki nöron sayısını girişten çıkışa azalacak şekilde ayarladık ve 50 iterasyon boyunca ağı eğittik.

TABLO-1

KATMAN SAYISI	EĞİTİM SÜRESİ	DOĞRU SINIFLANDIRILAN VERİ
1	250 saniye	%54,57
2	251 saniye	%56,21
4	300 saniye	%53,61
6	301 saniye	%53,49
8	351 saniye	%50,6
10	351saniye	%43,81

Tabloda görüldüğü üzere katman sayısı artarken doğruluk biraz arttıktan sonra düşmeye başlamıştır. Katman başına düşen nöron sayısı arttıkça doğruluk ikinci katmandan sonra azalmıştır. Ayrıca katman sayısının artması ile eğitim süresi uzamıştır.

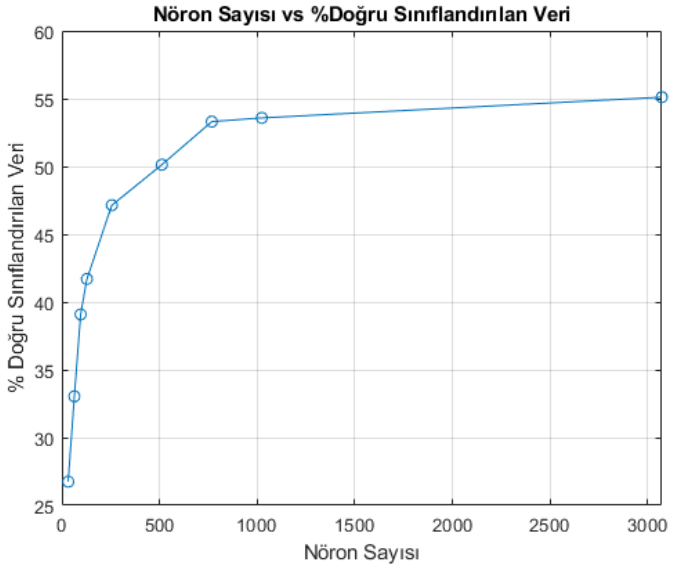
- **NÖRON SAYISININ ÖĞRENME SÜRECİNE ETKİSİ**

Katman sayısını 4 olarak sabit tutup nöron sayısının değişimi ile sınıflandırma başarımının değişimini ve eğitim süresinin değişimi aşağıdaki tabloda (tablo-2) gösterilmiştir. Ağ tüm gizli katmanlarda eşit sayıda nöron olacak şekilde oluşturulmuştur.

TABLO-2		
NÖRON SAYISI	**EĞİTİM SÜRESİ**	**DOĞRU SINIFLANDIRILAN VERİ**
32	161 saniye	%26,75
64	161 saniye	%33,04
96	162 saniye	%39,08
128	160 saniye	%41,7
256	161 saniye	%47,14
512	159 saniye	%50,13
768	160 saniye	%53,31
1024	160 saniye	%53,58
3072	163 saniye	%55,11

Tablo-2'de görüldüğü üzere nöron sayısının artışı eğitim süresinde kayda değer bir fark yaratmamıştır. Ayrıca nöron sayısı artışı ile bir artış veya azalma eğilimi de yoktur. Buna karşılık nöron sayısının artması ile sınırlandırma başarısı artış göstermiştir. Nöron sayısı ile sınıflandırma başarısının değişimi aşağıdaki figürde (figür-6) gösterilmiştir.

FİGÜR-6



Figür-6'da görüldüğü üzere doğru sınıflandırılan veri sayısı nöron sayısı arttıkça artış gösterse de bir süre artış hızı azalmakta ve bir limit değere yaklaşılmaktadır. Bu sınıflandırma problemimizde çok katmanlı algılayıcının performansının sınırlarında olduğumuzu göstermektedir.

- **AKTİVASYON FONKSİYONLARININ EĞİTİM SÜRECİNE ETKİSİ**

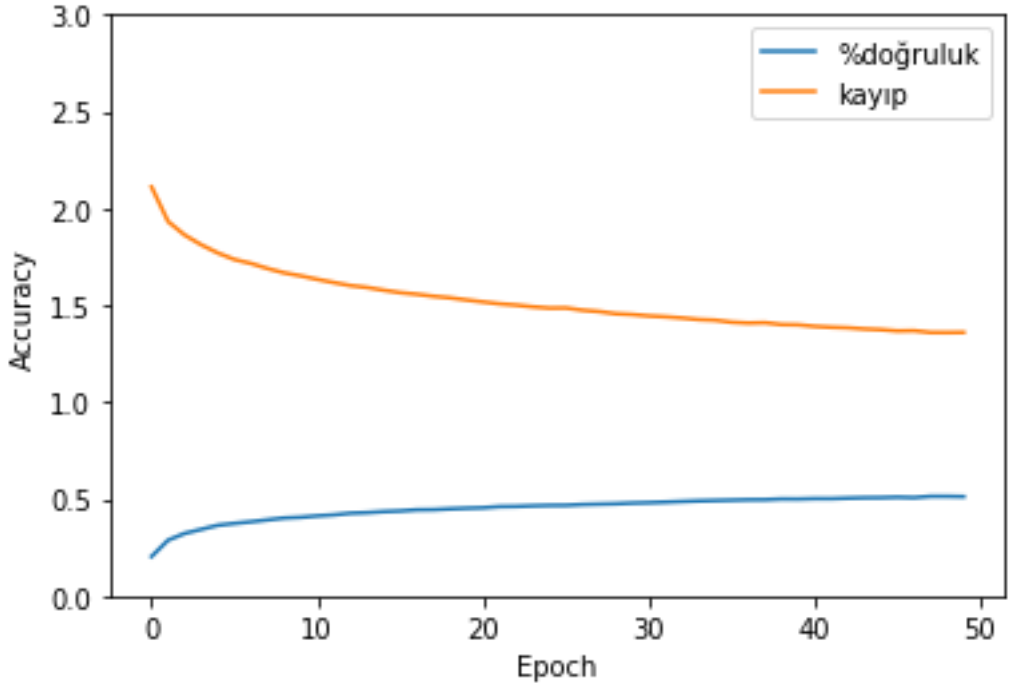
Farklı aktivasyon fonksiyonlarının öğrenme sürecini nasıl etkilediğini imcelemek için 4 gizli katmanı bulunan ve 512 nöronu olan bir ağda farklı aktivasyon fonksiyonları kullanılarak doğru sınıflandırılan verilerin oranı ve eğitim süresi incelenmiştir. Sonuçlar aşağıdaki tabloda (tablo-3) gösterilmiştir.

TABLO-3

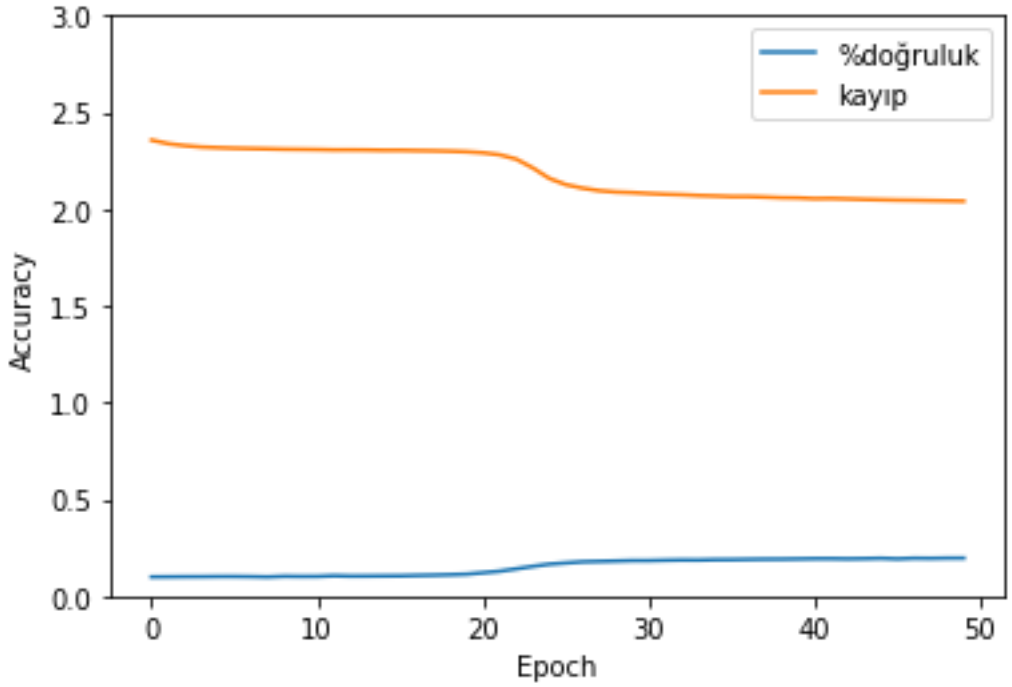
GİZLİ KATMAN AKTİVASYON FONKSİYONU	ÇIKIŞ KATMANI AKTİVASYON FONKSİYONU	EĞİTİM SÜRESİ	DOĞRU SINIFLANDIRILAN VERi
ReLU	SoftMax	176 sn	%51
Sigmoid	SoftMax	176 sn	%22,08
Sigmoid	Sigmoid	174 sn	%19,49
ReLU	Sigmoid	179 sn	%50,67
ReLU	ReLU	169 sn	%15,96
Sigmoid	ReLU	167 sn	%10
SoftMax	SoftMax	177 sn	%10
SoftMax	Sigmoid	171 sn	%10

Tablo-3'de görüldüğü üzere gizli katmanlarda aktivasyon fonksiyonu olarak ReLU kullanılırsa sınıflandırma başarısında önemli bir artış olmaktadır. Ayrıca çıkış katmanında SoftMax ya da sigmoid kullanılması sınıflandırma için önemlidir. Çıkış katmanında ReLU kullanıldığı durumlarda eğitim gerçekleşmemiştir diyebiliriz. Ayrıca gizli katmanlarda SoftMax kullanılması da eğitimi olumsuz etkilemektedir. Farklı aktivasyon fonksiyonlarının kullanılması eğitim süresini 8 saniyeden fazla değiştirmemiştir. Ara katmanlarda ReLU ve çıkış katmanında SoftMax kullanıldığı durumda, ara katmanlarda sigmoid ve çıkış katmanında SoftMax kullanılan bir durumda iterasyon başına kayıp(loss) ve doğruluk değişimi aşağıdaki figürlerde (figür-7,8) gösterilmiştir. Ayrıca çıkışta ReLU kullanılması durumunda kayıp (loss) değeri salınım yapmaktadır

FİGÜR-7 (ReLU & SoftMax)



FİGÜR-8 (Sigmoid & SoftMax)



- **KAYIP (LOSS) FONKSİYONUNUN EĞİTİM SÜRECİNE ETKİSİ**

Kullanılabilecek farklı kayıp fonksiyonlarının etkisini incelemek için 4 gizli katmanı bulunan ve 512 nörondan oluşan bir ağın kayıp fonksiyonları değiştirilerek veri sınıflandırmadaki başarının değişimi incelenmiştir. Elde edilen veriler tablo-4'de gösterilmiştir.

TABLO-4

KAYIP FONKSİYONU	EĞİTİM SÜRESİ	DOĞRU SINIFLANDIRILAN VERİ
Caterogorical Cross Entropy	162 saniye	%51,00
Kullback-Leibler Divergence	165 saniye	%52,61
Mean Absolute Error	159 saniye	%31,24
Mean Squared Error	160 saniye	%36,78
Poisson	157 saniye	%46,17
Cosine Similarity	171 saniye	%49,7
Hinge	186 saniye	%26,95
Huber	170 saniye	%31,81

Farklı kayıp fonksiyonlarının kullanılmasının eğitim sürecini önemli ölçüde etkilediği görülmüştür. Hinge fonksiyonu ile %26,95 başarım sağlanabilirken Kullback-Leibler Divergence kullanıldığında başarım 52,61 seviyelerine çıkmaktadır. Kayıp fonksiyonlarını hesaplarken oluşan karmaşıklık farkı eğitim süresini etkilemektedir.

- **GERİYE YAYILIM İÇİN KULLANILABİLECEK ALGORİTMALARIN EĞİTİM SÜRECİNE ETKİSİ**

Ağırlıkların güncellenmesi, bir yerel minimumun bulunması için farklı algoritmalar kullanılabilir. Bu algoritmalar arasındaki fark hem başarımı hem de eğitim sürecini etkileyecektir. Bu durumu gözlemlemek için 4 katmanı ve 512 nöronu bulunan bir ağ Kullback-Leibler Divergence kayıp fonksiyonu kullanılarak farklı algoritmalar ile çalıştırılıp doğru sınıflandırılan veri sayısı ve eğitim süresi incelenmiştir. Sonuçlar tablo-5'de gösterilmiştir.

TABLO-5		
ALGORİTMA	**EĞİTİM SÜRESİ**	**DOĞRU SINIFLANDIRILAN VERİ**
Stokastik Gradyan İniş	162 saniye	%51,48
Adamax	177 saniye	%43,96
Adagrad	170 saniye	%45,91
Adadelta	175 saniye	%32,42
Adam	171 saniye	%37,94
RMSProp	191 saniye	%27,31
Ftrl	202 saniye	%10
Nadam	276 saniye	%26,94

Tablo-5'de görüldüğü üzere eğitim süreleri ve başarım algoritmadan algoritmaya fark göstermektedir. Bu tablodan CIFAR10 verilerini sınıflandırma problemi için SGD, Adagrad ve Adamax kullanılabileceğini görüyoruz. Gerek başarım gerekse hızın yüksek olması dolayısıyla SGD algoritmasını kullanmayı tercih edilmiştir.

- **DROPOUT KULLANIMININ EĞİTİM SÜRECİNE ETKİSİ**

Dropout kullanımı katmanda belirli sayıda nöronun belirli bir olasılık ile bir iterasyon için bağlantı dışı bırakılması ise gerçekleşir. Bu olasılığı değiştirerek başarımın değişimini incelenmiştir. Veriler tablo-6'ya işlenmiştir.

TABLO-6	
DROPOUT OLASILIĞI	**DOĞRU SINIFLANDIRILAN VERİ**
0.00	%50,82
0.05	%52,87
0.10	%52,7
0.15	%52,44
0.2	%52,09
0.35	%46,33
0.5	%27,89

Küçük olasılık değerleri için dropout etkisi hemen hemen sabit kalırken 0.2 değerinden yükseldikçe öğrenmedeki başarım düşmektedir. Bunun sebebi yüksek olasılıklarda nöronların sıkça yok sayılmaya başlanması ile aşırı uyumun engellenmesi yerine sürece dahil olmayan nöron sıklığının artması sonucu eğitimin kesilmesidir.

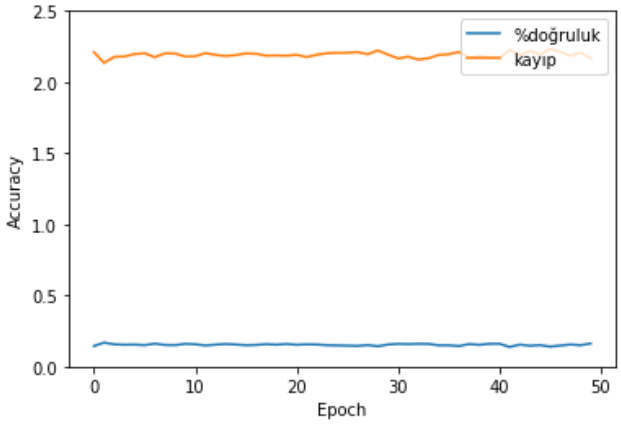
- **ÖĞRENME HIZININ EĞİTİM SÜRECİNE ETKİSİ**

Öğrenme hızı gradyan azalmasına dayalı metotlarda adım aralığını etkileyen bir faktördür. Öğrenme hızının artışı ile adım aralığı da artar. Öğrenme hızının büyük olduğu durumlarda salınım yapma veya ıraksama olasılıkları ortaya çıkarken çok küçük olduğu durumlarda ise yerel bir minimuma takılma olasılığı ortaya çıkar. Ayrıca küçük öğrenme hızları ağırlıklarda küçük değişimlere tekabül ettiğinden dolayı öğrenme hızının azalması eğitimin uzaması ile kompanse edilir. Öğrenme hızının eğitime etkisini incelemek için farklı öğrenme hızlarında doğru sınıflandırılan veri oranı incelenmiştir. Sonuçlar tablo-7'de gösterilmektedir.

TABLO-7	
ÖĞRENME HIZI	**DOĞRU SINIFLANDIRILAN VERİ**
0.001	%48,10
0.005	%52,23
0.01	%53,26
0.05	%51,72
0.1	%49,06
0.2	%42,74
0.4	%16.54

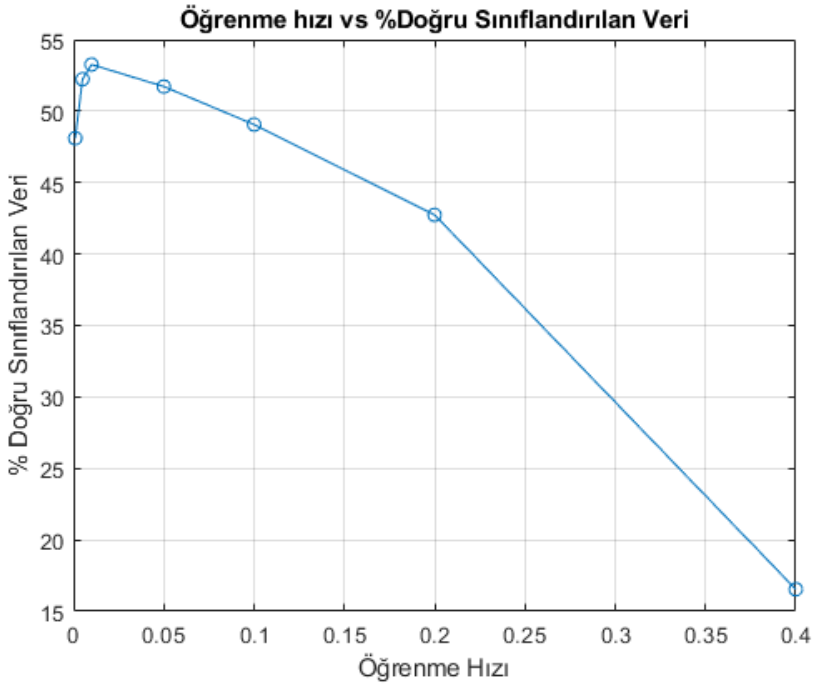
Düşük öğrenme hızlarında adım aralığı küçül olduğu için doğru sınıflandırılan veri sayısı daha yüksek hızlara göre küçük kalmışken yüksek öğrenme hızlarında SGD algoritmasındaki salınım yüzünden başarım düşmüştür. Bu salınımın bir örneği aşağıda 0.4 öğrenme hızı için kayıp (loss) değerinin her iterasyonda değişimini gösteren figürde (figür-9) gözükmektedir.

FİGÜR-9



Momentum terimi olmadan öğrenme hızına göre başarımın değişimi aşağıdaki figürde (figür-10) gösterilmektedir.

FİGÜR-10



Salınım ve yerel minimumun önüne geçmek için katsayısı 0.7 olan bir momentum terimi eklendiğinde olan değişiklikler aşağıdaki tabloda (tablo-8) gösterilmiştir.

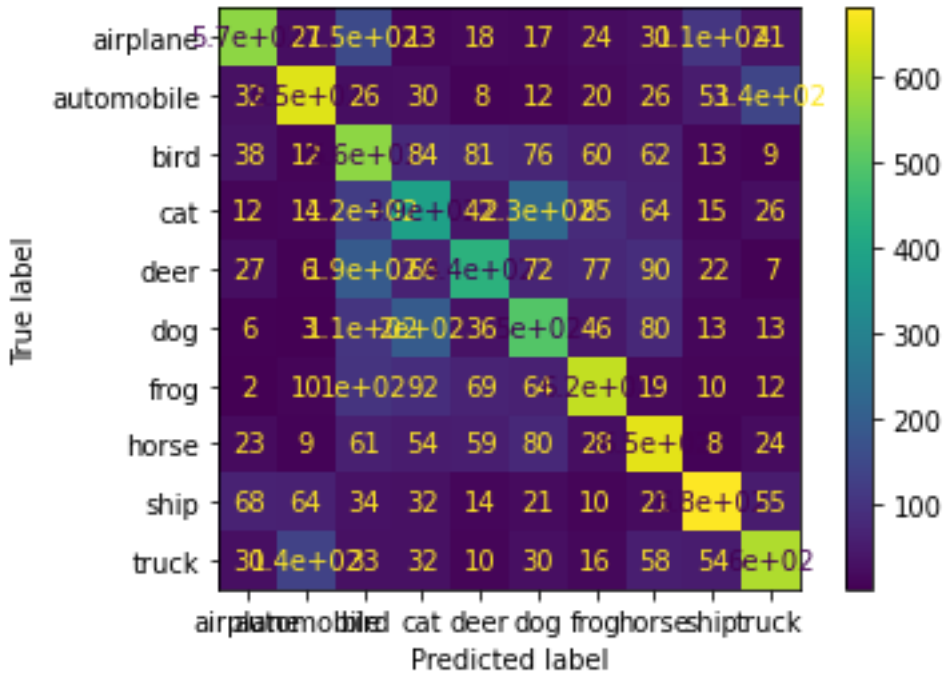
TABLO-8

ÖĞRENME HIZI	DOĞRU SINIFLANDIRILAN VERİ
0.001	%52,71
0.005	%53,38
0.01	%52,02
0.05	%45,09
0.1	%16,19
0.2	%10
0.4	%10

Momentum terimi küçük öğrenme hızlarında başarımı az miktarda arttırmıştır ancak yüksek hızlarda başarımı ciddi miktarda düşürmüştür.

Şu ana kadar yapılan karşılaştırmalar sonucunda sınıflandırma problemi için en yüksek başarıyı sağlamak için tablolar incelenerek daha iyi sonuç vermesi beklenen katsayılar, fonksiyonlar kullanılmıştır.

Algoritma olarak stokastik gradyan azalması, kayıp fonksiyonu olarak Kullback-Leibler Divergence kullanılmıştır. Ağ toplam 3072 nörondan ve 3 gizli katmandan oluşmaktadır. Her gizli katmanda eşit sayıda nöron vardır. Çıkış katmanında aktivasyon fonksiyonu olarak SoftMax kullanılmıştır. Diğer katmanlarda kullanılan aktivasyon fonksiyonu ReLU olarak seçilmiştir. Dropout olasılığı 0.1 alınmıştır. Öğrenme hızı 0.005 ve momentum terimi katsayısı 0 alınmıştır. Bu başlangıç koşulları ile ağ 120 iterasyon boyunca çalıştırıldığında verilerin %56,62'si doğru sınıflandırılmıştır ve bu durumda oluşan confussion matrisi aşağıdaki gibidir.

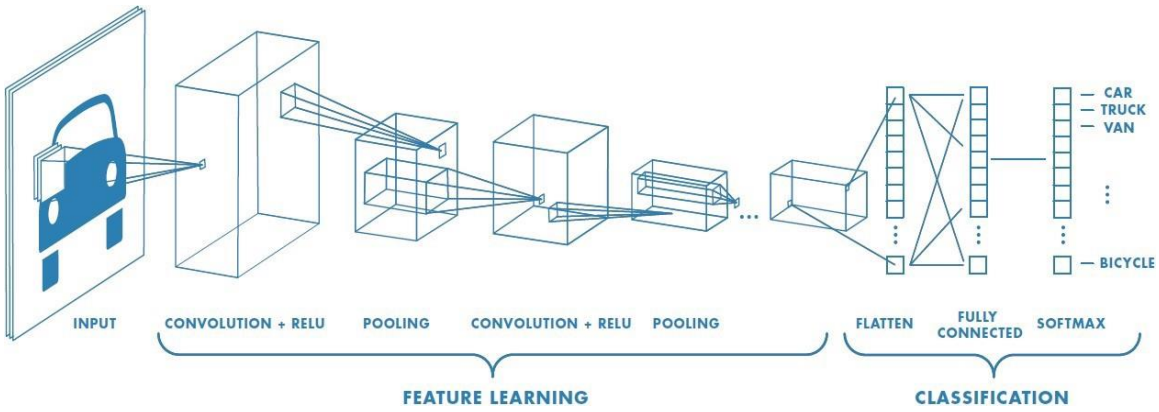


FİGÜR-11

2) EVRİŞİMLİ SİNİR AĞI KULLANILARAK VERİLERİN SINIFLANDIRILMASI

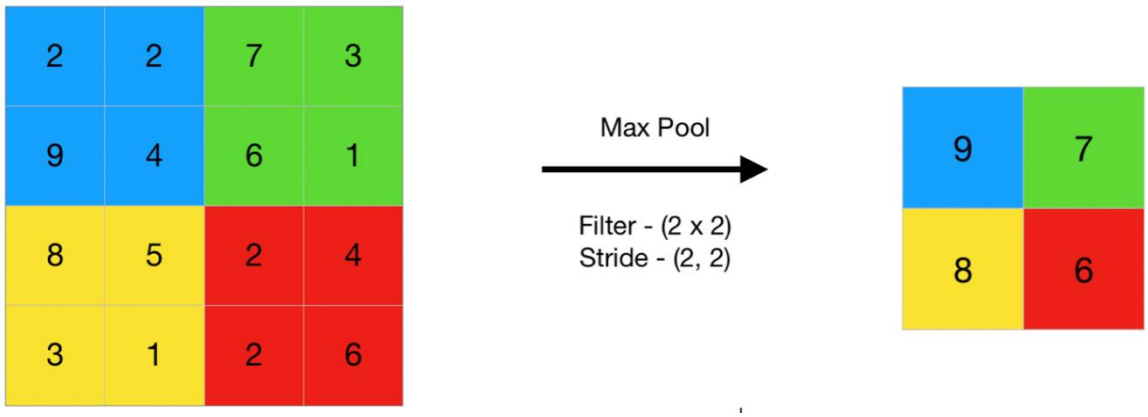
Evrişimli sinir ağları görüntü işleme, bilgisayar görüsü gibi çeşitli alanlarda kullanılan ve yapıları bu amaca yönelik tasarlanmış yapay sinir ağlarıdır. Aşağıdaki figürde (figür-12) evrişimli sinir ağlarının genel yapısı gösterilmektedir.

FİGÜR-12



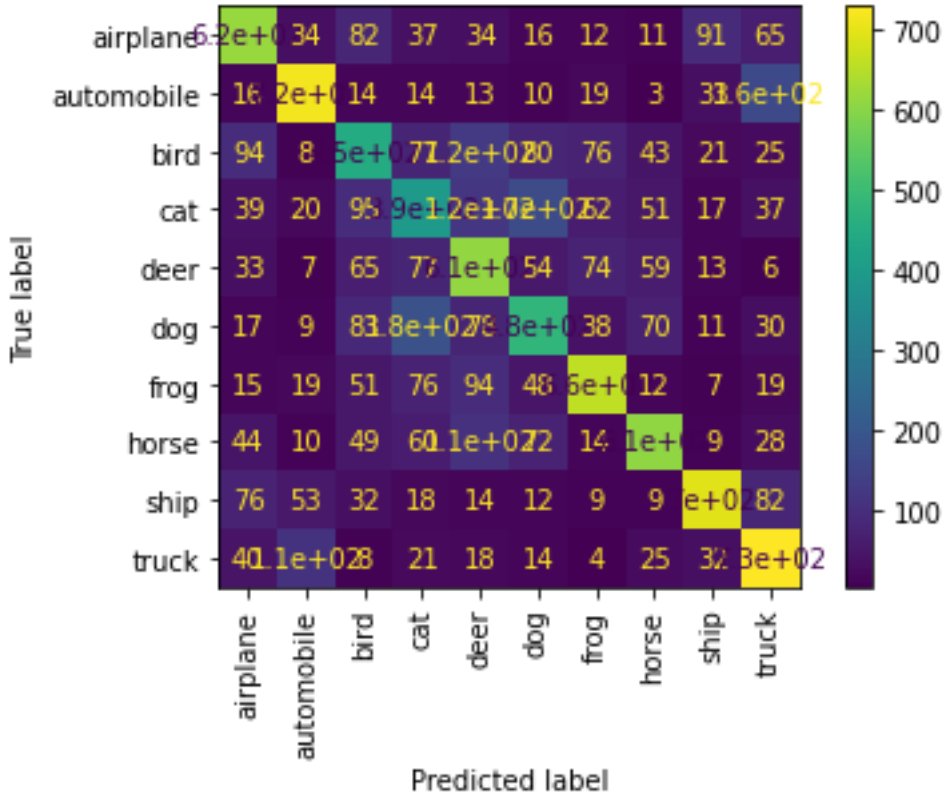
Evrişimli sinir ağları girdi olarak verilen resmin bilgilerini bir matris olarak alır ve ilk önce özellik çıkarma işleminin gerçekleştiği katmanda bu işlem gerçekleşir. Girdi olarak alınan matrise ilk olarak yapılması istenen işleme göre (öbekleme, sınıflandırma vb.) seçilen sayısal filtreler uygulanır. Bu işlem köşe belirginleştirme, bulanıklaştırma gibi farklılaşabileceği için uygulanan filtreler değişir. Bu filtre uygulama işlemi konvolüsyon katmanında gerçekleşir. Konvolüsyon katmanının ardından aktivasyon katmanı gelir. Bu katmanda konvolüsyon katmanı çıktılarına ReLU fonksiyonu uygulanır. Bu sayede nonlineerlik arttırılır ve sınıflandırma işlemleri kolaylaşır. Bu katmanın ardından pooling katmanı gelir. Bu katmanda verilen girişin alt-örneklemesi gerçekleştirilir. Bu işleme pooling denir. Ortalama pooling, maksimum pooling gibi farklı pooling türleri vardır ancak verilen girdinin en belirgin özelliklerini öne çıkardığı için maksimum pooling işlemini tercih ettik. Bu işlemde verilen görüntüyü temsil eden matris alt bloklara ayrılır ve bu bloklardaki verilerin en büyüğü seçilerek daha küçük boyutlu bir matris oluşturulur. Bu işlem devam eden figürde (figür-12) gösterilmiştir.

FİGÜR-13



Pooling katmanın çıktısı bir matristir ancak sınıflandırma işleminde kullanılmak için düzleştirme işlemine tabi tutulur ve bu matris sınıflandırma işleminde kullanılacak vektöre dönüştürülür. Bu vektör ise sınıflandırmayı yapacak ağa girdi olarak verilir. Tasarladığımız ağda toplamda 2 tane konvolüsyon katmanı ve 1 tane maksimum pooling katmanı bulunmaktadır. Devamında ise 128 nöronu bulunan ve ReLU aktivasyon fonksiyonunu kullanılan bir giriş ile SoftMax aktivasyon fonksiyonunu kullanan bir çıkış katmanı ekledik. Öğrenme hızını 0.01 ve momentum terimi katsayısını 0.9 alarak ağı 10 iterasyon boyunca eğittik.

Evrişimli sinir ağımızın eğitimi 1484 saniye sürdü. Evrişimli sinir ağı çok katmanlı algılayıcıya göre daha uzun sürede eğitilmektedir ancak sadece üç özellik çıkartma ve üç sınıflandırma katmanından oluşan bu ağ 10 iterasyon eğitilmesine rağmen %64,28 doğrulukla verileri doğru sınıfladı. Bu durum ağ mimarisinin yapılması istenilen işe göre şekillendirmenin önemini gösteriyor. Çok katmanlı algılayıcı %57 üzerinde bir başarımı çeşitli denemelere ve iyileştirmelere rağmen asla gösterememiştir. Evrişimli sinir ağına ilişkin Confussion matrisi aşağıdaki şekilde gösterilmiştir.


airplane
automobile
bird
cat
deer
dog
frog
horse
ship
truck
True label
Predicted label
700
600
500
400
300
200
100

FİGÜR-14

3) KÜTÜPHANE KULLANILMADAN YAZILAN ÇOK KATMANLI ALGILAYICI KODU İLE VERİLERİN SINIFLANDIRILMASI

CİFAR10 veri seti yukarıda detaylıca bahsedildiği üzere paket programlar kullanılarak çok katmanlı algılayıcı modeli ve evrişimli sinir ağı modeli ile sınıflandırılmaya çalışıldı. Aynı veri setinin sınıflandırmak için daha önce paket program kullanılmadan yazılmış olan çok katmanlı algılayıcı kodu ile sınıflandırılması istendi. Bu veriyi daha etkin biçimde sınıflandırabilmek için bu koda bazı değişiklikler ve eklemeler yapıldı. Bu değişiklikler başlıca: Veri setinin okunması ve ağa hazır hale getirilmesi, ReLU ve Softmax aktivasyon fonksiyonlarının eklenmesi, geriye yayılım algoritmasında kullanılacağı için ReLU ve Softmax aktivasyon fonksiyonlarının türevlerinin fonksiyon olarak koda eklenmesi, kayıp (loss) fonksiyonu olarak sınıfsal karşılıklı entropi yani categorical cross entropy fonksiyonunun koda eklenmesi, aşırı öğrenmeyi (overfitting) engelleyen dropout yönteminin koda eklenmesidir. Yine test sonunda başarım oranını görmek için doğruluk değerine ek olarak karışıklık matrisi (confusion matrix) bastırılmıştır. ReLU, Softmax fonksiyonları, categorical cross entropy loss fonksiyonu, dropout tekniği ve confusion matrisinden bahsedilmiştir.

Son halinde çıkış katmanı için ayrı, diğer katmanlar için ayrı bir aktivasyon fonksiyonu belirlenerek kullanılabilmektedir. Kullanılabilen aktivasyon fonksiyonları Sigmoid, Softmax ve ReLU'dur. Aktivasyon fonksiyonlarının farkını gözlemleyebilmek adına paket programlı çözümlerde kullanılmış olan ara katmanlarda ReLU, çıkış katmanında Softmax aktivasyon fonksiyonu kombinasyonunun yanı sıra her katmanda sigmoid ve ara katmanlarda ReLU çıkışta sigmoid olmak üzere toplamda üç farklı aktivasyon fonksiyonu kombinasyonu kullanıldı.
ReLU ve sigmoid ya da sadece sigmoid kullanılan durumlarda mantıklı kayıp ve doğruluk değerleri belirlenirken ve sınıflandırılan veri sayısı belli bir iterasyon sayısı boyunca artarken softmax fonksiyonu kullanılan durumlarda kayıp değerinin oldukça yüksek çıktığı, sınıflandırma yüzdesinin ise %10'u pek aşmadığı genel olarak tüm fotoğrafları aynı nesneymiş gibi sınıflandırdığı görüldü. Bunun sebebinin, paket programlarda softmax kullanılan çıkışların verimler sonuçlar verdiği de göz önünde bulundurulduğunda, softmax ya da softmax fonksiyonunun türevinin formülasyonunda ya da koda adapte edilmesinde bir yanlışlık olabileceği düşünüldü. Belirlenen sonuçlar aşağıda bir tablo halinde verilmiştir. Bu

işlemler yapılırken 3072 girişli ve 10 çıkışlı ağın ara katmanlarında sırasıyla 20 ve 15 nöron bulundurulmuştur. Öğrenme hızı 0.001 ve momentum terimi 0.9 olarak alınmıştır. 20 iterasyon yapılmış, işlem beş kere tekrarlanmış, ortalama değerler alınmıştır.

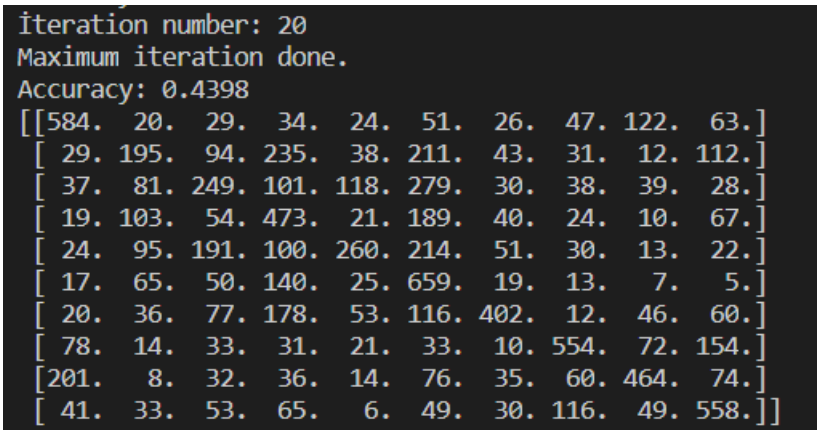
Aktivasyon Fonksiyonu	Eğitim sonunda loss	Eğitim sonunda doğruluk	Test verisi için doğruluk
Sigmoid	1.872	%34.13	%31.8
ReLU + Sigmoid çıkış	1.639	%45.39	%44.8
ReLU + Softmax çıkış	30.763	%10	%10

Görülebileceği gibi çıkış katmanı haricinde ReLU kullanıldığında sadece sigmoid kullanımına göre sınıflama başarısı artmıştır. Çıkışta ReLU kullanılmamasının sebebi çıkış vektörümüzün 1'den büyük değerler almasının gerekmemesi olarak düşünülerek bu kombinasyon kullanıldı. Softmax fonksiyonu yazıldıktan sonra yapılan denemelerde Softmax değerinin çok küçülürken sıfıra çok yakın bir sayının çok büyük bir sayıya bölünmesi sebebiyle "nan" yani tanımsız olduğu ve bunun ağın çalışmasını engellediği görüldü. Bu etkiden kurtulmak için maximum normalization tekniği kullanıldı. Buradaki yöntem Softmax fonksiyonunun parametresi olan vektörün yani, aktivasyon fonksiyonuna sokulmamış nöron girişlerinin oluşturduğu vektörün değerlerini vektörün en yüksek değeri kadar kaydırarak normalize etmektir. Vektörü X olarak kabul edersek X_i değerleri X_i-max(X) olarak değiştirilmektedir. Bu sayede her terim kaydırıldığından sonuç değişmemekte fakat "nan" problemi de ortadan kalkmaktadır. Maksimum normalizasyonu yapılmış Softmax aktivasyon fonksiyonu ve türevi aşağıda gösterilmektedir.

```
def softmax(x,i):
    # x : Nöronların girişindeki dot product değerlerinin oluşturduğu vektör
    # i : Hesaplanan nöronun indisi
    return np.exp(x[i]-np.max(x))/np.exp(x-np.max(x)).sum()
#Softmax fonksiyonunun türevi
def softmax_derivative(x,i):
    return softmax(x,i)*(1-softmax(x,i))
```

FİGÜR 15

Yukarıda açıklanmış olan confusion matrisi, sınıfların ikili olarak nasıl sınıflandırıldığını görmemizi sağlar. Örneğin yukarıda da görüldüğü gibi birbirine de benzer araçların ayırt edilmesi zor olduğundan iki sınıfın kesiştiği değerler diğerlerine nazaran daha büyüktür. Bunun sebebi o iki sınıfın iyi ayırt edilememiş olması ve yanlış tahmin edilmesi yani bir anlamda karıştırılmasıdır. Aksine birbirinden oldukça farklı sınıfların kesişiminin oldukça düşük olduğu bunun sebebinin de aynı mantıkla iki farklı cismin daha başarılı sınıflandırıldığıdır. Confusion matrisinin köşegeni, yani iki aynı sınıfın birleştiği yerdeki değerler doğru sınıflandırılan verileri simgeler. Yani ideal bir confusion matrisi birim matrisin sınıf başına örnek sayısı katı olarak düşünülebilir. Koda eklenen confusion matrisi bastırma kısmının örnek bir çıktısı aşağıdaki gibidir.



FİGÜR 16

Görüldüğü gibi, 20 iterasyon sonucunda test edilmiş verinin oluşturduğu confusion matrisi, toplamda 4398 veri doğru sınıflandırılmış.
Bu kodun öğrenme hızı, momentum terimi, katman-nöron sayıları gibi diğer farklı parametrelere bağlı testleri de yapılması planlanmış, fakat yeterli zaman olmaması ve kodun yeteri kadar hızlı çalışmaması sebebiyle yapılamamıştır.

SONUÇ

CIFAR10 veri seti veriden öğrenmeye dayalı hazır kütüphaneler kullanılarak çok katmanlı algılayıcı ve evrişimli sinir ağları yapısı ile ve bu kütüphanelerden bağımsız sıfırdan yazılan yine birçok katmanlı algılayıcı yapısı ile sınıflandırılmaya çalışılmıştır. Çeşitli parametreler değiştirilerek sonuçlar gözlemlenmiş ve tartışılmıştır. Hazır kütüphaneler ile yapılan sınıflandırmalar karşılaştırıldığında CNN yani evrişimli sinir ağlarının görüntü sınıflandırma problemleri için daha uygun olduğu ve daha başarılı sonuçlar verdiği açıktır. Bunun yanış ıra her ne kadar Softmax fonksiyonu istenildiği gibi çalışmasa da kütüphanesiz yazılan çok katmanlı algılayıcının %45-50 civarında, paket program çok katmanlı algılayıcısına yakın doğru veri sınıflandırdığı görülmektedir. Paket program kullanıldığında daha yüksek başarım elde edilmesinin sebeplerin birisi de kullanılan optimize edicilerdir (optimizer).

KAYNAKÇA

Neural Networks and Learning Machines, Simon Haykin

https://keras.io/api/losses

https://keras.io/api/optimizers

https://keras.io/api/layers/core_layers

https://www.geeksforgeeks.org/cnn-introduction-to-pooling-layer/#:~:text=Pooling%20layers%20are%20used%20to,generated%20by%20a%20convolution%20layer.

https://heartbeat.fritz.ai/a-beginners-guide-to-convolutional-neural-networks-cnn-cf26c5ee17ed

https://towardsdatascience.com/understanding-input-and-output-shapes-in-convolution-network-keras-f143923d56ca

https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks

http://rinterested.github.io/statistics/softmax.html

https://en.wikipedia.org/wiki/Softmax_function

https://peltarion.com/knowledge-center/documentation/modeling-view/build-an-ai-model/loss-functions/categorical-crossentropy

https://en.wikipedia.org/wiki/Stochastic_gradient_descent

https://towardsdatascience.com/stochastic-gradient-descent-clearly-explained-53d239905d31

https://medium.com/datadriveninvestor/introduction-to-how-cnns-work-77e0e4cde99b#:~:text=One%20of%20the%20main%20parts,Convolutional