Lab Report – 6

Transfer Learning

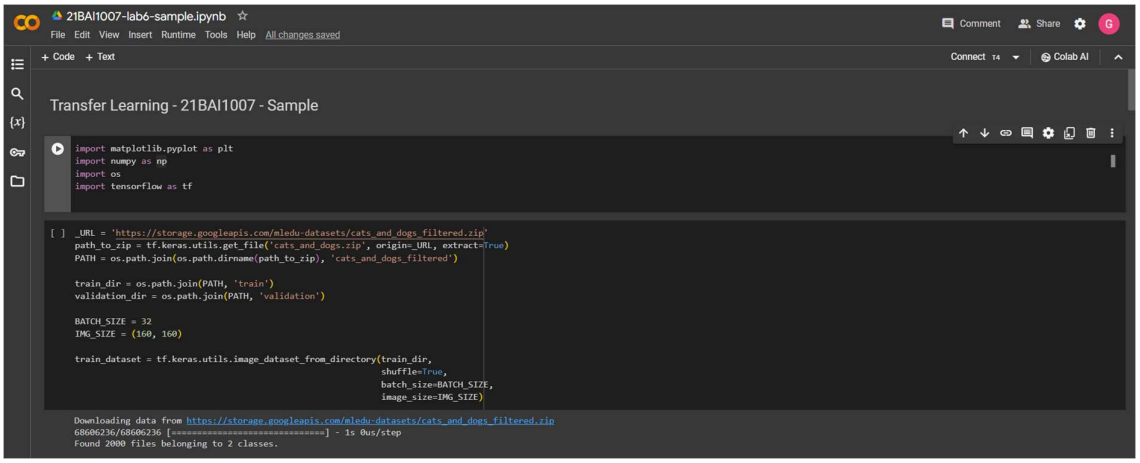
Goutham Krishnan

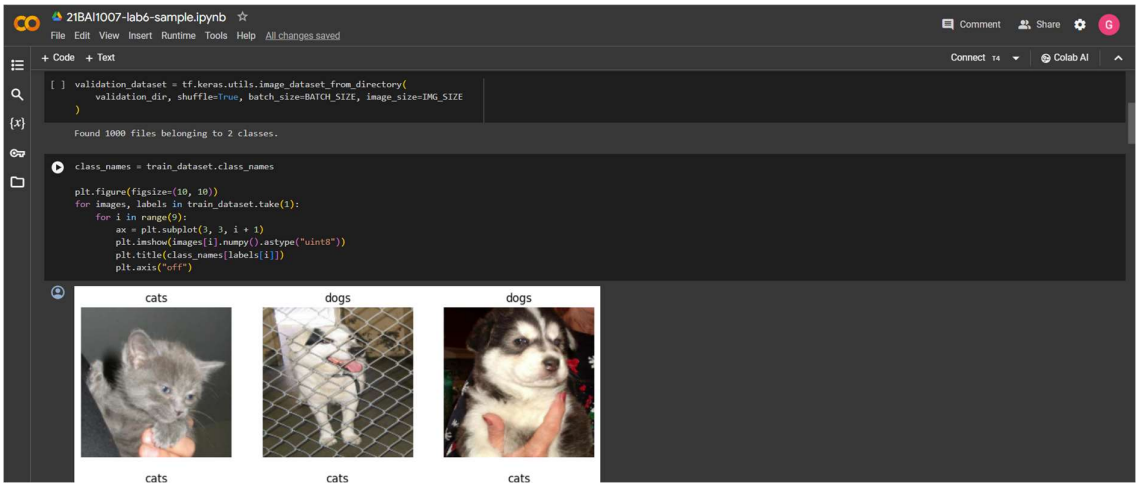
21BAI1007

Aim

1. Sample Network – Classifying the Cats v/s Dogs dataset using MobilenetV2 pretrained model.
2. Exploring the model structure and analysis for some common pretrained models in PyTorch – AlexNet, ConvNeXt, VGG16 and ResNet.
3. Implementation of transfer learning on classification of Flowers dataset using pretrained models VGG16, ResNet50 and ResNet152 and TensorFlow.

Code and Output for Aim 1 – Sample Network





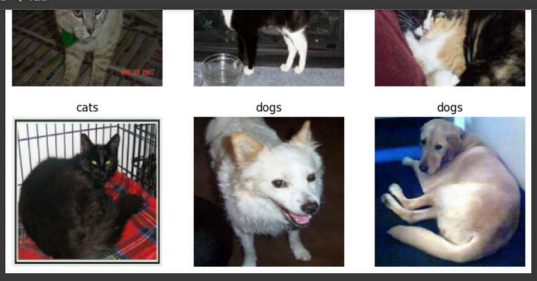
21BAI1007-lab6-sample.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share G

+ Code + Text

Connect T4 Colab AI




```
val_batches = tf.data.experimental.cardinality(validation_dataset)
test_dataset = validation_dataset.take(val_batches // 5)
validation_dataset = validation_dataset.skip(val_batches // 5)
```

```
print(
    "Number of validation batches: %d"
    % tf.data.experimental.cardinality(validation_dataset)
)
print("Number of test batches: %d" % tf.data.experimental.cardinality(test_dataset))
```

```
Number of validation batches: 26
Number of test batches: 6
```

21BAI1007-lab6-sample.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share G

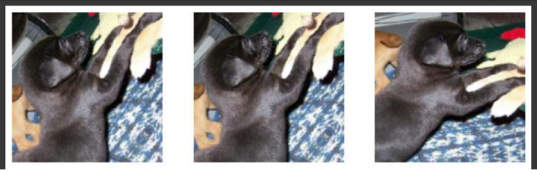
+ Code + Text

Connect T4 Colab AI

```
AUTOTUNE = tf.data.AUTOTUNE

train_dataset = train_dataset.prefetch(buffer_size=AUTOTUNE)
validation_dataset = validation_dataset.prefetch(buffer_size=AUTOTUNE)
test_dataset = test_dataset.prefetch(buffer_size=AUTOTUNE)
```

```
data_augmentation = tf.keras.Sequential(
    [
        tf.keras.layers.RandomFlip("horizontal"),
        tf.keras.layers.RandomRotation(0.2),
    ]
)
```

```
for image, _ in train_dataset.take(1):
    plt.figure(figsize=(10, 10))
    first_image = image[0]
    for i in range(9):
        ax = plt.subplot(3, 3, i + 1)
        augmented_image = data_augmentation(tf.expand_dims(first_image, 0))
        plt.imshow(augmented_image[0] / 255)
        plt.axis("off")
```



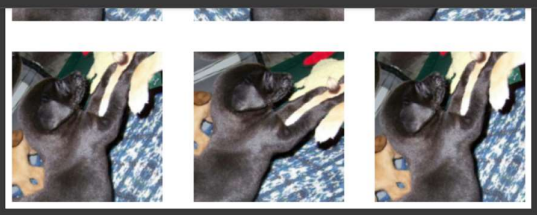
21BAI1007-lab6-sample.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share G

+ Code + Text

Connect T4 Colab AI



```
preprocess_input = tf.keras.applications.mobilenet_v2.preprocess_input
```

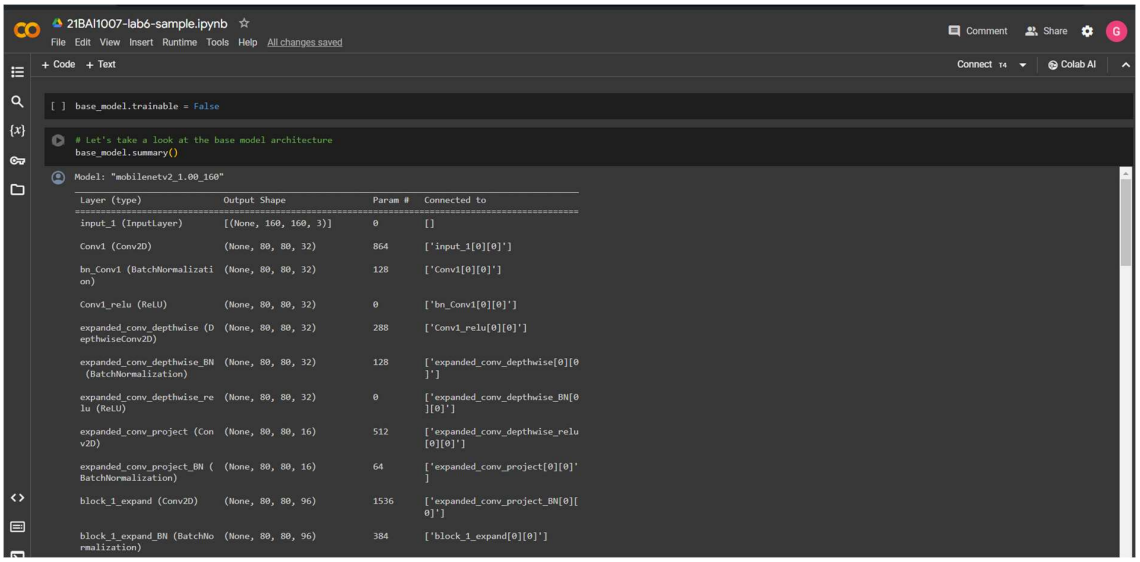
```
rescale = tf.keras.layers.Rescaling(1.0 / 127.5, offset=-1)
```

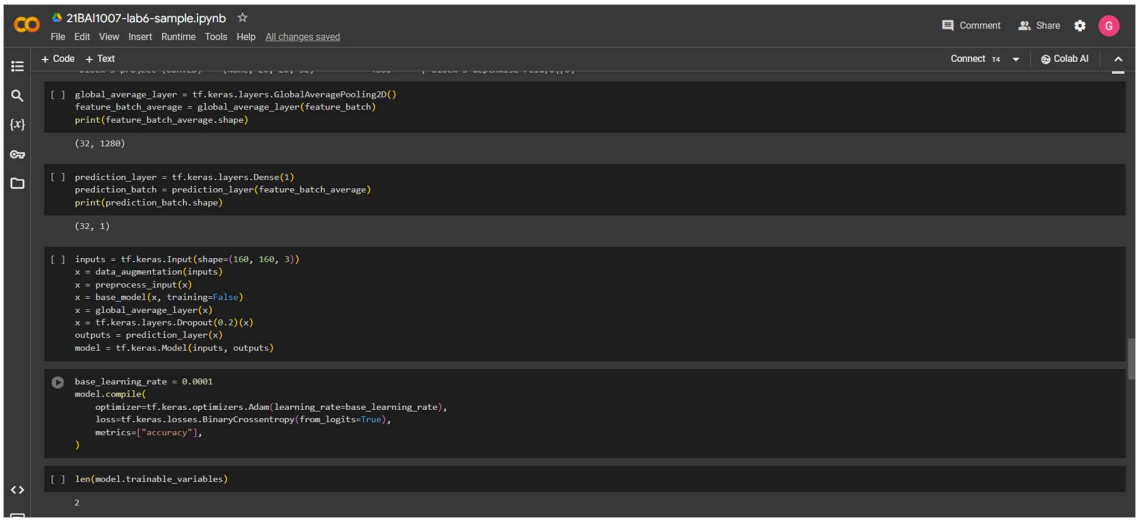
```
# Create the base model from the pre-trained model MobileNet V2
IMG_SHAPE = IMG_SIZE + (3,)
base_model = tf.keras.applications.MobileNetV2(
    input_shape=IMG_SHAPE, include_top=False, weights="imagenet"
)
```

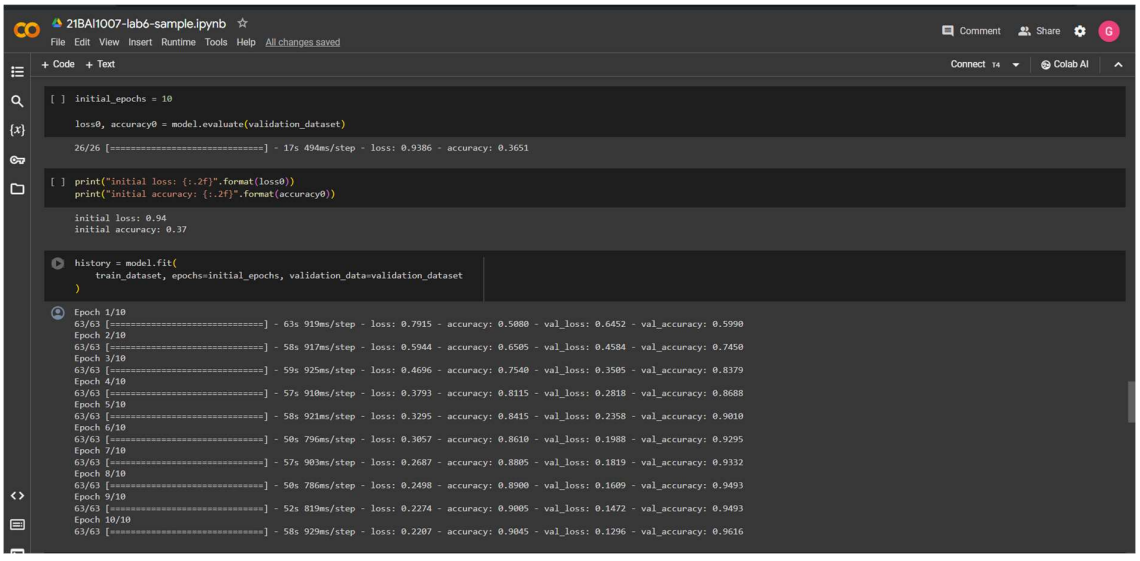
```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/mobilenet_v2/mobilenet_v2_weights_tf_dim_ordering_tf_kernels_1.0_160_no_top.h5
9406464/9406464 [==============================] - 0s 0us/step
```

```
image_batch, label_batch = next(iter(train_dataset))
feature_batch = base_model(image_batch)
print(feature_batch.shape)
```

```
(32, 5, 5, 1280)
```


21BAI1007-lab6-sample.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
+ Code + Text
Connect T4 Colab AI
[ ] base_model.trainable = False
# Let's take a look at the base model architecture
base_model.summary()
Model: "mobilenetv2_1.00_160"
Layer (type) Output Shape Param # Connected to
input_1 (InputLayer) [(None, 160, 160, 3)] 0 []
Conv1 (Conv2D) (None, 80, 80, 32) 864 ['input_1[0][0]']
bn_Conv1 (BatchNormalization) (None, 80, 80, 32) 128 ['Conv1[0][0]']
Conv1_relu (ReLU) (None, 80, 80, 32) 0 ['bn_Conv1[0][0]']
expanded_conv_depthwise (DepthwiseConv2D) (None, 80, 80, 32) 288 ['Conv1_relu[0][0]']
expanded_conv_depthwise_BN (BatchNormalization) (None, 80, 80, 32) 128 ['expanded_conv_depthwise[0][0]']
expanded_conv_depthwise_relu (ReLU) (None, 80, 80, 32) 0 ['expanded_conv_depthwise_BN[0][0]']
expanded_conv_project (Conv2D) (None, 80, 80, 16) 512 ['expanded_conv_depthwise_relu[0][0]']
expanded_conv_project_BN (BatchNormalization) (None, 80, 80, 16) 64 ['expanded_conv_project[0][0]']
block_1_expand (Conv2D) (None, 80, 80, 96) 1536 ['expanded_conv_project_BN[0][0]']
block_1_expand_BN (BatchNormalization) (None, 80, 80, 96) 384 ['block_1_expand[0][0]']


21BAI1007-lab6-sample.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
+ Code + Text
Connect T4 Colab AI
[ ] global_average_layer = tf.keras.layers.GlobalAveragePooling2D()
feature_batch_average = global_average_layer(feature_batch)
print(feature_batch_average.shape)
(32, 1280)
[ ] prediction_layer = tf.keras.layers.Dense(1)
prediction_batch = prediction_layer(feature_batch_average)
print(prediction_batch.shape)
(32, 1)
[ ] inputs = tf.keras.Input(shape=(160, 160, 3))
x = data_augmentation(inputs)
x = preprocess_input(x)
x = base_model(x, training=False)
x = global_average_layer(x)
x = tf.keras.layers.Dropout(0.2)(x)
outputs = prediction_layer(x)
model = tf.keras.Model(inputs, outputs)
base_learning_rate = 0.0001
model.compile(
optimizer=tf.keras.optimizers.Adam(learning_rate=base_learning_rate),
loss=tf.keras.losses.BinaryCrossentropy(from_logits=True),
metrics=["accuracy"],
)
[ ] len(model.trainable_variables)
2


21BAI1007-lab6-sample.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
+ Code + Text
Connect T4 Colab AI
[ ] initial_epochs = 10
loss0, accuracy0 = model.evaluate(validation_dataset)
26/26 [==============================] - 17s 494ms/step - loss: 0.9386 - accuracy: 0.3651
[ ] print("initial loss: {:.2f}".format(loss0))
print("initial accuracy: {:.2f}".format(accuracy0))
initial loss: 0.94
initial accuracy: 0.37
history = model.fit(
train_dataset, epochs=initial_epochs, validation_data=validation_dataset
)
Epoch 1/10
63/63 [==============================] - 63s 919ms/step - loss: 0.7915 - accuracy: 0.5080 - val_loss: 0.6452 - val_accuracy: 0.5990
Epoch 2/10
63/63 [==============================] - 58s 917ms/step - loss: 0.5944 - accuracy: 0.6505 - val_loss: 0.4584 - val_accuracy: 0.7450
Epoch 3/10
63/63 [==============================] - 59s 925ms/step - loss: 0.4696 - accuracy: 0.7540 - val_loss: 0.3505 - val_accuracy: 0.8379
Epoch 4/10
63/63 [==============================] - 57s 910ms/step - loss: 0.3793 - accuracy: 0.8115 - val_loss: 0.2818 - val_accuracy: 0.8688
Epoch 5/10
63/63 [==============================] - 58s 921ms/step - loss: 0.3295 - accuracy: 0.8415 - val_loss: 0.2358 - val_accuracy: 0.9010
Epoch 6/10
63/63 [==============================] - 50s 796ms/step - loss: 0.3057 - accuracy: 0.8610 - val_loss: 0.1988 - val_accuracy: 0.9295
Epoch 7/10
63/63 [==============================] - 57s 903ms/step - loss: 0.2687 - accuracy: 0.8805 - val_loss: 0.1819 - val_accuracy: 0.9332
Epoch 8/10
63/63 [==============================] - 50s 786ms/step - loss: 0.2498 - accuracy: 0.8900 - val_loss: 0.1609 - val_accuracy: 0.9493
Epoch 9/10
63/63 [==============================] - 52s 819ms/step - loss: 0.2274 - accuracy: 0.9005 - val_loss: 0.1472 - val_accuracy: 0.9493
Epoch 10/10
63/63 [==============================] - 58s 929ms/step - loss: 0.2207 - accuracy: 0.9045 - val_loss: 0.1296 - val_accuracy: 0.9616

21BAI1007-lab6-sample.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

Connect T4 Colab AI

```
[ ] base_model.trainable = True
```

```
[ ] model.compile(
        loss=tf.keras.losses.BinaryCrossentropy(from_logits=True),
        optimizer=tf.keras.optimizers.RMSprop(learning_rate=base_learning_rate / 10),
        metrics=["accuracy"],
    )
```

```
[ ] len(model.trainable_variables)
```

```
158
```

```
fine_tune_epochs = 10
total_epochs = initial_epochs + fine_tune_epochs

history_fine = model.fit(
    train_dataset,
    epochs=total_epochs,
    initial_epoch=history.epoch[-1],
    validation_data=validation_dataset,
)
```

```
Epoch 10/20
63/63 [==============================] - 160s 2s/step - loss: 0.1525 - accuracy: 0.9355 - val_loss: 0.0495 - val_accuracy: 0.9864
Epoch 11/20
63/63 [==============================] - 145s 2s/step - loss: 0.1063 - accuracy: 0.9560 - val_loss: 0.0507 - val_accuracy: 0.9777
Epoch 12/20
63/63 [==============================] - 144s 2s/step - loss: 0.0952 - accuracy: 0.9650 - val_loss: 0.0493 - val_accuracy: 0.9802
Epoch 13/20
63/63 [==============================] - 145s 2s/step - loss: 0.0925 - accuracy: 0.9585 - val_loss: 0.0395 - val_accuracy: 0.9839
Epoch 14/20
63/63 [==============================] - 152s 2s/step - loss: 0.0741 - accuracy: 0.9745 - val_loss: 0.0415 - val_accuracy: 0.9851
Epoch 15/20
63/63 [==============================] - 143s 2s/step - loss: 0.0774 - accuracy: 0.9690 - val_loss: 0.0418 - val_accuracy: 0.9827
Epoch 16/20
63/63 [==============================] - 136s 2s/step - loss: 0.0749 - accuracy: 0.9680 - val_loss: 0.0378 - val_accuracy: 0.9814
Epoch 17/20
```

21BAI1007-lab6-sample.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

Connect T4 Colab AI

```
# Retrieve a batch of images from the test set
image_batch, label_batch = test_dataset.as_numpy_iterator().next()
predictions = model.predict_on_batch(image_batch).flatten()

# Apply a sigmoid since our model returns logits
predictions = tf.nn.sigmoid(predictions)
predictions = tf.where(predictions < 0.5, 0, 1)

print("Predictions:\n", predictions.numpy())
print("Labels:\n", label_batch)

plt.figure(figsize=(10, 10))
for i in range(9):
    ax = plt.subplot(3, 3, i + 1)
    plt.imshow(image_batch[i].astype("uint8"))
    plt.title(class_names[predictions[i]])
    plt.axis("off")
```

```
Predictions:
 [0 1 1 1 1 0 0 0 1 1 1 0 0 0 0 0 0 0 0 1 0 1 1 0 1 0 1 1 1 0 0 1]
Labels:
 [0 1 1 1 1 0 0 0 0 1 1 0 0 0 0 0 0 0 1 1 0 1 1 0 0 0 1 1 1 0 0 1]
```

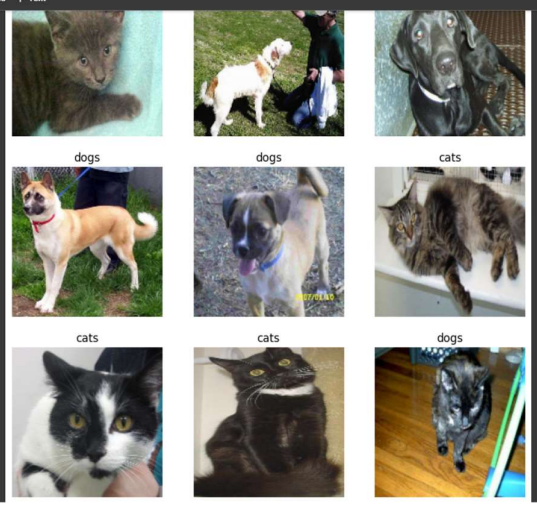
21BAI1007-lab6-sample.ipynb

File Edit View Insert Runtime Tools Help All changes saved

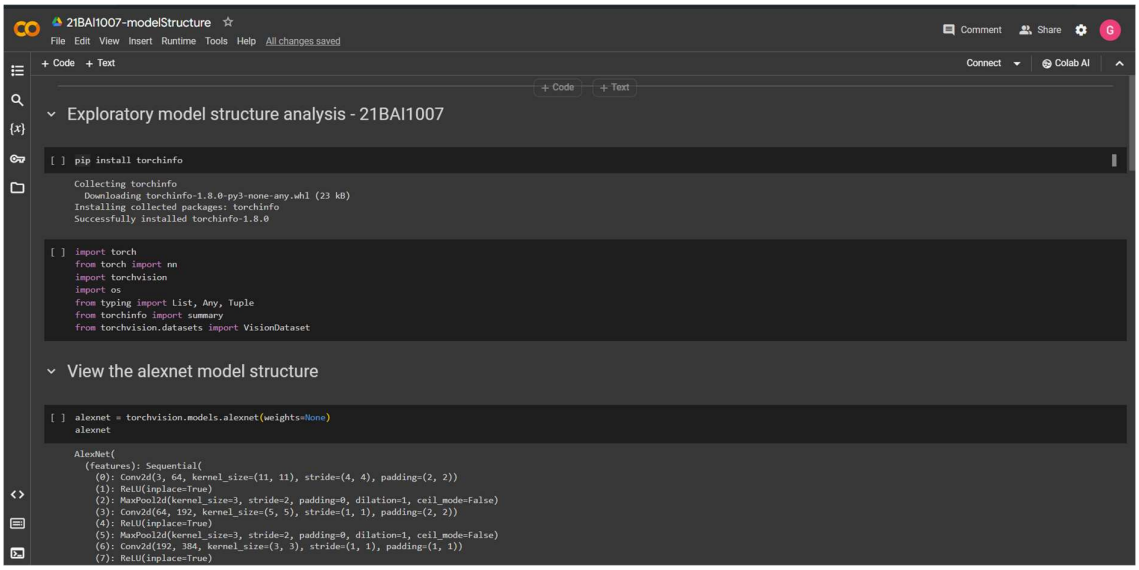
Comment Share

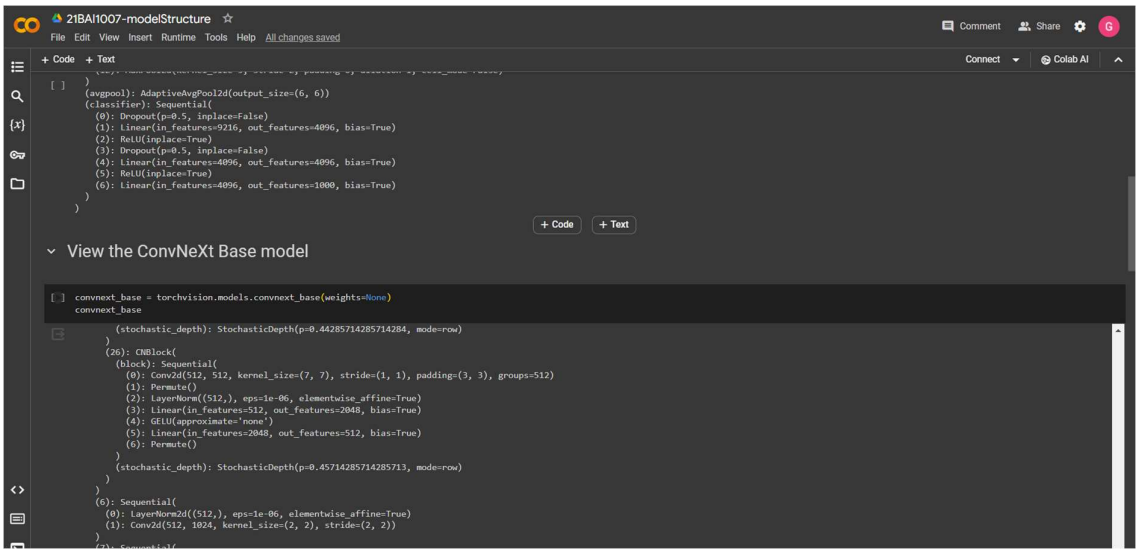
+ Code + Text

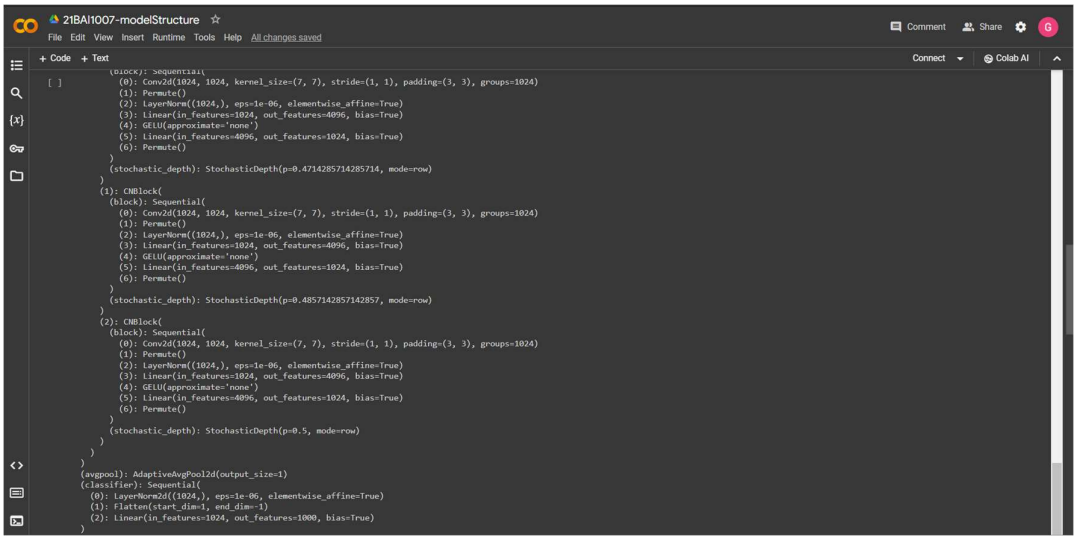
Connect T4 Colab AI

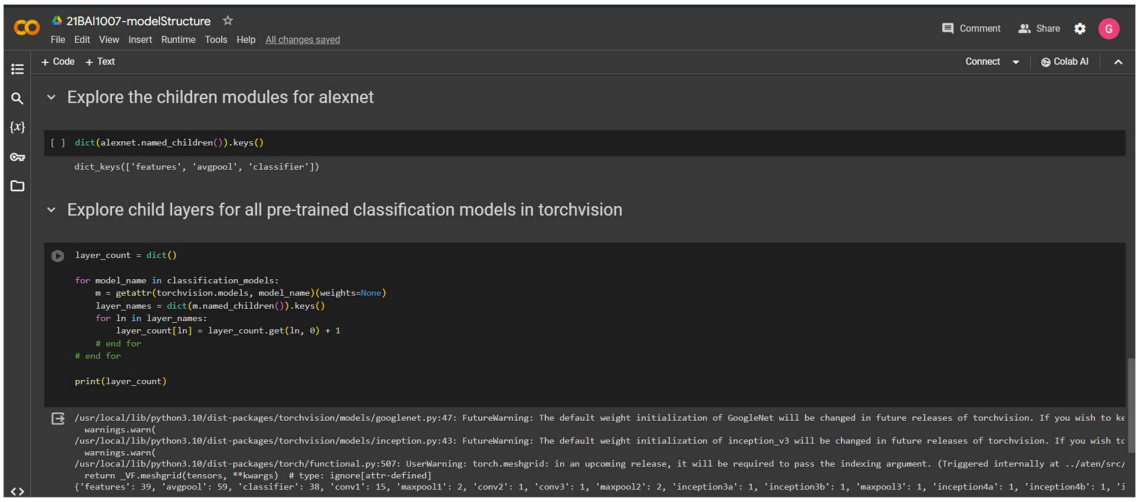


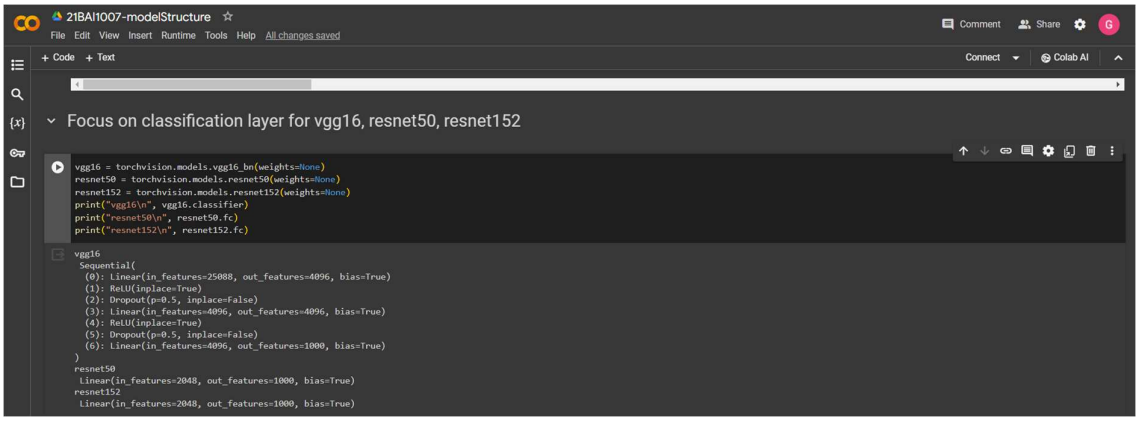
Code and Output for Aim 2 – Model Structure Analysis



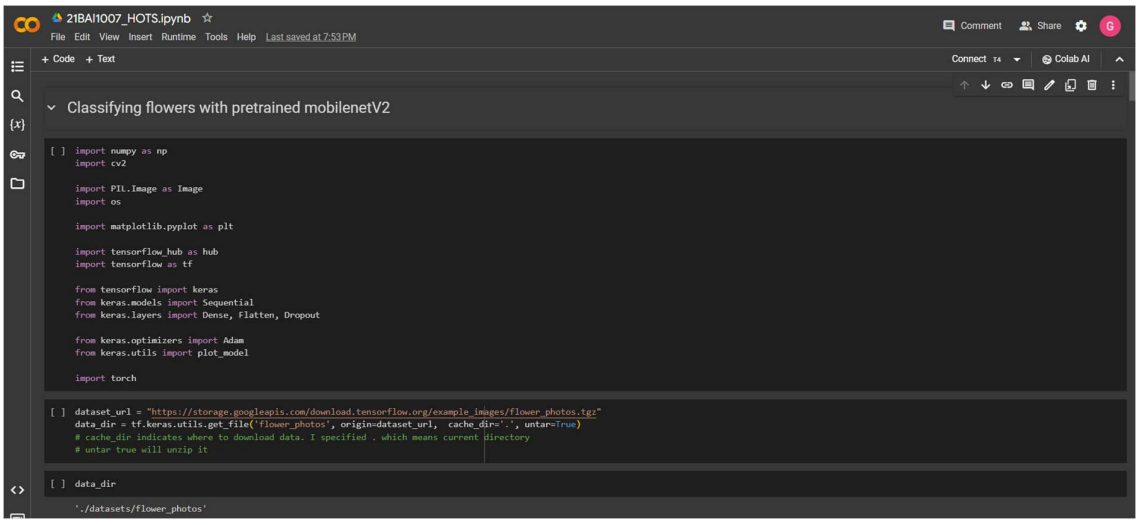








Code and Output for Aim 3 – VGG16, ResNet50, ResNet152



21BAI1007_HOTS.ipynb

File Edit View Insert Runtime Tools Help Last saved at 7:53 PM

Comment Share G

+ Code + Text

Connect T4 Colab AI

```
[ ] import pathlib
    data_dir = pathlib.Path(data_dir)
    data_dir
```

```
PosixPath('datasets/flower_photos')
```

```
img_height , img_width = 180, 180
batch_size = 32

train_ds = tf.keras.preprocessing.image_dataset_from_directory(
    data_dir,
    label_mode = 'categorical',
    validation_split = 0.2,
    subset = 'training',
    seed = 123,
    image_size = (img_height, img_width),
    batch_size = batch_size
)
```

```
Found 3670 files belonging to 5 classes.
Using 2936 files for training.
```

+ Code + Text

```
[ ] # Preprocessing and load the validation data
val_ds = tf.keras.preprocessing.image_dataset_from_directory(
    data_dir,
    label_mode ='categorical',
    validation_split =0.2,
    subset= 'validation',
    seed =123,
    image_size=(img_height,img_width),
    batch_size=batch_size
)
```

```
Found 3670 files belonging to 5 classes.
Using 734 files for validation.
```

21BAI1007_HOTS.ipynb

File Edit View Insert Runtime Tools Help Last saved at 7:53 PM

Comment Share G

+ Code + Text

Connect T4 Colab AI

```
[ ] ## Class names list that sotres the class names of the flowers dataset
    class_names = ['daisy', 'dandelion', 'roses', 'sunflowers', 'tulips']

    print(class_names)
```

```
['daisy', 'dandelion', 'roses', 'sunflowers', 'tulips']
```

```
import matplotlib.pyplot as plt
plt.figure(figsize=(10,10))
for images,labels in train_ds.take(1):
  for i in range(6):
    ax = plt.subplot(3,3,i+1)
    plt.imshow(images[i].numpy().astype('uint8'))
    plt.axis('off')
```

21BAI1007_HOTS.ipynb

File Edit View Insert Runtime Tools Help Last saved at 7:53 PM

Comment Share G

+ Code + Text

Connect T4 Colab AI

```
[ ]
```

Resnet50 Model

```
def resnet50_model(in_shape, num_classes):

    ###Importing the model
    resnet_model = Sequential()
    pretrained_model = keras.applications.ResNet50(include_top=False, input_shape=in_shape, pooling='avg', classes=num_classes, weights='imagenet')

    for layer in pretrained_model.layers:
      layer.trainable = False
    resnet_model.add(pretrained_model)

    # Add dense layers for classification
    resnet_model.add(Dense(512, activation='relu'))
    resnet_model.add(Dropout(0.6))
    resnet_model.add(Dense(256, activation='relu'))
    resnet_model.add(Dense(5, activation='softmax'))

    resnet_model.compile(loss='categorical_crossentropy', optimizer=Adam(learning_rate=0.001), metrics='accuracy')

    return resnet_model
```

21BAI1007_HOTS.ipynb

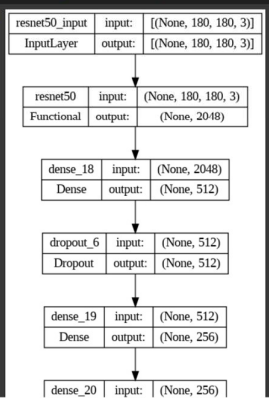
File Edit View Insert Runtime Tools Help Last saved at 7:53 PM

Comment Share G

+ Code + Text

Connect T4 Colab AI

```
input_shape = (180, 180, 3)
num_classes = 5

resnet50 = resnet50_model(input_shape, num_classes)

plot_model(resnet50, show_shapes=True, show_layer_names=True)
```



21BAI1007_HOTS.ipynb

File Edit View Insert Runtime Tools Help Last saved at 7:53 PM

Comment Share G

+ Code + Text

Connect T4 Colab AI

```
[ ] # Train upto 10 epochs
    # Fit the model in a variable called history for further analysis
    history = resnet50.fit(train_ds, validation_data=val_ds, epochs=10,)
```

```
Epoch 1/10
92/92 [==============================] - 14s 111ms/step - loss: 0.9398 - accuracy: 0.6846 - val_loss: 0.4361 - val_accuracy: 0.8406
Epoch 2/10
92/92 [==============================] - 9s 96ms/step - loss: 0.5409 - accuracy: 0.7997 - val_loss: 0.3913 - val_accuracy: 0.8420
Epoch 3/10
92/92 [==============================] - 12s 129ms/step - loss: 0.4702 - accuracy: 0.8317 - val_loss: 0.4308 - val_accuracy: 0.8365
Epoch 4/10
92/92 [==============================] - 9s 98ms/step - loss: 0.3930 - accuracy: 0.8573 - val_loss: 0.3921 - val_accuracy: 0.8515
Epoch 5/10
92/92 [==============================] - 10s 101ms/step - loss: 0.3908 - accuracy: 0.8525 - val_loss: 0.3453 - val_accuracy: 0.8706
Epoch 6/10
92/92 [==============================] - 9s 93ms/step - loss: 0.3499 - accuracy: 0.8702 - val_loss: 0.3476 - val_accuracy: 0.8760
Epoch 7/10
92/92 [==============================] - 9s 98ms/step - loss: 0.3322 - accuracy: 0.8798 - val_loss: 0.3315 - val_accuracy: 0.8719
Epoch 8/10
92/92 [==============================] - 9s 97ms/step - loss: 0.2613 - accuracy: 0.9026 - val_loss: 0.3165 - val_accuracy: 0.8719
Epoch 9/10
92/92 [==============================] - 9s 99ms/step - loss: 0.2636 - accuracy: 0.9040 - val_loss: 0.3281 - val_accuracy: 0.8896
Epoch 10/10
92/92 [==============================] - 9s 91ms/step - loss: 0.2397 - accuracy: 0.9108 - val_loss: 0.3458 - val_accuracy: 0.8774
```

21BAI1007_HOTS.ipynb

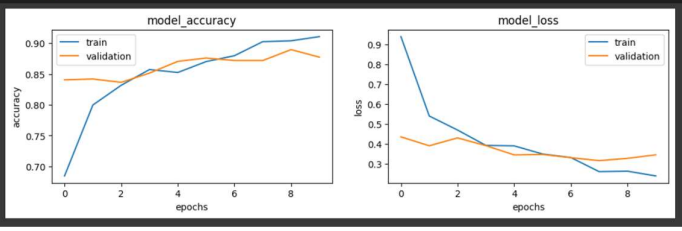
File Edit View Insert Runtime Tools Help Last saved at 7:53 PM

Comment Share G

+ Code + Text

Connect T4 Colab AI

```
plt.figure(1, figsize=(12, 3))

plt.subplot(121)
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('model_accuracy')
plt.ylabel('accuracy')
plt.xlabel('epochs')
plt.legend(['train', 'validation'])


plt.subplot(122)
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model_loss')
plt.ylabel('loss')
plt.xlabel('epochs')
plt.legend(['train', 'validation'])

plt.show()
```



21BAI1007_HOTS.ipynb

File Edit View Insert Runtime Tools Help Last saved at 7:53 PM

Comment Share G

+ Code + Text

Connect T4 Colab AI

VGG16 Model

```
def vgg16_model(in_shape, num_classes):

    ###Importing the model
    vgg16 = Sequential()
    pretrained_model = keras.applications.vgg16.VGG16(include_top=False, input_shape=in_shape, pooling='avg', classes=num_classes, weights='imagenet')

    for layer in pretrained_model.layers:
      layer.trainable = False
    vgg16.add(pretrained_model)

    # Add dense layers for classification
    vgg16.add(Dense(512, activation='relu'))
    vgg16.add(Dropout(0.6))
    vgg16.add(Dense(256, activation='relu'))
    vgg16.add(Dense(5, activation='softmax'))

    vgg16.compile(loss='categorical_crossentropy', optimizer=Adam(learning_rate=0.001), metrics='accuracy')

    return vgg16
```

```
input_shape = (180, 180, 3)
num_classes = 5

vgg16 = vgg16_model(input_shape, num_classes)

plot_model(vgg16, show_shapes=True, show_layer_names=True)
```

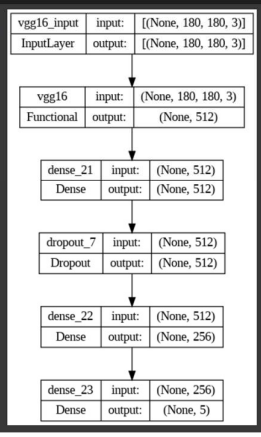
21BAI1007_HOTS.ipynb

File Edit View Insert Runtime Tools Help Last saved at 7:53 PM

Comment Share G

+ Code + Text

Connect T4 Colab AI



21BAI1007_HOTS.ipynb

File Edit View Insert Runtime Tools Help Last saved at 7:53 PM

Comment Share G

+ Code + Text

Connect T4 Colab AI

```
[ ] history2 = vgg16.fit(train_ds, validation_data=val_ds, epochs=10,)

    Epoch 1/10
    92/92 [==============================] - 14s 124ms/step - loss: 2.2218 - accuracy: 0.6226 - val_loss: 0.6371 - val_accuracy: 0.8052
    Epoch 2/10
    92/92 [==============================] - 12s 124ms/step - loss: 0.8778 - accuracy: 0.7599 - val_loss: 0.5077 - val_accuracy: 0.8229
    Epoch 3/10
    92/92 [==============================] - 12s 125ms/step - loss: 0.6729 - accuracy: 0.7909 - val_loss: 0.4966 - val_accuracy: 0.8270
    Epoch 4/10
    92/92 [==============================] - 12s 125ms/step - loss: 0.5422 - accuracy: 0.8232 - val_loss: 0.4888 - val_accuracy: 0.8297
    Epoch 5/10
    92/92 [==============================] - 12s 128ms/step - loss: 0.4566 - accuracy: 0.8454 - val_loss: 0.4512 - val_accuracy: 0.8447
    Epoch 6/10
    92/92 [==============================] - 12s 126ms/step - loss: 0.3883 - accuracy: 0.8634 - val_loss: 0.4455 - val_accuracy: 0.8447
    Epoch 7/10
    92/92 [==============================] - 12s 129ms/step - loss: 0.3657 - accuracy: 0.8781 - val_loss: 0.4714 - val_accuracy: 0.8297
    Epoch 8/10
    92/92 [==============================] - 12s 128ms/step - loss: 0.3074 - accuracy: 0.8873 - val_loss: 0.4607 - val_accuracy: 0.8406
    Epoch 9/10
    92/92 [==============================] - 12s 126ms/step - loss: 0.2860 - accuracy: 0.8988 - val_loss: 0.5120 - val_accuracy: 0.8365
    Epoch 10/10
    92/92 [==============================] - 12s 123ms/step - loss: 0.2709 - accuracy: 0.9060 - val_loss: 0.4950 - val_accuracy: 0.8460
```

21BAI1007_HOTS.ipynb

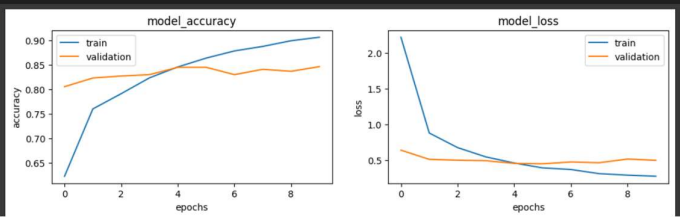
File Edit View Insert Runtime Tools Help Last saved at 7:53 PM

Comment Share

+ Code + Text

Connect Colab AI

```
plt.figure(1, figsize=(12, 3))

plt.subplot(121)
plt.plot(history2.history['accuracy'])
plt.plot(history2.history['val_accuracy'])
plt.title('model_accuracy')
plt.ylabel('accuracy')
plt.xlabel('epochs')
plt.legend(['train', 'validation'])


plt.subplot(122)
plt.plot(history2.history['loss'])
plt.plot(history2.history['val_loss'])
plt.title('model_loss')
plt.ylabel('loss')
plt.xlabel('epochs')
plt.legend(['train', 'validation'])

plt.show()
```



21BAI1007_HOTS.ipynb

File Edit View Insert Runtime Tools Help Last saved at 7:53 PM

Comment Share

+ Code + Text

Connect Colab AI

Resnet 152 Model

```
def resnet152_model(in_shape, num_classes):

    ###Importing the model
    resnet152 = Sequential()
    pretrained_model = keras.applications.ResNet152(include_top=False, input_shape=in_shape, pooling='avg', classes=num_classes, weights='imagenet')

    for layer in pretrained_model.layers:
      layer.trainable = False
    resnet152.add(pretrained_model)

    # Add dense layers for classification
    resnet152.add(Dense(512, activation='relu'))
    resnet152.add(Dropout(0.6))
    resnet152.add(Dense(256, activation='relu'))
    resnet152.add(Dense(5, activation='softmax'))

    resnet152.compile(loss='categorical_crossentropy', optimizer=Adam(learning_rate=0.001), metrics='accuracy')

    return resnet152
```

```
[ ] input_shape = (180, 180, 3)
    num_classes = 5

    resnet152 = resnet152_model(input_shape, num_classes)

    plot_model(resnet152, show_shapes=True, show_layer_names=True)
```

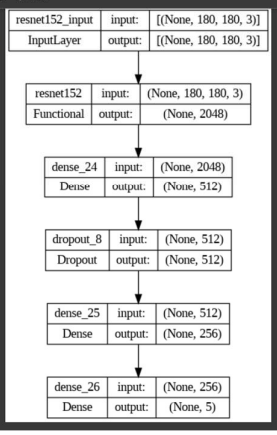
21BAI1007_HOTS.ipynb

File Edit View Insert Runtime Tools Help Last saved at 7:53 PM

Comment Share

+ Code + Text

Connect Colab AI



21BAI1007_HOTS.ipynb

File Edit View Insert Runtime Tools Help Last saved at 7:53 PM

Comment Share

+ Code + Text

Connect T4 Colab AI

```
history3 = resnet152.fit(train_ds, validation_data=val_ds, epochs=10,)
```

```
Epoch 1/10
92/92 [==============================] - 36s 292ms/step - loss: 0.9514 - accuracy: 0.6863 - val_loss: 0.4774 - val_accuracy: 0.8365
Epoch 2/10
92/92 [==============================] - 21s 229ms/step - loss: 0.5539 - accuracy: 0.8014 - val_loss: 0.3609 - val_accuracy: 0.8706
Epoch 3/10
92/92 [==============================] - 21s 220ms/step - loss: 0.4403 - accuracy: 0.8403 - val_loss: 0.3513 - val_accuracy: 0.8706
Epoch 4/10
92/92 [==============================] - 22s 231ms/step - loss: 0.4102 - accuracy: 0.8467 - val_loss: 0.3628 - val_accuracy: 0.8706
Epoch 5/10
92/92 [==============================] - 21s 220ms/step - loss: 0.3651 - accuracy: 0.8668 - val_loss: 0.3267 - val_accuracy: 0.8719
Epoch 6/10
92/92 [==============================] - 21s 219ms/step - loss: 0.3149 - accuracy: 0.8866 - val_loss: 0.3349 - val_accuracy: 0.8883
Epoch 7/10
92/92 [==============================] - 21s 225ms/step - loss: 0.2925 - accuracy: 0.8900 - val_loss: 0.3344 - val_accuracy: 0.8815
Epoch 8/10
92/92 [==============================] - 20s 218ms/step - loss: 0.3138 - accuracy: 0.8859 - val_loss: 0.3512 - val_accuracy: 0.8665
Epoch 9/10
92/92 [==============================] - 21s 225ms/step - loss: 0.2732 - accuracy: 0.9009 - val_loss: 0.3289 - val_accuracy: 0.8842
Epoch 10/10
92/92 [==============================] - 21s 230ms/step - loss: 0.2307 - accuracy: 0.9172 - val_loss: 0.3386 - val_accuracy: 0.8828
```

21BAI1007_HOTS.ipynb

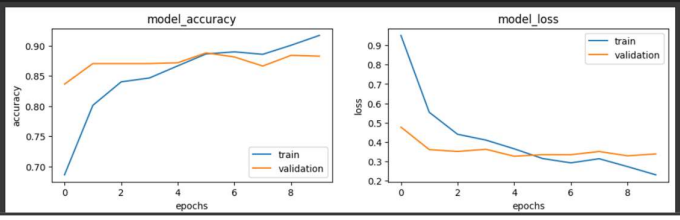
File Edit View Insert Runtime Tools Help Last saved at 7:53 PM

Comment Share

+ Code + Text

Connect T4 Colab AI

```
plt.figure(1, figsize=(12, 3))

plt.subplot(121)
plt.plot(history3.history['accuracy'])
plt.plot(history3.history['val_accuracy'])
plt.title('model_accuracy')
plt.ylabel('accuracy')
plt.xlabel('epochs')
plt.legend(['train', 'validation'])


plt.subplot(122)
plt.plot(history3.history['loss'])
plt.plot(history3.history['val_loss'])
plt.title('model_loss')
plt.ylabel('loss')
plt.xlabel('epochs')
plt.legend(['train', 'validation'])

plt.show()
```



21BAI1007_HOTS.ipynb

File Edit View Insert Runtime Tools Help Last saved at 7:53 PM

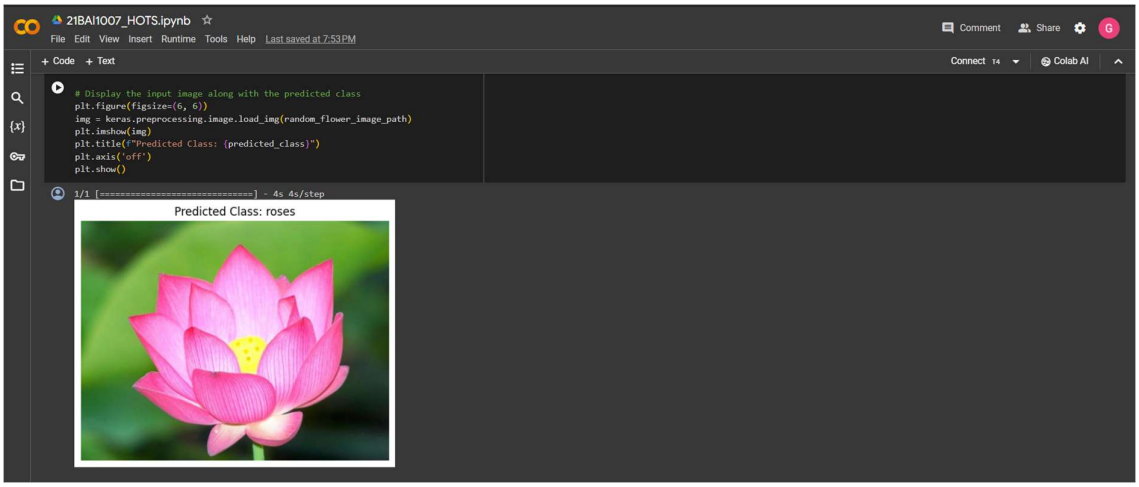
Comment Share

+ Code + Text

Connect T4 Colab AI

Predicting an image using the best model - Resnet152

```
# Function to preprocess the input image
def preprocess_image(image_path, target_size=(180, 180)):
    img = keras.preprocessing.image.load_img(image_path, target_size=target_size)
    img_array = keras.preprocessing.image.img_to_array(img)
    img_array = tf.expand_dims(img_array, 0)  # Create a batch
    return img_array

# Function to make predictions using the model
def predict_flower(image_path, model):
    # Preprocess the input image
    img = preprocess_image(image_path)

    # Make predictions
    predictions = model.predict(img)

    # Decode predictions
    predicted_class = class_names[np.argmax(predictions)]

    return predicted_class

# Path to the random flower image
random_flower_image_path = "test_images/flower_test.jpg"  # Replace with the path to your image

# Predict the flower class
predicted_class = predict_flower(random_flower_image_path, resnet152)

# Display the input image along with the predicted class
plt.figure(figsize=(6, 6))
img = keras.preprocessing.image.load_img(random_flower_image_path)
plt.imshow(img)
plt.title(f"Predicted Class: {predicted_class}")
plt.axis('off')
plt.show()
```



Comparison between ResNet50, ResNet150 and VGG16 pretrained models

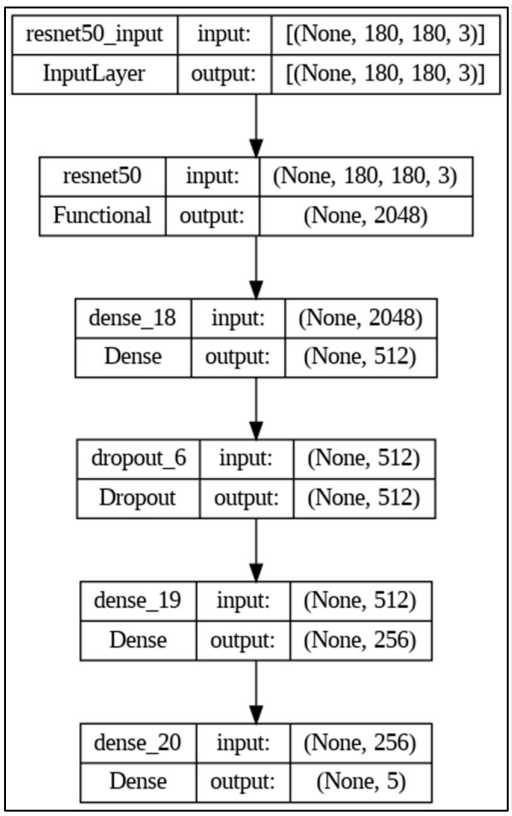


Fig: ResNet50

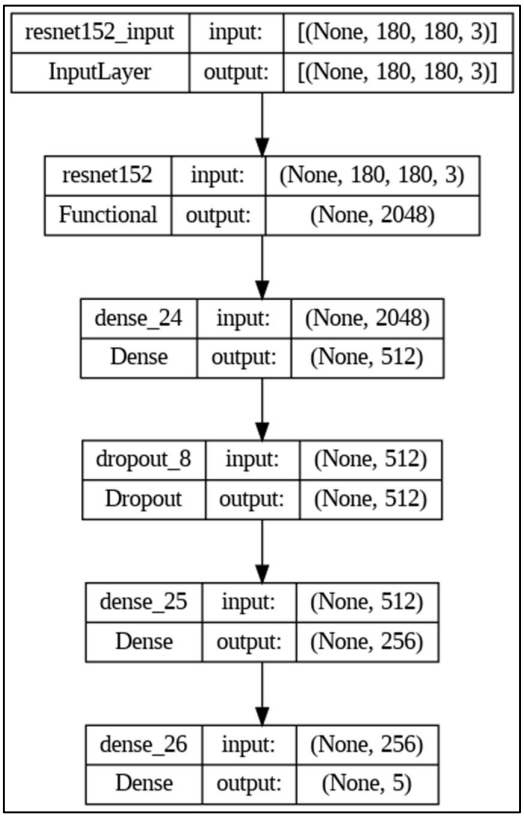


Fig: ResNet152

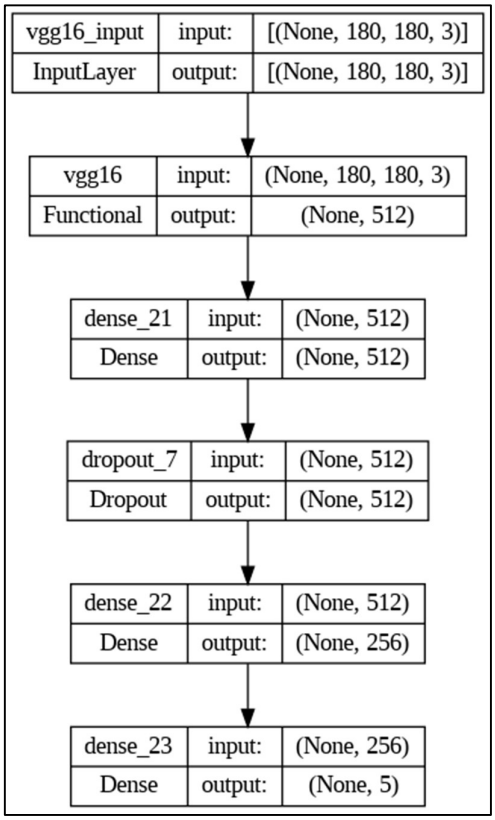


Fig: VGG16

Comparison based on training performance on Flowers Dataset

Model	VGG16	ResNet50	ResNet152
Training Loss	0.2709	0.2397	0.2307
Training Accuracy	0.9060	0.9108	0.9172
Validation Loss	0.4950	0.3458	0.3386
Validation Accuracy	0.8460	0.8774	0.8828

Results

Hence by comparing the three models based on its performance on classifying the Flowers dataset, ResNet152 has performed the best with an accuracy of 88.28% compared to 87.74% for ResNet50 and 84.60% for VGG16.