Yayın Tarihi: 16 Haziran 2022

# Yapay Sinir Ağlarıyla Görüntü Tanıma (Haziran 2022)

**Ayşe Rabia Soylu[1], 152120181044, ve Emin Cingöz[2], 152120181036**
[1]Bilgisayar Mühendisliği, Eskişehir Osmangazi Üniversitesi, 26040 Türkiye
[2]Bilgisayar Mühendisliği, Eskişehir Osmangazi Üniversitesi, 26040 Türkiye

[1]e-posta: ayserabiasoylu@gmail.com
[2]e-posta: emincingoz@gmail.com

Bu çalışma Eskişehir Osmangazi Üniversitesinde görevli Doktor Öğretim Üyesi Eyüp Çınar'ın 2021-2022 eğitim öğretim yılı bahar döneminde vermiş olduğu 'Neural Networks' dersi kapsamında yapılmıştır.

Google Drive Link: https://drive.google.com/drive/folders/1kmypFBrjoAFck2k08VlVfSBDvAURLgto?usp=sharing

Github Link: https://github.com/emincingoz/Cifar_10_Convolutional_Neural_Network

**ÖZET** Cifar-10 veri seti kullanılarak basit evrişimsel sinir ağı kullanılan modeller geliştirildi. İlk uygulanan model, oldukça fazla aşırı öğrenmeye uğradığı için ikinci bir model oluşturuldu. İkinci oluşturulan modelde, ilk modele ek olarak dropout, early stopping, batch size gibi yapılar kullanıldı. Bu yapılar değişik varyasyonları ile kullanılarak olabildiğince aşırı öğrenmenin önüne geçilmeye çalışıldı. Geliştirilen modeller çok basit düzeyde yapılardan oluştuğu için yeterince yüksek sonuçlar elde edebilmek mümkün görülmemiştir. Ancak model karmaşıklığı göz önüne alındığında olabilecek oldukça yüksek sonuçlar elde edildi. Test verisi doğruluk oranı, hiper parametrelerin ayarlanması ile %85 doğruluk oranına çekilebilmiştir ve yine test veri seti kayıp oranı ise %33 oranlarına indirilebilmiştir.

**ANAHTAR KELİMELER** Cifar10, Nesne Tanıma, Evrişim, Convolutional Neural Networks, Convolutional Layer, Pooling, Dropout, Early Stopping, Learning Rate, Aşırı Öğrenme, Overfitting, Converge, Optimizer, Loss Function, One Hot Encoding, Flatten, Train, Validation, Test, Accuracy, Loss, Hiper Parametre

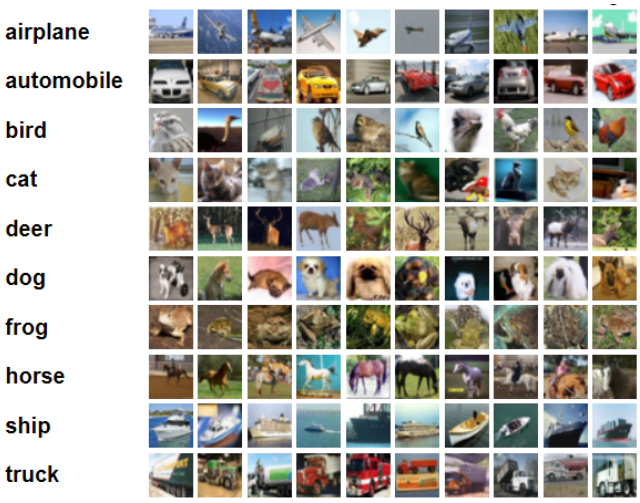
## I. GİRİŞ

Bu çalışma kapsamında Cifar-10 veri seti kullanılarak fotoğraf tipindeki görsel veriler kullanılarak nesne tanıma algoritmaları uygulanmaya çalışılmıştır. Çalışmada iki temel model üzerinde çalışılarak performans değerlendirme metrikleri yorumlanarak geliştirmeler yapılmıştır. İlk model 3 convolutional, 3 max pooling, 1 flatten ve 2 dense katmanından oluşmaktadır. Bu modelin loss functionı, optimizer'ı, learning rate'i, momentumu ve epoch sayısı değiştirilerek performansı gözlenmiştir. Bu hiper parametrelerle oynamak modelin aşı öğrenmeye maruz kalmasını engelleyememiştir. İlk modelde iyi sonuç verdiği gözlemlenen epoch sayısı, optimizer, loss function bilgileri kullanılarak ikinci bir model kuruldu. Bu modelde drop out, normalizasyon, early stopping, bach size üzerinde değişiklikler yapıldı. Tüm uğraşlar sonucunda %85 validation accuracy ve %33 loss değerlerine ulaşılmıştır.

## II. VERİ SETİ

CIFAR-10 veri seti 32x32 piksellik, 60.000 adet farklı fotoğraf içerir. Bu veri seti, 10 farklı sınıfın her birine ait 10.000'er farklı renkli fotoğraf örneğini içerir. 'Keras' derin öğrenme kütüphanesi bu veri setinin projeye doğrudan dahil edilerek kullanılabileceği bir method içermektedir. Veri seti [1], 50.000 eğitim; 10.000 test verisine ayrılmıştır. Şekil 1'de bu veri setinin her bir sınıfına ait rastgele çekilmiş 10'ar fotoğraf görünmektedir. Bu sınıflar: uçak, otomobil, kuş, kedi, geyik, köpek, kurbağa, at, gemi, ve kamyondur.

Şekil 1 - CIFAR-10 Veri Seti Örnekleri




## III. VERİ SETİ ÖN İŞLEME

Oluşturulan modelin performansını artırmak veya bazen kurulan modelden bağımsız olarak verinin kendisinin model performansını olumsuz etkilemesini önlemek için veri setine normalizasyon uygulanmıştır. Modelin öğrenmesini yanılgısız test edebilmek için model test edilirken eğitim esnasında hiç karşılaşmadığı örnekler sunmak için veri seti test ve eğitim olarak ikiye ayrılmıştır.

### *A. NORMALİZASYON*

Oluşturulan modellerin katmanlarında yapılan matematiksel hesaplamalar sonucunda fotoğraf verilerimizin R, G, B değerleri 0-255 değer aralığının dışına çıkabilir. Fotoğraflar aynı boyutta tutularak normalize edilmiş fotoğraf verileri, sınıflandırıcıların kararlılığını arttırdığı gözlemlenmiştir.[2]

### *B. VERİ SETİNİ EĞİTİM VE TEST İÇİN AYIRMA*

Tensorflow keras kütüphanesi fonksiyonu kullanarak projeye dahil edilen cifar-10 veri setinin %16.7'si test ve %83.3'ü test verisi olarak ayrılmıştır.

## IV. KURULAN MODELLER

Özellikle fotoğraf, resim, video gibi görüntü içeren kümelerde performansıyla öne çıkan Convolutional Neural Network (CNN) yapısı kullanılmıştır.

### *A. MODEL 1*

Resmin yüksek ve düşük nitelikli özelliklerini anlamak için convolutional layer'lar kullanmıştır. Model 1'de öznitelik haritası çıkarma amacıyla 32x32 lik bir input matrisi verisi için 3 adet 3x3'lük bir filtre (Convolutional Layer) kullanılmıştır.

Veri setinin örnekleri, 32 yükseklik, 32 genişlik ve 3 adet renk pikseli verisi tuttuğu için input katmanında input_shape = ( 32, 32, 3) olarak girilmiştir.

Çıkarılan bu özniteliklerden max pooling ile en alakalı olan özellikler 2x2 lik metris kapsamında taranarak bulunmuştur.

TABLO I - 1.MODEL / SON VERSIYON / MODEL.SUMMARY() ÇIKTISI

```
_________________________________________________________________
Model: "sequential_7"
_________________________________________________________________
 Layer (type)                 Output Shape              Param
=================================================================
 conv2d_21 (Conv2D)           (None, 30, 30, 32)        896

 max_pooling2d_13 (MaxPoolin  (None, 15, 15, 32)        0
 g2D)

 conv2d_22 (Conv2D)           (None, 13, 13, 64)        18496

 max_pooling2d_14 (MaxPoolin  (None, 6, 6, 64)          0
 g2D)

 conv2d_23 (Conv2D)           (None, 4, 4, 64)          36928

 max_pooling2d_15 (MaxPoolin  (None, 2, 2, 64)          0
 g2D)

 flatten_7 (Flatten)          (None, 1024)              0

 dense_14 (Dense)             (None, 64)                65600

 dense_15 (Dense)             (None, 10)                650

=================================================================
Total params: 122,570
Trainable params: 122,570
Non-trainable params: 0
```

Model 1'in ilk versiyonunda SGD (learning rate= 0.05, momentum=0.0) optimizer, sparse categorical crossentropy loss function kullanılmıştır. Epoch sayısı 20 olarak belirlenmiştir.

Ancak validation accuracy %53-56 arasında 20 epoch boyunca gidip gelmiştir. İlerleme kat edilemediği için learning rate = 0.08, momentum = 0.1 yapılarak denenmiş ancak accuracy değerinin ilerleyememesi önlenememiştir. Bu hiper parametreler sonucunda validation accuracy %60 dolaylarına sabitlenmiştir.

Epoch sayısının 20'den 10'a azalması overfitting'i azaltsa bile modelin doğruluk performansını arttırmıyordu.

Bu sorunu aşmak için optimizer'ı değiştirdik. Adams optimizer'da modelin öğrenme hızı Şekil 2' de görüldüğü üzere birden arttı.

*Şekil 2*

```
loss: 1.5236 - accuracy: 0.4461 - val_loss: 1.2280 - val_accuracy: 0.5571
loss: 1.1701 - accuracy: 0.5864 - val_loss: 1.0773 - val_accuracy: 0.6243
loss: 1.0270 - accuracy: 0.6411 - val_loss: 1.0533 - val_accuracy: 0.6325
loss: 0.9265 - accuracy: 0.6761 - val_loss: 0.9503 - val_accuracy: 0.6735
loss: 0.8543 - accuracy: 0.7020 - val_loss: 0.9037 - val_accuracy: 0.6900
loss: 0.7974 - accuracy: 0.7203 - val_loss: 0.9894 - val_accuracy: 0.6594
loss: 0.7529 - accuracy: 0.7381 - val_loss: 0.9002 - val_accuracy: 0.6931
loss: 0.7083 - accuracy: 0.7523 - val_loss: 0.8821 - val_accuracy: 0.7035
loss: 0.6694 - accuracy: 0.7640 - val_loss: 0.8704 - val_accuracy: 0.7075
loss: 0.6305 - accuracy: 0.7798 - val_loss: 0.8859 - val_accuracy: 0.7095
```

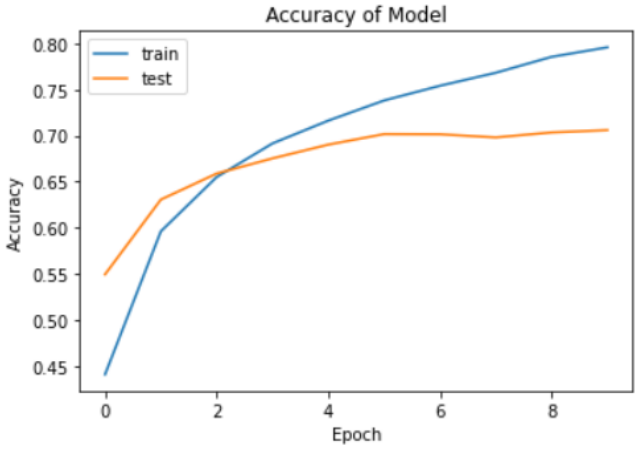
Daha yüksek sonuçları elde edebilmeyi hedefleyerek epoch sayısını 20'ye yükselttik ancak Şekil 3'de görüldüğü üzere yine %70 civarını geçemedik.

*Şekil 3*

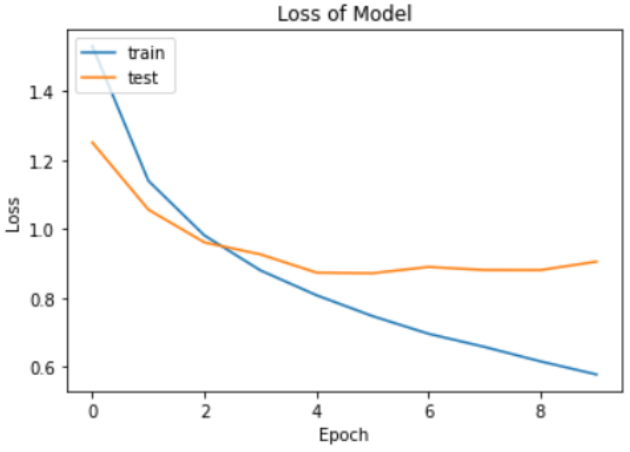
```
loss: 0.2068 - accuracy: 0.9257 - val_loss: 1.7319 - val_accuracy: 0.6861
loss: 0.1992 - accuracy: 0.9289 - val_loss: 1.8010 - val_accuracy: 0.6780
loss: 0.1862 - accuracy: 0.9330 - val_loss: 1.7993 - val_accuracy: 0.6841
loss: 0.1781 - accuracy: 0.9353 - val_loss: 1.8974 - val_accuracy: 0.6847
loss: 0.1789 - accuracy: 0.9361 - val_loss: 1.8136 - val_accuracy: 0.6841
loss: 0.1682 - accuracy: 0.9406 - val_loss: 1.9338 - val_accuracy: 0.6850
loss: 0.1632 - accuracy: 0.9406 - val_loss: 2.0535 - val_accuracy: 0.6819
loss: 0.1679 - accuracy: 0.9405 - val_loss: 1.9732 - val_accuracy: 0.6910
loss: 0.1530 - accuracy: 0.9458 - val_loss: 2.0138 - val_accuracy: 0.6874
loss: 0.1563 - accuracy: 0.9448 - val_loss: 2.1327 - val_accuracy: 0.6762
```

İlk conv2d katmanından sonra max pooling uygulanmaması overfitting'i arttıran bir unsurdur. Validation ve train accuracy değerleri arasında, her epoch sonucunda %7' den başlayıp %15 e kadar yükselen bir artış gözlemlenmektedir.

*Şekil 4(a)*




*Şekil 4(b)*




Şekil 4a ve Şekil 4b de görüldüğü üzere bu modelde önlenemeyen overfitting sorunu yüzünden yeni bir cnn modeli kurulmuştur.

Model 1'de test accuracy %70'i geçemeyip, train ve test accuracy ve loss arasındaki fark atmaya devam ettiği için Tablo 2' de görüldüğü üzere model 2' ye convolutional katmanlarından sonra normalizasyon adımı ve sinir ağından sonra dropout eklenmiştir. Bu sayede overfitting'in önlenmesi amaçlanmıştır.

### B. MODEL 2

Geliştirilen ikinci modelin model mimari yapılandırması aşağıda tablo 2'de verilmiştir. Modelde "input_shape"e verilen (32, 32, 3) boyutu, kullanılan renkli resimlerden oluşmuş olan cifar10 veri setinin boyutunu ifade etmektedir.

TABLO 2
2.MODEL MODEL.SUMMARY() ÇIKTISI

```
_________________________________________________________________
Model: "sequential"
_________________________________________________________________
 Layer (type)                 Output Shape              Param
=================================================================
 input_1 (InputLayer)         [(None, 32, 32, 3)]       0
 conv2d (Conv2D)              (None, 32, 32, 32)        896
 batch_normalization          (None, 32, 32, 32)        128
 conv2d_1 (Conv2D)            (None, 32, 32, 32)        9248
 batch_normalization_1        (None, 32, 32, 32)        128
 max_pooling2d (MaxPooling2D  (None, 16, 16, 32)        0
 conv2d_2 (Conv2D)            (None, 16, 16, 64)        18496
 batch_normalization_2 (Batc  (None, 16, 16, 64)        256
 hNormalization)
 conv2d_3 (Conv2D)            (None, 16, 16, 64)        36928
 batch_normalization_3 (Batc  (None, 16, 16, 64)        256
 hNormalization)
 max_pooling2d_1 (MaxPooling  (None, 8, 8, 64)          0
 2D)
 conv2d_4 (Conv2D)            (None, 8, 8, 128)         73856
 batch_normalization_4 (Batc  (None, 8, 8, 128)         512
 hNormalization)
 conv2d_5 (Conv2D)            (None, 8, 8, 128)         147584
 batch_normalization_5 (Batc  (None, 8, 8, 128)         512
 hNormalization)
 max_pooling2d_2 (MaxPooling  (None, 4, 4, 128)         0
 2D)
 flatten (Flatten)            (None, 2048)              0
 dropout (Dropout)            (None, 2048)              0
 dense (Dense)                (None, 512)               1049088
 dropout_1 (Dropout)          (None, 512)               0
 dense_1 (Dense)              (None, 10)                5130
=================================================================
Total params: 1,343,018
Trainable params: 1,342,122
Non-trainable params: 896
_________________________________________________________________
```

Model eğitilirken geri yayılım (back propagation) yapılmaktadır. Bu işlem sırasında her katman hatasını ayrı ayrı düzeltmeye çalıştığı için katmanların eğitiminde kaymaların oluştuğunu ifade etmek için kullanılan terim olan dahili ortak değişken kayması (internal covariate shift) oluşur ve eğitim süresi oldukça artar.[3] Eğitim süresini artıran bu durumdan kaçınmak için batch normalization işlemi uygulanır. Batch normalization, sadece uygulandığı katmanda etkili olduğu için geliştirilen modelde, her bir konvolüsyon katmanından sonra kullanılmıştır. Ayrıca batch normalization, çok daha yüksek öğrenme oranı (learning rate) kullanabilmemizi mümkün kılar.[3]

Konvolüsyon katmanlarında çekirdek boyutu (kernel size) olarak (3, 3) kullanılmıştır, bunun sebebi incelenen örneklerde uygulamasının oldukça yaygın olması ve denenilen (5, 5) gibi diğer boyutlardan daha yüksek doğruluk değerleri elde edilebilmesidir. Ayrıca aktivasyon fonksiyonu olarak görüntü sınıflandırma uygulamalarında yaygın olarak kullanımı bulunan "relu" kullanılmıştır ve padding eklenmesi gerekli görülmemiştir. [4]

Her iki konvolüsyon bloğundan sonra havuzlama katmanı (pooling layer) eklenmiştir. Görüntü sınıflandırma problemlerindeki kullanım yaygınlığından dolayı max pooling kullanılmıştır.[4] "pool_size" parametresi, büyük boyutlar verildiğinde model eğitim sonuçlarını düşürdüğü için (2, 2) boyutu uygun görüldü. "pool_size" parametresi büyük rakamlar girildiğinde düzleştirme işlemi (flatten) sonrasında oluşan çıktı boyutunu düşürmektedir. "cifar10" veri setinin geliştirdiğimiz model ile eğitimi sırasında overfitting sık sık gözlemlenmiştir. Bu boyutu artırmak overfitting üzerinde yeterince olumlu etki sağlayamamıştır. Bu yüzden (2, 2) model için uygun görülmektedir.

Konvolüsyon blokları ile özellik çıkarımları gerçekleştirildikten sonra sınıflandırma işlemini gerçekleştirmek için düzleştirme (flatten) katmanı kullanılmıştır.[2] Bunun sebebi özellik çıkarımında oluşan son çıktının boyutu bir matris özelliği taşımasıdır ancak sınıflandırma aşamasında girdilerin tek boyutu olması beklenmektedir. Düzleştirme katmanı kullanılmadığı takdirde uygulama hata verecektir.

Eğitim sırasında overfitting sık gözlemlenmesinden dolayı model karmaşıklığının düşük tutulması gerekmektedir. Bu noktada, dense katmanlarının çok fazla olması ya da nöron sayısının fazla olması tercih edilmemiştir.[5] Eğitim sonucunda elde edilen validasyon kaybı ve doğruluk oranları bu görüşün doğruluğunu teyit etmektedir. Modele fazla sayıda dense katmanı eklenmesi ya da nöron sayılarının artırılması model karmaşıklığını artırdığı için eğitim kümesinde doğruluk oranı ne kadar artıp, kayıp oranı ne kadar azalsa da test veri setinin doğruluk oranı yeterince artış göstermemiş ve kayıp oranı da oldukça yüksek kalmıştır.[5]

Sınıflandırma aşamasında kullanılan dense katmanının da "relu" aktivasyon fonksiyonuna sahip olması, "relu" fonksiyonunun güçlü yönleri sebebiyle uygun görülmüştür.[2]

Sınıflandırma sonunda çıktı boyutu "cifar10" veri setinin boyutu olan 10 seçilmiştir ve "softmax" aktivasyon fonksiyonu kullanılması uygun görülmüştür. Bu seçimin sebebi, çok sınıf bulunan sınıflandırma problemlerinde, her sınıf için [0, 1] aralığında sonuçlar üretiyor olmasıdır ve daha sonra bahsedilecek olan "sparse_categorical_crossentropy" kayıp (loss) fonksiyonu ile uyumlu çalışmasıdır.[2]

Oluşturulan modelin derlenmesi (compile) aşamasında, optimizer olarak "adam" kullanılmıştır. Adam optimizer, öğrenme hızını kendisi ayarlayan kompleks bir optimizer fonksiyonudur ve görüntü sınıflandırma problemlerinde ve birçok problemde günümüzde yaygın olarak kullanılmaktadır.[2]

Model derlenmesinde kayıp fonksiyonu olarak "sparse_categorical_crossentropy" tercih edildi.[6] Bu seçimin sebebi, modele eğitilmesi için verilen sınıf etiketlerinin [0, 9] aralığında tutulmasıdır.[6] Sınıf etiketleri, one hot encoding işleminden geçirildiğinde genel olarak daha doğru sonuçlar vermektedir. Bunun sebebi, encode edilmemiş etiketlerin aslında bir büyüklük ve öncelik belirtmesinden kaynaklı olarak eğitimin tarafsız olamamasıdır. Ancak "sparse_categorical_crossentropy" kayıp fonksiyonu kullanıldığında, istemsiz gerçekleştirilen bu taraflı yaklaşımın önüne geçilmektedir. Encode edilmiş sınıf etiketlerinin kullanılmaması hesaplama maliyetini düşürdüğü için bu şekilde kullanılması doğru bulunmuştur.[6]

Kayıp fonksiyonu seçiminde diğer bir tercih, "categorical_crossentropy" seçilebilmesidir. Ancak bu kayıp fonksiyonunun seçilebilmesi için yukarıda da bahsedildiği gibi sınıf etiketlerinin one hot encoding işleminden geçirilerek her bir sınıf etiketinin 0 ve 1 ile ifade edilmesi gerekmektedir. One hot encoding işlemi örnek çıktısı şekil 5'de gösterilmiştir.

| Color |
|---|
| Red |
| Red |
| Yellow |
| Green |
| Yellow |

| Red | Yellow | Green |
|---|---|---|
| 1 | 0 | 0 |
| 1 | 0 | 0 |
| 0 | 1 | 0 |
| 0 | 0 | 1 |

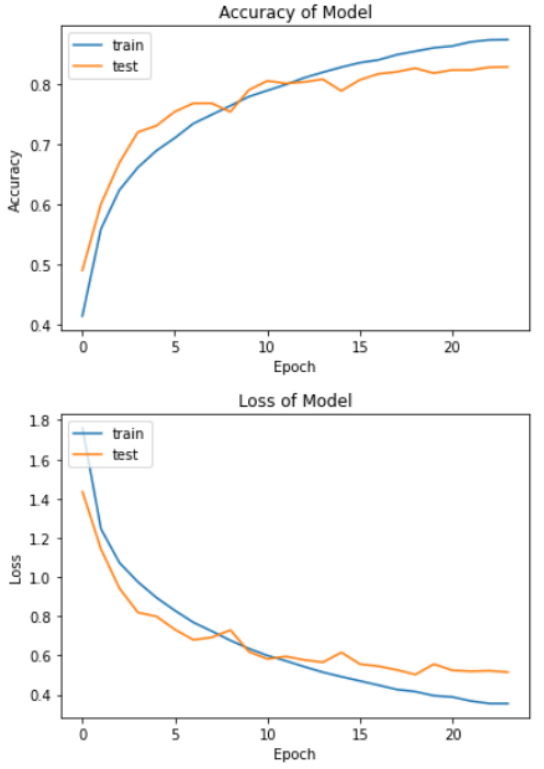
*Şekil 5- One Hot Encoding Output*

Model eğitilirken overfitting gözlenmesinden dolayı, sınıflandırma aşamasında dense katmanlarından sonra dropout kullanılmıştır. Dropout oranı, 0,2 ve 0,6 aralığında değişen değerlerde kullanıldığında, bu değerin artırılmasının aşırı öğrenmenin (overfitting) önüne geçtiği gözlemlenmiştir. Daha yüksek oranlarda drop out edilmesinin yüksek nitelikli özelliklerin de sıfırlanmasına sebep olabildiği için model performansını düşürdüğü gözlemlenmiştir.[7]
Model eğitilmesi için 32, 64 boyutlarında "batch_size" parametreleri denendi. Birbirlerine çok yakın değerler oldukları için model doğruluğu üzerinde yeterli etkiler oluşturmadığı gözlemlendi. Ancak batch_size parametresi verilmediği takdirde, aşırı öğrenme durumu daha fazla engellendiği gözlemlendi. Yine de modelin farklı kurgulandığı durumlarda "batch_size" parametresi girilmesinin model eğitimine katkı sağladığı görüldü.

Modelde, yakınsama (converge) istendiği gibi gerçekleşmediği durumlarda eğitimi yarıda kesmek gerekebileceği için early stopping callback'i kullanıldı. "validation_loss" çıktısı gözlemlendi ve "patience" parametresi 3 ve 5 değerlerinde kullanıldı. Bu kullanılan parametrelere göre "validation_loss" çıktısı 3 veya 5 kereden fazla azalma göstermediği zaman model eğitimi otomatik olarak durdurulmaktadır. patience = 3 kullanılması daha az aşırı öğrenme gerçekleşmiş sonuçlar demek olduğu için tercih edildi.

Gerçekleştirilen farklı eğitimler sonucunda elde edilen doğruluk değerleri olabildiğince az aşırı öğrenme olması hedeflendiğinde, test veri seti doğruluk oranı, %82-%85 aralığında kalmaktadır. (Şekil 6 - Accuracy of Model)

Gözlemlenen eğitim veri seti doğruluk oranları, %87-%91, eğitim veri seti eğitim kaybı %32-%36 ve test veri seti kaybı, %49-%53 aralıklarında elde edilmiştir. Bu sonuçlar olabildiğince aşırı öğrenmeden uzak sonuçlardır. (Şekil 6 - Loss of Model)




*Şekil 6 - Second Model Result*

## V. SONUÇ

Cifar-10 veri seti kullanılarak basit evrişimli sinir ağ (CNN) modelleri kuruldu. Batch size, epoch size, vb. çeşitli hiper parametreler model kayıp ve doğruluk oranlarına göre güncellenerek test veri kümesi doğruluk oranı olabildiğince 1'e yaklaştırılmaya çalışıldı. Uygulanan değişiklikler ile maksimum genelde %82-%85 aralığında test veri seti doğruluk oranları elde edildi. Hiper parametrelerde yapılan değişikliklerde aşırı öğrenme dikkate alınarak bu durumdan olabildiğince kaçınılmaya çalışıldı. Yapılan diğer çalışmalara bakıldığında cifar-10 veri seti ile görüntü sınıflandırma konusunda kurulan modellerin doğası gereği genellikle elde edilebilecek yüksek oranlar elde edilmiştir.

Model karmaşıklığı artırılarak veya farklı yaklaşımlar kullanılarak görüntü sınıflandırmada elde edilen doğruluk oranlarının yükseltilmesi mümkündür. Örnek vermek gerekirse cifar-10 veri seti kullanılan çalışmalarda [8], vision transformer kullanılarak maksimum %99,4 oranında doğruluk oranları elde edilebilmiştir [9]. Yine yapılan başka bir çalışmada Big Transfer (BiT) yöntemi kullanılarak %99.37 doğruluk oranları elde edilebilmiştir. [10]

Sonuç olarak bizim tarafımızdan kullanılan yöntemlerin cifar-10 veri setini sınıflandırmada yeterince yetenekli olmadıkları gözlemlenmiştir. Bunun sebebi, modellerin oldukça basit düzeyde olması ve veri setinin daha kompleks yapılara ihtiyaç duyması olarak gösterilebilir.

## VI. KAYNAKÇA

[1] http://www.cs.toronto.edu/~kriz/cifar.html (Veri Seti)

[2] Neural Network Dönem İçi Dersleri ve PDF'leri

[3] Batch Normalization
https://coimer.medium.com/batch-normalization-b7d73c9cc6df

[4]https://machinelearningmastery.com/how-to-develop-a-cnn-from-scratch-for-cifar-10-photo-classification/

[5]https://datascience.stackexchange.com/questions/43191/validation-loss-is-not-decreasing

[6]https://stats.stackexchange.com/questions/326065/cross-entropy-vs-sparse-cross-entropy-when-to-use-one-over-the-other

[7]https://medium.com/@tuncerergin/evrisimsel-sinir-aginda-nicin-dropout-kullanmamalisiniz-7e31941f8bb0

[8]https://paperswithcode.com/sota/image-classification-on-cifar-10

[9]https://paperswithcode.com/paper/going-deeper-with-image-transformers

[10]https://paperswithcode.com/paper/large-scale-learning-of-general-visual