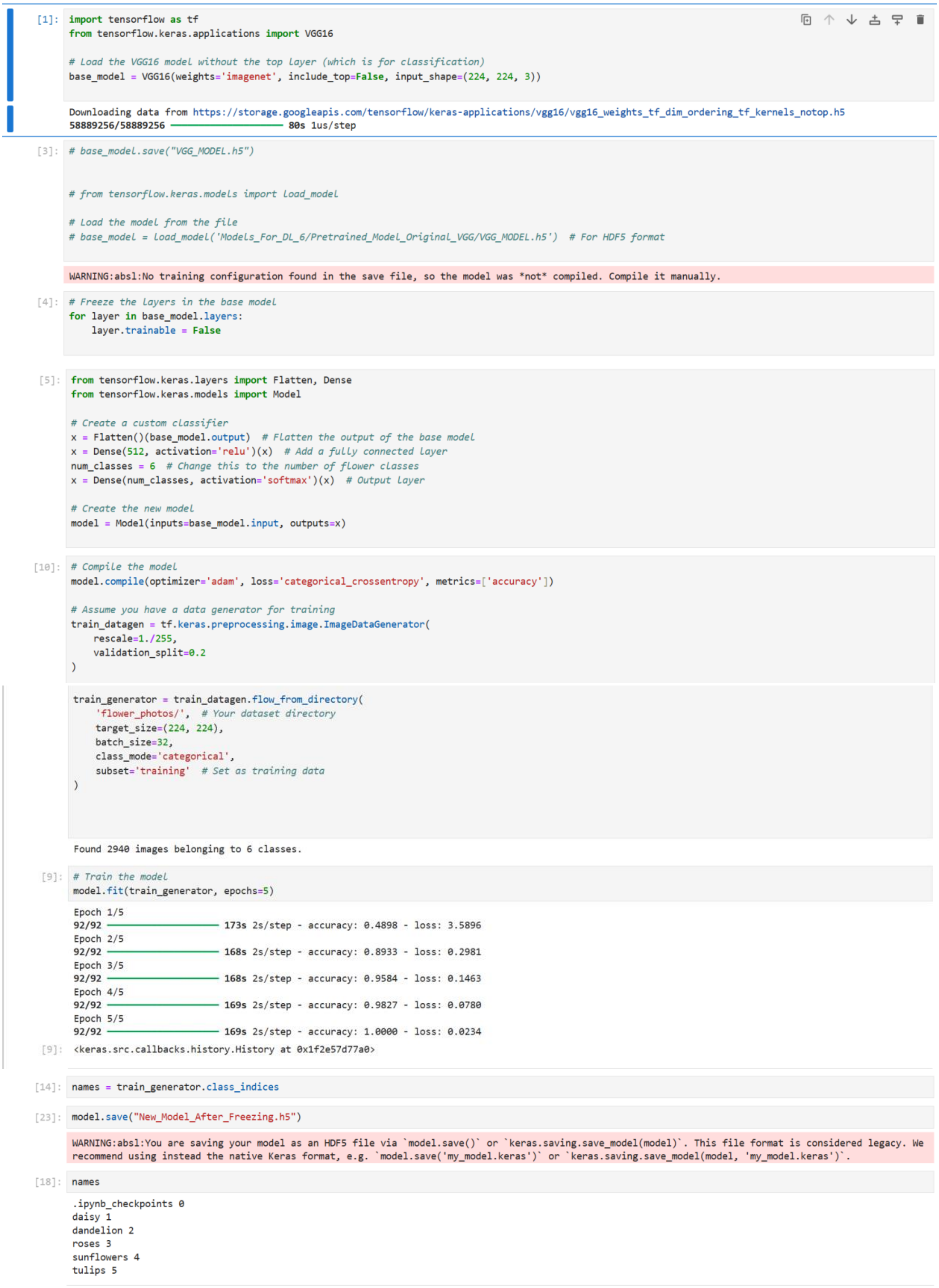


```
[1]: import tensorflow as tf
     from tensorflow.keras.applications import VGG16

     # Load the VGG16 model without the top layer (which is for classification)
     base_model = VGG16(weights='imagenet', include_top=False, input_shape=(224, 224, 3))
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg16/vgg16_weights_tf_dim_ordering_tf_kernels_notop.h5
58889256/58889256 ━━━━━━━━━━━━━━━━━━━━ 80s 1us/step
```

```
[3]: # base_model.save("VGG_MODEL.h5")


     # from tensorflow.keras.models import load_model

     # Load the model from the file
     # base_model = load_model('Models_For_DL_6/Pretrained_Model_Original_VGG/VGG_MODEL.h5')  # For HDF5 format
```

```
WARNING:absl:No training configuration found in the save file, so the model was *not* compiled. Compile it manually.
```

```
[4]: # Freeze the layers in the base model
     for layer in base_model.layers:
         layer.trainable = False
```

```
[5]: from tensorflow.keras.layers import Flatten, Dense
     from tensorflow.keras.models import Model

     # Create a custom classifier
     x = Flatten()(base_model.output)  # Flatten the output of the base model
     x = Dense(512, activation='relu')(x)  # Add a fully connected layer
     num_classes = 6  # Change this to the number of flower classes
     x = Dense(num_classes, activation='softmax')(x)  # Output layer

     # Create the new model
     model = Model(inputs=base_model.input, outputs=x)
```

```
[10]: # Compile the model
      model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

      # Assume you have a data generator for training
      train_datagen = tf.keras.preprocessing.image.ImageDataGenerator(
          rescale=1./255,
          validation_split=0.2
      )
```

```
      train_generator = train_datagen.flow_from_directory(
          'flower_photos/',  # Your dataset directory
          target_size=(224, 224),
          batch_size=32,
          class_mode='categorical',
          subset='training'  # Set as training data
      )
```

```
Found 2940 images belonging to 6 classes.
```

```
[9]: # Train the model
     model.fit(train_generator, epochs=5)
```

```
Epoch 1/5
92/92 ━━━━━━━━━━━━━━━━━━━━ 173s 2s/step - accuracy: 0.4898 - loss: 3.5896
Epoch 2/5
92/92 ━━━━━━━━━━━━━━━━━━━━ 168s 2s/step - accuracy: 0.8933 - loss: 0.2981
Epoch 3/5
92/92 ━━━━━━━━━━━━━━━━━━━━ 168s 2s/step - accuracy: 0.9584 - loss: 0.1463
Epoch 4/5
92/92 ━━━━━━━━━━━━━━━━━━━━ 169s 2s/step - accuracy: 0.9827 - loss: 0.0780
Epoch 5/5
92/92 ━━━━━━━━━━━━━━━━━━━━ 169s 2s/step - accuracy: 1.0000 - loss: 0.0234
[9]: <keras.src.callbacks.history.History at 0x1f2e57d77a0>
```

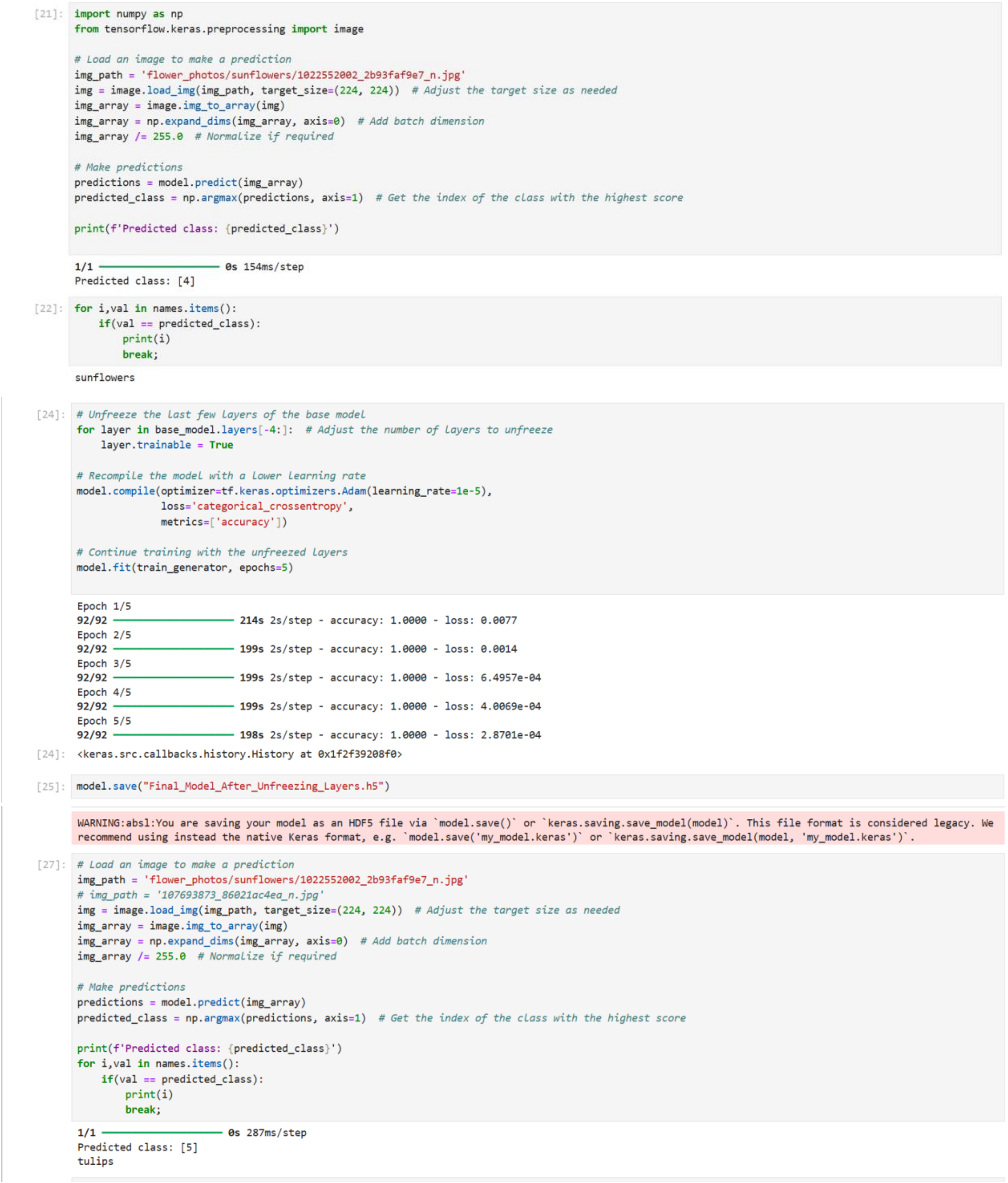
```
[14]: names = train_generator.class_indices
```

```
[23]: model.save("New_Model_After_Freezing.h5")
```

```
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file format is considered legacy. We
recommend using instead the native Keras format, e.g. `model.save('my_model.keras')` or `keras.saving.save_model(model, 'my_model.keras')`.
```

```
[18]: names
```

```
.ipynb_checkpoints 0
daisy 1
dandelion 2
roses 3
sunflowers 4
tulips 5
```



```
[21]: import numpy as np
      from tensorflow.keras.preprocessing import image

      # Load an image to make a prediction
      img_path = 'flower_photos/sunflowers/1022552002_2b93faf9e7_n.jpg'
      img = image.load_img(img_path, target_size=(224, 224))  # Adjust the target size as needed
      img_array = image.img_to_array(img)
      img_array = np.expand_dims(img_array, axis=0)  # Add batch dimension
      img_array /= 255.0  # Normalize if required

      # Make predictions
      predictions = model.predict(img_array)
      predicted_class = np.argmax(predictions, axis=1)  # Get the index of the class with the highest score

      print(f'Predicted class: {predicted_class}')
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 154ms/step
Predicted class: [4]
```

```
[22]: for i,val in names.items():
          if(val == predicted_class):
              print(i)
              break;
```

```
sunflowers
```

```
[24]: # Unfreeze the last few layers of the base model
      for layer in base_model.layers[-4:]:  # Adjust the number of layers to unfreeze
          layer.trainable = True

      # Recompile the model with a lower learning rate
      model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=1e-5),
                    loss='categorical_crossentropy',
                    metrics=['accuracy'])

      # Continue training with the unfreezed layers
      model.fit(train_generator, epochs=5)
```

```
Epoch 1/5
92/92 ━━━━━━━━━━━━━━━━━━━━ 214s 2s/step - accuracy: 1.0000 - loss: 0.0077
Epoch 2/5
92/92 ━━━━━━━━━━━━━━━━━━━━ 199s 2s/step - accuracy: 1.0000 - loss: 0.0014
Epoch 3/5
92/92 ━━━━━━━━━━━━━━━━━━━━ 199s 2s/step - accuracy: 1.0000 - loss: 6.4957e-04
Epoch 4/5
92/92 ━━━━━━━━━━━━━━━━━━━━ 199s 2s/step - accuracy: 1.0000 - loss: 4.0069e-04
Epoch 5/5
92/92 ━━━━━━━━━━━━━━━━━━━━ 198s 2s/step - accuracy: 1.0000 - loss: 2.8701e-04
[24]: <keras.src.callbacks.history.History at 0x1f2f39208f0>
```

```
[25]: model.save("Final_Model_After_Unfreezing_Layers.h5")
```

```
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file format is considered legacy. We recommend using instead the native Keras format, e.g. `model.save('my_model.keras')` or `keras.saving.save_model(model, 'my_model.keras')`.
```

```
[27]: # Load an image to make a prediction
      img_path = 'flower_photos/sunflowers/1022552002_2b93faf9e7_n.jpg'
      # img_path = '107693873_86021ac4ea_n.jpg'
      img = image.load_img(img_path, target_size=(224, 224))  # Adjust the target size as needed
      img_array = image.img_to_array(img)
      img_array = np.expand_dims(img_array, axis=0)  # Add batch dimension
      img_array /= 255.0  # Normalize if required

      # Make predictions
      predictions = model.predict(img_array)
      predicted_class = np.argmax(predictions, axis=1)  # Get the index of the class with the highest score

      print(f'Predicted class: {predicted_class}')
      for i,val in names.items():
          if(val == predicted_class):
              print(i)
              break;
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 287ms/step
Predicted class: [5]
tulips
```