Fundations of Deep Learning

Flowers Photo Classifier

Pasquale Formicola

```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

Connect to Google Drive

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount
```

Loading Data

Creation of Training Set and Validation Set

```
image_size = (224, 224)  # Dimensione desiderata delle immagini
batch_size=32
datagen = ImageDataGenerator(
    rescale=1.0/255.0,  # Normalizzazione dei valori dei pixel tra 0 e 1
    validation_split=0.2,  # Percentuale di dati da utilizzare per la validazione
)

train_generator = datagen.flow_from_directory(
    ('/content/drive/MyDrive/Colab Notebooks/flower_photos'),
    target_size=image_size,
    class_mode='categorical',
    batch_size=batch_size,
    subset='training'  # Utilizza solo il subset di addestramento
)

validation_generator = datagen.flow_from_directory(
    ('/content/drive/MyDrive/Colab Notebooks/flower_photos'),
    target_size=image_size,
    class_mode='categorical',
    batch_size=batch_size,
    subset='validation'  # Utilizza solo il subset di validazione
)
```

```
Found 2955 images belonging to 5 classes.
Found 735 images belonging to 5 classes.
```

Training

```
import numpy as np

class_indices = train_generator.class_indices
num_classes = len(class_indices)

class_counts = train_generator.classes
class_counts = np.bincount(class_counts, minlength=num_classes)

for class_name, class_index in class_indices.items():
    print(f"Class: {class_name} - Size of subsample: {class_counts[class_index]}")
```

```
Class: daisy - Size of subsample: 507
Class: dandelion - Size of subsample: 719
Class: roses - Size of subsample: 513
Class: sunflowers - Size of subsample: 560
Class: tulips - Size of subsample: 656
```

Validation

```
class_indices = validation_generator.class_indices
num_classes = len(class_indices)

class_counts = validation_generator.classes
class_counts = np.bincount(class_counts, minlength=num_classes)
```

```
for class_name, class_index in class_indices.items():
    print(f"Class: {class_name} - Size of subsample: {class_counts[class_index]}")
```

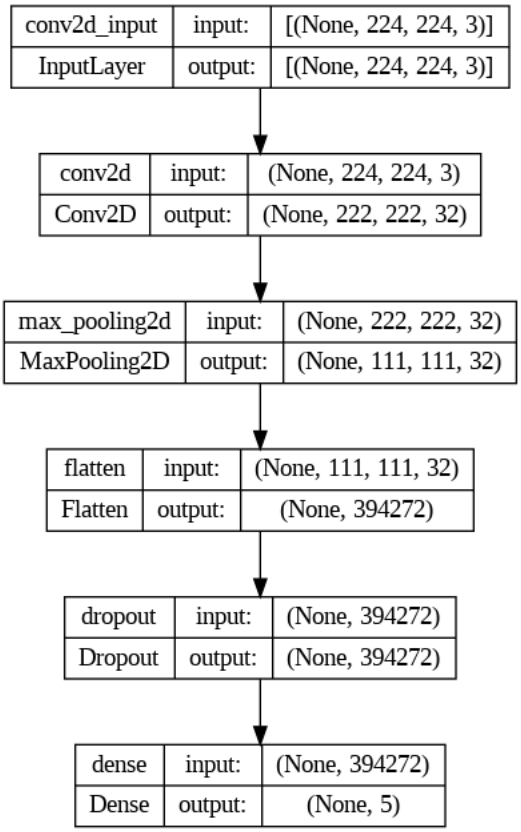
```
Class: daisy - Size of subsample: 126
Class: dandelion - Size of subsample: 179
Class: roses - Size of subsample: 128
Class: sunflowers - Size of subsample: 139
Class: tulips - Size of subsample: 163
```

First Architecture

- Conv2D
- MaxPooling2D
- Flatten
- Dropout
- Dense
- SoftMax Activation

```
from tensorflow import keras

model = keras.models.Sequential()
model.add(keras.layers.Conv2D(32, (3, 3), activation='relu', input_shape=(224, 224, 3)))
model.add(keras.layers.MaxPooling2D((2, 2)))
model.add(keras.layers.Flatten())
model.add(keras.layers.Dropout(0.5))  # dropout rate 0,5
model.add(keras.layers.Dense(5, activation='softmax'))
```

```
import tensorflow as tf
from tensorflow.keras.utils import plot_model

plot_model(model, to_file='model.png', show_shapes=True, show_layer_names=True)
```



Training the Model

Loss Function = Categorical CrossEntropy, with Adam Optimizer

```
import matplotlib.pyplot as plt
```

```
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

history = model.fit(train_generator, batch_size=64, epochs=10, validation_data=validation_generator)
```

```
Epoch 1/10
93/93 [==============================] - 588s 6s/step - loss: 2.8776 - accuracy: 0.4802 - val_loss: 1.1527 - val_accuracy
Epoch 2/10
93/93 [==============================] - 20s 210ms/step - loss: 0.8442 - accuracy: 0.6843 - val_loss: 1.1078 - val_accura
Epoch 3/10
93/93 [==============================] - 21s 222ms/step - loss: 0.4735 - accuracy: 0.8589 - val_loss: 1.1114 - val_accura
Epoch 4/10
93/93 [==============================] - 20s 212ms/step - loss: 0.2503 - accuracy: 0.9411 - val_loss: 1.2679 - val_accura
Epoch 5/10
93/93 [==============================] - 21s 227ms/step - loss: 0.1284 - accuracy: 0.9739 - val_loss: 1.3509 - val_accura
Epoch 6/10
93/93 [==============================] - 20s 217ms/step - loss: 0.0708 - accuracy: 0.9909 - val_loss: 1.5552 - val_accura
Epoch 7/10
93/93 [==============================] - 21s 224ms/step - loss: 0.0359 - accuracy: 0.9970 - val_loss: 1.6125 - val_accura
Epoch 8/10
93/93 [==============================] - 20s 218ms/step - loss: 0.0248 - accuracy: 0.9990 - val_loss: 1.5582 - val_accura
Epoch 9/10
93/93 [==============================] - 21s 224ms/step - loss: 0.0172 - accuracy: 0.9983 - val_loss: 1.7325 - val_accura
Epoch 10/10
93/93 [==============================] - 19s 210ms/step - loss: 0.0158 - accuracy: 0.9990 - val_loss: 1.6490 - val_accura
```

```
model.save('/content/drive/MyDrive/Deep Learning/Flower_Classifier_v1.0', overwrite=True, save_format="h5")
```

```
# Plot
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Accuracy del Modello')
plt.xlabel('Epoca')
plt.ylabel('Accuratezza')
plt.legend(['Train', 'Validation'], loc='upper left')
plt.show()
```

54,69% Accuracy on Validation Set.

Data Augmentation

Rotation, Zoom, Shift and Horizontal Flip of Images trying to achive more accuracy

```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import matplotlib.pyplot as plt
import numpy as np

datagen = ImageDataGenerator(
    rescale=1.0/255.0,
    validation_split=0.2,
    rotation_range=20,  # Angoli di rotazione casuali fino a 20 gradi
    zoom_range=0.2,  # Zoom casuale fino al 20%
    width_shift_range=0.2,  # Spostamento orizzontale casuale fino al 20% della larghezza dell'immagine
    height_shift_range=0.2,  # Spostamento verticale casuale fino al 20% dell'altezza dell'immagine
    horizontal_flip=True  # Ribaltamento orizzontale casuale delle immagini
)

train_generator = datagen.flow_from_directory(
    ('/content/drive/MyDrive/Colab Notebooks/flower_photos'),
    target_size=image_size,
    batch_size=batch_size,
    class_mode='categorical',
    subset='training'
)

validation_generator = datagen.flow_from_directory(
    ('/content/drive/MyDrive/Colab Notebooks/flower_photos'),
    target_size=image_size,
    batch_size=batch_size,
    class_mode='categorical',
    subset='validation'
)
```

```
Found 2955 images belonging to 5 classes.
Found 735 images belonging to 5 classes.
```

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import matplotlib.pyplot as plt

# Create an ImageDataGenerator with augmentation settings
datagen = ImageDataGenerator(
    rescale=1.0/255.0,
    validation_split=0.2,
    rotation_range=20,
    zoom_range=0.2,
    width_shift_range=0.2,
    height_shift_range=0.2,
    horizontal_flip=True
)

# Specify the path to the directory containing the images
image_directory = '/content/drive/MyDrive/Colab Notebooks/flower_photos'

# Generate batches of original and augmented images
batch_size = 1
original_images = datagen.flow_from_directory(
    directory=image_directory,
    target_size=(224, 224),
    batch_size=batch_size,
    class_mode=None,
    subset='validation',
    shuffle=False
)

augmented_images = datagen.flow_from_directory(
    directory=image_directory,
    target_size=(224, 224),
    batch_size=batch_size,
    class_mode=None,
    subset='validation',
    shuffle=False
)

# Retrieve the first image from each batch
original_image = original_images.next()[0]
augmented_image = augmented_images.next()[0]

# Visualize the original and augmented images side by side
fig, axes = plt.subplots(1, 2)
axes[0].imshow(original_image)
axes[0].set_title('Original Image')
axes[0].axis('off')
axes[1].imshow(augmented_image)
axes[1].set_title('Augmented Image')
axes[1].axis('off')

plt.show()
```

```
Found 735 images belonging to 5 classes.
Found 735 images belonging to 5 classes.
```

Model Improvement

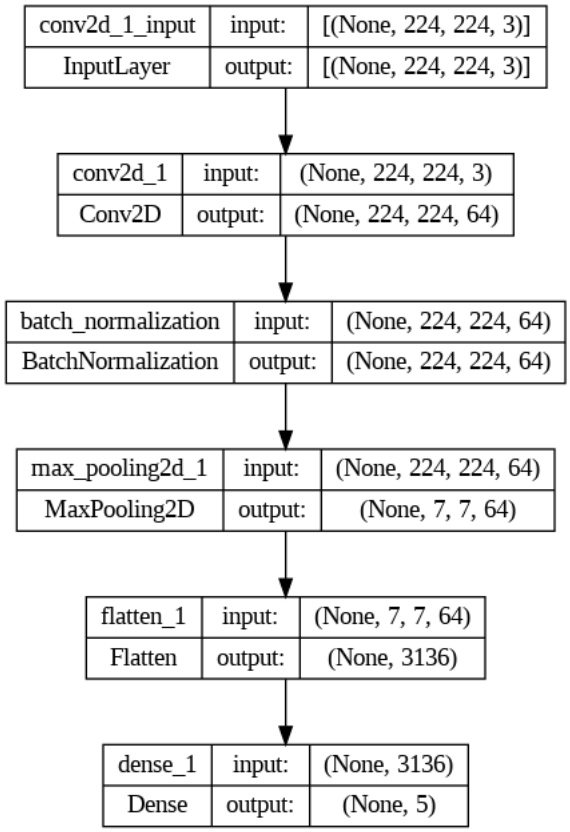
New Architecture

- Bigger Kernel size and Pooling Window
- Add Batch Normalization to stabilize the training

```
from tensorflow import keras
```

```
model = keras.models.Sequential()
model.add(keras.layers.Conv2D(64, (24, 24), activation='relu', padding='same', input_shape=(224, 224, 3)))
model.add(keras.layers.BatchNormalization())
model.add(keras.layers.MaxPooling2D((32, 32)))
model.add(keras.layers.Flatten())
model.add(keras.layers.Dense(5, activation='softmax'))

# Decrease the learning rate
optimizer = keras.optimizers.Adam(learning_rate=1e-4)

model.compile(optimizer=optimizer, loss='categorical_crossentropy', metrics=['accuracy'])
```

```
plot_model(model, to_file='model.png', show_shapes=True, show_layer_names=True)
```




```
history = model.fit(train_generator, batch_size=64, epochs=15, validation_data=validation_generator)
```

```
Epoch 1/15
93/93 [==============================] - 51s 462ms/step - loss: 1.5837 - accuracy: 0.3939 - val_loss: 1.4829 - val_accura
Epoch 2/15
93/93 [==============================] - 35s 372ms/step - loss: 1.1013 - accuracy: 0.5519 - val_loss: 1.3739 - val_accura
Epoch 3/15
93/93 [==============================] - 34s 369ms/step - loss: 0.9831 - accuracy: 0.6122 - val_loss: 1.2168 - val_accura
Epoch 4/15
93/93 [==============================] - 35s 371ms/step - loss: 0.9115 - accuracy: 0.6393 - val_loss: 1.3573 - val_accura
Epoch 5/15
93/93 [==============================] - 34s 368ms/step - loss: 0.8561 - accuracy: 0.6616 - val_loss: 1.0861 - val_accura
Epoch 6/15
93/93 [==============================] - 35s 373ms/step - loss: 0.7749 - accuracy: 0.7083 - val_loss: 1.1024 - val_accura
Epoch 7/15
93/93 [==============================] - 34s 366ms/step - loss: 0.7297 - accuracy: 0.7201 - val_loss: 1.0778 - val_accura
Epoch 8/15
93/93 [==============================] - 35s 373ms/step - loss: 0.6870 - accuracy: 0.7415 - val_loss: 1.1370 - val_accura
Epoch 9/15
93/93 [==============================] - 34s 367ms/step - loss: 0.6617 - accuracy: 0.7567 - val_loss: 1.8199 - val_accura
Epoch 10/15
93/93 [==============================] - 35s 375ms/step - loss: 0.6065 - accuracy: 0.7810 - val_loss: 1.2978 - val_accura
Epoch 11/15
93/93 [==============================] - 34s 367ms/step - loss: 0.5779 - accuracy: 0.7854 - val_loss: 1.1755 - val_accura
Epoch 12/15
93/93 [==============================] - 35s 373ms/step - loss: 0.5573 - accuracy: 0.8020 - val_loss: 1.1585 - val_accura
Epoch 13/15
93/93 [==============================] - 34s 368ms/step - loss: 0.5368 - accuracy: 0.8108 - val_loss: 1.1198 - val_accura
Epoch 14/15
93/93 [==============================] - 35s 373ms/step - loss: 0.4939 - accuracy: 0.8284 - val_loss: 1.1853 - val_accura
Epoch 15/15
93/93 [==============================] - 34s 368ms/step - loss: 0.4496 - accuracy: 0.8416 - val_loss: 1.1772 - val_accura
```

```
model.save('/content/drive/MyDrive/Deep Learning/Flower_Classifier_v2.0', overwrite=True, save_format="h5")


# Plot
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Accuracy del Modello')
plt.xlabel('Epoca')
plt.ylabel('Accuratezza')
plt.legend(['Train', 'Validation'], loc='upper left')
plt.show()
```

53,7% Validation Accuracy Achived

- The Data Augmentation techniques used didn't improve the model
- Neither the new architecture

Deeper Neural Network

We're going to make a Deep Net.

We have also find out in literature that a Dropout Rate of 20% for Image Classification is preferred, in general we tried this way since this value, in papers we've found, should stay between 0.2 and 0.5

Also images dimension has been changed to 112x112px

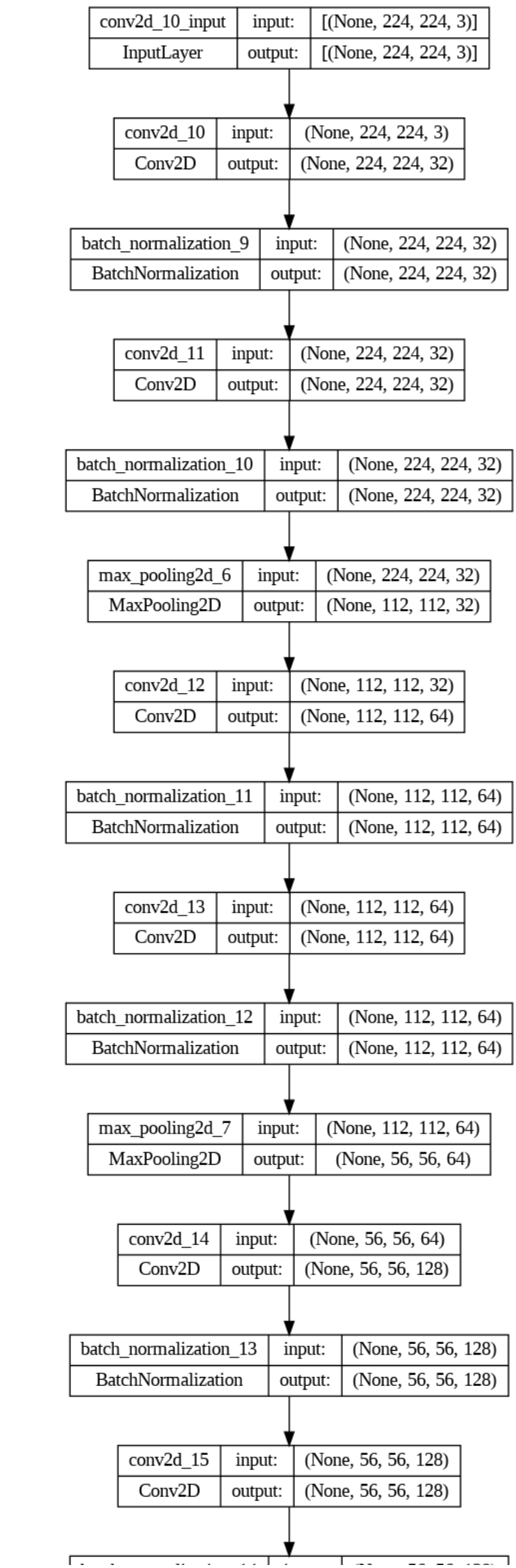
- Preventing Overfitting
- Achive a better Generalization

```
from tensorflow import keras
import tensorflow as tf

# Riduzione della risoluzione delle immagini

image_size = (112, 112)  # Desired image size
batch_size = 32

datagen = keras.preprocessing.image.ImageDataGenerator(
    rescale=1.0/255.0,  # Normalize pixel values between 0 and 1
    validation_split=0.2,  # Percentage of data to use for validation
)

train_generator = datagen.flow_from_directory(
    '/content/drive/MyDrive/Colab Notebooks/flower_photos',
    target_size=image_size,
    class_mode='categorical',
    batch_size=batch_size,
    subset='training'  # Use only the training subset
)

validation_generator = datagen.flow_from_directory(
    '/content/drive/MyDrive/Colab Notebooks/flower_photos',
    target_size=image_size,
    class_mode='categorical',
    batch_size=batch_size,
    subset='validation'  # Use only the validation subset
)

model = keras.models.Sequential()

model.add(keras.layers.Conv2D(32, (3, 3), activation='relu', padding='same', input_shape=(112, 112, 3)))
model.add(keras.layers.BatchNormalization())
model.add(keras.layers.Conv2D(32, (3, 3), activation='relu', padding='same'))
model.add(keras.layers.BatchNormalization())
model.add(keras.layers.MaxPooling2D((2, 2)))

model.add(keras.layers.Conv2D(64, (6, 6), activation='relu', padding='same'))
model.add(keras.layers.BatchNormalization())
model.add(keras.layers.Conv2D(64, (6, 6), activation='relu', padding='same'))
model.add(keras.layers.BatchNormalization())
model.add(keras.layers.MaxPooling2D((2, 2)))

model.add(keras.layers.Conv2D(128, (9, 9), activation='relu', padding='same'))
model.add(keras.layers.BatchNormalization())
model.add(keras.layers.Conv2D(128, (9, 9), activation='relu', padding='same'))
model.add(keras.layers.BatchNormalization())
model.add(keras.layers.MaxPooling2D((2, 2)))

model.add(keras.layers.Conv2D(256, (18, 18), activation='relu', padding='same'))
model.add(keras.layers.BatchNormalization())
```

```
model.add(keras.layers.Conv2D(256, (18, 18), activation='relu', padding='same'))
model.add(keras.layers.BatchNormalization())
model.add(keras.layers.MaxPooling2D((2, 2)))

model.add(keras.layers.Flatten())
model.add(keras.layers.Dense(512, activation='relu'))
model.add(keras.layers.Dropout(0.2))
model.add(keras.layers.Dense(5, activation='softmax'))

optimizer = keras.optimizers.Adam(learning_rate=1e-4)

model.compile(optimizer=optimizer, loss='categorical_crossentropy', metrics=['accuracy'])
```

```
Found 2955 images belonging to 5 classes.
Found 735 images belonging to 5 classes.
```

```
from tensorflow.keras.utils import plot_model

# Visualizza il modello
plot_model(model, to_file='model.png', show_shapes=True, show_layer_names=True)
```


conv2d_10_input | input: | [(None, 224, 224, 3)]
InputLayer | output: | [(None, 224, 224, 3)]
conv2d_10 | input: | (None, 224, 224, 3)
Conv2D | output: | (None, 224, 224, 32)
batch_normalization_9 | input: | (None, 224, 224, 32)
BatchNormalization | output: | (None, 224, 224, 32)
conv2d_11 | input: | (None, 224, 224, 32)
Conv2D | output: | (None, 224, 224, 32)
batch_normalization_10 | input: | (None, 224, 224, 32)
BatchNormalization | output: | (None, 224, 224, 32)
max_pooling2d_6 | input: | (None, 224, 224, 32)
MaxPooling2D | output: | (None, 112, 112, 32)
conv2d_12 | input: | (None, 112, 112, 32)
Conv2D | output: | (None, 112, 112, 64)
batch_normalization_11 | input: | (None, 112, 112, 64)
BatchNormalization | output: | (None, 112, 112, 64)
conv2d_13 | input: | (None, 112, 112, 64)
Conv2D | output: | (None, 112, 112, 64)
batch_normalization_12 | input: | (None, 112, 112, 64)
BatchNormalization | output: | (None, 112, 112, 64)
max_pooling2d_7 | input: | (None, 112, 112, 64)
MaxPooling2D | output: | (None, 56, 56, 64)
conv2d_14 | input: | (None, 56, 56, 64)
Conv2D | output: | (None, 56, 56, 128)
batch_normalization_13 | input: | (None, 56, 56, 128)
BatchNormalization | output: | (None, 56, 56, 128)
conv2d_15 | input: | (None, 56, 56, 128)
Conv2D | output: | (None, 56, 56, 128)

batch_normalization_14	input:	(None, 56, 56, 128)
BatchNormalization	output:	(None, 56, 56, 128)

We have also increased the Batch Size to 128 to improve speed and balance the 40 epoch

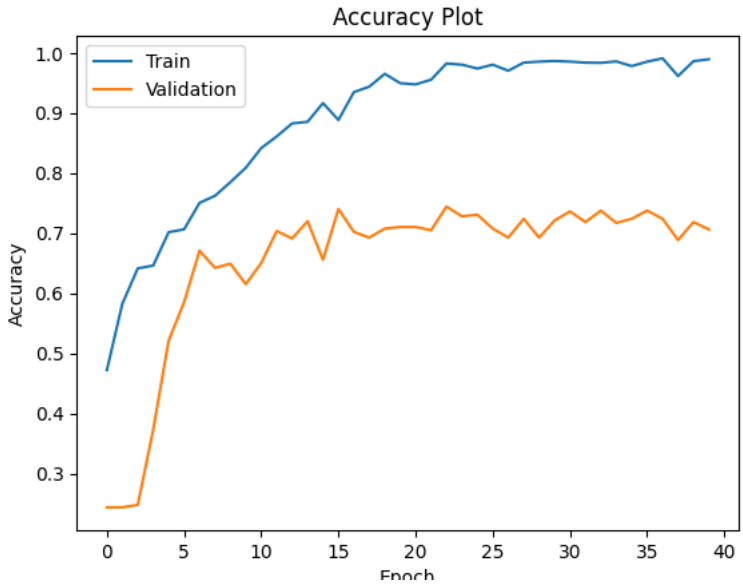
```
history = model.fit(train_generator, batch_size=128, epochs=40, validation_data=validation_generator)
```

```
Epoch 1/40
93/93 [==============================] - 38s 275ms/step - loss: 1.9157 - accuracy: 0.4071 - val_loss: 2.6577 - val_accura
Epoch 2/40
93/93 [==============================] - 23s 242ms/step - loss: 1.1555 - accuracy: 0.5357 - val_loss: 3.0710 - val_accura
Epoch 3/40
93/93 [==============================] - 22s 232ms/step - loss: 1.0053 - accuracy: 0.6010 - val_loss: 2.5271 - val_accura
Epoch 4/40
93/93 [==============================] - 21s 228ms/step - loss: 0.9236 - accuracy: 0.6247 - val_loss: 2.0044 - val_accura
Epoch 5/40
93/93 [==============================] - 20s 213ms/step - loss: 0.8250 - accuracy: 0.6785 - val_loss: 1.3474 - val_accura
Epoch 6/40
93/93 [==============================] - 21s 225ms/step - loss: 0.7592 - accuracy: 0.7107 - val_loss: 1.0110 - val_accura
Epoch 7/40
93/93 [==============================] - 20s 214ms/step - loss: 0.6743 - accuracy: 0.7367 - val_loss: 1.0650 - val_accura
Epoch 8/40
93/93 [==============================] - 21s 225ms/step - loss: 0.6423 - accuracy: 0.7557 - val_loss: 0.9169 - val_accura
Epoch 9/40
93/93 [==============================] - 20s 215ms/step - loss: 0.6034 - accuracy: 0.7692 - val_loss: 0.8656 - val_accura
Epoch 10/40
93/93 [==============================] - 21s 224ms/step - loss: 0.5240 - accuracy: 0.8007 - val_loss: 0.9299 - val_accura
Epoch 11/40
93/93 [==============================] - 20s 216ms/step - loss: 0.5115 - accuracy: 0.8064 - val_loss: 1.0330 - val_accura
Epoch 12/40
93/93 [==============================] - 21s 229ms/step - loss: 0.5290 - accuracy: 0.8054 - val_loss: 1.1980 - val_accura
Epoch 13/40
93/93 [==============================] - 20s 219ms/step - loss: 0.4561 - accuracy: 0.8379 - val_loss: 0.9351 - val_accura
Epoch 14/40
93/93 [==============================] - 21s 224ms/step - loss: 0.3839 - accuracy: 0.8596 - val_loss: 0.8406 - val_accura
Epoch 15/40
93/93 [==============================] - 20s 214ms/step - loss: 0.3295 - accuracy: 0.8782 - val_loss: 1.0947 - val_accura
Epoch 16/40
93/93 [==============================] - 22s 236ms/step - loss: 0.3059 - accuracy: 0.8910 - val_loss: 1.0396 - val_accura
Epoch 17/40
93/93 [==============================] - 20s 213ms/step - loss: 0.2650 - accuracy: 0.9056 - val_loss: 0.9378 - val_accura
Epoch 18/40
93/93 [==============================] - 21s 225ms/step - loss: 0.2239 - accuracy: 0.9218 - val_loss: 1.2524 - val_accura
Epoch 19/40
93/93 [==============================] - 20s 216ms/step - loss: 0.1845 - accuracy: 0.9303 - val_loss: 1.1793 - val_accura
Epoch 20/40
93/93 [==============================] - 22s 235ms/step - loss: 0.2179 - accuracy: 0.9262 - val_loss: 1.3398 - val_accura
Epoch 21/40
93/93 [==============================] - 20s 214ms/step - loss: 0.1512 - accuracy: 0.9499 - val_loss: 1.1435 - val_accura
Epoch 22/40
93/93 [==============================] - 21s 224ms/step - loss: 0.0990 - accuracy: 0.9668 - val_loss: 1.4504 - val_accura
Epoch 23/40
93/93 [==============================] - 20s 216ms/step - loss: 0.1016 - accuracy: 0.9635 - val_loss: 1.4187 - val_accura
Epoch 24/40
93/93 [==============================] - 22s 235ms/step - loss: 0.1362 - accuracy: 0.9550 - val_loss: 1.3716 - val_accura
Epoch 25/40
93/93 [==============================] - 20s 214ms/step - loss: 0.1250 - accuracy: 0.9577 - val_loss: 1.4233 - val_accura
Epoch 26/40
93/93 [==============================] - 21s 226ms/step - loss: 0.1015 - accuracy: 0.9682 - val_loss: 2.1720 - val_accura
Epoch 27/40
93/93 [==============================] - 20s 219ms/step - loss: 0.1249 - accuracy: 0.9624 - val_loss: 1.4191 - val_accura
Epoch 28/40
93/93 [==============================] - 21s 229ms/step - loss: 0.0771 - accuracy: 0.9760 - val_loss: 1.6374 - val_accura
Epoch 29/40
93/93 [==============================] - 20s 215ms/step - loss: 0.0637 - accuracy: 0.9807 - val_loss: 1.4541 - val_accura
```

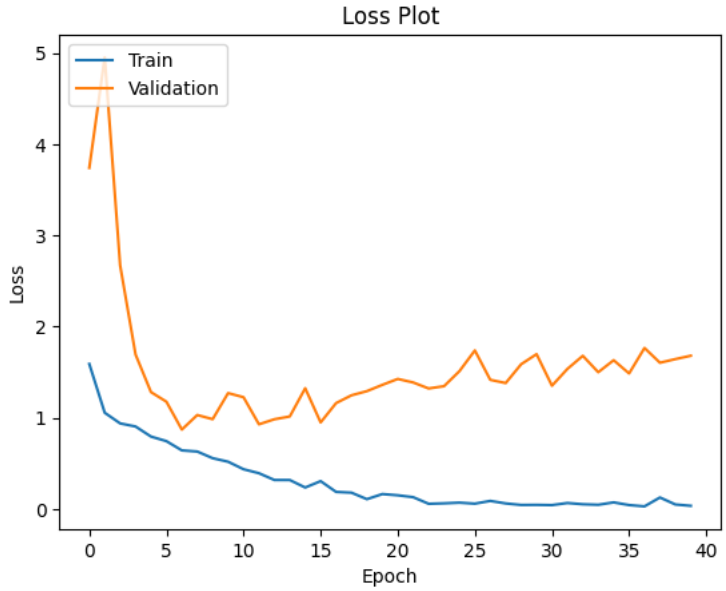
```
model.save('/content/drive/MyDrive/Deep Learning/Flower_Classifier_v3.0', overwrite=True, save_format="h5")
```

70,61% Accuracy

```
import matplotlib.pyplot as plt

# Accuracy Plot
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Accuracy Plot')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend(['Train', 'Validation'], loc='upper left')
plt.show()
```




```
# Loss Plot
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Loss Plot')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend(['Train', 'Validation'], loc='upper left')
plt.show()
```




```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 224, 224, 32)      896

 batch_normalization (BatchN  (None, 224, 224, 32)     128
 ormalization)

 conv2d_1 (Conv2D)           (None, 224, 224, 32)      9248

 batch_normalization_1 (Batc  (None, 224, 224, 32)     128
 hNormalization)

 max_pooling2d (MaxPooling2D  (None, 112, 112, 32)     0
 )

 conv2d_2 (Conv2D)           (None, 112, 112, 64)      73792

 batch_normalization_2 (Batc  (None, 112, 112, 64)     256
 hNormalization)

 conv2d_3 (Conv2D)           (None, 112, 112, 64)      147520
```

```
batch_normalization_3 (Batc  (None, 112, 112, 64)     256
hNormalization)

max_pooling2d_1 (MaxPooling  (None, 56, 56, 64)       0
2D)

conv2d_4 (Conv2D)           (None, 56, 56, 128)       663680

batch_normalization_4 (Batc  (None, 56, 56, 128)      512
hNormalization)

conv2d_5 (Conv2D)           (None, 56, 56, 128)       1327232

batch_normalization_5 (Batc  (None, 56, 56, 128)      512
hNormalization)

max_pooling2d_2 (MaxPooling  (None, 28, 28, 128)      0
2D)

conv2d_6 (Conv2D)           (None, 28, 28, 256)       10617088

batch_normalization_6 (Batc  (None, 28, 28, 256)      1024
hNormalization)

conv2d_7 (Conv2D)           (None, 28, 28, 256)       21233920

batch_normalization_7 (Batc  (None, 28, 28, 256)      1024
hNormalization)

max_pooling2d_3 (MaxPooling  (None, 14, 14, 256)      0
2D)

flatten (Flatten)           (None, 50176)             0
```

```
keras.utils.plot_model(model)
```