FIRAT ÜNİVERSİTESİ

BİLGİSAYAR MÜHENDİSİĞİ

İLERİ SAYISAL GÖRÜNTÜ İŞLEME

FİNAL ÖDEVİ

HATİCE ÖZTÜRK

201129111

VERİ SETİ TANITIMI:

Veri seti, 3670 adet görüntü ve daisy, dandelion, roses, sunflowers, tulips olmak üzere 5 farklı sınıftan oluşmaktadır. Daisy sınıfında 633, Dandelion 898, Roses 641, Sunflowers 699 ve Tulips 799 adet görüntü içermektedir. Sınıflara ait etiketleme işlem yapılarak daisy sınıfına 0, dandelion sınıfına 1, roses sınıfına 2, sunflowers sınıfına 3 ve tulips sınıfına 4 etiketi verildi. Aşağıda Şekil 1 ve Şekil 2 'de sınıflar ve etiketleri verilmiştir.

```
In [213]: pandas.unique(data_labels)
     ...: print(pandas.unique(data_labels))
['daisy' 'dandelion' 'roses' 'sunflowers' 'tulips']
```

Şekil 1

```
In [214]: pandas.unique(labels)
     ...: print(pandas.unique(labels))
[0 1 2 3 4]
```

Şekil 2

Veri seti örnekleri aşağıdaki gibidir.

Daisy Dandelion Roses Sunflowers Tulips

MODEL OLUŞTURMA

Sınıflandırma işlemi için CNN modeli oluşturuldu.

Birçok Convolutional Katmanı arka arkaya kondu ve her birinden sonra ReLU katmanı eklendi. Ve bundan sonra Pooling katmanları ve Flattening katmanı eklendi . Daha sonra ReLu katmanı kadar Fully-Connceted katmanı eklendi.

Overfitting'i azaltmak için bir teknik olan, bir düzenlenme biçimi olan Dropout katmanı modele eklendi. Dropout'u bir katmana uyguladığınızda, eğitim işlemi sırasında katmandan rastgele bir dizi çıktı birimini (etkinleştirmeyi sıfıra ayarlayarak) çıkarır. Bu işlem ile, katmandan rastgele çıktı birimlerinin % 20'si çıkarıldı.

Aktivasyon "softmax" dır. Softmax, çıkışın 1'e kadar çıkmasını sağlar, böylece çıkış sinyali olasılık olarak yorumlanabilir. Daha sonra model, hangi seçeneğin en yüksek olasılıklara sahip olduğuna bağlı olarak tahminini yapacaktır.

Convolutional Layer özellikleri saptanmak için kullanıldı. Resmin özelliklerini algılamaktan sorumludur. Bu katman, görüntüdeki düşük ve yüksek seviyeli özellikleri çıkarmak için resme bazı fitreler uygular.

Pooling (Downsampling) Layer: Ağırlık sayısını azaltır ve uygunluğu kontrol eder. Bu katmanın görevi, gösterimin kayma boyutunu ve ağ içindeki parametreleri ve hesaplama sayısını azaltmak içindir. Bu sayede ağdaki uyumsuzluk kontrol edilmiş olur. Birçok Pooling işlemleri vardır, bu modelde en popüler olan max pooling kullanıldı.

Flattening Layer : Bu katman Klasik Sinir Ağı için verileri hazırlar.

Fully-Connected Layer : Bu katman modelin son ve en önemli katmanıdır. Verileri Flattening işleminden alır ve Sinir ağı yoluyla öğrenme işlemini geçekleştirir.

Aşağıdaki şekil 3 'de uygulamada oluşturulan model verilmektedir.

```
#%%
model = tf.keras.models.Sequential()
model.add(tf.keras.layers.InputLayer(input_shape=(64,64,3,))) # Input layer
model.add(tf.keras.layers.Conv2D(64, kernel_size=(3,3), activation='relu')) # 2D Convolution layer
model.add(tf.keras.layers.MaxPool2D(pool_size = (2,2))) # Max Pool layer
model.add(tf.keras.layers.BatchNormalization()) # Normalization layer
model.add(tf.keras.layers.Conv2D(64, kernel_size=(3,3), strides = (1,1), activation='relu')) # 2D Convolution layer
model.add(tf.keras.layers.MaxPool2D(pool_size = (2,2))) # Max Pool layer
model.add(tf.keras.layers.BatchNormalization()) # Normalization layer
model.add(tf.keras.layers.Conv2D(128, kernel_size=(3,3), strides = (1,1), activation='relu')) # 2D Convolution laye
model.add(tf.keras.layers.MaxPool2D(pool_size = (2,2))) # Max Pool layer
model.add(tf.keras.layers.BatchNormalization()) # Normalization layer
model.add(tf.keras.layers.Conv2D(128, kernel_size=(3,3), strides = (1,1), activation='relu')) # 2D Convolution laye
model.add(tf.keras.layers.MaxPool2D(pool_size = (2,2))) # Max Pool layer
model.add(tf.keras.layers.GlobalMaxPool2D()) # Global Max Pool layer
model.add(tf.keras.layers.Flatten()) # Dense Layers after flattening the data
model.add(tf.keras.layers.Dense(128, activation='relu'))
model.add(tf.keras.layers.Dropout(0.2)) # Dropout
model.add(tf.keras.layers.Dense(64, activation='relu'))
model.add(tf.keras.layers.BatchNormalization()) # Normalization layer
model.add(tf.keras.layers.Dense(5, activation='softmax')) # Add Output Layer
```

Şekil 3

MODEL EĞİTİM İŞLEMİ

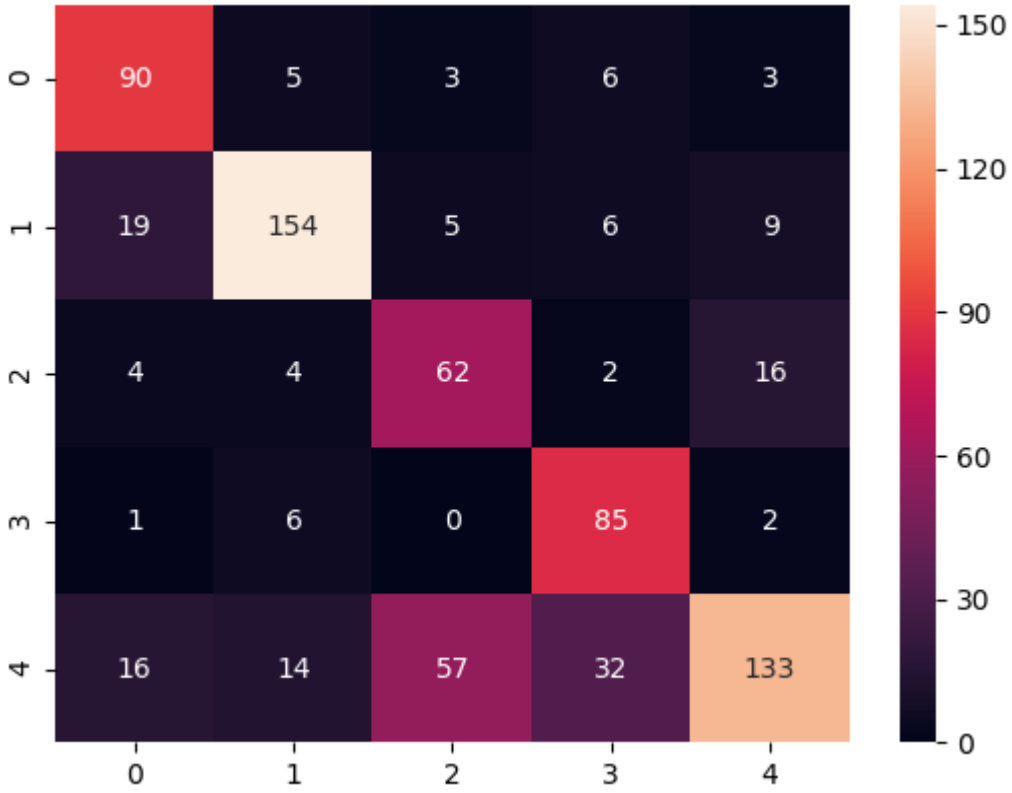
Model eğitimi için kullanılan veri seti %80-%20 oranında train ve test verisi olarak bölündü ve Epoch sayısı 20 olarak belirlendi. Aşağıdaki Şekil 4'de epoch sonuçları verilmiştir.

```
In [220]: model.fit(x_train,y_train, epochs=20)
Epoch 1/20
92/92 [==============================] - 51s 490ms/step - loss: 1.2903 - accuracy: 0.4980
Epoch 2/20
92/92 [==============================] - 43s 471ms/step - loss: 0.9791 - accuracy: 0.6172
Epoch 3/20
92/92 [==============================] - 43s 472ms/step - loss: 0.8261 - accuracy: 0.6785
Epoch 4/20
92/92 [==============================] - 45s 493ms/step - loss: 0.7850 - accuracy: 0.6982
Epoch 5/20
92/92 [==============================] - 52s 570ms/step - loss: 0.6721 - accuracy: 0.7548
Epoch 6/20
92/92 [==============================] - 46s 498ms/step - loss: 0.5964 - accuracy: 0.7762
Epoch 7/20
92/92 [==============================] - 44s 475ms/step - loss: 0.5123 - accuracy: 0.8099
Epoch 8/20
Epoch 8/20
92/92 [==============================] - 52s 570ms/step - loss: 0.4568 - accuracy: 0.8283
Epoch 9/20
92/92 [==============================] - 53s 572ms/step - loss: 0.3944 - accuracy: 0.8590
Epoch 10/20
92/92 [==============================] - 51s 549ms/step - loss: 0.3283 - accuracy: 0.8835
Epoch 11/20
92/92 [==============================] - 56s 607ms/step - loss: 0.2577 - accuracy: 0.9104
Epoch 12/20
92/92 [==============================] - 45s 487ms/step - loss: 0.2291 - accuracy: 0.9210
Epoch 13/20
92/92 [==============================] - 43s 471ms/step - loss: 0.1600 - accuracy: 0.9435
Epoch 14/20
92/92 [==============================] - 44s 475ms/step - loss: 0.1580 - accuracy: 0.9465
Epoch 15/20
92/92 [==============================] - 44s 475ms/step - loss: 0.1188 - accuracy: 0.9595
Epoch 16/20
92/92 [==============================] - 45s 485ms/step - loss: 0.0933 - accuracy: 0.9704
Epoch 17/20
92/92 [==============================] - 45s 491ms/step - loss: 0.0971 - accuracy: 0.9704
Epoch 18/20
92/92 [==============================] - 44s 482ms/step - loss: 0.0718 - accuracy: 0.9768
Epoch 19/20
92/92 [==============================] - 44s 483ms/step - loss: 0.0786 - accuracy: 0.9765
Epoch 20/20
92/92 [==============================] - 46s 497ms/step - loss: 0.0519 - accuracy: 0.9833
Out[220]: <tensorflow.python.keras.callbacks.History at 0x1ecbb92f668>
```

Şekil 4

CONFUSION MATRIX

Şekil 5'de modele ait confusion matrix'i verilmiştir.



Şekil 5

PRECISION, RECALL ,ACCURACY ve F1-SCORE

Şekil 6'da değerlendirme ölçütleri değerleri verilmiştir.

```
Accuracy: 0.71

Micro Precision: 0.71
Micro Recall: 0.71
Micro F1-score: 0.71

Macro Precision: 0.76
Macro Recall: 0.70
Macro F1-score: 0.71

Weighted Precision: 0.75
Weighted Recall: 0.71
Weighted F1-score: 0.72
```

Şekil 6