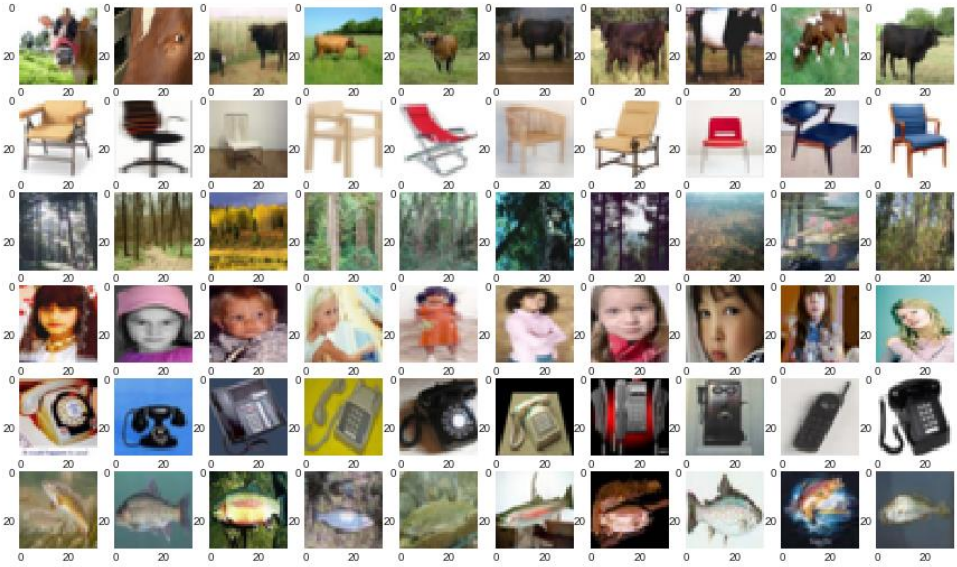
DERİN ÖĞRENME VE EVRİŞİMLİ SİNİR AĞLARI
PROJE ÖDEVİ RAPORU

BURAK BAGATARHAN
G151210101

1.ÖĞRETİM A GRUBU



Şekil 1 Sınıflar

Proje kapsamında cifar100 veri setinde bulunan 6 sınıf ile bir derin sinir ağı tasarlayıp bu ağı eğitmemiz isteniyor. Cifar verilerinden istenilen sınıfları alabilmek için cifar2png kütüphanesini kullandım. (https://github.com/knjcode/cifar2png)

```
width,height,channel = (32,32,3)
num_of_classes = 6
batch_size = 32
input_shape = (width,height,channel)
epoch = 20
train_path = "Dataset/train"
test_path = "Dataset/test"
```

Şekil 2 Kullanılan Bazı Parametreler

Projede **3 adım** bulunmaktadır.

1.Augmentation ve Dropout Kullanılmayan Model İle Eğitim

Elimizde bulunan verileri tasarladığımız bir **Convolutional Neural Network** (Evrişimsel Sinir Ağı) ile eğiteceğiz. Burada tasarladığımız modelde dropout katmanları bulunmayacak ve elimizde bulanan veri üzerinde augmentation işlemleri gerçekleştirilmeyecektir.

```
elif add_dropout==False:
    print("Dropout kullanılmayan model import edildi...")

    model.add(Conv2D(32, (3, 3), padding='same',input_shape=input_shape))
    model.add(Activation('relu'))
    model.add(Conv2D(32, (3, 3)))
    model.add(Activation('relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))

    model.add(Conv2D(64, (3, 3), padding='same'))
    model.add(Activation('relu'))
    model.add(Conv2D(64, (3, 3)))
    model.add(Activation('relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))

    model.add(Conv2D(128, (5, 5), padding='same'))
    model.add(Activation('relu'))
    model.add(Conv2D(128, (5, 5)))
    model.add(Activation('relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))

    model.add(Flatten())
    model.add(Dense(256))
    model.add(Activation('relu'))
    model.add(Dense(num_of_class))
    model.add(Activation('softmax'))

return model
```

Şekil 3 Dropout kullanılmayan model

Bunu yaparken öncelikle custon_dnn adında bir python dosyası oluşturdum. Oluşturulan bu dosyaya bir kütüphanede diyebiliriz. Burada bulunan build_model isimli metot kullanılarak gerekli parametrelerde (add_dropout,num_of_class,input_shape) verilerek modelimizi build etmiş oluruz.

Yukarıda belirtilen modelde 6 CNN 2 Dense katman bulunmaktadır. CNN katmanlarında nöron sayıları her gizli katmanda arttırılmıştır. En son çıkış dense katmanında sınıf sayısı kadar nöron bulunmaktadır. Bunun sebebi kullandığımız loss (categorical_crossentropy) ile alakalıdır. (6 çıkış her biri için oran veriyor diyebiliriz.) (sparse_categorical_crossentopy kullanmış olsaydık tek çıkış verirdik.)

Eğitim aşamasına geçildiğinde yine custom_dnn python dosyamızda bulunan başka bir metot olan data_fit metodunu çağırıyoruz. Bu metot da bulunan parametreler (data_augmentation, train_path, test_path, width,height,batch_size,epoch,model) bunlardır.

Elimizde bulunan veriyi göndermek için daha önceden cifar2png kullanarak indirdiğimiz cifar datalarında seçtiğimiz 6 sınıf yolunu (train_path ve val_path) ve augmentation yapılacak mı bunu belirtiyoruz.

```
def data_fit(data_augmentation,train_path,test_path,width,height,batch_size,epoch,model):
    optimizer = keras.optimizers.Adam(lr=0.001)

    if data_augmentation==False:
        print("Augmentation yapılmayacak....")

        datagen = ImageDataGenerator(rescale=1/255)
        X_train = datagen.flow_from_directory(train_path,class_mode="categorical",target_size=(width, height),batch_size=batch_size,shuffle=True)
        X_val = datagen.flow_from_directory(test_path,class_mode="categorical",target_size=(width, height),batch_size=batch_size,shuffle=True)

        train_step_size = X_train.n//batch_size
        test_step_size = X_val.n//batch_size


        model.compile(loss="categorical_crossentropy",optimizer=optimizer,metrics=['acc'])

        history = model.fit_generator(generator=X_train,steps_per_epoch=train_step_size, validation_data=X_val ,validation_steps=test_step_size,
                  epochs=epoch)
```

Şekil 4 Eğitimin yapılması

```
history = data_fit(False,train_path,test_path,width,height,batch_size,epoch,model)
```

Şekil 5 Eğitimin başlatılması

Şekil 4'te gösterilen kod parçasında augmentation yapılmayacak şekilde datagen ayarlanmıştır jupyter tarafında gönderilen train_path ve validation_path te bulanan veriler okunmuş ve scale edilmiştir. Model compile edilmiş ve fit_generator komutu ile eğitimi başlatılır. Eğitim sonucunda bilmek istediğimiz accuracy ve loss bilgileri history değişkenine yazılır. Bu değişken işlemler bittikten sonra return edilir.

```
acc = history.history["acc"]
val_accuracy = history.history["val_acc"]
loss = history.history["loss"]
val_loss = history.history["val_loss"]
```

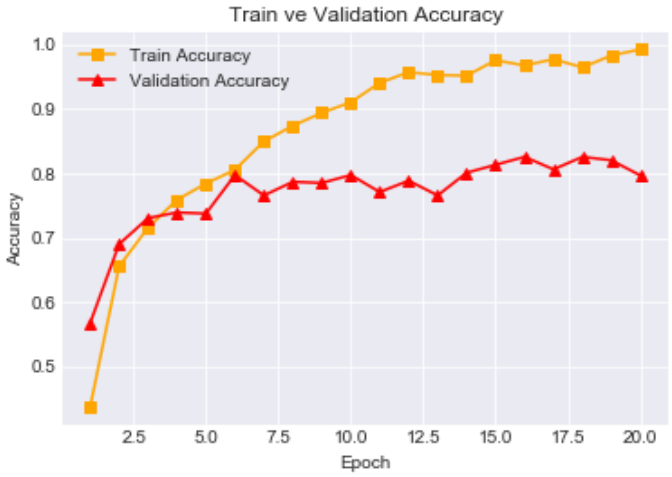
Şekil 6 Historye yazılan bilgilerin okunması

```
def plot(train_accuracy,val_accuracy,train_loss,val_loss,model_name=""):

    epochs = range(1, len(train_accuracy)+1)

    plt.plot(epochs, train_accuracy, "s-", color="orange",label = "Train Accuracy")
    plt.plot(epochs, val_accuracy, "^-",color="red" ,label = "Validation Accuracy")
    plt.title("Train ve Validation Accuracy")
    plt.xlabel("Epoch")
    plt.ylabel("Accuracy")
    plt.legend()

    plt.savefig(f'grafikler/{model_name}_accuracy.png')
    plt.figure()


    plt.plot(epochs, train_loss,"s-",color="orange", label = "Train Loss")
    plt.plot(epochs, val_loss, "^-",color="red", label = "Validation Loss")
    plt.title("Train ve Validation Loss ")
    plt.xlabel("Epoch")
    plt.ylabel("Loss")
    plt.legend()

    plt.savefig(f'grafikler/{model_name}_loss.png')
    plt.show()
```

Şekil 7 Grafik Çizdirme

History den okunan bilgiler doğrultusunda grafik çizdirmek için plot_graph adında bir python dosyası oluşturulmuştur. Bu python dosyası içerinde bulanan plot isimli metot da historyden okunan (acc,val_accuracy,loss,val_loss) bilgiler gönderildiğinde grafiklerimiz çizilecek ve save edilecektir.

1.1 Grafiklerin Yorumlanması



Şekil 8 1.Model Train ve Validation Accuracy

Yukarı da ki grafikte görüldüğü üzere validation accuracy 6. Epochtan sonra neredeyse sabit kalmıştır. Ancak train accuracy yükselmeye devam etmiştir. 6 epochtan sonra öğrenme kesilmiştir. Model ezberlemeye başlamıştır. Bu ezberleme durumuna overfit deriz. Modelin elimizde bulanan veriye ve sınıf sayısına göre oldukça karmaşık olması bu sonucu almamızda etkili olmuştur. Eğitim bir süre daha devam ettirilmesi durumda model 1.0 başarı göstermeye devam edecek ve zaman içerisinde validation accuracy düşmeye başlayabilir

.

Şekil 98 1. Model Train ve Validation Loss

Loss grafiğini incelediğimizde train datasıyla loss değerinin sürekli düştüğü gözlenmektedir. Hatta 20. Epochta loss değeri 0 olmuştur. Bu overfit olmayan bir modelde karşılaşacağımız bir sonuç değildir. Bunu yanında validation loss değeri düşme eğilimi göstersede bir noktadan sonra durağan olmayan sonuçlar vermiştir. Sonuncu epochta pik noktasına ulaşmıştır. Buda eğitimlerde beklediğimiz bir durum değildir.

2.Dropout Kullanılan Model İle Eğitim

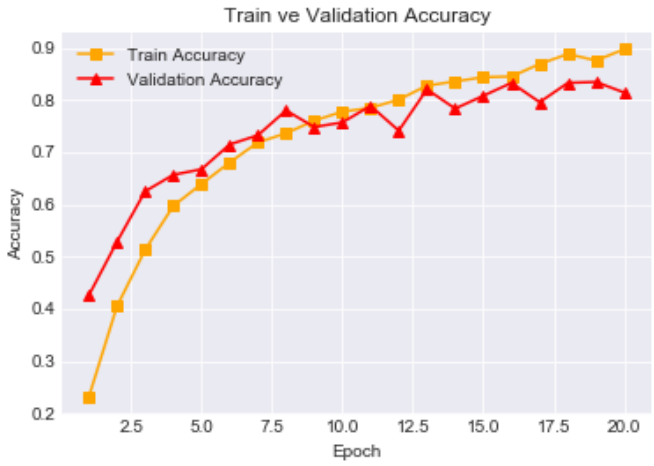
```python
def build_model(add_dropout,num_of_class,input_shape):

    model = Sequential()

    # dropout kullanılan model
    if add_dropout==True:
        print("Dropout kullanılan model import edildi...")
        model.add(Conv2D(32, (3, 3), padding='same',input_shape=input_shape))
        model.add(Activation('relu'))
        model.add(Conv2D(32, (3, 3)))
        model.add(Activation('relu'))
        model.add(MaxPooling2D(pool_size=(2, 2)))
        model.add(Dropout(0.1))

        model.add(Conv2D(64, (3, 3), padding='same'))
        model.add(Activation('relu'))
        model.add(Conv2D(64, (3, 3)))
        model.add(Activation('relu'))
        model.add(MaxPooling2D(pool_size=(2, 2)))
        model.add(Dropout(0.3))

        model.add(Conv2D(128, (5, 5), padding='same'))
        model.add(Activation('relu'))
        model.add(Conv2D(128, (5, 5)))
        model.add(Activation('relu'))
        model.add(MaxPooling2D(pool_size=(2, 2)))
        model.add(Dropout(0.5))

        model.add(Flatten())
        model.add(Dense(256))
        model.add(Activation('relu'))
        model.add(Dropout(0.6))
        model.add(Dense(num_of_class))
```

Burada bizden istenilen 1 adımda oluşturduğumuz modele dropout katmanları ekleyerek overfit problemini aşmayı denememizdir. Burada da yukarıda belirtilen kodlar kullanışmışır. Ancak burada custom_dnn içerisinde bulunan build_model metodunda bulnan add_dropout true olarak gönderilmiştir ve dropoutlu model import edilmiştir.

Şekil 10 Dropout modelin import edilmesi

Bu modelde de yukarıda belirtilen şekilde data_fit çalıştırılmıtır. Gerekli parametreler gönderilmiş ve eğitim başlatılmıştır. Yine eğitim bilgileri history nesnesinde tutulmaktadır.

2.1 Grafiklerin Yorumlanması



Şekil 11 Dropout Kullanılan Model Accuracy

Yukarıda bulunan grafikte görüleceği üzere dropout kullanılan modelde train ve validation accuracy değerleri birbirlerine yakın sonuçlar almışlardır. 1. Modelde karşılaştıımız overfit probleminde ki gibi 1 başarı değil daha makul bir sonuç olan 0.9 başarı alınmıştır. Validation da overfit olmadığımızı doğrular şekilde 0.85 dolaylarında bir başarı elde etmiştir. Buradan çıkaracağımız sonuç overfit problemi ile başa çıkmada en sık başvurabilceğimiz yollaradan birinin dropout katmanı kullanmak olduğudur.

Şekil 12 Dropout kullanılan model Loss

Dropout kullanılan modelin loss grafiğini incelediğimizde de train lossun epochlar boyunca düştüğü ve validation eğrisinin de bir düşüş gösterdiği görülmektedir. Arada sapmalar yaşanmıştır ama bu kısmen kabul edilebilecek bir sonuçtur. Eğitim yeterince başarılı görünüyor. Dropout oranları arttırarak daha iyi sonuçlar alınabilir.

3. Sadece Augmentation Kullanılan Model ile Eğitim

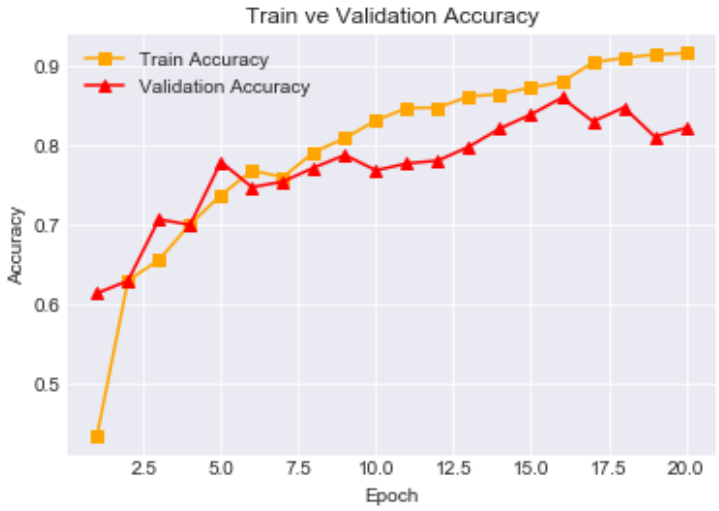
Sadece augmentation (veri zenginleştirme) kullanılan modelde dropout kullanılması istenmiyor add_dropout = False parametresi ile bunu sağlarız. data_fit metodumuzda parametre olarak data_augmentation = True şeklinde göndereceğiz.

```
elif data_augmentation==True:

    print("Augmentation yapılacak...")


    datagen = ImageDataGenerator(
        featurewise_center=True,
        samplewise_center=True,
        zca_whitening=True,
        zca_epsilon=1e-06,
        rotation_range=5,
        width_shift_range=0.1,
        height_shift_range=0.1,
        shear_range=0.1,
        zoom_range=0.1,
        fill_mode='nearest',
        horizontal_flip=True,
        vertical_flip=False,
        rescale=1/255,
      )

    val_datagen = ImageDataGenerator(rescale=1/255)

    X_train = datagen.flow_from_directory(train_path,class_mode="categorical",target_size=(width, height),batch_size=batch_size)
    X_val = datagen.flow_from_directory(test_path,class_mode="categorical",target_size=(width, height),batch_size=batch_size)

    train_step_size = X_train.n//batch_size
    test_step_size = X_val.n//batch_size

    model.compile(loss="categorical_crossentropy",optimizer=optimizer,metrics=['acc'])

    history = model.fit_generator(
        X_train,
        steps_per_epoch=train_step_size,
        epochs=epoch,
        validation_data=X_val,
        validation_steps=test_step_size,
      )

return history
```

Şekil 13 Augmentation yapılması ve fit edilme

Augmentation yapmak için ImageDataGenerator kullanarak ve gerekli parametreler verilerek bir data generator oluşturuyoruz. Augmentation işlemini sadece eğitim aşamasında kullacağımzı için bir de test için augmentation yapılmayan bir datagen oluşturuyoruz. Yine yukarılarda belirttiğimiz gibi grafikleri çizdirmek için history değişkenini döndürüyoruz

3.1 Grafiklerin Yorumlanması



Şekil 14 Augmentation kullanılan model Accuracy

Augmentation kullanılan modelde 1. modele göre daha iyi sonuçlar aldığımız söylenebilir. Train ve validation accuracy birlikte artış gösteriyorlar. Ancak 16 epochtan sonra validation accuracy artmayı kesiyor hatta azalamaya başlıyor. Belki eğitimi 16. Epochta durdurarak model overfit olmadan kesebiliriz (Early Stopping). Ancak tam bir ezberleme durumunda söz etmek çok doğru olmayacaktır. Bunun yanında belki augmentation parametreleri ile oynayıp overfit yönelimi biraz daha düşürülebilir.

Şekil 15 Augmentation kullanılan model loss

Augmentation kullanılan modelin loss grafiğini incelediğimizde train loss zaman içerisinde düştüğü görebiliriz. Validation loss değeri de zamanla azalmıştır. Ancak train loss eğrisi gibi çok stabil bir grafiği olduğu söylenemez. Bunun sebebi elimizde bulunan verinin azlığında kaynaklı olabileceğine düşünmekteyim. Yine 16 Epochtan sonra modelin durdurulması iyi bir tercih olacaktır. Bu durumun önüne geçmek için öğrenme katsayısının küçültülmesi denenebilir. Bu koşulda epoch sayısınında artırılması gerekir.

4.Genel Yorum

Tüm aşamalar grafikler incelendiğinde en başarılı modelin dropout kullanılan model olduğu söylenebilir. Ancak augmentation parametreleri ile oynayarak iyi sonuçlar alınabilir. En doğru hamle dropout kullanılan modelde augmentation veri kullanmak olacaktır. Overfit problemi ile karşılaştığımızda dropout ve augmentation etkili sonuçlar verdiği çıkarımını yapabiliriz.

Derin öğrenme problemlerinde kesin bir çözümden söz edemeyiz. Seçilen optimizasyon algoritması, katmanlar da ki nöron sayısı, katman sayısı, dropout oranı, öğrenme katsayısı, augmentation yapılırken kullanılan (yakınlaştırma, eksenlerde kaydırma, yatay-dikey döndürme) parametreler bunların her biri ile denemeler yapılır optimum sonuca ulaşılabilir. Grid Search kullanarak da denemeler yapılabilir.