Klasifikasi Gambar Pistachio

Fatiya Quzza (2108107010042)

Dataset yang digunakan

LDataset Pistachio Image merupakan kumpulan citra pistachio yang digunakan untuk pengembangan model klasifikasi berbasis citra dan kecerdasan buatan. Dataset ini berasal dari penelitian yang dipublikasikan dalam jurnal "Progress in Nutrition" dan "Electronics". Dataset ini mencakup dua jenis pistachio utama: Kirmizi dan Siirt. Setiap citra pistachio dilengkapi dengan atribut morfologis, bentuk, dan warna sebagai fitur klasifikasi. Atribut morfologis mencakup area, perimeter, dan berbagai ukuran lainnya. Atribut bentuk melibatkan faktor bentuk seperti aspect ratio dan roundness. Sementara itu, atribut warna mencakup statistik warna seperti mean, standard deviation, skewness, dan kurtosis.

Disini saya hanya menggunakan struktur direktori "Pistachio_Image_Dataset/Pistachio_Image_Dataset," dimana terdapat dua folder utama: "Kirmizi_pistachio" dan "Siirt_pistachio". Setiap folder ini berisi gambar-gambar pistachio untuk masing-masing jenis. Disini tidak disertakan atribut morfologis, bentuk, atau warna secara eksplisit. Sebaliknya, fokusnya lebih pada representasi visual melalui citra pistachio.

Link Dataset :
https://www.kaggle.com/datasets/muratkokludataset/pistachio-image-dataset/data

Jumlah Fitur

```
Train set
class 0: 924
class 1: 687

Test set
class 0: 308
class 1: 229
```

Disini terdapat total 2149 fitur.

Jumlah Label

Pada Dataset ini terdapat dua label, yaitu Kirmizi Pistachio dan Siirt Pistachio, dimana Kiirmizi Pistachio di-encoding menjadi 0 dan Siirt Pistachio menjadi 1

```
class_label_encoding = {
    'Kirmizi_Pistachio': 0,
    'Siirt_Pistachio': 1
}
```

✓ 0.0s

Jenis Jaringan Saraf Tiruan yang Digunakan

Model jaringan saraf tiruan yang digunakan dalam klasifikasi ini adalah Convolutional Neural Network atau CNN. CNN dirancang khusus untuk menangani data berstruktur grid, seperti citra. CNN terdiri dari beberapa lapisan konvolusional yang mampu mengekstrak fitur-fitur penting dari citra secara hierarkis.

Pertama-tama, model ini memiliki beberapa lapisan konvolusional, dimulai dengan Conv2D yang menggunakan filter 3x3 untuk mengekstrak pola-pola dari citra input dengan aktivasi ReLU (Rectified Linear Unit) yang mengenalkan unsur non-linearitas. MaxPooling2D kemudian digunakan untuk mereduksi dimensi citra dan mengekstrak fitur-fitur utama. Proses ini diulang beberapa kali dengan penambahan Dropout untuk mencegah overfitting, yaitu melewatkan sebagian neuron pada saat pelatihan untuk meningkatkan generalisasi model.

Selanjutnya, hasil fitur dari konvolusi diekstraksi dan diratakan (flatten) menjadi vektor satu dimensi. Model ini kemudian memiliki beberapa lapisan Dense (fully connected) yang bertanggung jawab untuk melakukan klasifikasi. Dense layer ini memiliki fungsi aktivasi ReLU dan Dropout untuk penanganan overfitting. Kemudian, output layer menggunakan satu neuron dengan fungsi aktivasi sigmoid, yang mana cocok untuk tugas klasifikasi biner.

Jenis Optimisasi

Dalam model tersebut, jenis optimisasi yang digunakan adalah Adam, yang merupakan singkatan dari Adaptive Moment Estimation. Adam adalah metode optimisasi yang efisien dan seringkali digunakan dalam pelatihan model jaringan saraf tiruan. Adam menggabungkan konsep Momentum dan RMSprop.

Algoritma Adam menghitung dan memanfaatkan momen pertama (mean) dan momen kedua (mean kuadrat) dari gradien parameter selama iterasi pelatihan. Dengan menggunakan informasi ini, Adam kemudian menghasilkan pembaruan parameter untuk meningkatkan efisiensi konvergensi model. Kelebihan utama dari Adam adalah kemampuannya dalam menyesuaikan laju pembelajaran secara adaptif berdasarkan momen kedua, memungkinkan algoritma untuk secara efektif dan cepat beradaptasi terhadap perubahan dalam gradien. Keistimewaan ini membuat Adam menjadi pilihan yang baik untuk berbagai masalah pelatihan model, dan seringkali menghasilkan konvergensi yang lebih cepat

Jenis Fungsi Aktivasi yang digunakan

Fungsi aktivasi yang digunakan di lapisan konvolusional adalah ReLU (Rectified Linear Unit), ditunjukkan oleh parameter 'relu'. ReLU adalah pilihan umum karena sifat non-linearitasnya yang sederhana namun efektif. Fungsi ini mengonversi semua nilai negatif menjadi nol dan mempertahankan nilai positif, membantu model untuk mempelajari pola-pola yang kompleks dalam data.

Lapisan Dense pada akhir model menggunakan fungsi aktivasi sigmoid, yang merupakan fungsi aktivasi umum untuk tugas klasifikasi biner. Fungsi sigmoid menghasilkan keluaran antara 0 dan 1, yang dapat diinterpretasikan sebagai probabilitas kelas positif. Ini sangat sesuai untuk tugas seperti klasifikasi jenis pistachio, di mana model diarahkan untuk menghasilkan prediksi biner (positif atau negatif) terkait jenis pistachio.

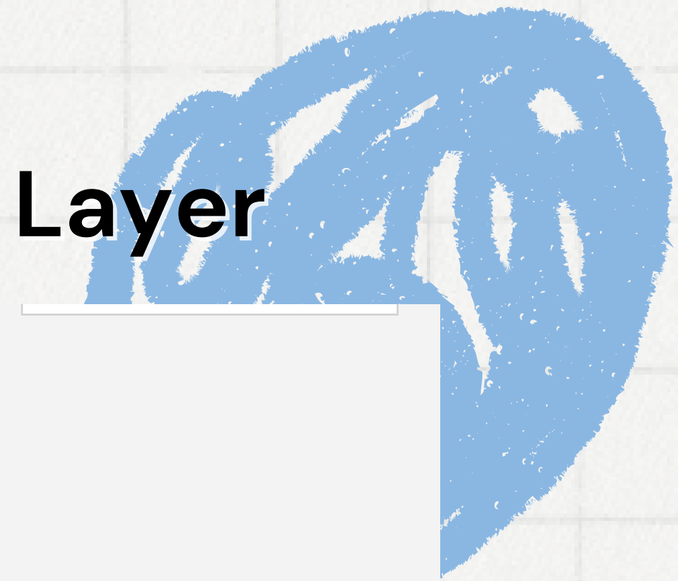
Dengan kombinasi fungsi aktivasi ReLU dan sigmoid, model ini dirancang untuk memanfaatkan keuntungan dari ReLU dalam mengekstrak fitur-fitur yang kompleks dan kemampuan sigmoid dalam menghasilkan prediksi probabilitas biner. Kombinasi ini dapat meningkatkan kemampuan model untuk mengklasifikasikan gambar jenis pistachio dengan akurasi yang baik.

Jumlah Hidden Layer

```
model = Sequential()

model.add(layers.Conv2D(32,(3,3),activation='relu',input_shape=(200,200,1)))
model.add(layers.Conv2D(64,(3,3),activation='relu'))
model.add(layers.MaxPooling2D())

model.add(layers.Conv2D(64,(3,3),activation='relu'))
model.add(layers.Conv2D(128,(3,3),activation='relu'))
model.add(layers.MaxPooling2D())


model.add(layers.Conv2D(256,(3,3),activation='relu'))
model.add(layers.Conv2D(128,(3,3),activation='relu'))
model.add(layers.MaxPooling2D())

model.add(layers.Dropout(0.2))

model.add(layers.Flatten())

model.add(layers.Dense(1024,activation='relu'))
model.add(layers.Dropout(0.2))
model.add(layers.Dense(512,activation='relu'))
model.add(layers.Dropout(0.1))
model.add(layers.Dense(64,activation='relu'))
model.add(layers.Dense(1,'sigmoid'))
```

Disini terdapat 8 hidden layer, pada code diatas yang termasuk hidden layer adalah layer Conv2D (selain yang pertama dan terakhir) dan juga layer Dense, untuk MaxPooling2D, Dropout, dan Flatten bukan termasuk hidden layer

Jumlah Total Hidden Node per Layer



```
#Jumlah Total Hidden Node per Layer

conv2D = 32
conv2D_1 = 64
conv2D_2 = 64
conv2D_3 = 128
conv2D_4 = 256
conv2D_5 = 128
conv2D_6 = 128
dense = 1024
dense_1 = 512
dense_2 = 64
dense_3 = 1

total = conv2D + conv2D_1 + conv2D_2 + conv2D_3 + conv2D_4 + conv2D_5 + conv2D_6 + dense
+ dense_1 + dense_2 + dense_3

print('total hidden node = ', total)
```

✓ 0.0s Pythor

```
total hidden node =  2401
```

Jumlah Total Bobot

```
# Menampilkan jumlah total bobot
total_weights = sum([var.numpy().flatten().shape[0] for var in model.trainable_variables])

print("Jumlah Total Bobot pada Model:", total_weights)
```

3] ✓ 0.5s Python

```
Jumlah Total Bobot pada Model: 59081281
```

Thank you very much!