
Tasarladığım Ağ 4 Konvolüsyon ve 2 Tane Dense Katmanından Oluşmaktadır. Ağ için verileri önce cifar100 veri setini import ederek bilgisayarıma PNG olarak kaydeden bunun kontrolü yapan kod yazdım ve benim için verilen 6 sınıfın her birinin 500 adet eğitim 100 adet test verisini kaydettim. Toplamda 3000 eğitim 600 test verisi bulunmaktadır.

## Verilerin Elde Edilmesi ile İlgili Kodlar

## Dosyaların Ayarlanması

Tüm train dosyalarında 500'er Adet resim var.Bu Resimler cifar100'de ilgili sınıfa aitse o klasöre resim kaydediliyor. Train Datadaki verilerin dosya içindeki Kontrolü kolay olsun diye hepsine bir değişken tanımladım ve dosya isimlerine göre kontrol yaptım

```
sayi = train_data.shape[0]
baby=0
caterpillar=0
mushroom=0
pickup_truck=0
shark=0
turtle=0
#Tüm veriseti dolaşılıyor.İlgili sınıflar ilgili dosyalara kaydediliyor.
from PIL import Image
```

```
for i in  range(sayi):

    im = Image.fromarray(train_data[i])
    if train_targets[i]==2:
        im.save("train\\baby\{}.png".format(baby))
        baby+=1

    if train_targets[i]==18:
        im.save("train\\caterpillar\{}.png".format(caterpillar))
        caterpillar+=1
    if train_targets[i]==51:
        im.save("train\\mushroom\{}.png".format(mushroom))
        mushroom+=1
    if train_targets[i]==58:
        im.save("train\\pickup_truck\{}.png".format(pickup_truck))
        pickup_truck+=1
    if train_targets[i]==73:
        im.save("train\\shark\{}.png".format(shark))
        shark+=1
    if train_targets[i]==93:
        im.save("train\\turtle\{}.png".format(turtle))
        turtle+=1
```

Bütün işlemler Bu Sefer test datası için yapılıyor. Buradaki degişkenler kontrol için sıfırlanıyor ve her tüm test dosyalarında 100'er adet resim var.

Bilgisayardaki train klasöründeki her bir dizine girilip resimler toplanır
Toplanan resimlerin verisinden train_data oluşturuluyor.

```
Dosyalar = [os.path.join("train",fname) for fname in os.listdir("train")]
Say=0
Index=0
#Sınıf Sayısı Belirleniyor
sinifSayisi = 6
TrainTarget = []

for dosya in Dosyalar:
    TrainResimler = [os.path.join(dosya,fname) for fname in os.listdir(dosya)]
    if Say==0:
        x = np.array([np.array(image.load_img(res,target_size=(32,32))) for res in TrainResimler])
        TrainData = x
        if dosya == "train\\baby":
```

```
        for Index in range(500):
          #Gelen Resimin dosya ismine göre o indexte eğitim hedefi dolduruluyor.
            TrainTarget.append([1])

    if dosya == "train\\caterpillar":
        for Index in range(500):
            TrainTarget.append([2])

    if dosya == "train\\shark":
        for Index in range(500):
            TrainTarget.append([3])

    if dosya == "train\\mushroom":
        for Index in range(500):
            TrainTarget.append([4])

    if dosya == "train\\pickup_truck":
        for Index in range(500):
            TrainTarget.append([5])

    if dosya == "train\\turtle":
        for Index in range(500):
            TrainTarget.append([0])

    Say+=1
else:

    if dosya == "train\\baby":
        for Index in range(500):
            TrainTarget.append([1])

    if dosya == "train\\caterpillar":
        for Index in range(500):
            TrainTarget.append([2])

    if dosya == "train\\shark":
        for Index in range(500):
            TrainTarget.append([3])

    if dosya == "train\\mushroom":
        for Index in range(500):
            TrainTarget.append([4])

    if dosya == "train\\pickup_truck":
        for Index in range(500):
```

```
            TrainTarget.append([5])

    if dosya == "train\\turtle":
        for Index in range(500):
            TrainTarget.append([0])
```

#Ayrı Ayrı alınan resimlerin verileri Tek bir dizide TrainData da birleştiriliyor.

```
x = np.array([np.array(image.load_img(res,target_size=(32,32))) for res in TrainResimler])
TrainData = np.vstack((TrainData, x))
```

Aynı işlemler test verileri içinde yapılıyor.Test Data Dolduruyor.

Veriler hazırlandıktan sonra Ağ Oluşturuluyor.Ağda 4 Konvolüsyon Katmanı ve 2 Adet Dense Katmanı Vardır.

```
model=models.Sequential()
model.add(layers.Conv2D(32,(3,3),activation="relu",input_shape=(32,32,3)))

model.add(layers.Conv2D(64,(2,2),activation="relu"))
model.add(layers.MaxPooling2D(2,2))
model.add(layers.Conv2D(64,(2,2),activation="relu"))
model.add(layers.MaxPooling2D(2,2))
model.add(layers.Conv2D(128,(2,2),activation="relu"))

model.add(layers.Flatten())
```

Dropout İlk Eğitimde Eklenmedi Sonraki Eğitimde Buraya Eklendi.

```
#model.add(layers.Dropout(0.5))
model.add(layers.Dense(64,activation="relu"))
```

#Karar katmani oldugundan Sinif Sayisi parametre olarak veriliyor.

```
model.add(layers.Dense(sinifSayisi,activation="softmax"))
model.summary()
```

#Optimizasyon Birden fazla sınıf olduğu için categorical_crossentropy olarak belirlendi.

```
model.compile(optimizer="adam",
              loss='categorical_crossentropy',
              metrics=['accuracy'])
```

# Ağ Özeti:

```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 30, 30, 32)        896
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 29, 29, 64)        8256
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 14, 14, 64)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 13, 13, 64)        16448
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 6, 6, 64)          0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 5, 5, 128)         32896
_________________________________________________________________
flatten_1 (Flatten)          (None, 3200)              0
_________________________________________________________________
dense_1 (Dense)              (None, 64)                204864
_________________________________________________________________
dense_2 (Dense)              (None, 6)                 390
=================================================================
Total params: 263,750
Trainable params: 263,750
Non-trainable params: 0
```

# İlk Ağ Eğitimi

```
TrainTargetArray = np.array(TrainTarget)
TestTargetArray = np.array(TestTarget)
#Categorical Crossentropy Kullanabilmek icin Hedef Verileri Categorilendiriliyor.
TrainTargets = np_utils.to_categorical(TrainTargetArray, sinifSayisi)
TestTargets = np_utils.to_categorical(TestTargetArray, sinifSayisi)
```

Dosyalardan Çektiğimiz için veriler sıralı haldeydi daha doğru bir eğitim için eğitim ve test verileri kendi aralarında karıştırılıyor. Buna bağlı olarak Targets değerleri de aynı şekilde karıştırılıyor. Böylelikle değerin karşılık geldiği Hedef değeri değişmemiş oluyor.

```
TrainTargets,TrainData = shuffle(TrainTargets,TrainData, random_state=0)
TestData,TestTargets = shuffle(TestData,TestTargets,random_state=0)
h=model.fit(TrainData,
        TrainTargets,
        epochs=15,
        batch_size=4,
        verbose=1,
        validation_data=(TestData, TestTargets))
```

# Veri Zenginlestirme

## Test ve Train konumları ayarlanıyor.

```
train_dir = 'train'
test_dir = 'test'
train_datagen = ImageDataGenerator(zoom_range=0.3, #0.3 Oraninda Yaklastirma
                        rotation_range=25,                  #25 Derece Dondurme Uygulaniyor
                        horizontal_flip=True)               #Yatay Ters Cevirme
test_datagen = ImageDataGenerator(zoom_range=0.3,
                        rotation_range=25,
                        horizontal_flip=True)
train_generator = train_datagen.flow_from_directory(train_dir,
                                batch_size=10,
                                target_size=(32,32),
                                class_mode="categorical",   #Birden fazla sınıf Categorical Ayarlanıyor
                                shuffle=True)               #Veriler sıralı çekildiği için karıştırılıyor.
test_generator = test_datagen.flow_from_directory(test_dir,
                                  target_size=(32,32),
                                  batch_size=10,
                                  class_mode="categorical",
                                  shuffle=True)
```

## 3000 Veri Olduğu İçin batch_size = 10 , steps_per_epoch=300 Olarak Belirlendi.

## 500 Veri Olduğu İçin batch_size = 10 , steps_per_epoch=50 Olarak Belirlendi.

## Ağ Eğitimi Yapılıyor.

```
h = model.fit_generator(train_generator,
                    steps_per_epoch=300,
                    epochs=15,
                    validation_data=train_generator,
                    validation_steps=50)
```

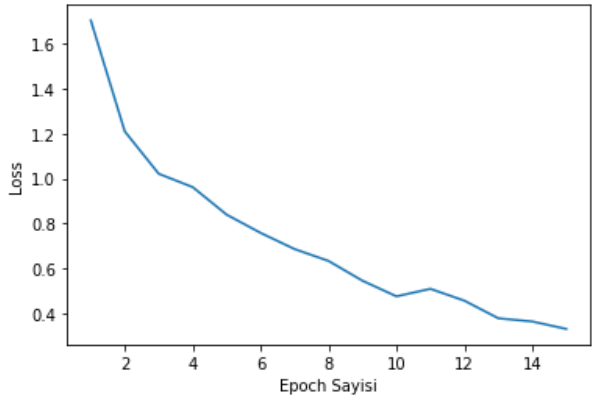
Veri Zenginleştirme ile eğitilen ağ test ediliyor.

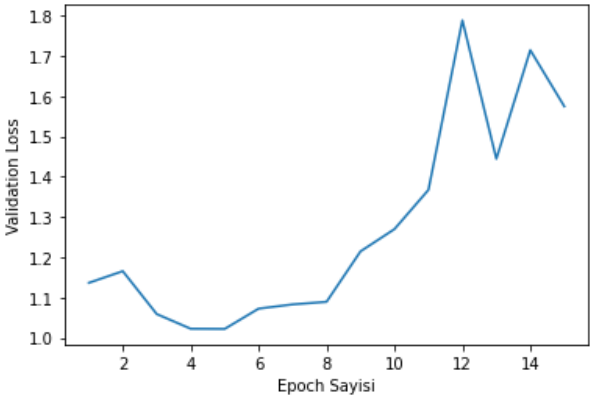
model.evaluate_generator(generator=test_generator, steps=20)

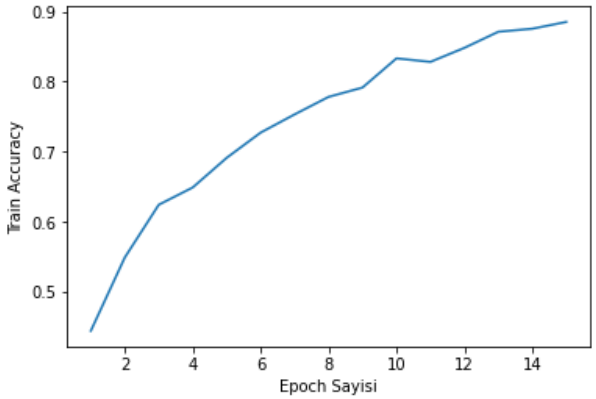
# Sonuç Grafikleri

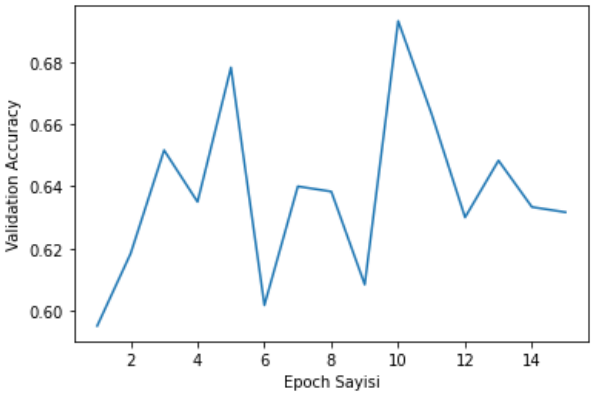
## İlk Test

Ağın ilk eğitiminde sonuçlar :Validation Accuracy %60 ve üzeri değerlere ulaştı. Kullandığım bilgisayar yavaş olduğundan dolayı süreyi kısaltmak için epoch sayısını 15 olarak aldım. Validation Loss anlamsız bir şekilde artıp azalırken Validation Accuracy kararlılığını korudu.



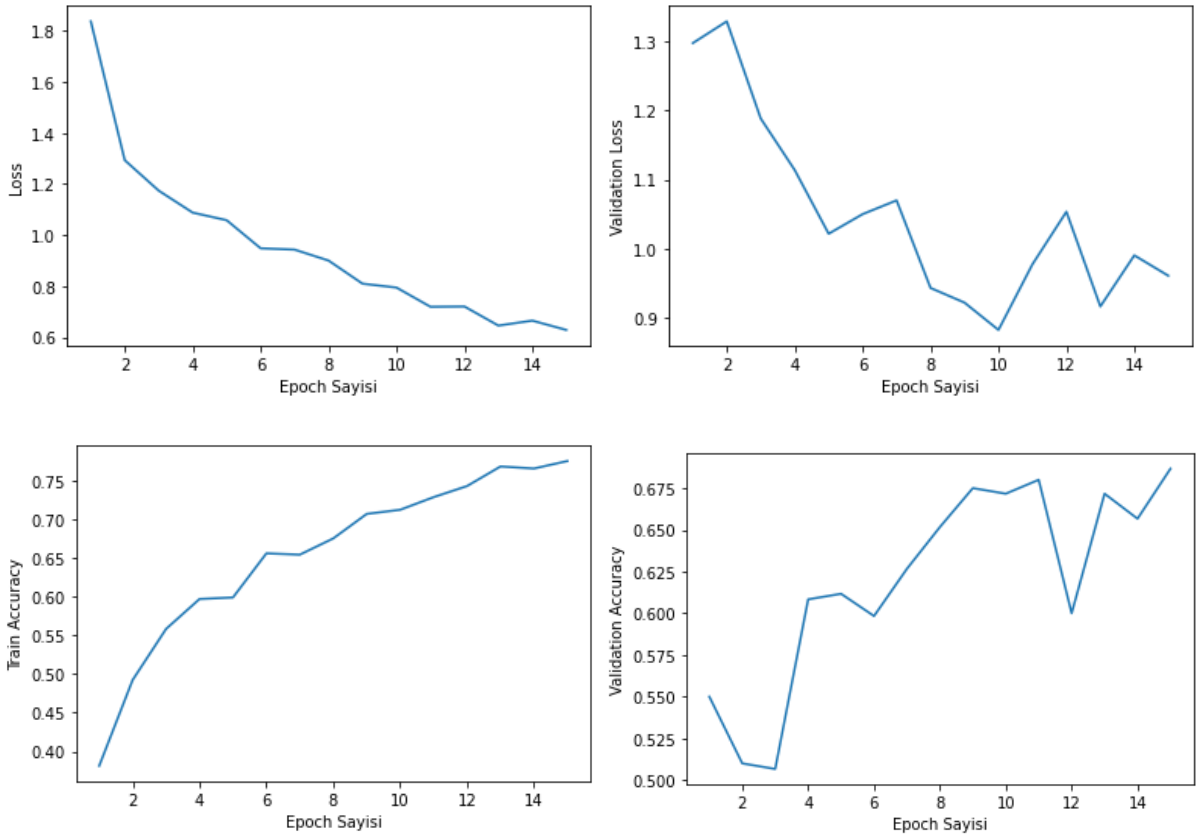



# Dropout

Dropout eklenmeden önce Eğitim Doğruluğu %90 civarlarına ulaşmıştı epoch biraz daha fazla olsaydı eğitimde ezberlemeye sebep olacaktı.Dropout kullanarak eğitim doğruluğu %90 iken aynı epoch sayısında %75 doğruluğa düşürülüp ağın ezberlemesi engellendi. Dropout eklenince ağın test doğruluğu artmış oldu.



# Veri Zenginleştirme

Veri  zenginleştirme ile tekrardan eğitim yapıldığında ağın test doğruluğu belirgin oranda arttı. Buna karşın eğitim doğruluğu ise düşüş yaşadı. Böylelikle ezberleme yapması da engellenmiş oldu. Veri zenginleştirmede resimlerin 25 derece döndürülmesi yatay ters çevrilmesi ve 0.3 oranında yaklaştırılması işlemleri yapıldı.