GÖRÜNTÜ İŞLEME

PROJE

CEREN YAŞAR
14011020
ELİF ŞAHİN
16011615

Yöntem

Learning Rate	0.001	0.0001	0.00001
Tümör	%100	%100	%100
GHIM	%52	%43	%22

Learning rate de ki belirtilen değişikliklere bakıldığında GHIM dosyasında learning rate değeri düştükçe başarı oranı da azalmış. Ancak Tümör dosyasında başarı oranında bir düşüş olmamış.

Learning rate değeri ile ilgili öğrendiğimiz kadarıyla bu değer ile amacımız training step aşamalarında loss değerinin olabildiğince 0 değerine yakın olmasının sağlamak. 0 değerine ne kadar yakınsa bulma başarısı da o kadar artıyor. Başka bir öğrendiğimiz bilgi de learning rate aslında öğrenme hızını da belirliyor.

Veri tabanlarındaki sınıflandırma başarısı tablodan da görüldüğü gibi learning rate 0.001 değeri üzerinden bakıldığında 20 ayrı sınıfı olan 10000 resimden oluşan GHIM veri tabanında yarıya düşüyor. Hem sınıf sayısının artması hem de verilerin artışı ağımızdaki başarı da düşüşe neden oldu. Epoch değerini arttırıp 20 yaptığımızda başarı oranı %10 kadar arttı.

Tümör veri tabanında 2 ayrı sınıf vardı ve başarı burada genellikle %100 değerinde oluyor. Learning rate değeri çok büyümediği sürece de başarı oranı korunuyor.

Uygulama

Epoch:*10*
Katman Sayısı:*6(512,256,192,128,64,32)*
Tümör

Learning Rate	0.001	0.0001	0.00001
Başarı	%100	%100	%100
Zaman (2.5 Ghz & 8 GB Ram Sistem de)	11 dk 56 sn	12 dk 5 sn	12 dk 25 sn

GHIM

	Learning Rate	0.001	0.0001	0.00001
(2.5 GHz & 8 GB Ram Sistem de)	Başarı	%49	%49	
	Zaman	2 saat 5 dk 7 sn	2 saat 18 dk 22 sn	
(2.8 GHz & 16 GB Ram Sistem de)	Başarı	%52	%43	%22
	Zaman	1 saat 3 dk 9 sn	1 saat47 sn	1 saat 2 dk 26 sn

Learning rate değeri azaldıkça GHIM veri tabanı sınıflandırma başarı oranında düşüş oluyor. Ama Tümör veri tabanı sınıflandırma başarı oranın da bir düşüş meydana gelmemiş.

Tabi bu başarı oranlarında değişiklik olabilir. Çünkü programı her çalıştırdığımızda farklı bir train ve test set oluşturmuş olacak. Program veri tabanındaki verileri kümelerin oranları sabit olsa da rastgele olarak bu kümelere ayırıyor. Bu da başarı oranlarında tüm parametreler aynı olsa dahi değişikliğe neden oluyor.

Tümör
Epoch:*10*
Learning Rate: *0.001*

Epoch	1	5	10
Başarı	%30	%100	%100
Zaman (2.5 Ghz & 8 GB Ram Sistem de)	1 dk 20 sn	7 dk 2 sn	11 dk 56 sn

GHIM

Epoch	1	5	10
Başarı	%22	%38	%52
Zaman (2.8 GHz & 16 GB Ram Sistem de)	7 dk 27 sn	33 dk 30 sn	1 saat 3 dk 9 sn

Epoch değeri arttıkça öğrenme süresi uzamış. Bundan dolayı her iki küme için de genel olarak bakıldığında sınıflandırma başarısı bu değerle doğru orantılı olarak artmış.

Tümör veri tabanı daha küçük olduğu için daha küçük bir epoch değerinden sonra %100'e ulaşmış. Bu yüzden bu değerden sonra başarı da bir artış olmamış.

Tümör
Learning Rate: *0.001*
Katman Sayısı:*6(512,256,192,128,64,32)*

Katmanlar	4(512,256,192,128)	5(512,256,192,128,64)	6(512,256,192,128,64,32)
Başarı	%40	%100	%100
Zaman (2.5 Ghz & 8 GB Ram Sistem de)	14 dk 21 sn	11 dk 58 sn	11 dk 56 sn

GHIM

Katmanlar	4(512,256,192,128)	5(512,256,192,128,64)	6(512,256,192,128,64,32)
Başarı	%6	%62	%52
Zaman (2.8 GHz & 16 GB Ram Sistem de)	1 saat 3 dk 22 sn	1 saat 4 dk 30 sn	1 saat 3 dk 9 sn

Tümör veri tabanında 5 katmandan sonra başarı değeri %100 e ulaşmış. GHIM veri tabanı için ise 5 katmanda en yüksek başarı oranına ulaşmış, fakat 4. Katmanda %6 gibi düşük bir başarı oranı gözlemleniyor. Burdan çıkardığımız sonuç katman sayısı ile başarı oranı arasında doğrusal veya ters orantısal bir ilişki bulunmadığı gibi görünse de bundan tam emin olamayız çünkü program her çalıştığında başarının test edildiği data değişmekte. Bundan dolayı da etkileniyor olabilir.

Zaman performansı

Program GPU kullanmadan iki ayrı sistemde denendiğinde GHIM veri tabanı 2.6 GHz & 8 GB Ram sisteminde 2 saat sürerken 2.8 GHz &16GB Ram sisteminde 1 saat civarı sürmüş. Tümör veri tabanı ise 2.6 GHz & 8 GB Ram sisteminde yaklaşık olarak 12 dakika sürmüş.

Sonuç

1. Learning rate değeri düştüğünde Tümör veri tabanı sınıflandırma başarısında değişiklik olmazken GHIM veri tabanının sınıflandırma başarısında düşüş meydana gelmiş.
2. Epoch değeri arttıkça her iki veri tabanında da sınıflandırma başarısı artmış.
3. Katman sayısı ile başarı arasında net bir ilişki gözlemlenemedi. 4 katmanda çok düşükken 5 katmanda görülen en yüksek başarı görüldü.
4. Program tümör veri tabanı için 12 dakika, GHIM veri tabanı içinde aynı sistemde 2 saat sürmüş.
5. Genel olarak bakıldığında algoritma Tümör veri tabanı için oldukça başarılı iş çıkarmış. Ancak GHIM deki başarı oranı ve zaman performansına bakıldığında bu veri tabanını için başarılı bir performans sergileyememiş.

Program Kodu

```
import tflearn
from tflearn.layers.conv import conv_2d, max_pool_2d
from tflearn.layers.core import input_data, dropout, fully_connected
from tflearn.layers.estimator import regression
import cv2
import numpy as np
import os
from random import shuffle
from tqdm import tqdm
import tensorflow as tf

from time import time
start_time = time()


defcreateTrain(folder):
    train=[]
    number_of = len(os.listdir(folder))#number_of sinif sayisidir.
    index=0
for c_name in tqdm(os.listdir(folder)):# c_name sinif adlarini ifade eder.
        path = os.path.join(folder,c_name)
        label=np.zeros((number_of -1,), dtype=np.int)#bir sinifi tanimlayan
label array olusturuluyor.
        label=np.insert(label, index,1)# Sinifa uygun yere 0 lardan olusan
label arrayine 1 sayisi ekleniyor.
        index +=1
for image in tqdm(os.listdir(path)):
            in_path=os.path.join(path,image)
            img=cv2.imread(in_path, cv2.IMREAD_GRAYSCALE)
            img=cv2.resize(img,(boyut, boyut))
            train.append([np.array(img), label, image, c_name])
return train,number_of

#hangi dosyayi okuyalim? Buradan ayarliyoruz.
ghim='GHIM20'
tumor_teshis='tumorTeshis'
boyut=50
train,number_of=createTrain(tumor_teshis)

shuffle(train)
print(train)

test_img=train[:int(len(train)/100)]
test_image_number=int((len(train)-int(len(train)/100))/5)
trains = train[int(len(train)/100):-1*test_image_number]
tests = train[-1*test_image_number:]

train_matrix=np.array([i[0]for i in trains]).reshape(-1, boyut, boyut,1)
train_target=[i[1]for i in trains]
test_matrix=np.array([i[0]for i in tests]).reshape(-1, boyut, boyut,1)
test_target=[i[1]for i in tests]


tf.reset_default_graph()

net = input_data(shape=[None, boyut, boyut,1], name='input')

net = conv_2d(net,512,5, activation='relu')
```

```
net = max_pool_2d(net,5)

net = conv_2d(net,256,5, activation='relu')
net = max_pool_2d(net,5)

net = conv_2d(net,192,5, activation='relu')
net = max_pool_2d(net,5)

net = conv_2d(net,128,5, activation='relu')
net = max_pool_2d(net,5)

net = conv_2d(net,64,5, activation='relu')
net = max_pool_2d(net,5)

net = conv_2d(net,32,5, activation='relu')
net = max_pool_2d(net,5)

net = fully_connected(net,1024, activation='relu')

net = dropout(net,0.8)

net = fully_connected(net, number_of, activation='softmax')


net = regression(net, optimizer='adam', learning_rate=0.001,
loss='categorical_crossentropy',
                     name='targets')
#learning rate ne kadar hızlı öğreneceğimizi anlamaya yarayan bir sistem.

model = tflearn.DNN(net, tensorboard_dir='log', tensorboard_verbose=0)

model.fit({'input': train_matrix},{'targets': train_target},
n_epoch=10,validation_set=({'input': test_matrix},{'targets':
test_target}),snapshot_step=500, show_metric=True, run_id="Resim tanima")

#saver = tf.train.Saver()
#saver.save(, 'my_test_model',global_step=1000)

success=0
for i in test_img:
    mat=i[0].reshape(boyut, boyut,1)
    output=model.predict([mat])

#output labelini bulma
    k=0
while(train[k][1][np.argmax(output[0])]!=1):
        k+=1
    label=train[k][3]


print("bulunan label: "+ label)
print("gercek label: "+ i[3])
print("out:"+str(output))
print("gercek:"+str(i[1]))
print("ad:"+ i[2])
print("--------------------------")
if i[1][np.argmax(output[0])]==1:
        success+=1
print("Basari: %"+ str((success/len(test_img))*100))

end_time = time()
```

```
time_taken = end_time - start_time
hours, rest = divmod(time_taken,3600)
minutes, seconds = divmod(rest,60)
print( str(hours)+" hours "+str(minutes)+" minutes "+str(seconds)+" seconds
")
```