



**YILDIZ TEKNİK ÜNİVERSİTESİ  
ELEKTRİK-ELEKTRONİK FAKÜLTESİ  
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ**

**GÖRÜNTÜ İŞLEME**

**ÖDEV - 3**

**Öğr. Üyesi: Prof. Dr. Mine Elif KARSLIGİL**

**Hazırlayan: 19011065 Elif Mertoğlu**

# 1-) Yöntem

Problemi çözmek için öncelikle yapmam gerekenleri sıraladım. Bunlar sırasıyla: veri setini yüklemek, 'Train' içinde yer alan görüntülerin %20'sini validasyon için ayırmak, class isimlerini nümerik sayıdan binary hale getirmek, modelleri oluşturmak, en başarılı modeli belirlemek, 'Test' datası içinde yer alan verilerle en başarılı modeli test etmek ve confusion matrix'i çıkartmak, her class'tan seçilen random 3 görüntü ile modeli test etmek ve confusion matrix'ini çıkartmak.

**a. Veri setini yüklemek:**

CIFAR-10 dataseti, derin öğrenme konularında çalışmak isteyenlerin sık kullandığı bir dataset olduğu için Keras'ın datasets kütüphanesi içinde yer almaktadır. İlgili kütüphaneyi import ettikten sonra `cifar10.load_data()` fonksiyonu ile veri setini kod içine train ve test olarak çektim.

**b. 'Train' içinde yer alan görüntülerin %20'sini validasyon için ayırmak:**

Bunun için `train_test_split()` fonksiyonunu kullandım. %20 koşulunu sağlayabilmek için `test_size` parametresini 0.20 olarak tanımladım.

**c. Class isimlerini nümerik sayıdan binary hale getirmek:**

Veri setindeki class'lardan 'airplane', 0; 'truck', 9 ile temsil edildiğinde derin öğrenme modeli tarafından 'truck' sınıfı daha çok öneme sahipmiş gibi algılanmasın diye class'ları one hot encoder kullanarak etiketledim. Bu durumda artık 'truck' sınıfı 9 ile gösterilmek yerine [0, 0, 0, 0, 0, 0, 0, 0, 0, 1] şeklinde gösterilmiş olacaktır.

**d. Modelleri oluşturmak:**

Bu adımda ödev raporunda istenilenlere göre 8 farklı model oluşturdum. Hiçbir model için padding uygulamadım. Tüm modellerde stride değerini 1 olarak aldım. Aktivasyon fonksiyonu olarak 'ReLU' kullandım. 3 katmanlı modellerim için her conv katmanından sonra Dropout kullandım fakat değerini 0.05 artırarak kullandım. Dropout için belirlediğim ilk değerse 0.20 idi. Tüm konvolüsyon katmanlarından sonra bir kere (2,2) MaxPooling kullandım. Epoch için yaptığım denemeler sonucu en iyi değer 35 olduğunu gördüm.

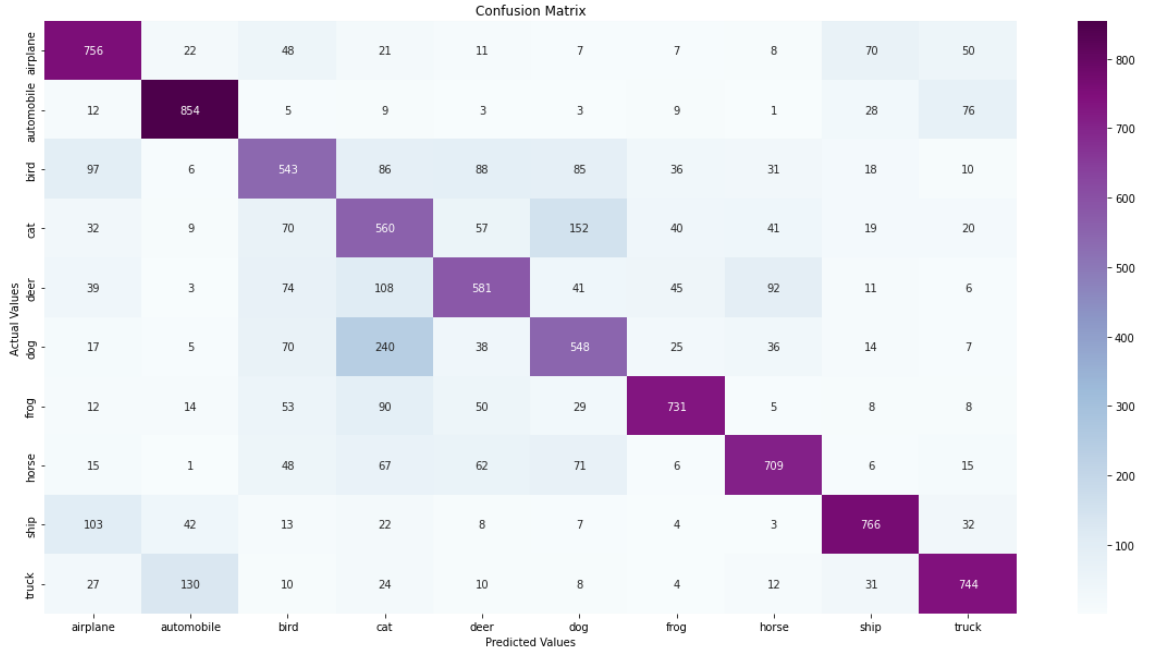
<b>Model – 0</b>	3 katman	32 filtre	3x3
<b>Model - 1</b>	3 katman	64 filtre	3x3
<b>Model – 2</b>	3 katman	32 filtre	5x5
<b>Model – 3</b>	3 katman	64 filtre	5x5
<b>Model – 4</b>	5 katman	32 filtre	3x3
<b>Model – 5</b>	5 katman	64 filtre	3x3
<b>Model – 6</b>	5 katman	32 filtre	5x5
<b>Model - 7</b>	5 katman	64 filtre	5x5

**e. En başarılı modeli belirleme:**

En başarılı modeli belirlerken, modelin validasyon datası üzerinde verdiği sonuca göre karar verdim. Bunun için her bir epoch'ta yer alan val\_accuracy değerine ve modelin eğitimi tamamlandıktan sonra model.evaluate() fonksiyonu ile test datası üzerinde verdiği sonuca baktım. Değerlendirmeler sonucu en başarılı modelin %x doğrulukla Model – x olduğunu görmüş oldum.

**f. 'Test' datası içinde yer alan verilerle en başarılı modeli test etmek ve confusion matrix'i çıkartmak:**




Bu adım için öncelikle class\_names dizisine sınıfların etiketlerini atadım. Bu diziyi confusion matrix hesaplarken axis'lerde kullanacağım. Test datası üzerinde tahmin yapmak için model.predict() fonksiyonunu kullanarak tahmin sonuçlarını başka bir dizide sakladım. Bu sonuçlar içinde en yüksek olasılığa sahip olan sınıfı farklı bir değişkene çektim. Bu değişkende asıl sonuç yer almaktadır. Confusion matrix için Seaborn kütüphanesini kullandım. confusion\_matrix() fonksiyonuna tahmin edilenlerle gerçek değerleri verdim.









**g. Her class'tan seçilen random 3 görüntü ile modeli test etmek ve confusion matrix'ini çıkartmak:**

Öncelikle for içinde 0'dan 9'a kadar gidip her bir sınıfı gezdiğimden emin oldum. Sonrasında her bir sınıf için y\_test'te sıfırdan başlayarak ilgili sınıfa karşılık gelip gelmediğine baktım. Eğer aynı sınıfa ait değilse (0,100) arası random bir sayı üretip indisi o kadar artırıp tekrar aynı sınıf mı diye kontrol sağlıyorum. Böyle 3 tane aynı sınıftan resim bulduğumda while döngüsünden çıkıp diğer bir sınıf için bakmaya geçiyorum. Random görüntüleri elde ettikten sonra bunlar için de tekrar tahmin yapıp confusion matrix elde ettim.

## 2-) Uygulama

 <p>Class: Airplane Predicted: Airplane</p>	airplane 0.999733	bird 0.000221	cat 0.000038	dog 0.000003	automobile 0.000002
 <p>Class: Airplane Predicted: Ship</p>	ship 0.871624	airplane 0.068540	bird 0.028531	deer 0.020772	cat 0.008258
 <p>Class: Airplane Predicted: Horse</p>	horse 0.421964	airplane 0.269515	cat 0.163934	truck 0.059652	ship 0.041916


 <p>Class: Automobile Predicted: Automobile</p>	automobile 0.984334	truck 0.015666	any 0.000000	any 0.000000	any 0.000000
 <p>Class: Automobile Predicted: Automobile</p>	automobile 0.999997	truck 0.000003	any 0.000000	any 0.000000	any 0.000000
 <p>Class: Automobile Predicted: Automobile</p>	automobile 1.000000	any 0.000000	any 0.000000	any 0.000000	any 0.000000




 <p>Class: Bird Predicted: Bird</p>	bird 0.974529	cat 0.010474	deer 0.006948	airplane 0.005232	horse 0.001834
 <p>Class: Bird Predicted: Bird</p>	bird 0.778562	cat 0.155964	frog 0.024973	horse 0.022480	dog 0.009872
 <p>Class: Bird Predicted: Bird</p>	bird 0.873357	airplane 0.126638	ship 0.000004	any 0.000000	any 0.000000

 <p>Class: Cat Predicted: Cat</p>	cat 0.904490	dog 0.056439	airplane 0.010084	frog 0.007448	ship 0.006360
 <p>Class: Cat Predicted: Cat</p>	cat 0.824156	bird 0.055692	deer 0.046578	dog 0.040610	airplane 0.020547
 <p>Class: Cat Predicted: Ship</p>	ship 0.419960	cat 0.352375	dog 0.070043	deer 0.050099	automobile 0.048913









 <p>Class: Deer Predicted: Bird</p>	bird 0.446754	airplane 0.272201	ship 0.110289	deer 0.061028	cat 0.047535
 <p>Class: Deer Predicted: Frog</p>	frog 0.558926	cat 0.117414	deer 0.109190	dog 0.106674	airplane 0.040795
 <p>Class: Deer Predicted: Deer</p>	deer 0.910776	bird 0.045397	cat 0.024058	dog 0.014180	frog 0.003491

 <p>Class: Dog Predicted: Bird</p>	bird 0.613812	cat 0.358068	dog 0.012389	deer 0.005330	airplane 0.001910
 <p>Class: Dog Predicted: Dog</p>	dog 0.525621	horse 0.319428	cat 0.100751	bird 0.024321	deer 0.014074
 <p>Class: Dog Predicted: Cat</p>	cat 0.614143	bird 0.037819	frog 0.153670	dog 0.125740	deer 0.044038

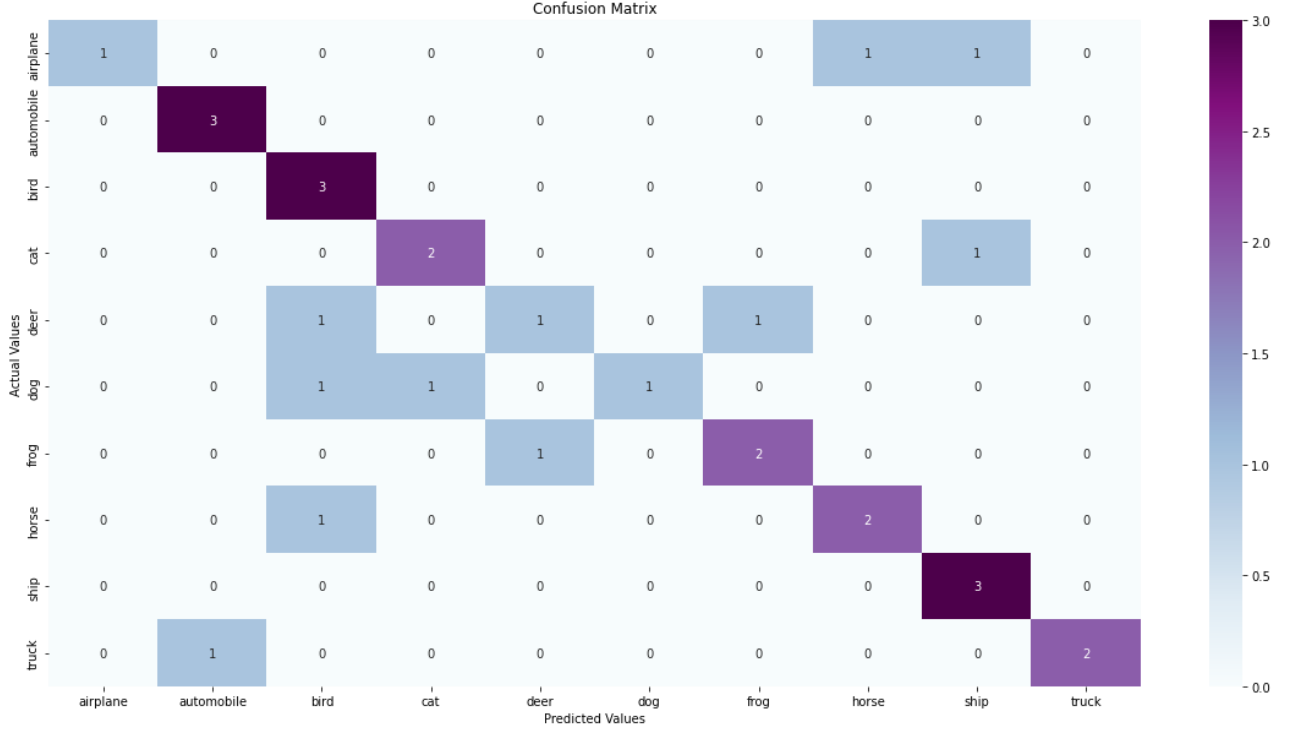
 <p>Class: Frog Predicted: Frog</p>	frog 0.997944	deer 0.001917	bird 0.000133	airplane 0.000005	cat 0.000001
 <p>Class: Frog Predicted: Deer</p>	deer 0.664768	frog 0.305585	cat 0.023484	bird 0.004395	dog 0.001693
 <p>Class: Frog Predicted: Frog</p>	frog 0.732302	bird 0.263639	airplane 0.003955	cat 0.000093	automobile 0.000006

 <p>Class: Horse Predicted: Horse</p>	<p>horse 0.999821</p>	<p>dog 0.000145</p>	<p>bird 0.000023</p>	<p>cat 0.000009</p>	<p>deer 0.000002</p>
 <p>Class: Horse Predicted: Bird</p>	<p>bird 0.733797</p>	<p>airplane 0.071780</p>	<p>horse 0.065643</p>	<p>cat 0.056406</p>	<p>dog 0.055384</p>
 <p>Class: Horse Predicted: Horse</p>	<p>horse 0.973686</p>	<p>deer 0.014220</p>	<p>dog 0.006054</p>	<p>airplane 0.004564</p>	<p>truck 0.001087</p>

 <p>Class: Ship Predicted: Ship</p>	<p>ship 0.614517</p>	<p>truck 0.123186</p>	<p>bird 0.088209</p>	<p>cat 0.042090</p>	<p>airplane 0.039974</p>
 <p>Class: Ship Predicted: Ship</p>	<p>ship 0.337618</p>	<p>cat 0.158588</p>	<p>truck 0.119021</p>	<p>automobile 0.110507</p>	<p>airplane 0.087854</p>
 <p>Class: Ship Predicted: Ship</p>	<p>ship 0.998164</p>	<p>truck 0.001113</p>	<p>automobile 0.000497</p>	<p>airplane 0.000221</p>	<p>cat 0.000004</p>

 <p>Class: Truck Predicted: Automobile</p>	automobile 0.771657	truck 0.228343	any 0.000000	any 0.000000	any 0.000000
 <p>Class: Truck Predicted: Truck</p>	truck 0.997500	automobile 0.001245	airplane 0.001239	ship 0.000016	any 0.000000
	truck 0.814796	automobile 0.180912	cat 0.003928	cat 0.003928	frog 0.000109

30 resim için Confusion Matrix:



### 3-) Sonuç

Eğittiğim modeller içerisinde en yüksek başarımları 5 katmanlı olanlarda aldım. Katman sayısının belirli bir yere kadar artması, modelin özelliklerden daha iyi çıkarım yapmasını sağlar. Fakat sürekli katman eklemek doğruluğu her zaman artırmaz çünkü bir yerden sonra ezberleme oluşur. Bizim örneğimiz için 5 katmanın herhangi bir ezberleme yaratmadığını görmüş oldum.

Modeller arasında filtre sayısının başarıma nasıl etki ettiğini incelediğimde kesin bir sonuca varamadım. Kimi modellerde katman ve filtre boyutu aynı tutulup filtre sayısı değiştirildiğinde 64 filtre sayısı için yüksek başarı görürken, kimisinde 32 filtre için daha iyi sonuç gördüm. Araştırmalarım göre en başarılı sonuçların önce küçük filtre sayısı ile başlayıp katmanlar ilerledikçe artırılmasıyla elde edildiğini öğrendim. Bunun sebebi ilk başta görüntüde noise'un fazla olması ve fazla filtre koymanın çok da iyi sonuç vermeyip zamandan yemesi olabilir.

Filtre boyutuna baktığımda 3x3 filtrelerin, 5x5 filtrelerden daha iyi sonuç verdiğini gördüm. Bunun sebebi bu projede kullanılan görüntülerin boyutlarıyla ilgili olabilir. 5x5 filtre için kenarlardan 2 piksellik bir kayıp olduğunu düşündüğümüzde 32x32 görüntüler için bu yüksek bir kayıp olmaktadır.

Bu parametreler dışında kullanılan Dropout katmanı, kullanıldığı yere ve aldığı değere göre başarıyı artırabilmektedir. Araştırmalarım göre Dropout'un input katmanları için yaklaşık 0.20

değeriyle ve son katmanlar için 0.50-0.80 arası değerlerle kullanılabileceğini gördüm ve ben de kendi modellerimde bu şekilde kullandım.

Eğittiğim 8 modelin içerisinde en başarılı sonucu veren model %67 doğruluk ile , 5 katman, 3x3'lük 64 adet filtre kullanan 5. modeldi.

Video linki: <https://youtu.be/XDaAPdYkei8>