T.C.
SAKARYA ÜNİVERSİTESİ
BİLGİSAYAR VE BİLİŞİM BİLİMLERİ FAKÜLTESİ

BSM 432 DERİN ÖĞRENME VE EVRİŞİMLİ SİNİR AĞLARI

B161210074 -Ahmet Yasir AKBAL

Bölüm : BİLGİSAYAR MÜHENDİSLİĞİ
Dersi Veren : Doç.Dr. DEVRİM AKGÜN

2019-2020 Bahar Dönemi

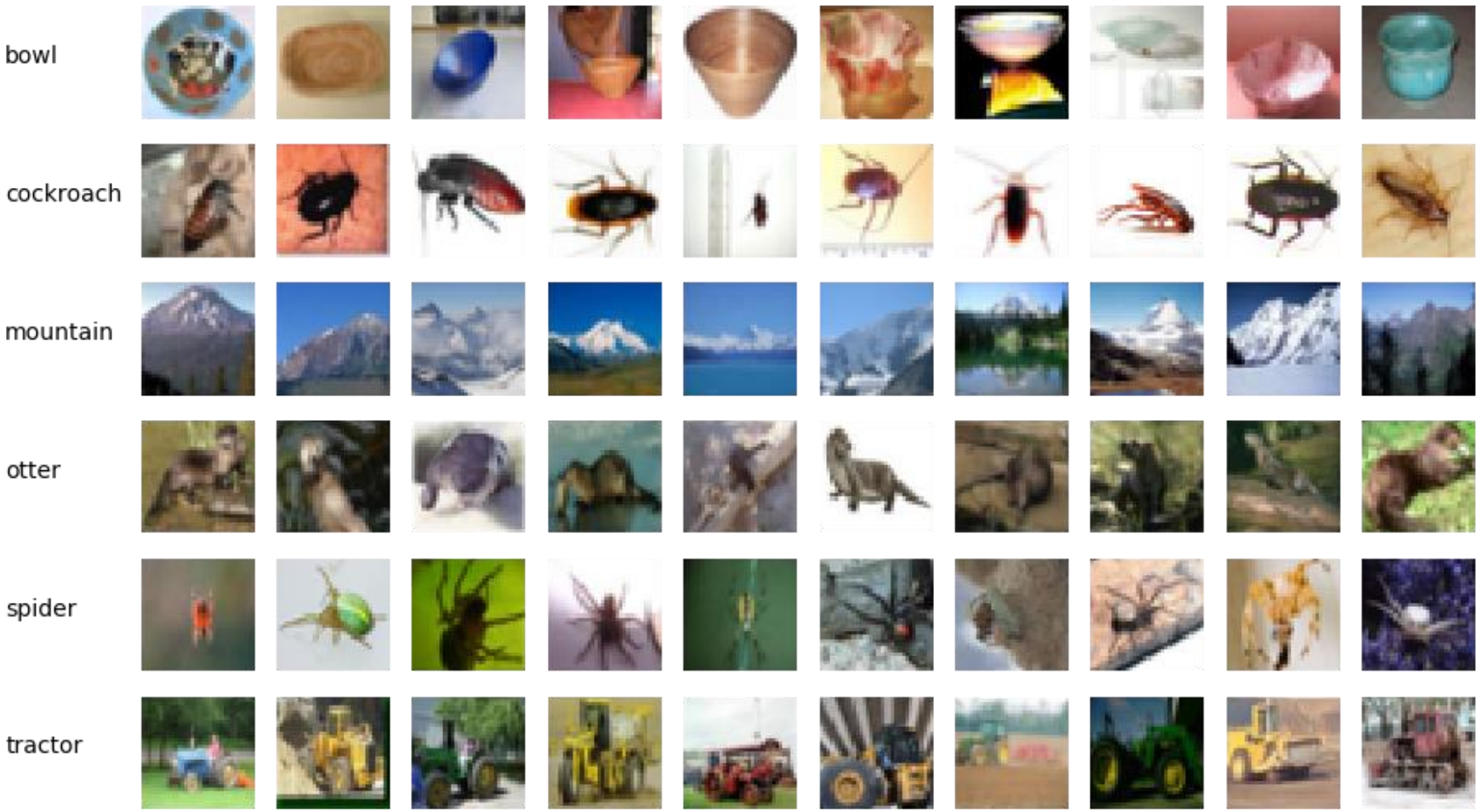
BÖLÜM 1. VERİSETİNİ TANIYALIM VE DÜZENLEYELİM

Kullandığım veriseti Cifar100 olup 20 üst sınıf ve herbir üst sınıftaki 5'er sınıf olmak üzere toplam 100 ayrı sınıftan oluşan bir resim verisetidir. Verisetinde toplam 60 bin resim olmakla birlikte bunların 50 bini eğitim, 10 bini test aşaması için ayrılmıştır. Her sınıftan toplamda 600(500 train,100 test) tane örnek bulunmaktadır. Herbir resim 32x32 boyutunda olup 3(RGB) kanallıdır.

Superclass	Classes
aquatic mammals	beaver, dolphin, otter, seal, whale
fish	aquarium fish, flatfish, ray, shark, trout
flowers	orchids, poppies, roses, sunflowers, tulips
food containers	bottles, bowls, cans, cups, plates
fruit and vegetables	apples, mushrooms, oranges, pears, sweet peppers
household electrical devices	clock, computer keyboard, lamp, telephone, television
household furniture	bed, chair, couch, table, wardrobe
insects	bee, beetle, butterfly, caterpillar, cockroach
large carnivores	bear, leopard, lion, tiger, wolf
large man-made outdoor things	bridge, castle, house, road, skyscraper
large natural outdoor scenes	cloud, forest, mountain, plain, sea
large omnivores and herbivores	camel, cattle, chimpanzee, elephant, kangaroo
medium-sized mammals	fox, porcupine, possum, raccoon, skunk
non-insect invertebrates	crab, lobster, snail, spider, worm
people	baby, boy, girl, man, woman
reptiles	crocodile, dinosaur, lizard, snake, turtle
small mammals	hamster, mouse, rabbit, shrew, squirrel
trees	maple, oak, palm, pine, willow
vehicles 1	bicycle, bus, motorcycle, pickup truck, train
vehicles 2	lawn-mower, rocket, streetcar, tank, tractor

Şekil 1.1. Cifar100 Veriseti Üst ve Alt Sınıflar

Benim projemde kullandığım 6 sınıf ise : bowl(kase), cockroach(hamam böceği), mountain(dağ), otter(su samuru), spider(örümcek), tractor(traktör)'dür. Her sınıftan 10'ar tane örneğin listelendiği görsel aşağıdadır:



Şekil 1.2. Kullandığım sınıflardan örnek

Verisetinin yüklenmesi, bölünmesi, görselleştirilmesi ve düzenlenmesi

```
(x_train,y_train),(x_test,y_test) = cifar100.load_data()

siniflarim = dict({10:"bowl",24:"cockroach",49:"mountain",55:"otter",79:"spider",89:"tractor"})
```

Şekil 1.3. Verisetinin yüklenmesi

Yukarıdaki kod parçacığında keras datasets modülünden import ettiğim cifar100 modülünden verisetimi load_data metodu ile belleğime yüklüyorum. Ayrıca daha sonraki aşamalarda verisetinden benim projemde kullanacağım sınıfları seçmemi kolaylaştıracak sözlük tipinden "siniflarim" değişkenimi oluşturuyorum.

```
genislik = x_train.shape[1]
uzunluk = x_train.shape[2]
kanal_sayisi = x_train.shape[3]
batch_size = 50
sinif_sayisi = len(siniflarim)
epochs = 75
```

Şekil 1.4. Önemli değerlerin değişken olarak tanımlanması

Şekil 1.4.'deki kod parçacığında ilerleyen zamanlarda kullanacağım bilgileri kolaylık olması açısından değişken olarak tanımladım.

```
kucuk_veriseti_yolu = "kucukcifar100"

os.mkdir(kucuk_veriseti_yolu)

train_dir = os.path.join(kucuk_veriseti_yolu, 'train')
os.mkdir(train_dir)
test_dir = os.path.join(kucuk_veriseti_yolu, 'test')
os.mkdir(test_dir)

resimleri_kaydet(x_train,y_train,siniflarim,train_dir)
resimleri_kaydet(x_test,y_test,siniflarim,test_dir)
```

Şekil 1.5. Klasörlerin oluşturulması

Şekil 1.5.'de diskte "kucukcifar100" adında bir ana klasor ve bunun içinde "train" ve "test" adında 2 alt klasor oluşturuyorum. Daha sonra "resimleri_kaydet" metodunu

train ve test kısmı için projemde kullanacağım sınıflara ait resimleri diske kaydetmesi için ilgili parametreleri vererek çağırıyorum.

```
def resimleri_kaydet(x,y,siniflar,dosya_yol):
    for k,v in siniflar.items():
        sinif_yol = os.path.join(dosya_yol, v)
        os.mkdir(sinif_yol)
        resimler = x[y[:,0] == k]
        for i,resim in enumerate(resimler):
            plt.imsave(sinif_yol +"\\"+ str(i)+".png",resim)
```

Şekil 1.6. resimleri_kaydet metodu

Şekil 1.6'daki metod ile verilen parametreler ile projemde kullanacağım verileri seçiyor ve bunları herbir sınıfa ait örnekler kendi sınıf adına ait klasör içinde bulunacak şekilde diske kaydediyorum.

```
fig,ax = plt.subplots(6,11,figsize=(16,9))

labels = list(siniflarim.keys())

for i in range(6):
    resimler = x_train[y_train[:,0] == labels[i]]
    ax[i,0].axis("off")
    ax[i,0].text(0,0.5,siniflarim[labels[i]],fontdict={"size":14})
    for j in range(1,11):
        ax[i,j].axis("off")
        ax[i,j].imshow(resimler[j])

plt.show()
```

Şekil 1.7. Her Sınıftan 10'ar Resmin Gösterilmesi

Bu kod parçacağında ödevde istenildiği üzere Şekil 1.2.'deki veri örneklerinin görselleştirilmesi işlemini yapıyorum.

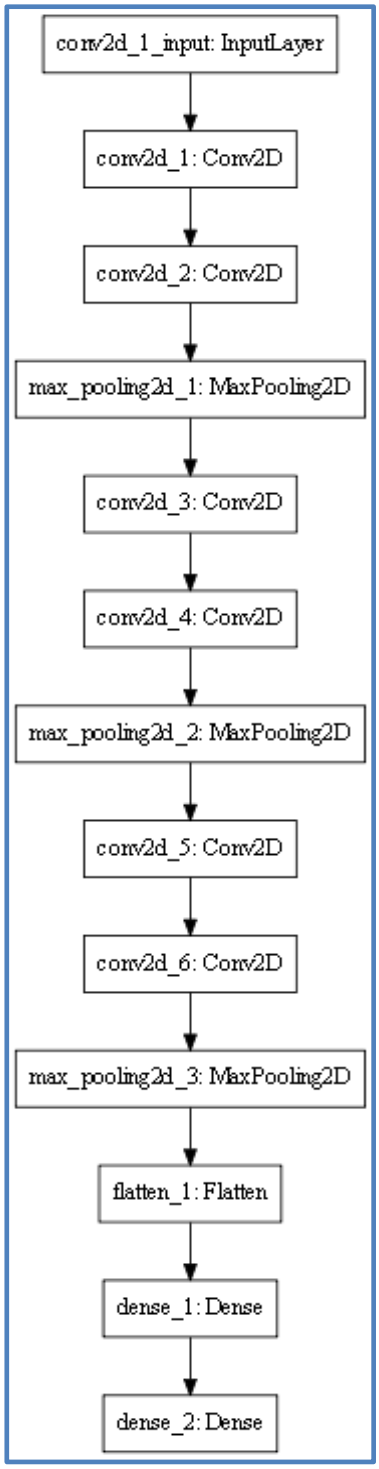
Image Generator-Data Augmentation ve DataGeneratorlerin tanımlanması

```
def generator_olustur(train_dir,test_dir,augmentation=False):
    if(augmentation):
        train_datagen = image.ImageDataGenerator(rescale=1./255,
        rotation_range=30,
        width_shift_range=0.15,
        height_shift_range=0.15,
        shear_range=0.15,
        zoom_range=0.15,
        horizontal_flip=True)
    else:
        train_datagen = image.ImageDataGenerator(rescale=1./255)

    test_datagen = image.ImageDataGenerator(rescale=1/255.)

    train_gen = train_datagen.flow_from_directory(train_dir,target_size=(genislik,uzunluk),
                                                  class_mode="categorical",batch_size=batch_size)
    test_gen = test_datagen.flow_from_directory(test_dir,target_size=(genislik,uzunluk),
                                                  class_mode="categorical",batch_size=batch_size)

    return train_gen,test_gen
```

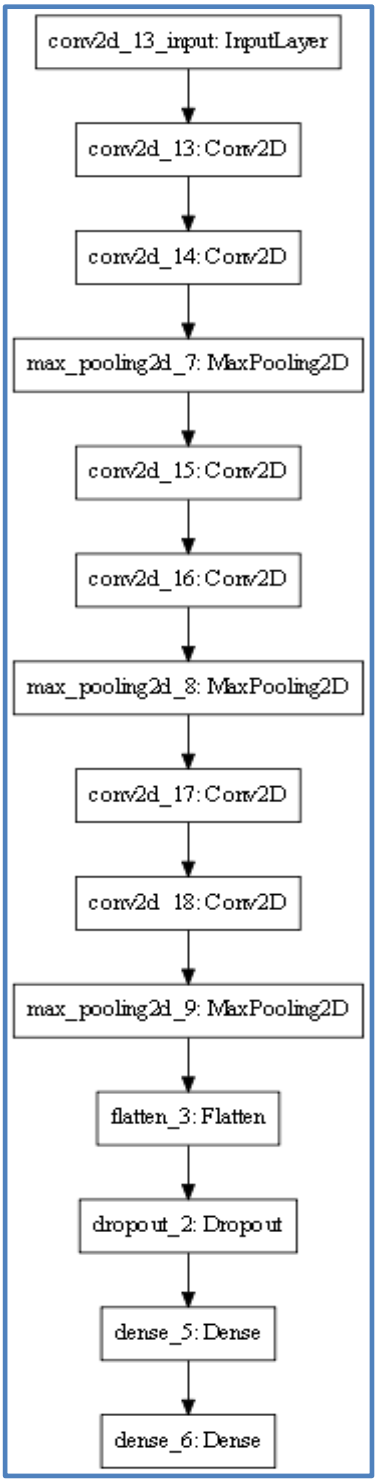
Şekil 1.7. generator_olustur metodu

Şekil 1.7.'deki metotta parametre olarak alınan augmentation değişkeninin değerine göre bir veri zenginleştirme(data augmentation) işlemi yapılıp yapılmayacağına karar veriyorum. Sonrasında yine parametre olarak verilen train ve test dosya yollarına göre datageneratorlerimi oluşturuyorum. Son olarak oluşturduğum generatorleri metottan geri dönüyorum. Veri zenginleştirme olarak resmi döndürme, dikey ve yatay eksende kaydırma, yatay eksende ters çevirme, yakınlaştırma, makaslama gibi işlemler yaptım.

2. KULLANDIĞIM MODEL VE PARAMETRELER



Şekil 2.1. Dropoutsuz Model



Şekil 2.2. Dropoutlu Model

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 32, 32, 32)	896
conv2d_2 (Conv2D)	(None, 32, 32, 32)	9248
max_pooling2d_1 (MaxPooling2	(None, 16, 16, 32)	0
conv2d_3 (Conv2D)	(None, 16, 16, 64)	18496
conv2d_4 (Conv2D)	(None, 16, 16, 64)	36928
max_pooling2d_2 (MaxPooling2	(None, 8, 8, 64)	0
conv2d_5 (Conv2D)	(None, 8, 8, 128)	73856
conv2d_6 (Conv2D)	(None, 8, 8, 128)	147584
max_pooling2d_3 (MaxPooling2	(None, 4, 4, 128)	0
flatten_1 (Flatten)	(None, 2048)	0
dense_1 (Dense)	(None, 256)	524544
dense_2 (Dense)	(None, 6)	1542

Total params: 813,094
Trainable params: 813,094
Non-trainable params: 0

Şekil 2.3. Dropoutsuz Model

Şekillerde görüldüğü üzere resmimi direkt olarak orjinal boyutunda girdi katmanından aldım. Daha sonra buna 2 kere 3x3 lük 32 tane filtreyle konvolüsyon işlemi uyguladım ve ardından 2x2 boyutunda filtremle stride sayısını 2 seçerek matrisime MaxPooling işlemi uyguladım. Bu 2 konvolüsyon işlemi ve havuzlama işlemini bir katman olarak sayarsak bu katmanı 2 kez daha sırasıyla 64 ve 128 adet filtre sayılarıyla tekrarladım. Bu işlemlerden elimdeki son matrisin boyutu 4x4x128 oldu. Havuzlama işlemleriyle parametre sayısını istediğim şekilde ayarladıktan sonra artık Flatten işlemiyle Tam Bağlı Katman(Fully Connected Layer/Dense Layer) kısmına geçtim. Flatten katmanından sonra 256 nöronlu bir katman daha ekleyerek onun çıkışına da sınıf sayım kadar nöron içeren çıktı katmanımı bağladım.

```
def model_olustur(dropout=False):
    model = models.Sequential()
    model.add(layers.Conv2D(32,(3,3),padding="same",activation="relu",
                            input_shape=(genislik,uzunluk,kanal_sayisi)))
    model.add(layers.Conv2D(32,(3,3),padding="same",activation="relu"))
    model.add(layers.MaxPooling2D()) #16x16x3
    model.add(layers.Conv2D(64,(3,3),padding="same",activation="relu"))
    model.add(layers.Conv2D(64,(3,3),padding="same",activation="relu"))
    model.add(layers.MaxPooling2D()) #8x8x3
    model.add(layers.Conv2D(128,(3,3),padding="same",activation="relu"))
    model.add(layers.Conv2D(128,(3,3),padding="same",activation="relu"))
    model.add(layers.MaxPooling2D()) #4x4x3

    model.add(layers.Flatten())
    if(dropout):
        model.add(layers.Dropout(0.5))
    model.add(layers.Dense(256,activation="relu"))
    model.add(layers.Dense(sinif_sayisi,activation="softmax"))

    model.compile(optimizer=optimizers.Adam(lr=0.0001),
                  loss="categorical_crossentropy",metrics=["acc"])
    model.summary()

    return model
```

Şekil 2.4. Model Oluşturma Metodum

Şekil 2.4'de görüldüğü üzere konvolüsyon işlemi sırasında padding olarak "same" işlemini seçtim yani konvolüsyon işlemi sonucu boyutun değişmesini engelledim. Konvolüsyon katmanlarında maliyeti diğer fonksiyonlara göre çok daha düşük ve başarısı iyi bir seviyede olduğu için aktivasyon fonksiyonu olarak "relu" kullandım. Flatten katmanında sonra metoda verilen parametreye göre bir Dropout katmanının eklenip eklenmeyeceğinin seçimini yaptım. Son katmanda modele Multi-Class Classfication işlemi yaptırmak istediğim için "softmax" aktivasyon fonksiyonunu kullandım.

Optimizasyon algoritması olarak yakınsama hızı ve başarımı yüksek olan Adam algoritmasını kullandım. Loss fonksiyonu için ise yine Multi-Class Classfication işlemi yaptığım için loss fonksiyonu olarak categorical_crossentropy kullandım. Bir sınıflandırma işlemi yaptığım ve verisetim düzenli bir şekilde dağıldığı için metrik olarak accuracy değeri üzerinden değerlendime yaptım.

3. MODELİ EĞİTME

```
train_gen,test_gen = generator_olustur(train_dir,test_dir)
model = model_olustur()
historyNormal = model.fit_generator(train_gen,
                steps_per_epoch=train_gen.samples//batch_size,
                epochs=epochs,validation_data=test_gen,
                validation_steps=test_gen.samples//batch_size)
```

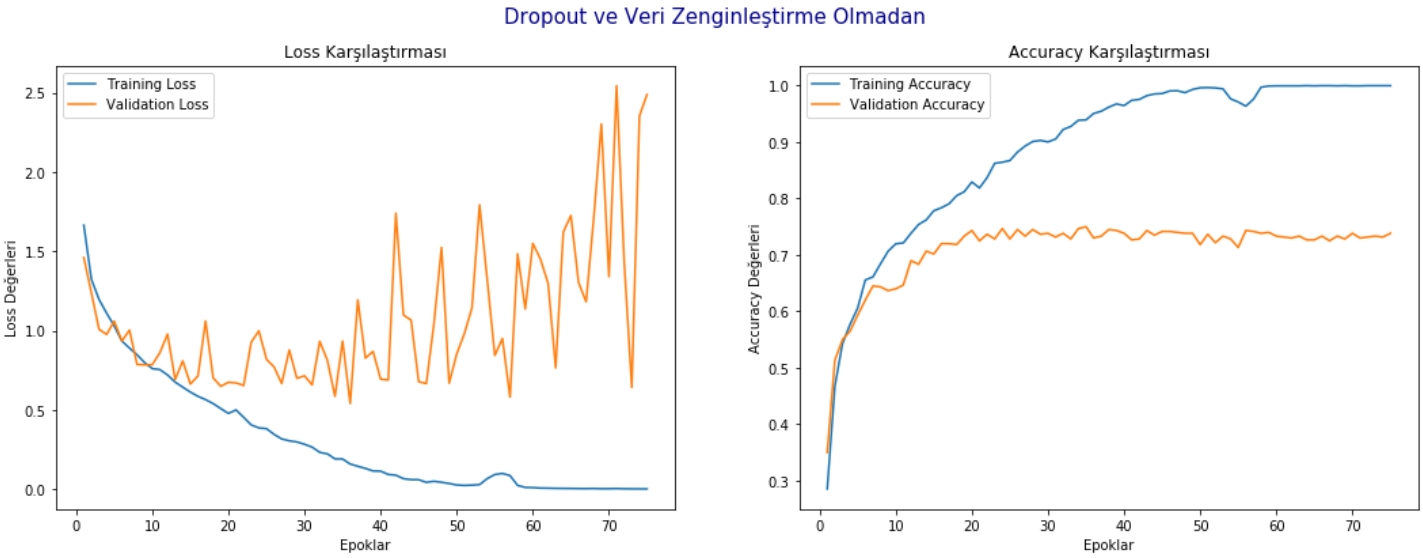
Şekil 3.1. Dropoutsuz ve Augmentationsuz Modelin Eğitilmesi

Şekil 3.1.'de dropoutsuz ve data augmentationsuz modelin eğitilmesi için gerekli kodlar gözükmedir. Dropout ve data augmentation kullanılan modeller için tek fark metotların parametrelerindedir.

4. MODELLERİN KARŞILAŞTIRILMASI

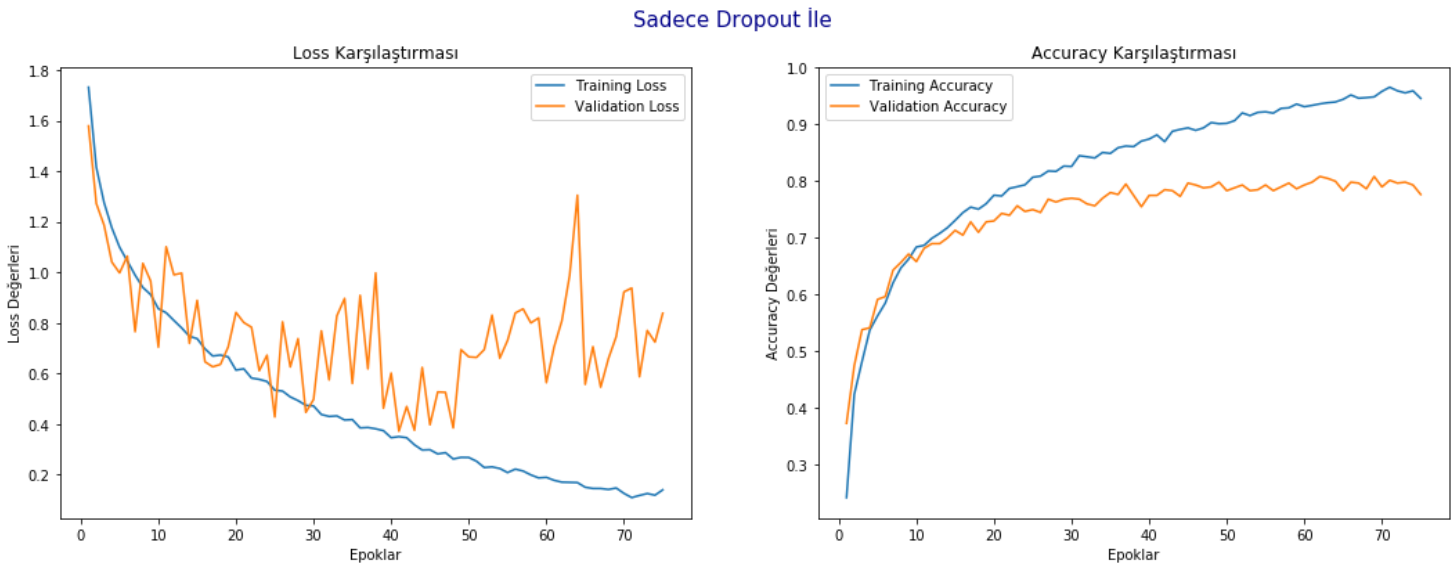
Tablo 4.1. Modellerin Maksimum Train ve Validation Başarımları

MODEL	TRAIN ACCURACY(%)	VALİDATİON ACCURACY(%)
Normal Model	100,00	75,00
Dropotlu	96,57	80,83
Augmentationli	81,60	82,50



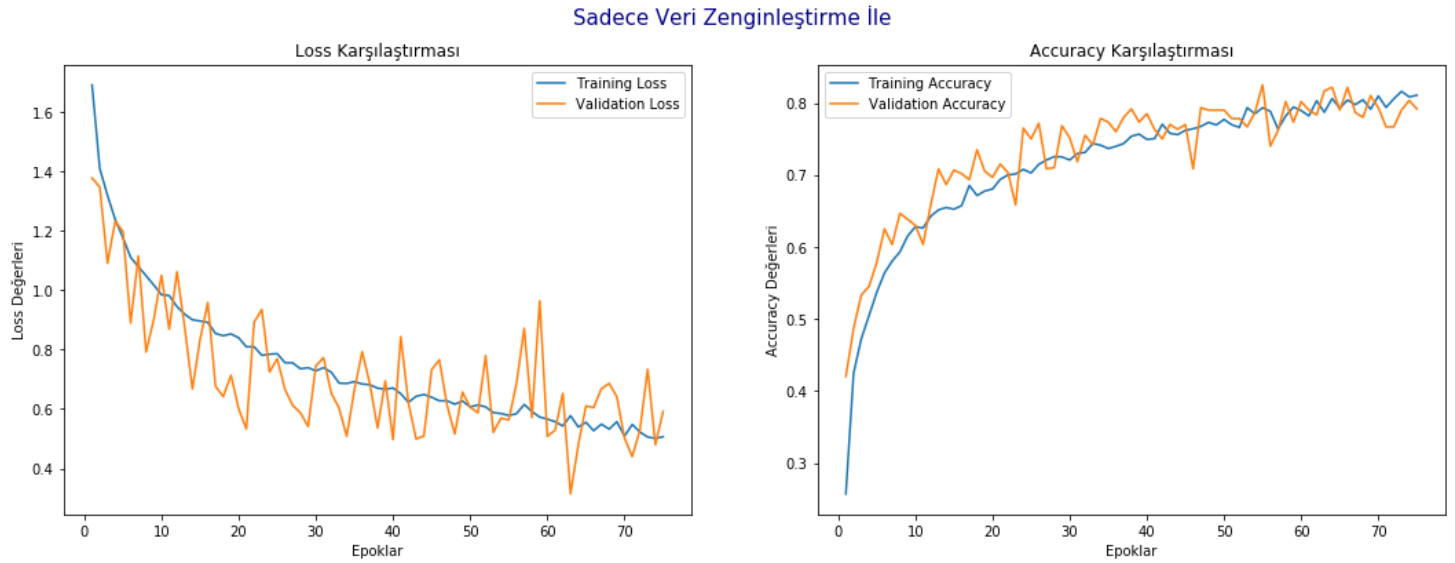
Şekil 4.1. Dropout ve Augmentationsiz Modelin Verileri

Şekil 4.1.'deki grafiklere bakarak modelin aşırı uydurduğunu söylebiliriz. Train accuracy sürekli olarak artarak 1 değerine ulaşırken, train loss sürekli azalarak 0 değerine ulaşırken, validation accuracy azalmamış ve validation accuracy aynı şekilde artamamıştır. Bunun sonucu olarak varyans çok yükselmiş yani aşırı uydurma sorunu yaşanmıştır. Modelde 20. epoktan sonra validation accuracy değerinde çok ufak değişiklikler olmuştur. Dolayısıyla bu modeli 20 epok eğitmek yeterli olacaktır.



Şekil 4.2. Dropoutlu Modelin Verileri

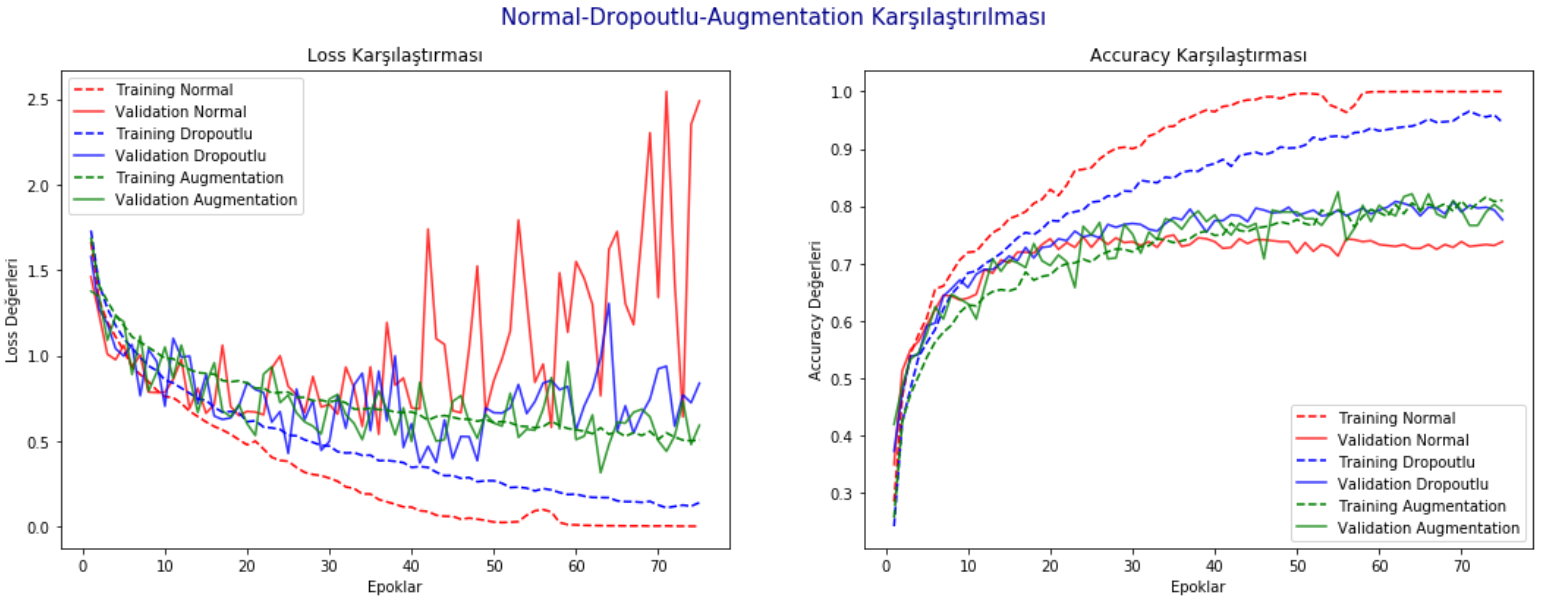
Şekil 4.2.deki grafiği yorumlayacak olursak dropout katmanının olduğu bu modelde de aşırı uydurma sorununun olduğunu göreceğiz. Train lossumuz 0'a, train accuracyimiz ise 1'e çok yaklaşırken aynı şey validation için gözlemlenmemiştir. Bu modeli de yaklaşık olarak 35-40 epoch kadar eğitmek yeterli olacaktır ya da early stopping işlemi uygulanabilir.



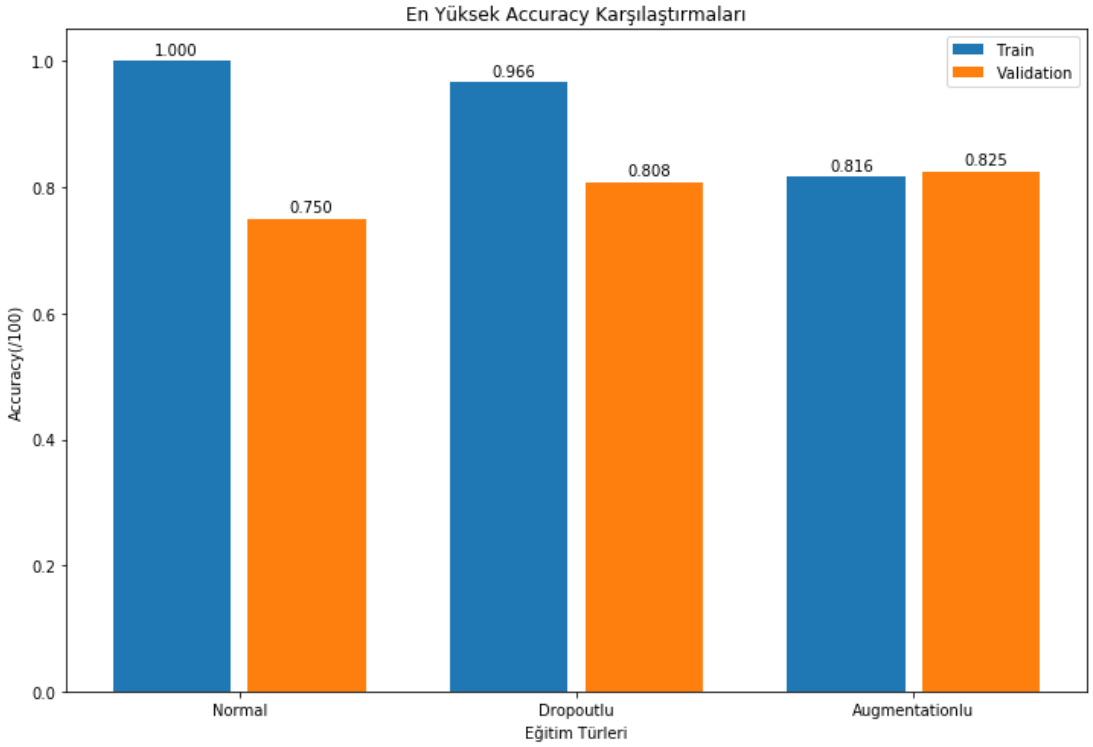
Şekil 4.3. Veri Zenginleştirme ile Eğitilen Modelin Verileri

Şekil 4.3.'deki modelin aşırı uydurma sorunu olmadığı açıkca görülmektedir. Validation loss çoğu zaman train loss ile aynı seviyede hatta kimi zaman daha düşük seviyededir. Aynı şekilde validation accuracy de train accuracy ile çoğu yerde aynı

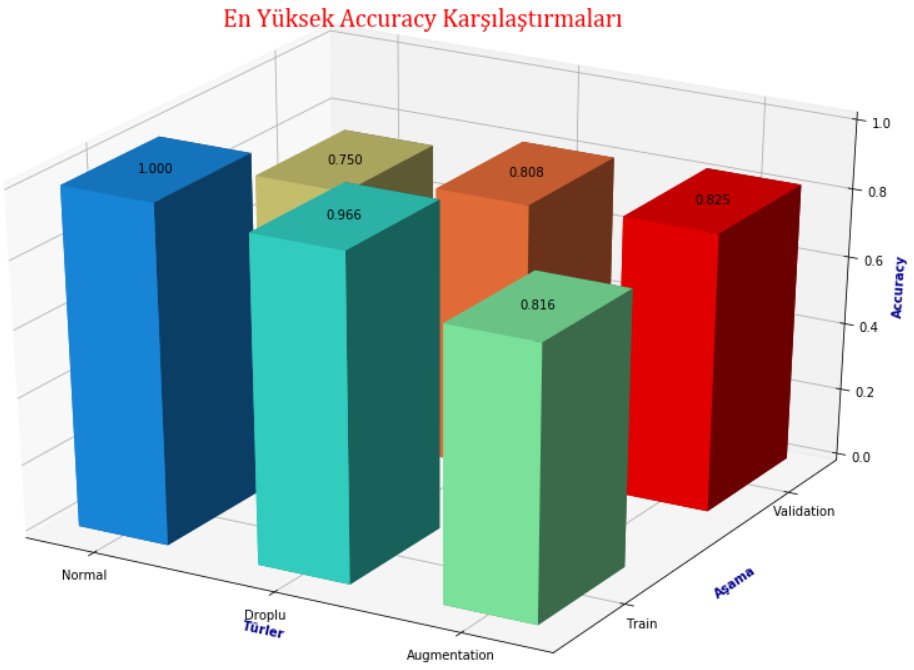
seviyede olmakla birlikte bazı yerlerde de daha yüksektir. Train ve validation accuracy değerleri birbirine çok yakın olduğu için varyans da sıfıra yakın olacaktır. Bu bilgiler sonucucu olarak aşırı uydurma probleminin olmadığı sonucana vardık.



Şekil 4.4. Tüm Modellerin Karşılaştırılması(çizgi grafiği ile)



Şekil 4.5. Tüm Modellerin Karşılaştırılması(sutün grafiği ile)

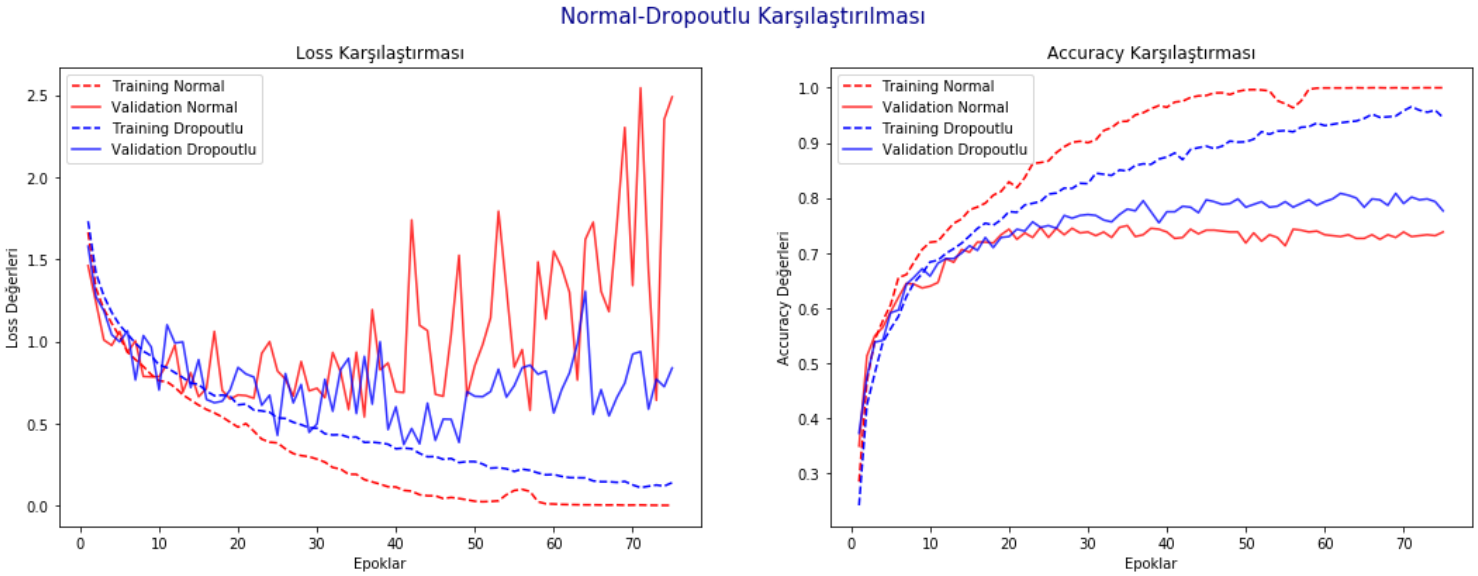


Şekil 4.6. Tüm Modellerin Karşılaştırılması(3 boyutlu sutün grafiği ile)

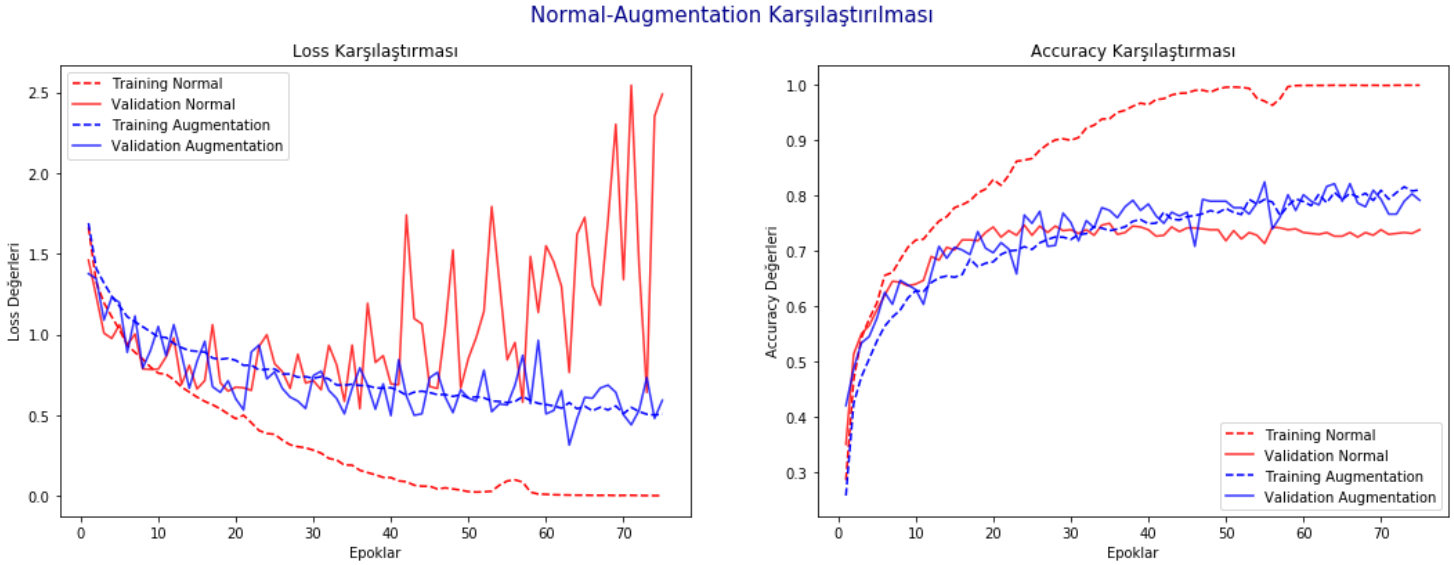
Şekil 4.4, 4.5 ve 4.6 incelenirse train aşamasında en iyi başarımın dropout ve data augmentation işlemlerini olmadığı modelde olduğu görülmektedir. Fakat bizim asıl istediğimiz modelin görmediği veriler üzerindeki başarımıdır. Bu noktada ise dropout ve data augmentation işlemlerinin uygulanmadığı modelin başarımı(validation başarımı) diğerlerine göre oldukça düşüktür. Daha önce bahsedildiği üzere model aşırı uydurma yapmıştır.

Sonuç olarak veri zenginleştirme ve dropout işlemleri aşırı uydurma sorununu çözmeye yardımcı olacak yöntemlerdendir. Bu modelde veri zenginleştirme işlemi overfitting problemini çözmede ve validation accuracyi arttırmada dropout işleminden daha iyi sonuç vermiştir. Belki daha fazla dropout katmanı eklenerek dropoutun verdiği fayda arttırabilirdi. Daha önceki bilgilerim ve tecrübelerim ile bir yorum yapacak olursam dropout işlemi model öğreneceği veri için gereğinden çok fazla karmaşıksa daha yüksek fayda vermektedir. Model hali hazırda zaten verisetini en iyi öğrenecek karmaşıklıkta(büyüklükte) ise dropout kullanmak fayda sağlamayacaktır. Fakat çoğu zaman verisetini en iyi şekilde öğrenecek büyüklükte modeli tasarlamak mümkün olmamaktadır. Bu sebepten ötürü dropout fayda oranı

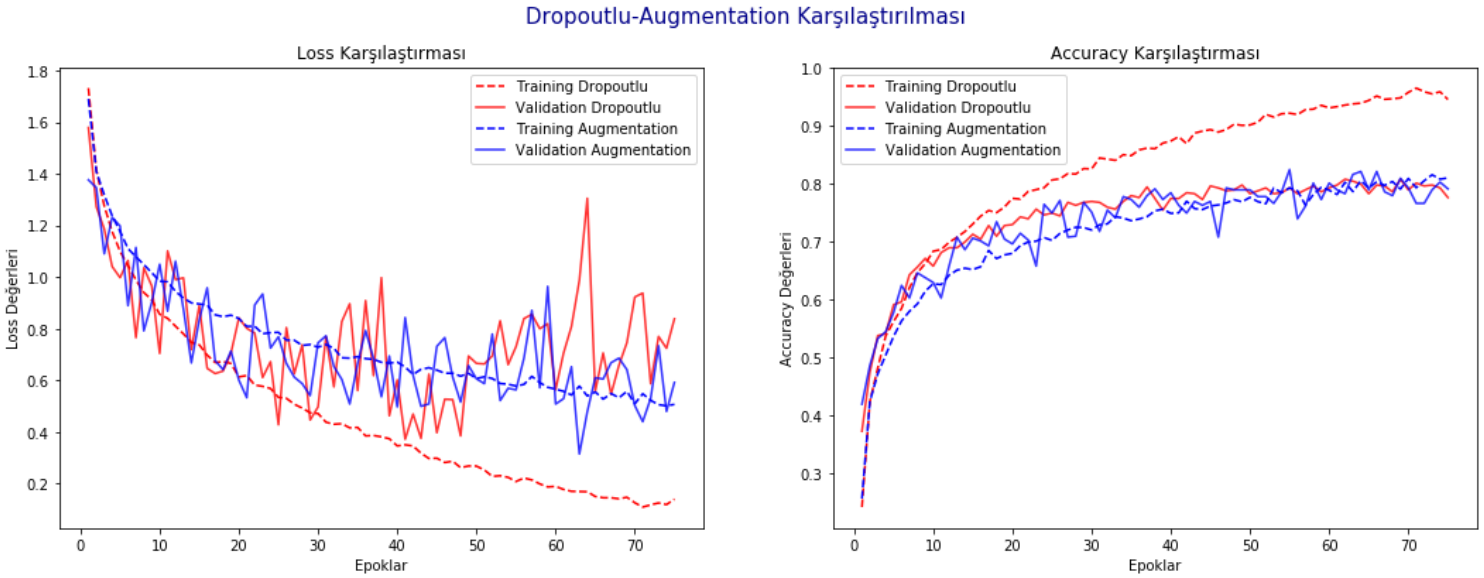
değişken olmakla birlikte aşırı uydurma problemini çözmek için iyi bir yoldur. Bunun yanında elimizdeki verisetinde bulunan örnek sayısı yeterli sayıda olmadığı için resimler üzerinde farklı işlemler yaparak yani veri zenginleştirme kullanarak ağımızı daha fazla veri ile eğitmek başarımı büyük oranda attırmıştır.



Şekil 4.7. Normal ve Dropoutlu Modellerin Karşılaştırılması



Şekil 4.8. Normal ve Veri Zenginleştirilme Kullanılan Modellerin Karşılaştırılması



Şekil 4.8. Dropout ve Veri Zenginleştirilme Kullanılan Modellerin Karşılaştırılması

Grafik Çizdirme Kodlarım

```
def grafik_ciz(history,suptitle):
    epoklar = range(1,len(history["loss"])+1)

    fig,axx = plt.subplots(1,2,figsize=(18,6))

    axx[0].plot(epoklar,history["loss"],label="Training Loss")
    axx[0].plot(epoklar,history["val_loss"],label="Validation Loss")
    axx[0].set_title("Loss Karşılaştırması")
    axx[0].set_xlabel("Epoklar")
    axx[0].set_ylabel("Loss Değerleri")
    axx[0].legend()

    axx[1].plot(epoklar,history["acc"],label="Training Accuracy")
    axx[1].plot(epoklar,history["val_acc"],label="Validation Accuracy")
    axx[1].set_title("Accuracy Karşılaştırması")
    axx[1].set_xlabel("Epoklar")
    axx[1].set_ylabel("Accuracy Değerleri")
    axx[1].legend()

    fig.suptitle(suptitle,fontsize=15,color="darkblue")

    plt.show()
```

Şekil 4.9. Tek Bir Modelin Loss ve Accuracy Değerlerini Çizdiren Metot

```python
def grafik_ciz2(historyList,aciklamalar):
    epoklar = range(1,len(historyList[0]["loss"])+1)

    fig,axx = plt.subplots(1,2,figsize=(18,6))

    colors = ["red","blue","green"]

    for i,history in enumerate(historyList):
        axx[0].plot(epoklar,history["loss"],label="Training " + aciklamalar[i],linestyle="--",color=colors[i])
        axx[0].plot(epoklar,history["val_loss"],label="Validation " + aciklamalar[i],color=colors[i],alpha=0.8)

    axx[0].set_title("Loss Karşılaştırması")
    axx[0].set_xlabel("Epoklar")
    axx[0].set_ylabel("Loss Değerleri")
    axx[0].legend()

    for i,history in enumerate(historyList):
        axx[1].plot(epoklar,history["acc"],label="Training " + aciklamalar[i],linestyle="--",color=colors[i])
        axx[1].plot(epoklar,history["val_acc"],label="Validation " + aciklamalar[i],color=colors[i],alpha=0.8)
    axx[1].set_title("Accuracy Karşılaştırması")
    axx[1].set_xlabel("Epoklar")
    axx[1].set_ylabel("Accuracy Değerleri")
    axx[1].legend()

    fig.suptitle("-".join(aciklamalar) + " Karşılaştırılması",fontsize=15,color="darkblue")

    plt.show()
```

Şekil 4.10. Birden Fazla Modelin Verilerinin Çizdirilmesi

```python
accMaxTrain = [np.max(historyNormal.history["acc"]),
               np.max(historyDroplu.history["acc"]),
               np.max(historyAug.history["acc"])]

accMaxVal = [np.max(historyNormal.history["val_acc"]),
             np.max(historyDroplu.history["val_acc"]),
             np.max(historyAug.history["val_acc"])]

x = np.arange(len(accMaxTrain))
```

```python
plt.figure(figsize=(12,8))

a = x-0.2
b = x+0.2
plt.bar(a,accMaxTrain,width=0.35)
plt.bar(b,accMaxVal,width=0.35)

plt.xticks(x,["Normal","Dropoutlu","Augmentationlu"])
plt.xlabel("Eğitim Türleri")
plt.ylabel("Accuracy(/100)")
plt.title("En Yüksek Accuracy Karşılaştırmaları")

plt.legend(["Train","Validation"])

for i in range(len(x)):
    plt.text(a[i]-0.05,accMaxTrain[i]+0.01,str("{:.3f}".format(accMaxTrain[i])))
    plt.text(b[i]-0.05,accMaxVal[i]+0.01,str("{:.3f}".format(accMaxVal[i])))

plt.show()
```

Şekil 4.11. Sutün Grafiğini Çizen Kodlar