**İMDB VERİLERİ KULLANILARAK COLAB’DA NÖRAL AĞIN EĞİTİLMESİ ÜZERİNE İNCELEMELER**

1. **GİRİŞ**

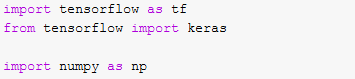
Bu çalışmada “binary” ve “text” sınıflandırma kullanılarak ham veriler ile eğitme ve test süreçleri yürütülmektedir. Film incelemeleri ve yorumları üzerinden elde edilen veriler ile ağ kümesinin eğitilmesi, eğitim sonunda başarım ve eğitime etki eden faktörler incelenmektedir.

“Binary” sınıflandırma olarak yapılacak eğitim süreçlerinde filmler hakkında yapılan yorumlar olumlu ve olumsuz değerlendirmeler sayısal 0-1 verileri kullanılırken “text” sınıflandırmalarında film hakkında yapılan yorumlar kullanılmaktadır. Ağ eğitim süreçlerinde eğitimlerin başarım oranları, eğitime etki edecek verilerin analizi ve ağ eğitim parametreleri değerlendirilerek sonuçlar karşılaştırılacaktır.

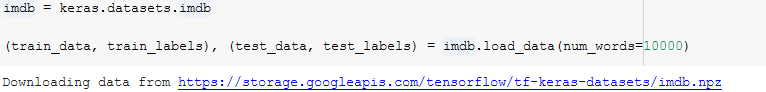
“Text” ve “Binary” sınıflandırma çalışmalarından sonra C# üzerinde hazırlanan bir program ile hazır sayısal veri seti ile ağın eğitilmesi uygulaması gerçekleştirilecektir. Ağın öğrenimine etki eden öğrenme oranı, momentum, eğitim ve test verileri üzerinden ağ üzerinde meydana gelen değişimler incelenecektir. Nöral ağdaki nöron verileri, nöral ağ bağlantılarındaki değişimler program üzerinde anlık olarak incelenerek ağın eğitilme süreci takip edilecektir.

1. **BİNARY CLASSIFICATION**

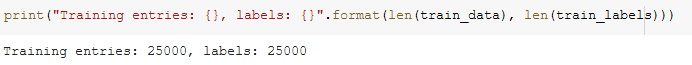
IMDB(Internet Movie Database) film yorumlarını ikili formatta değerlendirmeyi ele alan bu sınıflandırmada film yorumu veri seti kullanılarak ağın geliştirilmesi hedeflenmiştir. Olumsuz film yorumları 0, olumlu film yorumları 1 ile temsil edilerek 50.000 veri ile eğitim ortamı hazırlanmıştır. Örnek uygulama Tensorflow ile Colab üzerinde gerçekleştirilmektedir.



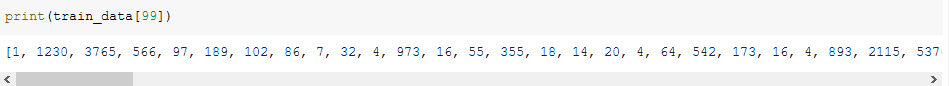
IMDB veri kümesi, içeriğinde bulunan kelime dizilerinin tamsayı dizilerine dönüştürmesiyle hazırlanmıştır. <https://ai.stanford.edu/~amaas/data/sentiment/aclImdb_v1.tar.gz> üzerinden alınan veri kümesi ile eğitim ve test süreçleri için alınmaktadır.

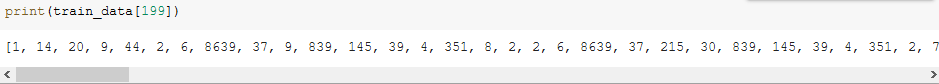


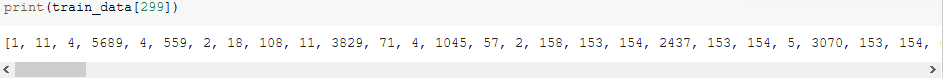
Filmlerin olumsuz değerlendirildiği etiketler 0, olumlu değerlendirildiği etiketler 1 olarak işaretlenerek 25.000 eğitim girdisi ve etiketi kullanılmaktadır.



Film incelemeleri sonucunda yapılan yorumlardaki her bir kelime, bir tamsayıyı ifade edilecek şekilde veri seti hazırlanmıştır. Yapılan film yorumlarına ait diziler train\_data[N] dizisi içerisinde bulunmaktadır. Burada N sayısı yapılan yorumun indeksini belirtmektedir. Örneğin yapılan 100, 200 ve 300. yorumlara ait sayısal dizi gösterimi şu şekilde görülmektedir:



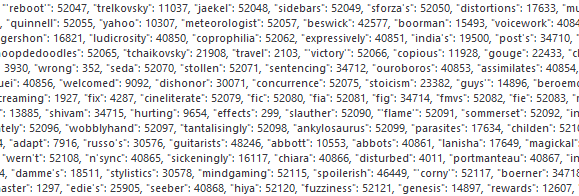




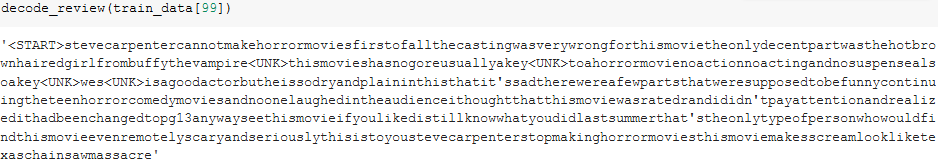
Yapılan yorumlarda kullanılan kelimelerin sayısında farklılık olabileceğinden dolayı 100, 200 ve 300. yorumlara ait dizilerin uzunlukları farklı olarak görüntülenmektedir:

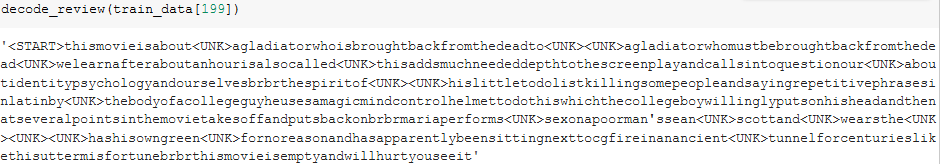


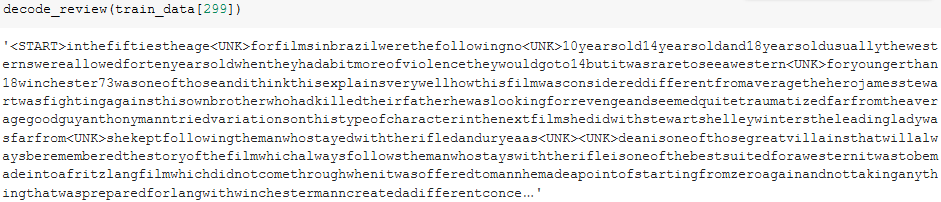
Yapılan yorumlarda kullanılan kelimeleri temsilen tamsayılar kullanılmıştır. Kelimelere karşılık gelecek tamsayı değerleri rastgele olarak atanabilmektedir. Kelimelerin karşılığı olan tamsayıları içeren dönüşüm tablosu .json dosyası içerisinde tanımlanmıştır ve aşağıda json dosyasının içeriğinden bir kesiti görülmektedir:



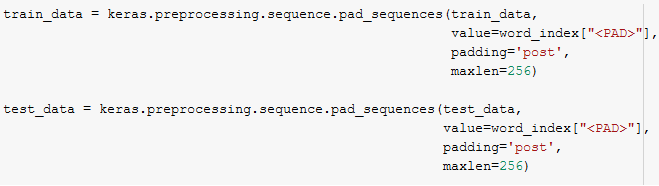
Tamsayıları kelimelere çevirmek için decode\_review[N] komutu kullanıldığında sayısal dizilerin karşılığı olan text dizisi elde edilebilmektedir. 100,200,300. sayısal dizilerine ait text dizilerinin elde edilmesi şu şekilde olmaktadır:







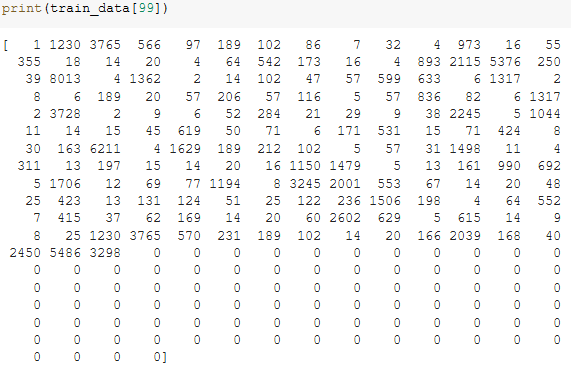
Görüldüğü gibi filmler için yapılan yorumlar farklı uzunlukta olabilmektedir. Nöral ağın eğitilebilmesi için aynı uzunluğa sahip veri kümelerinin kullanılması gerekmektedir. Farklı uzunluktaki veri kümelerinin aynı uzunluğa getirilmesi için pad\_sequences komutu kullanılmaktadır. Eğitim ve test verileri maksimum dizi boyutu 256 olacak şekilde yeniden oluşturulacakları bir önişlemeye tutulmaktadır:



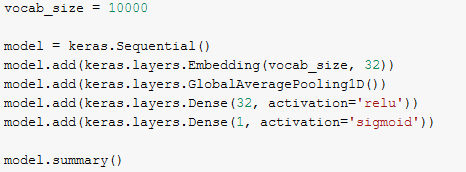
Eğitim dizisindeki 100, 200, ve 300. verilerin uzunlukları eşitlenmektedir:



Farklı uzunluğa sahip yorumlardaki kelimelerin sayısal veri tipien dönüştürülmesi sonrasında veri uzunluğunun 256’ya tamamlanabilmesi için veri dizinde boş kalan yerler 0 ile tamamlanmaktadır. 0 rakamı burada ilgili yerde kelime kullanılmadığını ifade etmektedir. Aynı boyuta getirilmiş veri dizileriyle ağın eğitilmesi mümkün olacaktır.



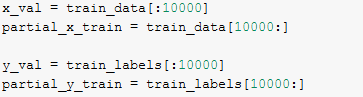
İlk katman embedding katmanıdır. Bu katman, tamsayı olarak kodlanmış kelimeleri alır ve her kelime dizini için embedding vektörünü arar. Bu vektörler model katarları olarak öğrenilir. Vektörler, çıktı dizisine bir boyut ekler. GlobalAveragePooling1D katmanı, sıra boyutunun ortalamasını alarak her örnek için sabit uzunlukta bir çıktı vektörü döndürür. Bu, modelin değişken uzunluktaki girdileri mümkün olan en basit şekilde işleyebilmesini sağlar. Bu sabit uzunluktaki çıkış vektörü, 32 gizli birim içeren tam bağlı bir katman aracılığıyla bağlantılanır. Son katman, tek bir çıkış düğümü olacak şekilde oluşturulur. Sigmoid aktivasyon fonksiyonunu kullanarak 0 ile 1 arasında çıkış üretilmektedir.



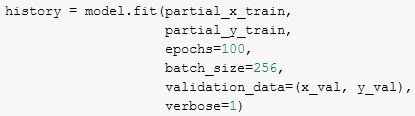
Bir model, bir kayıp işlevine ve eğitim için bir optimize ediciye ihtiyaç duyar. Bu bir ikili sınıflandırma problemi olduğundan ve bir olasılığın model çıktıları olduğundan (sigmoid aktivasyonlu tek birimlik bir katman), "binary\_crossentropy" kayıp fonksiyonu kullanılmaktadır.



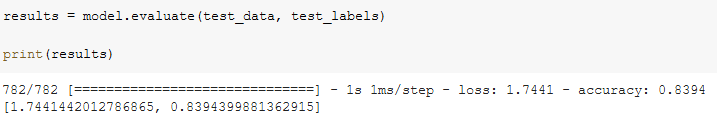
Eğitim verileri ile yapılan eğitim sonucunda eğitimin başarımını ölçmek için test verileri ile sınamanın yapılması gerekmektedir.



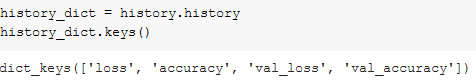
Model, verilerin 256 mini grup halinde ve 100 epoch(yineleme) ile eğitilmektedir:



Model eğitimi sonrasında elde edilen kayıp ve doğruluk değerleri elde edilmektedir:

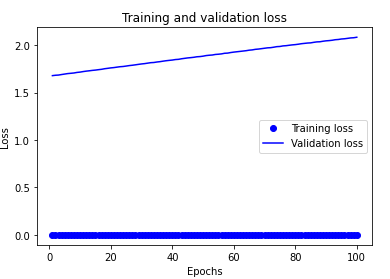


Eğitim sırasında değişen paramatrelerin değerlerinin görüntülenmesi için;



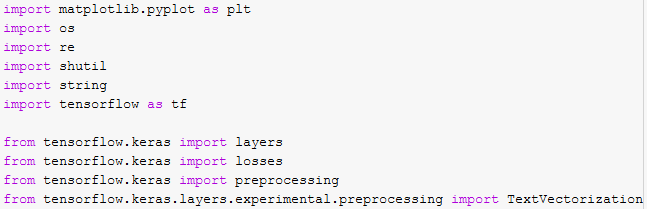
100 epoch sonunda edinilen paramatre değerlerinin değişimini gösteren grafiğin oluşturulması ile kayıp ve doğruluk değerlerindeki değişim gözlemlenmektedir:



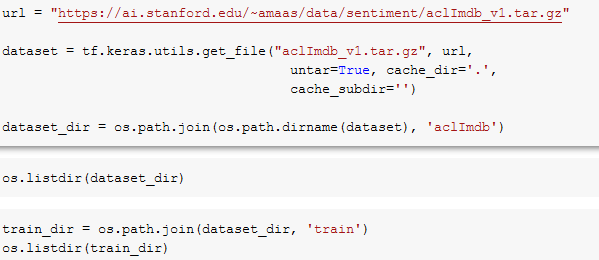


1. **TEXT CLASSIFICATON**

IMDB(Internet Movie Database) film yorumlarındaki düz metinleri kullanarak metin sınıflandırması üzerinden nöral ağın eğitilmesi amaçlanmaktadır. Modele göre olumlu ve olumsuz sınıflandırmak için 50.000 film incelemesine ait yorumlar kullanılmaktadır. Bu veriler eğitim için 25.000 ve test için 25.000 olarak ikiye ayrılmıştır. Önişlemler ve text vectorization işlemleri için tensorflow kullanılarak Colab üzerinde inceleme yapılmıştır.



IMDB verilerini içeren sıkıştırılmış .tar.gz uzantılı dosya ilgili adresten indirilerek açılmakta ve içerisinde yer alan metinler datasetlere alınmaktadır:



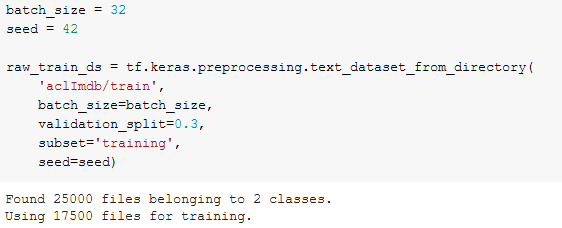
Film değerlendirmelerine ait pozitif ve negatif yorumları(pos/neg) içeren dosyalar ve içeriklerine aşağıdaki kodu kullanarak ulaşılabilmektedir:

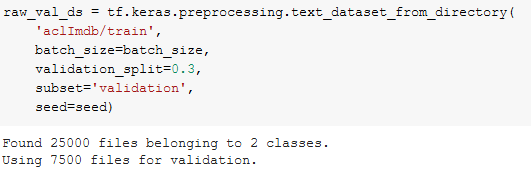


Pozitif ve negatif film değerlendirmeleriyle ilgili dosyalar işlenmeden önce sınıf-a ve sınıf-b klasörlerine ihtiyaç duyulacağından dolayı kullanılmayan klasörlerin silinmesi gerekmektedir:

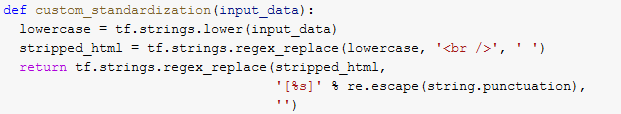


IMDB veri kümesi halihazırda eğitim ve test olmak üzere ikiye ayrılmış durumda, ancak bir doğrulama kümesi bulunmamaktadır. Aşağıdaki "validation\_split" bağımsız değişkenini kullanarak eğitim verilerinin 70:30'luk bir bölümünü kullanarak bir doğrulama kümesi oluşturulmaktadır. Bu da 25.000 dosya içinden 17.500 dosyanın eğitim ve 7500 verinin ise doğrulama için kullanılacağını ifade etmektedir.

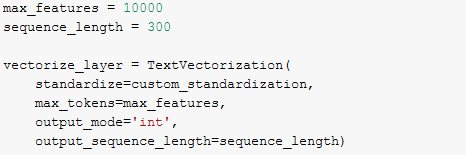




“Metin Vektorizasyon” katmanını kullanarak verileri standartlaştırılması, simgeleştirilmesi ve vektörleştirilmesi gerekmektedir. Standardizasyon, veri kümesini basitleştirmek için tipik olarak noktalama işaretlerini veya HTML öğelerini kaldırmak için metnin önceden işlenmesini ifade eder. Simgeleştirme, dizeleri belirteçlere bölmeyi ifade eder (örneğin, bir cümleyi boşluklara ayırarak tek tek kelimelere bölmek). Vektörizasyon, simgeleri sayılara dönüştürerek bir sinir ağına beslenmelerini ifade eder. Tüm bu görevler bu katmanla gerçekleştirilebilir.



Ayrıca model için model katmanın dizileri tam olarak belirli dizi uzunluk değerlerine doldurmasına veya dizilerin kesilmesini sağlayacak maksimum dizi uzunluğu gibi bazı sabitler de tanımlanmalıdır. Burada maksimum dizi uzunluğu 300 olarak tanımlanmıştır.

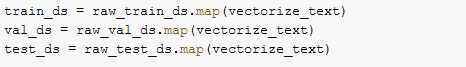


Önişleme katmanının durumunun veri kümesiyle eşitlenmesi için “adapt” kullanılmaktadır:

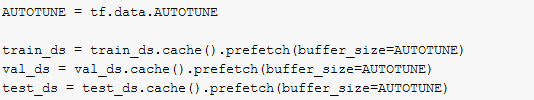


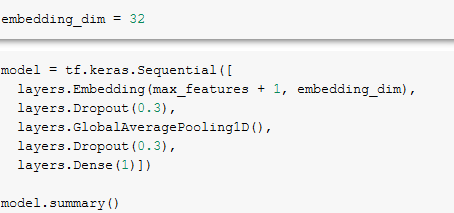


Modelin eitilmesinden hemen önce TextVectorizaton katmanı eğitim,test ve doğrulama veri kümelerine uygulanmalıdır:



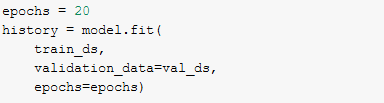
Farklı boyuttaki verilerin işlenebilmesi için önbellek oluşturma işleminin ardından model eğitimi sağlanmaktadır. Katmanlar, sınıflandırıcıyı oluşturmak için sırayla uygulanmaktadır. İlk katman bir embedding katmanıdır. Bu katman, tamsayı olarak kodlanmış verileri alır ve her kelime dizini için bir embedding vektörü arar. Bu vektörler model katarları olarak öğrenilir. Vektörler, çıktı dizisine bir boyut ekler. GlobalAveragePooling1D katmanı, sıra boyutunun ortalamasını alarak her örnek için sabit uzunlukta bir çıktı vektörü döndürür. Bu, modelin değişken uzunluktaki girdileri mümkün olan en basit şekilde işlemesini sağlar. Bu sabit uzunluktaki çıkış vektörü, 32 gizli birim içeren bir katman aracılığıyla çıkış katmanına bağlanır. Son katman, tek bir çıkış düğümünden oluşmaktadır.

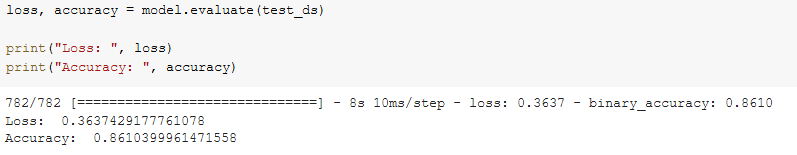




Bir modelin bir kayıp işlevine ve eğitim için bir optimize ediciye ihtiyacı vardır. Bu bir ikili sınıflandırma problemi olduğundan ve model bir olasılık çıkardığı için sigmoid aktivasyonlu tek birimlik bir katman kullanılmaktadır, daha sonra model 20 epoch ile eğitilme sürecine tutulmaktadır. Eğitimden sonra modelin performans değerlerinden olan kayıp ve doğruluk değerleri gözlemlenebilmektedir:



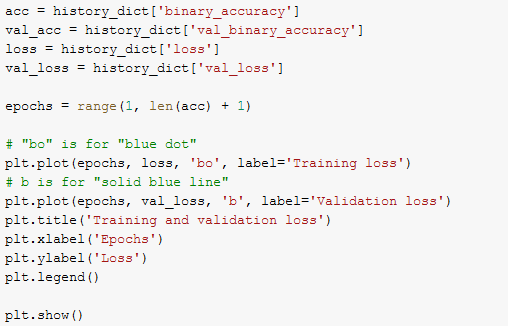


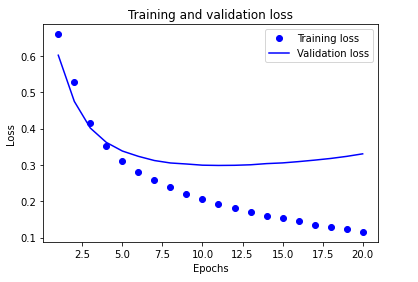


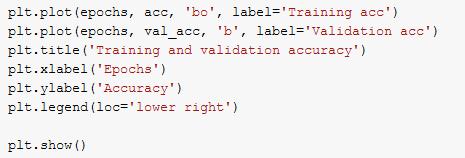
Eğitim sırasında değişen paramatrelerin değerlerinin görüntülenmesi için;



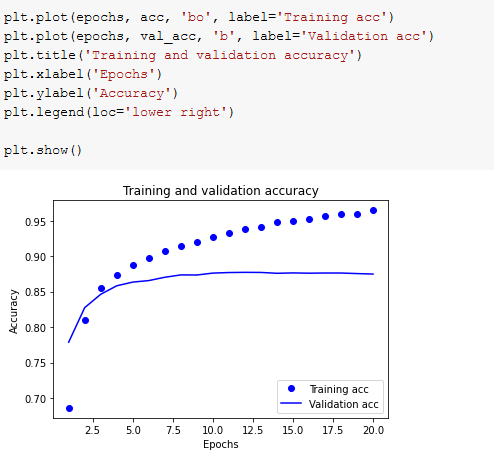
20 epoch eğitimden sonra kayıp değerlerinin 03-0.4 arasında olduğu gözlemlenmektedir. Burada ağın eğitiminde kullanılan bir parametre olan epoch miktarının az olması ağın eğitiminin kısıtlı olmasını sağlarken epochunyüksek bir değer girilmesi ağın öğrenmeden çok ezberleme özelliğini arttıracağından dolayı ağın başarısız eğitilmesine neden olmaktadır. Bundan dolayı denemeler ile uygun epoch miktarına karar verilmesi gerekmektedir.



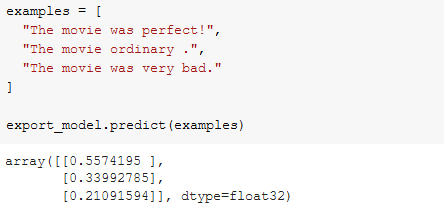




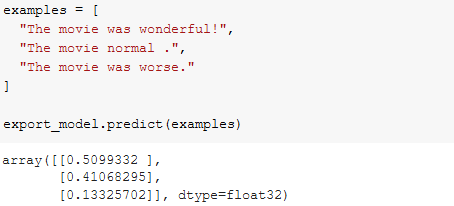
Modelin eğitilmesinden sonra eğitim doğruluğunun yaklaşık %86 olduğu gözlemlenmiştir.



Ağın kontrol edilmesi için eğitilen modele çeşitli film yorumları girilerek ağın bu yorumlara göre bir çıktı vermesi sağlanabilmektedir. Örnek olarak girilen üç yorumda film için çok iyi, orta ve kötü anlamında “perfect,ordinary ve very bad” kelimeleri girilerek modelin eğitimi sınanmıştır. Girilen text verilerine göre modelin ürettiği çıktılar %55, %33 ve %21 olarak elde edilmiştir.



2. örnekte girilen üç yorumda film için çok iyi, orta ve kötü anlamında “wonderful,normal ve worse” kelimeleri girilerek modelin eğitimi sınanmıştır. Girilen text verilerine göre modelin bu yorumlar için ürettiği çıktılar %50, %41 ve %13 olarak elde edilmiştir.



1. **SONUÇLAR**

Nöral ağın eğitilmesinde başarıyı etkileyen birkaç faktör bulunmaktadır. Örnek veri sayısının fazla olması ağın başarılı eğitilmesinde önemli bir unsurdur. Ağ eğitiminde eldeki verilerin bir kısmını ağın eğitimine dahil etmeden eğitilmiş ağın testinde kullanmak ağın eğitilme düzeyinin kontrolünde doğrulama başarısını öne çıkarmaktadır. Veriler yinelemeli olarak epoch miktarına bağlı olarak eğitime gönderilmektedir. Epoch değerinin düşük tutulması ağın eğitiminin yetersiz kalmasına neden olabileceği gibi epoch miktarının çok yüksek tutulması ağın öğrenmesinden çok verileri ezberlemesine, dolayısıyla yeni verilerle çalışırken hatalı sonuçlar üretmesine neden olabilmektedir. Bu yüzden uygun epoch değerinin bulunması için farklı değerler ile ağın eğitimi ve başarısı gözlemlenmelidir. Ağın yapısında kullanılan bağlantılar,nöron sayıları ve katman sayıları da ağın eğitilme başarısını etkileyen önemli etkenlerdir. Ancak uygun bir ağın yapısı belirten kesin bir formül bulunmamakla birlikte çok katmanlı ağların öğrenme başarısının daha az katmanlı ağlara göre daha fazla olduğu söylenebilmektedir. Çok yüksek sayıda katmana ve nörona sahip ağlardaki eğitim uzun süreli ve fazla bellek alanı gerektiren ağ sorunlarının çıkmasına da neden olabilmektedir.

1. **KAYNAK KODLAR**
   1. **Binary Classification**

import tensorflow as tf

from tensorflow import keras

import numpy as np

imdb = keras.datasets.imdb

(train\_data, train\_labels), (test\_data, test\_labels) = imdb.load\_data(num\_words=10000)

print("Training entries: {}, labels: {}".format(len(train\_data), len(train\_labels)))

print(train\_data[99])

print(train\_data[199])

print(train\_data[299])

len(train\_data[99]), len(train\_data[199]), len(train\_data[299])

word\_index = imdb.get\_word\_index()

# The first indices are reserved

word\_index = {k:(v+3) for k,v in word\_index.items()}

word\_index["<PAD>"] = 0

word\_index["<START>"] = 1

word\_index["<UNK>"] = 2  # unknown

word\_index["<UNUSED>"] = 3

reverse\_word\_index = dict([(value, key) for (key, value) in word\_index.items()])

def decode\_review(text):

    return ' '.join([reverse\_word\_index.get(i, '?') for i in text])

decode\_review(train\_data[99])

decode\_review(train\_data[199])

decode\_review(train\_data[299])

train\_data = keras.preprocessing.sequence.pad\_sequences

(train\_data, value=word\_index["<PAD>"],padding='post', maxlen=512)

test\_data = keras.preprocessing.sequence.pad\_sequences

(test\_data, value=word\_index["<PAD>"],padding='post', maxlen=512)

len(train\_data[99]), len(train\_data[199]), len(train\_data[299])

vocab\_size = 10000

model = keras.Sequential()

model.add(keras.layers.Embedding(vocab\_size, 32))

model.add(keras.layers.GlobalAveragePooling1D())

model.add(keras.layers.Dense(16, activation='relu'))

model.add(keras.layers.Dense(1, activation='sigmoid'))

model.summary()

model.compile(optimizer='adam',

              loss='binary\_crossentropy',

              metrics=['accuracy'])

x\_val = train\_data[:10000]

partial\_x\_train = train\_data[10000:]

y\_val = train\_labels[:10000]

partial\_y\_train = train\_labels[10000:]

history = model.fit(partial\_x\_train,

                    partial\_y\_train,

                    epochs=100,

                    batch\_size=256,

                    validation\_data=(x\_val, y\_val),

                    verbose=1)

sonuclar = model.evaluate(test\_data, test\_labels)

print(sonuclar)

history\_dict = history.history

history\_dict.keys()

import matplotlib.pyplot as plt

acc = history.history['accuracy']

val\_acc = history.history['val\_accuracy']

loss = history.history['loss']

val\_loss = history.history['val\_loss']

epochs = range(1, len(acc) + 1)

# "bo" is for "blue dot"

plt.plot(epochs, loss, 'bo', label='Training loss')

# b is for "solid blue line"

plt.plot(epochs, val\_loss, 'b', label='Validation loss')

plt.title('Training and validation loss')

plt.xlabel('Epochs')

plt.ylabel('Loss')

plt.legend()

plt.show()

plt.clf()   # clear figure

acc\_values = history\_dict['accuracy']

val\_acc\_values = history\_dict['val\_accuracy']

plt.plot(epochs, acc, 'bo', label='Training acc')

plt.plot(epochs, val\_acc, 'b', label='Validation acc')

plt.title('Training and validation accuracy')

plt.xlabel('Epochs')

plt.ylabel('Loss')

plt.legend()

plt.show()

* 1. Text CLassification

import matplotlib.pyplot as plt

import os

import re

import shutil

import string

import tensorflow as tf

from tensorflow.keras import layers

from tensorflow.keras import losses

from tensorflow.keras import preprocessing

from tensorflow.keras.layers.experimental.preprocessing import TextVectorization

url = "https://ai.stanford.edu/~amaas/data/sentiment/aclImdb\_v1.tar.gz"

dataset = tf.keras.utils.get\_file("aclImdb\_v1.tar.gz", url,

                                    untar=True, cache\_dir='.',

                                    cache\_subdir='')

dataset\_dir = os.path.join(os.path.dirname(dataset), 'aclImdb')

os.listdir(dataset\_dir)

train\_dir = os.path.join(dataset\_dir, 'train')

os.listdir(train\_dir)

sample\_file = os.path.join(train\_dir, 'pos/1181\_9.txt')

with open(sample\_file) as f:

  print(f.read())

remove\_dir = os.path.join(train\_dir, 'unsup')

shutil.rmtree(remove\_dir)

batch\_size = 32

seed = 42

raw\_train\_ds = tf.keras.preprocessing.text\_dataset\_from\_directory(

    'aclImdb/train',

    batch\_size=batch\_size,

    validation\_split=0.3,

    subset='training',

    seed=seed)

raw\_val\_ds = tf.keras.preprocessing.text\_dataset\_from\_directory(

    'aclImdb/train',

    batch\_size=batch\_size,

    validation\_split=0.3,

    subset='validation',

    seed=seed)

def custom\_standardization(input\_data):

  lowercase = tf.strings.lower(input\_data)

  stripped\_html = tf.strings.regex\_replace(lowercase, '<br />', ' ')

  return tf.strings.regex\_replace(stripped\_html,

                                  '[%s]' % re.escape(string.punctuation), '')

max\_features = 10000

sequence\_length = 250

vectorize\_layer = TextVectorization(

    standardize=custom\_standardization,

    max\_tokens=max\_features,

    output\_mode='int',

    output\_sequence\_length=sequence\_length)

train\_text = raw\_train\_ds.map(lambda x, y: x)

vectorize\_layer.adapt(train\_text)

def vectorize\_text(text, label):

  text = tf.expand\_dims(text, -1)

  return vectorize\_layer(text), label

train\_ds = raw\_train\_ds.map(vectorize\_text)

val\_ds = raw\_val\_ds.map(vectorize\_text)

test\_ds = raw\_test\_ds.map(vectorize\_text)

AUTOTUNE = tf.data.AUTOTUNE

train\_ds = train\_ds.cache().prefetch(buffer\_size=AUTOTUNE)

val\_ds = val\_ds.cache().prefetch(buffer\_size=AUTOTUNE)

test\_ds = test\_ds.cache().prefetch(buffer\_size=AUTOTUNE)

embedding\_dim = 32

model = tf.keras.Sequential([

  layers.Embedding(max\_features + 1, embedding\_dim),

  layers.Dropout(0.3),

  layers.GlobalAveragePooling1D(),

  layers.Dropout(0.3),

  layers.Dense(1)])

model.summary()

model.compile(loss=losses.BinaryCrossentropy(from\_logits=True),

              optimizer='adam',

              metrics=tf.metrics.BinaryAccuracy(threshold=0.0))

epochs = 20

history = model.fit(

    train\_ds,

    validation\_data=val\_ds,

    epochs=epochs)

loss, accuracy = model.evaluate(test\_ds)

print("Loss: ", loss)

print("Accuracy: ", accuracy)

history\_dict = history.history

history\_dict.keys()

acc = history\_dict['binary\_accuracy']

val\_acc = history\_dict['val\_binary\_accuracy']

loss = history\_dict['loss']

val\_loss = history\_dict['val\_loss']

epochs = range(1, len(acc) + 1)

# "bo" is for "blue dot"

plt.plot(epochs, loss, 'bo', label='Training loss')

# b is for "solid blue line"

plt.plot(epochs, val\_loss, 'b', label='Validation loss')

plt.title('Training and validation loss')

plt.xlabel('Epochs')

plt.ylabel('Loss')

plt.legend()

plt.show()

plt.plot(epochs, acc, 'bo', label='Training acc')

plt.plot(epochs, val\_acc, 'b', label='Validation acc')

plt.title('Training and validation accuracy')

plt.xlabel('Epochs')

plt.ylabel('Accuracy')

plt.legend(loc='lower right')

plt.show()

export\_model = tf.keras.Sequential([

  vectorize\_layer,

  model,

  layers.Activation('sigmoid')

])

export\_model.compile(

    loss=losses.BinaryCrossentropy(from\_logits=False), optimizer="adam", metrics=['accuracy']

)

# Test it with `raw\_test\_ds`, which yields raw strings

loss, accuracy = export\_model.evaluate(raw\_test\_ds)

print(accuracy)

ornekler = [

  "The movie was wonderful!",

  "The movie normal .",

  "The movie was worse."

]

export\_model.predict(ornekler)