GÖRÜNTÜ İŞLEME DERSİ PROJE ÖDEVİ

Öğrenci İsmi ve Numarası: İlker Bedir- 16011036

Ders Sorumlusu: Doç.Dr. M. Elif Karslıgil

Teslim Tarihi: 09.01.2020

Ödev Konusu: Konvolüsyonel Sinir Ağı Tasarımı Tabanlı Sınıflandırma ve Öğrenme Aktarımı Tabanlı Görüntü Erişimi Uygulaması

1-)YÖNTEM BÖLÜMÜ:

Yöntem bölümünde önce resmi okumak i.in Python'nun CV kütüphanesinin hazır fonksiyonu kullandım. Bu fonksiyon ile resim renkli olarak bir 3 boyutlu olarak matriste rgb bilgileri saklanmaktadır. Ardından resimlerin boyutunu (224,224) çevirdim.

```
# resim kategoriler
Categories = ["airplane", "automobile", "bird", "cat", "deer", "dog", "frog",
              "horse", "ship", "truck"]

# Train ve test dosyalarından okuma ve veri çekme.


def read_image():
    try:
        x = []
        y = []
        for category in Categories:
            path = os.path.join(DATADIR, category)
            print(path)
            os.chdir(path)
            label = Categories.index(category)
            i = 0
            for im in os.listdir(path):
                img = cv2.imread(im, cv2.IMREAD_COLOR)
                img = cv2.resize(img, (224, 224))
                x.append(img)
                y.append(label)
                # if i>5000:
                #   break
                i += 1
    except Exception as e:
        print(e)
        pass
    return x, y
```

Veri eğitimi yaparken ve test ederken oluşan sonuçlarının daha kesin olması için verimiz karıştırma işlemi.

```
# Gerekli Train Ve Test Datları karıştırmak için yazılan kod
c = list(zip(x_train, y_train))
random.shuffle(c)
x_train, y_train = zip(*c)
d = list(zip(x_test, y_test))
random.shuffle(d)
x_test, y_test = zip(*d)
```

Tüm resimleri eğitebilmek adına yapılan normalizasyon işlemi.

```
# Normalizasyon yapımı
x_train = np.array(x_train)  # Numpy arraye çevirme
y_train = np.array(y_train)
x_train = x_train.astype(np.float32)    # Bellek için Float32 ye çevirme
x_train = x_train/255
x_test = np.array(x_test)
y_test = np.array(y_test)
x_test = x_test.astype(np.float32)
x_test = x_test/255
```

Oluşturduğum katmanları aynı anda test edip görebilmek için bir fonksiyon yazdım. Burada verilen parametrelere göre katmanımı değiştiriyor.

```python
# bu benim oluşturduğum Convoluttional katmanın burada testleri teker teker denemek için fonksiyonel yazdım
def convolutional_layer(x_train, x_test, y_train, y_test, layer_count, filter_number, filter_size, initializer, activation_function, dropout_number, optimization_algorith
    i = 0
    model = Sequential()
    model.add(Conv2D(filter_number, filter_size,
                     activation=activation_function, input_shape=(224, 224, 3)))
    model.add(Dropout(dropout_number))
    i += 1
    while i < layer_count:
        model.add(ZeroPadding2D(1))
        model.add(Conv2D(filter_number, filter_size,
                         activation=activation_function))
        model.add(Dropout(dropout_number))
        i += 1
    model.add(Flatten())
    model.add(Dense(10, activation='softmax'))
    model.compile(optimizer=optimization_algorithm,
                  loss='categorical_crossentropy', metrics=['accuracy'])
    y_train = np_utils.to_categorical(y_train, 10)
    y_test = np_utils.to_categorical(y_test, 10)
    model.fit(x_train, y_train, epochs=10)
    y_pred = model.predict(x_test)
    y_pred = np.argmax(y_pred, axis=1)
    y_test = np.argmax(y_test, axis=1)
    print(confusion_matrix(y_test, y_pred))
    print(classification_report(y_test, y_pred))
```

Ödevin Soru-1 için verilen denemeleri de gerçekleştirdim.

```python
# 1.Soru için İstenen Çıktılar Fonksiyonu Buradadır.
convolutional_layer(x_train, x_test, y_train, y_test, 2, 32,
                    3, "glorot_uniform", 'relu', 0.2, 'adam')

convolutional_layer(x_train, x_test, y_train, y_test, 2, 32,
                    5, "glorot_uniform", 'relu', 0.2, 'adam')

convolutional_layer(x_train, x_test, y_train, y_test, 2, 32,
                    3, "glorot_uniform", 'relu', 0.7, 'adam')

convolutional_layer(x_train, x_test, y_train, y_test, 3, 32,
                    3, "glorot_uniform", 'relu', 0.2, 'adam')

convolutional_layer(x_train, x_test, y_train, y_test, 3, 32,
                    3, "glorot_uniform", 'relu', 0.7, 'adam')

convolutional_layer(x_train, x_test, y_train, y_test, 3, 32,
                    5, "glorot_uniform", 'relu', 0.2, 'adam')
```

Soru-2 için ise veri oluşturma, veriyi karıştırma ve normalizasyon için aynı adımlar yapılmıştır. Soru-2 için Soru-1'den farklı olarak bir model oluşturup modeli eğitmek aşağıdaki fonksiyonlarda anlatılmıştır. Model oluşumları aşağıdadır:

```
# VGG16 modelinin oluşması
model=tensorflow.keras.applications.VGG16(
    include_top=True,
    weights="imagenet",
    input_tensor=None,
    input_shape=(224, 224, 3),
    pooling=None,
    classes=1000,
    classifier_activation="softmax",
)
```

```
#Resnet50-Modelini oluşturulması
model_resnet=tensorflow.keras.applications.ResNet50(
    include_top=True,
    weights="imagenet",
    input_tensor=None,
    input_shape=None,
    pooling=None,
    classes=1000,
)
```

Model oluştuktan sonra ilk başta bizden istenen fully connected katmanını değiştirmek.

```
# Son katmanı yerine kendi Fully Connected ağımızın eklenmesi ve sınıflandırma
my_fc=Dense(1024, activation='relu', name='my_fc')(model.layers[-3].output)
pred=Dense(10, activation='softmax', name='myprediction')(my_fc)
my_model=Model(model.input, pred)
my_model.summary()
```

Ardından sadece son katmanı eğitilebilir yapma aşaması.

```
# Son katmanı Eğitilebilir yapma
for i in range(0, 21):
    my_model.layers[i].trainable=False
```

Test verileri kategorik veri formatında ayarlama.

```
#Test verilerini kategoriye çevirme
y_train=np_utils.to_categorical(y_train, 10)
y_test=np_utils.to_categorical(y_test, 10)
```

Son olarak verinin eğitimi ve test aşaması.

```
#Model eğitimi ve tamamlanması,test edilmesi
my_model.compile(optimizer='adam',
                 loss='categorical_crossentropy', metrics=['accuracy'])
my_model.fit(x_train, y_train, batch_size=64, epochs=10, verbose=1)
y_pred=my_model.predict(x_test)
y_pred=np.argmax(y_pred, axis=1)
y_test=np.argmax(y_test, axis=1)
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

Ayrıca bizden istenen 2. adım ise son 4 katmanı eğitilebilir yapıp tekrar denemek.Aşağıda yapılmıştır.

```
#Son 4KATMANI EĞİTİLEBİLİR YAPMA
for i in range(0, 18):
    my_model1.layers[i].trainable=False
my_model1.layers[18].trainable=True
my_model1.layers[19].trainable=True
my_model1.layers[20].trainable=True
```

2)UYGULAMA

SORU 1 İÇİN İSTENİLEN TESTLER:

1-)

```
convolutional_layer(x_train,x_test,y_train,y_test,2,32,3,"glorot_uniform",'relu',0.2,'adam')
```

```
Epoch 1/10
2813/2813 [==============================] - 20s 5ms/step - loss: 1.7373 - accuracy: 0.3696
Epoch 2/10
2813/2813 [==============================] - 13s 5ms/step - loss: 1.3545 - accuracy: 0.5177
Epoch 3/10
2813/2813 [==============================] - 14s 5ms/step - loss: 1.2188 - accuracy: 0.5688
Epoch 4/10
2813/2813 [==============================] - 14s 5ms/step - loss: 1.1358 - accuracy: 0.5985
Epoch 5/10
2813/2813 [==============================] - 14s 5ms/step - loss: 1.0671 - accuracy: 0.6214
Epoch 6/10
2813/2813 [==============================] - 14s 5ms/step - loss: 1.0019 - accuracy: 0.6461
Epoch 7/10
2813/2813 [==============================] - 14s 5ms/step - loss: 0.9500 - accuracy: 0.6633
Epoch 8/10
2813/2813 [==============================] - 14s 5ms/step - loss: 0.9072 - accuracy: 0.6763
Epoch 9/10
2813/2813 [==============================] - 14s 5ms/step - loss: 0.8713 - accuracy: 0.6914
Epoch 10/10
2813/2813 [==============================] - 14s 5ms/step - loss: 0.8443 - accuracy: 0.6993
```

KARIŞIKLIK MATRİSİ

```
[[5541  272  640  119  235   84   76  239 1352  442]
 [ 368 4971  188  147  154   83  105  196  687 2101]
 [ 558   88 3938  628 1299  497  925  417  573   77]
 [  95  111  976 2993 1443 1321 1219  503  218  121]
 [ 220   75  815  713 4537  530  505 1223  278  104]
 [ 103  104  994 1673 1592 2570  539 1062  234  129]
 [  54   58  870  731  636  237 6142  122  110   40]
 [ 156   77  336  463 1375  505  104 5630  157  197]
 [1053  429  448  220  296  153  154  278 5363  606]
 [ 364 1698  139  191  214  127   65  340  680 5182]]
```

	precision	recall	f1-score	support
0	0.65	0.62	0.63	9000
1	0.63	0.55	0.59	9000
2	0.42	0.44	0.43	9000
3	0.38	0.33	0.35	9000
4	0.39	0.50	0.44	9000
5	0.42	0.29	0.34	9000
6	0.62	0.68	0.65	9000
7	0.56	0.63	0.59	9000
8	0.56	0.60	0.58	9000
9	0.58	0.58	0.58	9000
accuracy			0.52	90000
macro avg	0.52	0.52	0.52	90000
weighted avg	0.52	0.52	0.52	90000

2-)

```
convolutional_layer(x_train,x_test,y_train,y_test,2,32,5,"glorot_uniform",'relu',0.2,'adam')
```

```
Epoch 1/10
2813/2813 [==============================] - 15s 5ms/step - loss: 1.7816 - accuracy: 0.3463
Epoch 2/10
2813/2813 [==============================] - 15s 5ms/step - loss: 1.4588 - accuracy: 0.4757
Epoch 3/10
2813/2813 [==============================] - 15s 5ms/step - loss: 1.3450 - accuracy: 0.5173
Epoch 4/10
2813/2813 [==============================] - 15s 5ms/step - loss: 1.2691 - accuracy: 0.5474
Epoch 5/10
2813/2813 [==============================] - 15s 5ms/step - loss: 1.2080 - accuracy: 0.5694
Epoch 6/10
2813/2813 [==============================] - 15s 5ms/step - loss: 1.1565 - accuracy: 0.5879
Epoch 7/10
2813/2813 [==============================] - 15s 5ms/step - loss: 1.1059 - accuracy: 0.6086
Epoch 8/10
2813/2813 [==============================] - 15s 5ms/step - loss: 1.0788 - accuracy: 0.6176
Epoch 9/10
2813/2813 [==============================] - 15s 5ms/step - loss: 1.0419 - accuracy: 0.6277
Epoch 10/10
2813/2813 [==============================] - 15s 5ms/step - loss: 1.0166 - accuracy: 0.6376
```

KARIŞIKLIK MATRİSİ

```
[[5585  331  616  139  161   85   94  154 1350  485]
 [ 367 5427  140  164  126   86  146  135  580 1829]
 [ 713  138 3814  933  932  468 1077  268  545  112]
 [ 140  171  947 3344 1098 1064 1365  367  273  231]
 [ 277  107 1045 1065 3777  482  816  954  330  147]
 [ 187  179 1060 1917 1418 2331  641  755  266  246]
 [  80   86  825  923  452  220 6132   68  151   63]
 [ 238  126  427  604 1459  582  155 4912  210  287]
 [1199  514  478  244  228  125  212  168 5200  632]
 [ 416 2224  140  190  170  108   94  229  665 4764]]
              precision    recall  f1-score   support

           0       0.61      0.62      0.61      9000
           1       0.58      0.60      0.59      9000
           2       0.40      0.42      0.41      9000
           3       0.35      0.37      0.36      9000
           4       0.38      0.42      0.40      9000
           5       0.42      0.26      0.32      9000
           6       0.57      0.68      0.62      9000
           7       0.61      0.55      0.58      9000
           8       0.54      0.58      0.56      9000
           9       0.54      0.53      0.54      9000

    accuracy                           0.50     90000
   macro avg       0.50      0.50      0.50     90000
weighted avg       0.50      0.50      0.50     90000
```

3-)

```
convolutional_layer(x_train,x_test,y_train,y_test,2,32,3,"glorot_uniform",'relu',0.7,'adam')
```

```
_________________________________________________________________
Epoch 1/10
2813/2813 [==============================] - 14s 5ms/step - loss: 1.8454 - accuracy: 0.3237
Epoch 2/10
2813/2813 [==============================] - 13s 5ms/step - loss: 1.5614 - accuracy: 0.4304
Epoch 3/10
2813/2813 [==============================] - 14s 5ms/step - loss: 1.5169 - accuracy: 0.4476
Epoch 4/10
2813/2813 [==============================] - 13s 5ms/step - loss: 1.4885 - accuracy: 0.4586
Epoch 5/10
2813/2813 [==============================] - 13s 5ms/step - loss: 1.4666 - accuracy: 0.4681
Epoch 6/10
2813/2813 [==============================] - 13s 5ms/step - loss: 1.4505 - accuracy: 0.4767
Epoch 7/10
2813/2813 [==============================] - 14s 5ms/step - loss: 1.4374 - accuracy: 0.4798
Epoch 8/10
2813/2813 [==============================] - 14s 5ms/step - loss: 1.4308 - accuracy: 0.4815
Epoch 9/10
2813/2813 [==============================] - 13s 5ms/step - loss: 1.4196 - accuracy: 0.4854
Epoch 10/10
2813/2813 [==============================] - 13s 5ms/step - loss: 1.4196 - accuracy: 0.4860
```

KARIŞIKLIK MATRİSİ

```
[[5210  306  443   81  313  121  316  304 1406  500]
 [ 301 5107   76  151  212   78  487  280  611 1697]
 [ 624   95 2215  374 1497  503 2879  268  507   38]
 [  89   82  305 1677 1066  938 4335  287  151   70]
 [ 174   47  366  377 3961  375 2538  896  207   59]
 [ 126  109  494 1077 1531 2106 2542  740  177   98]
 [  55   29  201  145  422  146 7841   67   86    8]
 [ 133   61  180  355 1714  501  696 5075  131  154]
 [1115  507  308  175  450  159  680  222 4827  557]
 [ 293 1823   72  223  354  105  431  418  588 4693]]
```

	precision	recall	f1-score	support
0	0.64	0.58	0.61	9000
1	0.63	0.57	0.60	9000
2	0.48	0.25	0.32	9000
3	0.36	0.19	0.25	9000
4	0.34	0.44	0.39	9000
5	0.42	0.23	0.30	9000
6	0.34	0.87	0.49	9000
7	0.59	0.56	0.58	9000
8	0.56	0.54	0.55	9000
9	0.60	0.52	0.56	9000
accuracy			0.47	90000
macro avg	0.50	0.47	0.46	90000
weighted avg	0.50	0.47	0.46	90000

4-)

```
convolutional_layer(x_train,x_test,y_train,y_test,3,32,3,"glorot_uniform",'relu',0.2,'adam')
```

```
Epoch 1/10
2813/2813 [==============================] - 18s 6ms/step - loss: 1.7459 - accuracy: 0.3625
Epoch 2/10
2813/2813 [==============================] - 17s 6ms/step - loss: 1.3288 - accuracy: 0.5241
Epoch 3/10
2813/2813 [==============================] - 17s 6ms/step - loss: 1.2018 - accuracy: 0.5727
Epoch 4/10
2813/2813 [==============================] - 17s 6ms/step - loss: 1.1301 - accuracy: 0.6006
Epoch 5/10
2813/2813 [==============================] - 17s 6ms/step - loss: 1.0652 - accuracy: 0.6177
Epoch 6/10
2813/2813 [==============================] - 17s 6ms/step - loss: 1.0149 - accuracy: 0.6419
Epoch 7/10
2813/2813 [==============================] - 18s 6ms/step - loss: 0.9825 - accuracy: 0.6517
Epoch 8/10
2813/2813 [==============================] - 17s 6ms/step - loss: 0.9480 - accuracy: 0.6620
Epoch 9/10
2813/2813 [==============================] - 17s 6ms/step - loss: 0.9219 - accuracy: 0.6719
Epoch 10/10
2813/2813 [==============================] - 17s 6ms/step - loss: 0.8934 - accuracy: 0.6790
```

KARIŞIKLIK MATRİSİ

```
[[5520  299  644   98  291  127  114  251 1240  416]
 [ 269 5570   92  107  134  130  111  176  549 1862]
 [ 448   75 3911  588 1453  624  947  366  510   78]
 [  88  132  654 2904 1588 1686 1136  474  211  127]
 [ 144   63  595  553 4971  673  435 1227  236  103]
 [ 103  135  690 1290 1773 3286  441  933  208  141]
 [  44   50  690  717  592  372 6329   79  107   20]
 [ 131  102  217  308 1335  641   65 5875  139  187]
 [ 702  434  393  170  371  161  181  239 5872  477]
 [ 312 1839  103  109  199  114   52  341  559 5372]]
```

	precision	recall	f1-score	support
0	0.71	0.61	0.66	9000
1	0.64	0.62	0.63	9000
2	0.49	0.43	0.46	9000
3	0.42	0.32	0.37	9000
4	0.39	0.55	0.46	9000
5	0.42	0.37	0.39	9000
6	0.65	0.70	0.67	9000
7	0.59	0.65	0.62	9000
8	0.61	0.65	0.63	9000
9	0.61	0.60	0.60	9000
accuracy			0.55	90000
macro avg	0.55	0.55	0.55	90000
weighted avg	0.55	0.55	0.55	90000

5-)

```
convolutional_layer(x_train,x_test,y_train,y_test,3,32,3,"glorot_uniform",'relu',0.7,'adam')
```

```
Epoch 1/10
2813/2813 [==============================] - 18s 6ms/step - loss: 1.9938 - accuracy: 0.2559
Epoch 2/10
2813/2813 [==============================] - 17s 6ms/step - loss: 1.6441 - accuracy: 0.3922
Epoch 3/10
2813/2813 [==============================] - 17s 6ms/step - loss: 1.5797 - accuracy: 0.4228
Epoch 4/10
2813/2813 [==============================] - 17s 6ms/step - loss: 1.5489 - accuracy: 0.4302
Epoch 5/10
2813/2813 [==============================] - 17s 6ms/step - loss: 1.5191 - accuracy: 0.4455
Epoch 6/10
2813/2813 [==============================] - 17s 6ms/step - loss: 1.5131 - accuracy: 0.4490
Epoch 7/10
2813/2813 [==============================] - 17s 6ms/step - loss: 1.5011 - accuracy: 0.4500
Epoch 8/10
2813/2813 [==============================] - 17s 6ms/step - loss: 1.4952 - accuracy: 0.4536
Epoch 9/10
2813/2813 [==============================] - 17s 6ms/step - loss: 1.4937 - accuracy: 0.4572
Epoch 10/10
2813/2813 [==============================] - 17s 6ms/step - loss: 1.4803 - accuracy: 0.4584
[[4751  346  932  189  300  101  779  242 1048  312]
 [ 200 4751  177  337  151   99 1763  220  354  948]
 [ 446   26 2133  970  689  225 4190  100  213    8]
 [  32   26  376 2753  349  408 4895   89   48   24]
 [  87   17  585 1026 2230  211 4333  420   71   20]
 [  61   45  583 2691  672 1175 3406  283   64   20]
 [  32   10  142  335  106   35 8300   12   19    9]
 [  67   32  279 1193 1511  465 1999 3362   49   43]
 [1078  462  729  432  309   97 1762  114 3740  277]
 [ 258 1979  175  503  212   92 1772  401  422 3186]]
```

	precision	recall	f1-score	support
0	0.68	0.53	0.59	9000
1	0.62	0.53	0.57	9000
2	0.35	0.24	0.28	9000
3	0.26	0.31	0.28	9000
4	0.34	0.25	0.29	9000
5	0.40	0.13	0.20	9000
6	0.25	0.92	0.39	9000
7	0.64	0.37	0.47	9000
8	0.62	0.42	0.50	9000
9	0.66	0.35	0.46	9000
accuracy			0.40	90000
macro avg	0.48	0.40	0.40	90000
weighted avg	0.48	0.40	0.40	90000

6-)

```
[13] convolutional_layer(x_train,x_test,y_train,y_test,3,32,5,"glorot_uniform",'relu',0.2,'adam')
```

```
_________________________________________________________________
Epoch 1/10
2813/2813 [==============================] - 20s 7ms/step - loss: 1.8318 - accuracy: 0.3170
Epoch 2/10
2813/2813 [==============================] - 19s 7ms/step - loss: 1.4832 - accuracy: 0.4598
Epoch 3/10
2813/2813 [==============================] - 19s 7ms/step - loss: 1.3781 - accuracy: 0.5043
Epoch 4/10
2813/2813 [==============================] - 19s 7ms/step - loss: 1.3190 - accuracy: 0.5248
Epoch 5/10
2813/2813 [==============================] - 19s 7ms/step - loss: 1.2739 - accuracy: 0.5404
Epoch 6/10
2813/2813 [==============================] - 19s 7ms/step - loss: 1.2306 - accuracy: 0.5584
Epoch 7/10
2813/2813 [==============================] - 19s 7ms/step - loss: 1.1995 - accuracy: 0.5701
Epoch 8/10
2813/2813 [==============================] - 19s 7ms/step - loss: 1.1746 - accuracy: 0.5801
Epoch 9/10
2813/2813 [==============================] - 19s 7ms/step - loss: 1.1595 - accuracy: 0.5869
Epoch 10/10
2813/2813 [==============================] - 19s 7ms/step - loss: 1.1420 - accuracy: 0.5904
[[5344  288  682  143  113  153  167  204 1556  350]
 [ 198 5450  188  218   74  209  247  149  615 1652]
 [ 482   55 4128  862  561  744 1301  272  540   55]
 [  53   81  938 3539  565 1662 1579  316  178   89]
 [ 163   47 1328 1014 3041  955  887 1192  287   86]
 [  72   89  955 1976  761 3304  720  831  192  100]
 [  41   25  757  890  177  347 6600   55   88   20]
 [ 120   60  458  660  864  991  174 5328  181  164]
 [ 889  419  607  315  154  250  323  185 5461  397]
 [ 306 1907  153  307   96  249  183  276  714 4809]]
```

	precision	recall	f1-score	support
0	0.70	0.59	0.64	9000
1	0.65	0.61	0.63	9000
2	0.40	0.46	0.43	9000
3	0.36	0.39	0.37	9000
4	0.47	0.34	0.39	9000
5	0.37	0.37	0.37	9000
6	0.54	0.73	0.62	9000
7	0.60	0.59	0.60	9000
8	0.56	0.61	0.58	9000
9	0.62	0.53	0.58	9000
accuracy			0.52	90000
macro avg	0.53	0.52	0.52	90000
weighted avg	0.53	0.52	0.52	90000

SORU 2 İÇİN İSTENİLEN TESTLER:

1-) VGG16 SADECE SON KATMAN EĞİTİLEBİLİR:

```
Epoch 1/10
79/79 [==============================] - 18s 134ms/step - loss: 11.4945 - accuracy: 0.1181
Epoch 2/10
79/79 [==============================] - 10s 126ms/step - loss: 2.0973 - accuracy: 0.2187
Epoch 3/10
79/79 [==============================] - 10s 126ms/step - loss: 1.9600 - accuracy: 0.2744
Epoch 4/10
79/79 [==============================] - 10s 126ms/step - loss: 1.9205 - accuracy: 0.2635
Epoch 5/10
79/79 [==============================] - 10s 126ms/step - loss: 1.8777 - accuracy: 0.2947
Epoch 6/10
79/79 [==============================] - 10s 126ms/step - loss: 1.8918 - accuracy: 0.3009
Epoch 7/10
79/79 [==============================] - 10s 127ms/step - loss: 1.8264 - accuracy: 0.3189
Epoch 8/10
79/79 [==============================] - 10s 126ms/step - loss: 1.7981 - accuracy: 0.3127
Epoch 9/10
79/79 [==============================] - 10s 125ms/step - loss: 1.6959 - accuracy: 0.3677
Epoch 10/10
79/79 [==============================] - 10s 125ms/step - loss: 1.7308 - accuracy: 0.3574
[[333   3   3  10   1   0   0  10 120  22]
 [ 48  82   0  22   0   1   0  20 136 193]
 [101   3  50 178  63   1   8  30  67   1]
 [ 25   3   3 312  40   4   5  23  67  20]
 [ 25   0   8 148 118   1   2  92  90  18]
 [ 19   2  10 252  53   6   0  80  58  22]
 [ 32   4  33 294  36   1  50  12  34   6]
 [ 21   2   3  78  31   2   0 251  57  57]
 [137   2   1  22   5   0   0  14 277  44]
 [ 21  10   0  18   2   0   0  32 130 289]]
```

	precision	recall	f1-score	support
0	0.44	0.66	0.53	502
1	0.74	0.16	0.27	502
2	0.45	0.10	0.16	502
3	0.23	0.62	0.34	502
4	0.34	0.24	0.28	502
5	0.38	0.01	0.02	502
6	0.77	0.10	0.18	502
7	0.45	0.50	0.47	502
8	0.27	0.55	0.36	502
9	0.43	0.58	0.49	502
accuracy			0.35	5020
macro avg	0.45	0.35	0.31	5020
weighted avg	0.45	0.35	0.31	5020

2-) VGG16 SADECE SON 4 KATMAN EĞİTİLEBİLİR :

```
Epoch 1/10
79/79 [==============================] - 16s 144ms/step - loss: 21.7987 - accuracy: 0.1573
Epoch 2/10
79/79 [==============================] - 11s 136ms/step - loss: 1.5563 - accuracy: 0.4248
Epoch 3/10
79/79 [==============================] - 11s 136ms/step - loss: 1.2669 - accuracy: 0.5454
Epoch 4/10
79/79 [==============================] - 11s 136ms/step - loss: 1.0317 - accuracy: 0.6315
Epoch 5/10
79/79 [==============================] - 11s 136ms/step - loss: 0.8715 - accuracy: 0.6950
Epoch 6/10
79/79 [==============================] - 11s 136ms/step - loss: 0.7524 - accuracy: 0.7314
Epoch 7/10
79/79 [==============================] - 11s 136ms/step - loss: 0.5803 - accuracy: 0.8025
Epoch 8/10
79/79 [==============================] - 11s 136ms/step - loss: 0.5277 - accuracy: 0.8234
Epoch 9/10
79/79 [==============================] - 11s 136ms/step - loss: 0.3741 - accuracy: 0.8752
Epoch 10/10
79/79 [==============================] - 11s 136ms/step - loss: 0.3321 - accuracy: 0.8893
[[396   2  36   7  16   8   0   7  20  10]
 [ 86 181  26  23  11  24   1  26  33  91]
 [ 34   0 279  68  50  45  11  13   2   0]
 [ 18   2  59 233  45  96  18  21   6   4]
 [ 18   0  57  94 209  44   7  65   4   4]
 [  7   0  52 168  52 163   6  49   3   2]
 [ 10   0 101 133  21  37 191   6   2   1]
 [ 17   1  25  35  48  36   1 324   8   7]
 [161   1  42  25  16  15   2  20 208  12]
 [ 61  39  13  34  12  24   2  46  26 245]]
```

	precision	recall	f1-score	support
0	0.49	0.79	0.60	502
1	0.80	0.36	0.50	502
2	0.40	0.56	0.47	502
3	0.28	0.46	0.35	502
4	0.44	0.42	0.43	502
5	0.33	0.32	0.33	502
6	0.80	0.38	0.52	502
7	0.56	0.65	0.60	502
8	0.67	0.41	0.51	502
9	0.65	0.49	0.56	502
accuracy			0.48	5020
macro avg	0.54	0.48	0.49	5020
weighted avg	0.54	0.48	0.49	5020

3-) RESNET SADECE SON KATMAN EĞİTİLEBİLİR:

```
Epoch 1/10
79/79 [==============================] - 18s 197ms/step - loss: 2.0506 - accuracy: 0.2254
Epoch 2/10
79/79 [==============================] - 16s 204ms/step - loss: 2.0335 - accuracy: 0.2431
Epoch 3/10
79/79 [==============================] - 16s 201ms/step - loss: 2.0225 - accuracy: 0.2473
Epoch 4/10
79/79 [==============================] - 16s 196ms/step - loss: 2.0108 - accuracy: 0.2503
Epoch 5/10
79/79 [==============================] - 15s 196ms/step - loss: 1.9940 - accuracy: 0.2559
Epoch 6/10
79/79 [==============================] - 16s 197ms/step - loss: 2.0246 - accuracy: 0.2374
Epoch 7/10
79/79 [==============================] - 16s 199ms/step - loss: 1.9971 - accuracy: 0.2509
Epoch 8/10
79/79 [==============================] - 16s 199ms/step - loss: 2.0209 - accuracy: 0.2318
Epoch 9/10
79/79 [==============================] - 16s 198ms/step - loss: 2.0115 - accuracy: 0.2363
Epoch 10/10
79/79 [==============================] - 16s 198ms/step - loss: 1.9989 - accuracy: 0.2375
[[270  10  24   0  16   9   6  21 110  36]
 [ 94  54   4   3  19   4  32  33  70 189]
 [ 43   6  69   3 101  81  36  39 103  21]
 [ 16   4  44   7 125 103  48  68  54  33]
 [ 21   1  73   2 118  89  28  66  76  28]
 [  8   0  41   8 115 129  32 102  41  26]
 [ 23   4  28   6 109  59 111  55  64  43]
 [ 27   4  16   3  85  73  28 156  47  63]
 [194  12   8   1  26  13  17  23 160  48]
 [ 92  18   2   2  21   7  37  44  53 226]]
```

	precision	recall	f1-score	support
0	0.34	0.54	0.42	502
1	0.48	0.11	0.18	502
2	0.22	0.14	0.17	502
3	0.20	0.01	0.03	502
4	0.16	0.24	0.19	502
5	0.23	0.26	0.24	502
6	0.30	0.22	0.25	502
7	0.26	0.31	0.28	502
8	0.21	0.32	0.25	502
9	0.32	0.45	0.37	502
accuracy			0.26	5020
macro avg	0.27	0.26	0.24	5020
weighted avg	0.27	0.26	0.24	5020

4-) RESNET SADECE SON 4 KATMAN EĞİTİLEBİLİR:

```
Epoch 1/10
79/79 [==============================] - 18s 196ms/step - loss: 1.9396 - accuracy: 0.2806
Epoch 2/10
79/79 [==============================] - 16s 203ms/step - loss: 1.9140 - accuracy: 0.2892
Epoch 3/10
79/79 [==============================] - 16s 200ms/step - loss: 1.9033 - accuracy: 0.3021
Epoch 4/10
79/79 [==============================] - 15s 195ms/step - loss: 1.8779 - accuracy: 0.2983
Epoch 5/10
79/79 [==============================] - 15s 194ms/step - loss: 1.8903 - accuracy: 0.3004
Epoch 6/10
79/79 [==============================] - 15s 196ms/step - loss: 1.9244 - accuracy: 0.2878
Epoch 7/10
79/79 [==============================] - 16s 199ms/step - loss: 1.8851 - accuracy: 0.3002
Epoch 8/10
79/79 [==============================] - 16s 199ms/step - loss: 1.9230 - accuracy: 0.2883
Epoch 9/10
79/79 [==============================] - 16s 198ms/step - loss: 1.9058 - accuracy: 0.2859
Epoch 10/10
79/79 [==============================] - 15s 196ms/step - loss: 1.9059 - accuracy: 0.2817
[[280  68   0   0  26   4  10  40  44  30]
 [ 40 235   0   1   9   0  26  65  20 106]
 [ 55  31   2   2 111  23  81 143  45   9]
 [ 11  44   1   6  87  28 113 179  15  18]
 [ 22  31   0   2 115  14  65 216  27  10]
 [ 13  23   0   4  90  32  59 259   8  14]
 [ 15  58   0   1  60   8 233 102  11  14]
 [ 20  41   0   0  38   7  24 333  10  29]
 [182  86   0   1  14   1  24  57  85  52]
 [ 42 165   0   1   5   1  16  84  12 176]]
```

	precision	recall	f1-score	support
0	0.41	0.56	0.47	502
1	0.30	0.47	0.37	502
2	0.67	0.00	0.01	502
3	0.33	0.01	0.02	502
4	0.21	0.23	0.22	502
5	0.27	0.06	0.10	502
6	0.36	0.46	0.40	502
7	0.23	0.66	0.34	502
8	0.31	0.17	0.22	502
9	0.38	0.35	0.37	502
accuracy			0.30	5020
macro avg	0.35	0.30	0.25	5020
weighted avg	0.35	0.30	0.25	5020

3-)SONUÇ BÖLÜMÜ:

Öncelikle orijinal bir konvolüsyonel bir ağ tasarladığımızda, ağa verilen hiper parametrelerin başarılı sınıflandırma oranlarında önemli bir etkisini gördük. Elde verilen verilere göre, tasarladığımız konvolüsyonel ağının başarı oranını arttırmak için daha fazla konvolüsyon katmanı, daha az dropout oranı filtrelerin daha küçük olması daha doğru tahmin etmemiz sağlar. Soru-1 için is en iyi sonuç, 3 katmanlı, 3x3 filtre boyutlu ve 0.2 Dropout değerine sahip ağ en iyi sonucu vermiştir.

Soru-2 için ise, uygulanan modeller arasında en başarılı model, VGG16 modeli görülmüştür. Ayrıca daha fazla parametre tahmin doğruluğu için daha iyi olduğu ama daha fazla zaman aldığı gözlemlenilmiştir.

Transfer Öğrenmeyi kullanmanın birçok avantajı vardır. Temel avantajları, eğitim süresinden tasarruf etmemizi ve sinir ağımızın çoğu durumda daha iyi çalışması ve çok fazla veriye ihtiyacımız olmadı. Model zaten önceden eğitildiği için nispeten küçük eğitim verileriyle iyi bir makine öğrenimi modeli oluşturabildik.

Ancak bazı modelleri formüllere uydurmak gerçekten de zordur. Bu da dezavantaj sayılabilir.