232137101 - Mohammad Amin ASLAMI

Final Ödevi

2024 / 06/ 29

Dersin adı: Yapay Zeka ve Uygulamaları (EST548)

Danışman: Prof.Dr.Sami EKİCİ

Problem1) input---conv---relu--sigmoid--FC1 (tam bağlantılı)---sigmoid---output yapısındaki bir CNN ağı için ileri besleme ve geri yayılım matematiksel ifadelerini elde etmenizi.

İleri Besleme (Forward Propagation):

1. Giriş: **X**

2. Konvolasiyon Katmanı:

$$Evrişim: z_1 = x * w_1 + b_1$$

$$(burada\ \boldsymbol{w_1}\ \boldsymbol{evrişim}\ çekirdeğidir\ ve\ \boldsymbol{b_1}\ \boldsymbol{bais}\ terimidir)$$

$$Aktivasyon\ (ReLU): a_1 = max * (0, b_1)$$

3. Sigmoid Katmanı:

$$z_2 = a_1$$

$$Aktivasyon\ (Sigmoid): a_2 = \sigma(z_1)\frac{1}{1+e^{-z_2}}$$

4. Tam Bağlantılı Katmanı:

$$z_3 = a_1 + w_2 + b_1$$

$$(burada\ \boldsymbol{w_2}\ \boldsymbol{ağırlık}\ matrisi\ ve\ \boldsymbol{b_1}\ \boldsymbol{bais}\ terimidir)$$

$$Aktivasyon\ (Sigmoid): a_3 = \sigma(z_3)\frac{1}{1+e^{-z_3}}$$

5. Çıkış:

$$\boldsymbol{y = a_3}$$

Geri Yayılım (Backpropagation):

Hata Hesaplaması:

$$\boldsymbol{y'nin}\ \text{öngörülen çıktı ve}\ \boldsymbol{y'nin}\ \text{gerçek çıktı olduğunu varsayalım.}$$

$$\text{Hata şu şekilde hesaplanır: } E = \frac{1}{2}\sum(y-y)^2)$$

Geri Yayılım (Backpropagation):

1. Çıkış Katmanı:

$$\text{Hata gradyanı: } \frac{\partial E}{\partial y} = y - y'$$

$$\text{Sigmoid türevi: } \frac{\partial \sigma}{\partial z_3} = \sigma(z_3) * (1 - \sigma(z_3))$$

$$\text{Ağırlık güncellemesi: } \frac{\partial E}{\partial w_3} = \frac{\partial E}{\partial y} * \frac{\partial y}{\partial z_3} * \frac{\partial z_3}{\partial w_2} = (y - y') * \sigma\,(z_3) * \big(1 - \sigma(z_3)\big) * a_2$$

$$\text{Bias güncellemesi: } \frac{\partial E}{\partial b_2} = \frac{\partial E}{\partial y} * \frac{\partial y}{\partial z_3} = (y - y') * \sigma\,(z_3) * \big(1 - \sigma(z_3)\big)$$

2. Tam Bağantılı Katmanı (FC1 Layer):

$$\text{Hata gradyanı: } \frac{\partial E}{\partial a_2} = \frac{\partial E}{\partial y} * \frac{\partial y}{\partial a_3} = (y - y') * \sigma\,(z_3) * \big(1 - \sigma(z_3)\big) * w_2$$

$$\text{Sigmoid türevi: } \frac{\partial \sigma}{\partial z_2} = \sigma(z_2) * (1 - \sigma(z_2))$$

$$\text{Ağırlık güncellemesi: } \frac{\partial E}{\partial w_1} = \frac{\partial E}{\partial a_2} * \frac{\partial a_2}{\partial a_2} * \frac{\partial z_2}{\partial w_1} = (y - y') * \sigma\,(z_3) * \big(1 - \sigma(z_3)\big) * w_2 * \sigma\,(z_2) * \big(1 - \sigma(z_2)\big) * x$$

$$\text{Bias güncellemesi: } \frac{\partial E}{\partial b_1} = \frac{\partial E}{\partial a_2} * \frac{\partial a_1}{\partial z_2} = (y - y') * \sigma\,(z_3) * \big(1 - \sigma(z_3)\big) * w_2 * \sigma\,(z_2) * \big(1 - \sigma(z_2)\big)$$

Geri yayılım sürecinin, her katmanın ağırlıkları ve önyargıları açısından kayıp fonksiyonunun hata gradyanlarını hesaplamayı ve ardından ağırlıkları ve önyargıları bu gradyanları kullanarak güncellemeyi içerdiğini unutmayın. İşlem, çıktı katmanından başlayarak ve girdi katmanına doğru geriye doğru hareket ederek her katman için tekrarlanır.

Problem2) VGG16 transfer öğrenme modeli ile herhangi bir imaj verisetini kullanarak imaj sınıflandırma uygulaması yapmanızı bekliyorum. Tüm adım ve işlemlerinize ait kanıtları (şekil, tablo vb.) tek bir belgede sisteme yükleyiniz. Veri setinizin %30'unu test işleminde kullanınız ve karışıklık tablosunu (confusion matrix) oluşturunuz. Uygulamanızı, matlab veya python ile yapabilirsiniz.

Değerli Hocam: ben 2. Sorunun çözümünü aşağıdaki link'te github hesabımda paylaştım, projeye daha net bir şekilde inceyelebilirsiniz.

https://github.com/aminaslami/Yapay-Zeka-ve-Uygulamalar-EST548-Final-Odevi/blob/main/vgg16-modeli-transfer-ogrenme-final-odevi.ipynb

Kan Hücresi Alt Tipi Sınıflandırması: Finetune **VGG16** modeli kullanılarak Transfer Öğrenme.

Kütüphanelerin yüklenmesi:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import os
import cv2
import seaborn as sns
from tqdm import tqdm
from sklearn.utils import shuffle
from sklearn import decomposition
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix
import tensorflow as tf
import keras
from keras.applications.vgg16 import VGG16
from keras.preprocessing import image
from keras.applications.vgg16 import preprocess_input
from keras.models import Sequential, Model
from keras.applications import DenseNet201
from keras.initializers import he_normal
from keras.layers import Lambda, SeparableConv2D, BatchNormalization, Dropout, MaxPooling2D,
from keras.callbacks import EarlyStopping, ReduceLROnPlateau, ModelCheckpoint
```

Verilerin yüklenmesi:

```
images, labels = load_data()
```

```
100%|██████████| 2478/2478 [00:23<00:00, 104.88it/s]
100%|██████████| 2499/2499 [00:23<00:00, 105.60it/s]
100%|██████████| 2483/2483 [00:23<00:00, 105.42it/s]
100%|██████████| 2497/2497 [00:24<00:00, 102.32it/s]
100%|██████████| 620/620 [00:04<00:00, 137.20it/s]
100%|██████████| 624/624 [00:04<00:00, 137.28it/s]
100%|██████████| 620/620 [00:04<00:00, 134.13it/s]
100%|██████████| 623/623 [00:04<00:00, 136.56it/s]
```

Test verisi için ayrılan yüzdelik (%30 Test)

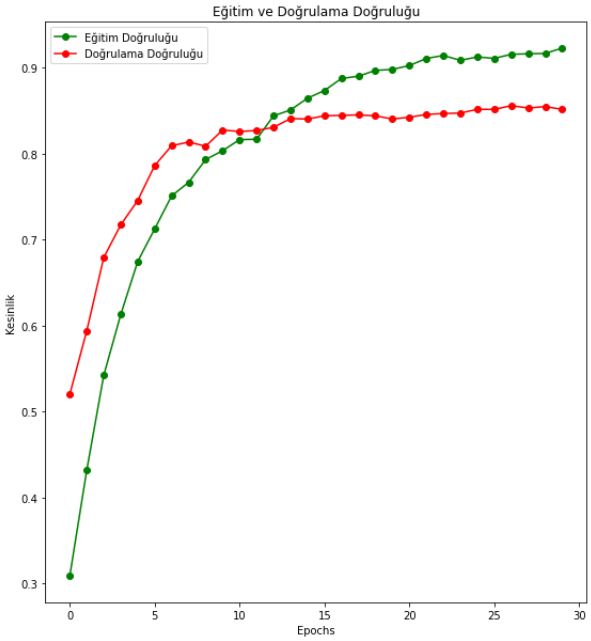
```
images, labels = shuffle(images, labels, random_state=5)

# Verileri eğitim ve doğrulama kümelerine bölünmesi (%70 eğitim, %30 test)
train_images, test_images, train_labels, test_labels = train_test_split(images, labels, test_size = 0.3)
test_images, val_images, test_labels, val_labels = train_test_split(test_images, test_labels, test_size = 0.5)
```

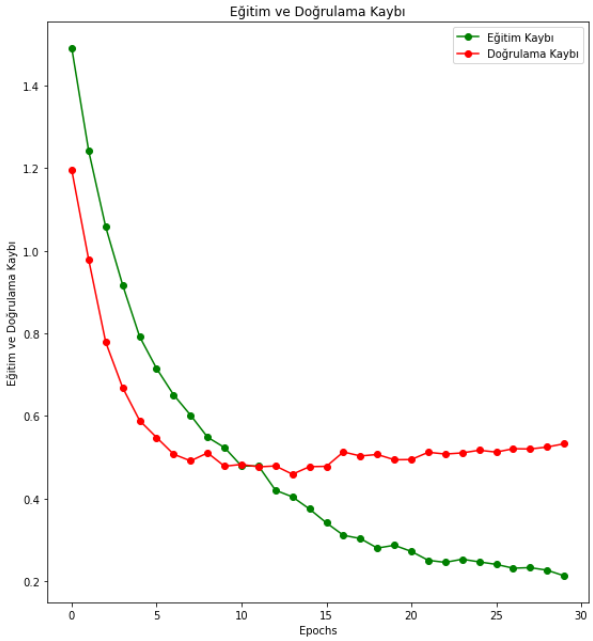
Model oluşturma ve eğitimi

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg16/vgg16_weights_tf_dim_ordering_tf_kerne
ls_notop.h5
58892288/58889256 [==============================] - 2s 0us/step
Model: "functional_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_2 (InputLayer)         [(None, 150, 150, 3)]     0
_________________________________________________________________
vgg16 (Functional)           (None, 4, 4, 512)         14714688
_________________________________________________________________
flatten (Flatten)            (None, 8192)              0
_________________________________________________________________
batch_normalization (BatchNo (None, 8192)              32768
_________________________________________________________________
dense (Dense)                (None, 256)               2097408
_________________________________________________________________
dropout (Dropout)            (None, 256)               0
_________________________________________________________________
batch_normalization_1 (Batch (None, 256)               1024
_________________________________________________________________
dense_1 (Dense)              (None, 128)               32896
_________________________________________________________________
dropout_1 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_2 (Dense)              (None, 64)                8256
_________________________________________________________________
dropout_2 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_3 (Dense)              (None, 4)                 260
=================================================================
Total params: 16,887,300
Trainable params: 2,155,716
Non-trainable params: 14,731,584
_________________________________________________________________
```

Modelin kayıp ve doğrulama doğruluğunun her birinin yakınsayıp yakınsamayacağını görmek için, dönem sayısını 10-15'ten (Bölüm 1 ve 2'de önceki iki modelde kullandığım) 30'a çıkarmaya karar verdik. Gerçekten de öyle oldu ve bu platoları aşağıdaki grafiklerde görselleştirebilirsiniz.

Model 2 için Doğruluk ve Kayıp grafikleri





Modelin Doğruluğu

```
# Test verileri üzerinde modeli değerlendirme

results = model3.evaluate(test_images, test_labels)

print("Modelin kaybı - ", results[0])
print("Modelin doğruluğu - ", results[1]*100, "%")
```

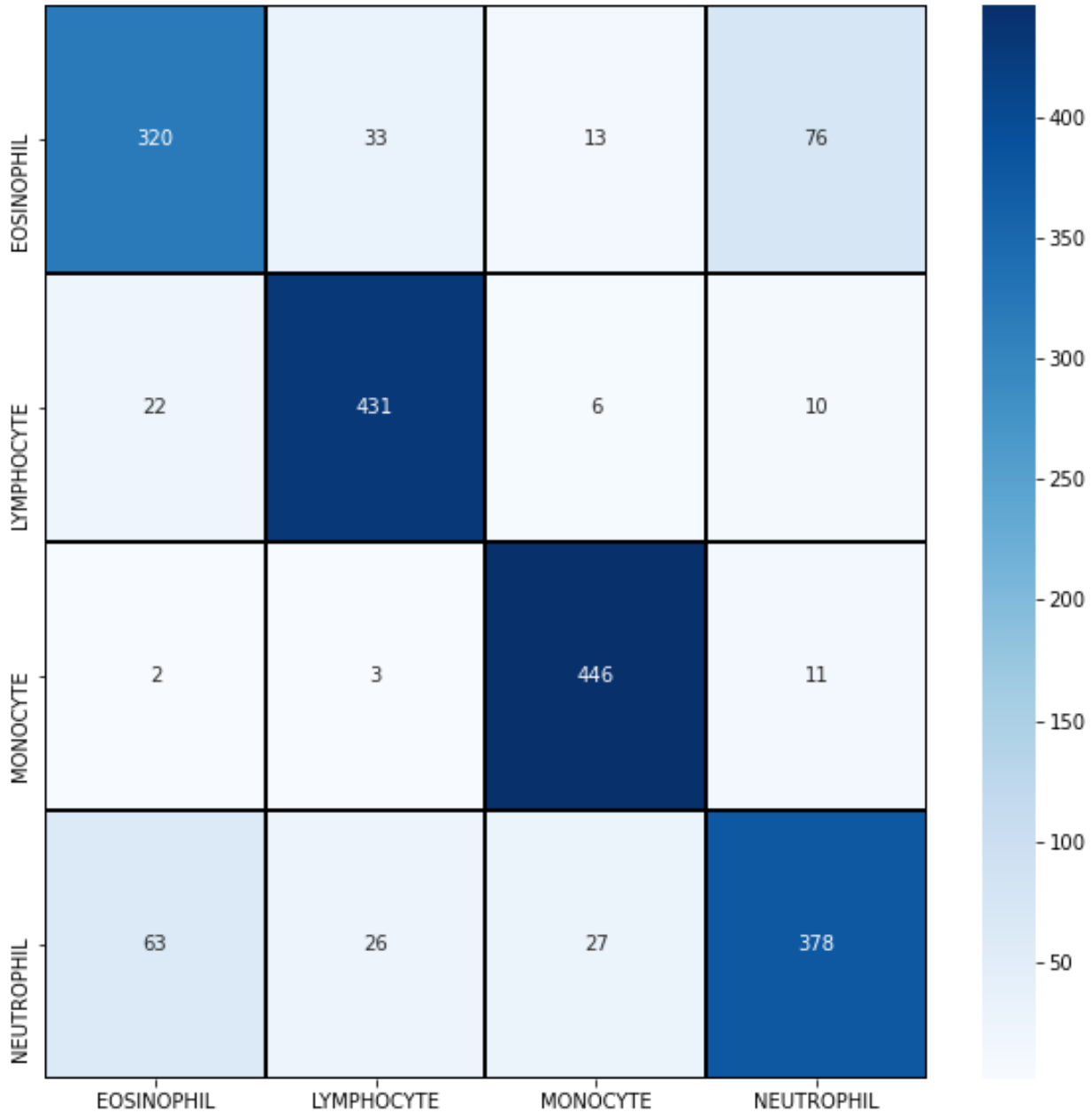
```
59/59 [==============================] - 2s 33ms/step - loss: 0.5589 - accuracy: 0.8436
Modelin kaybı -  0.558904230594635
Modelin doğruluğu -  84.35993790626526 %
```

Sınıflandırma Raporu

```
from sklearn.metrics import classification_report

print(classification_report(
    test_labels,
    predictions3,
    target_names = ['EOSINOPHIL (Class 0)', 'LYMPHOCYTE (Class 1)', 'MONOCYTE (Class 2)', 'NEUTROPHIL (Class 3)']))
```

```
                    precision    recall  f1-score   support

EOSINOPHIL (Class 0)       0.79      0.72      0.75       442
LYMPHOCYTE (Class 1)       0.87      0.92      0.90       469
  MONOCYTE (Class 2)       0.91      0.97      0.94       462
NEUTROPHIL (Class 3)       0.80      0.77      0.78       494

            accuracy                           0.84      1867
           macro avg       0.84      0.84      0.84      1867
        weighted avg       0.84      0.84      0.84      1867
```

Karmaşıklık Matrisi (Confusion Matrix)



VGG16 modeli (%84,3), sıfırdan oluşturulan CNN'den (%91,6) ve Densenet201 modelinden (%91,2) önemli ölçüde daha kötü performans gösterdi. Nötrofiller ve Eozinofilleri sınıflandırmaya çalışırken Lenfositler ve Monositler üzerinde aynı yüksek performans eğilimlerini tekrar görüyoruz.

Soruce: https://www.kaggle.com/code/kbrans/vgg16-model-83-36-acc