AI-Powered Nutrition Analyzer For Fitness Enthusiasts

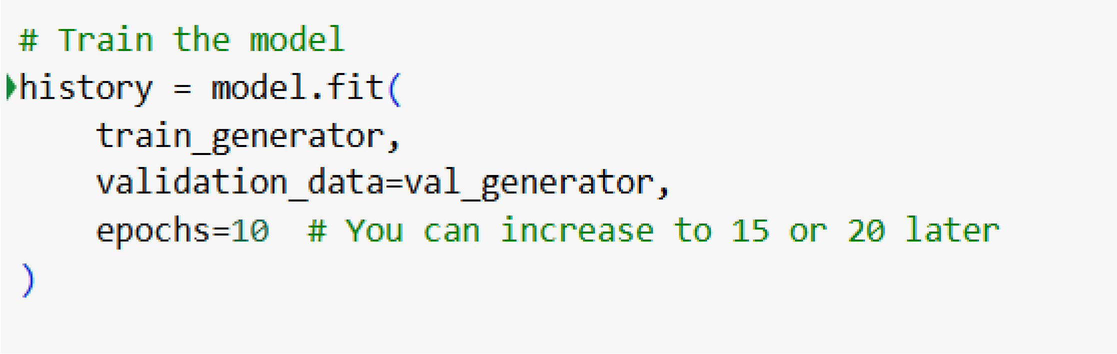
Date	21-06-2025
Team ID	SWTID1749893823
Project Title	AI-Powered Nutrition Analyzer For Fitness Enthusiasts
Maximum Marks	10 Marks

Initial Model Training Code, Model Validation and Evaluation Report

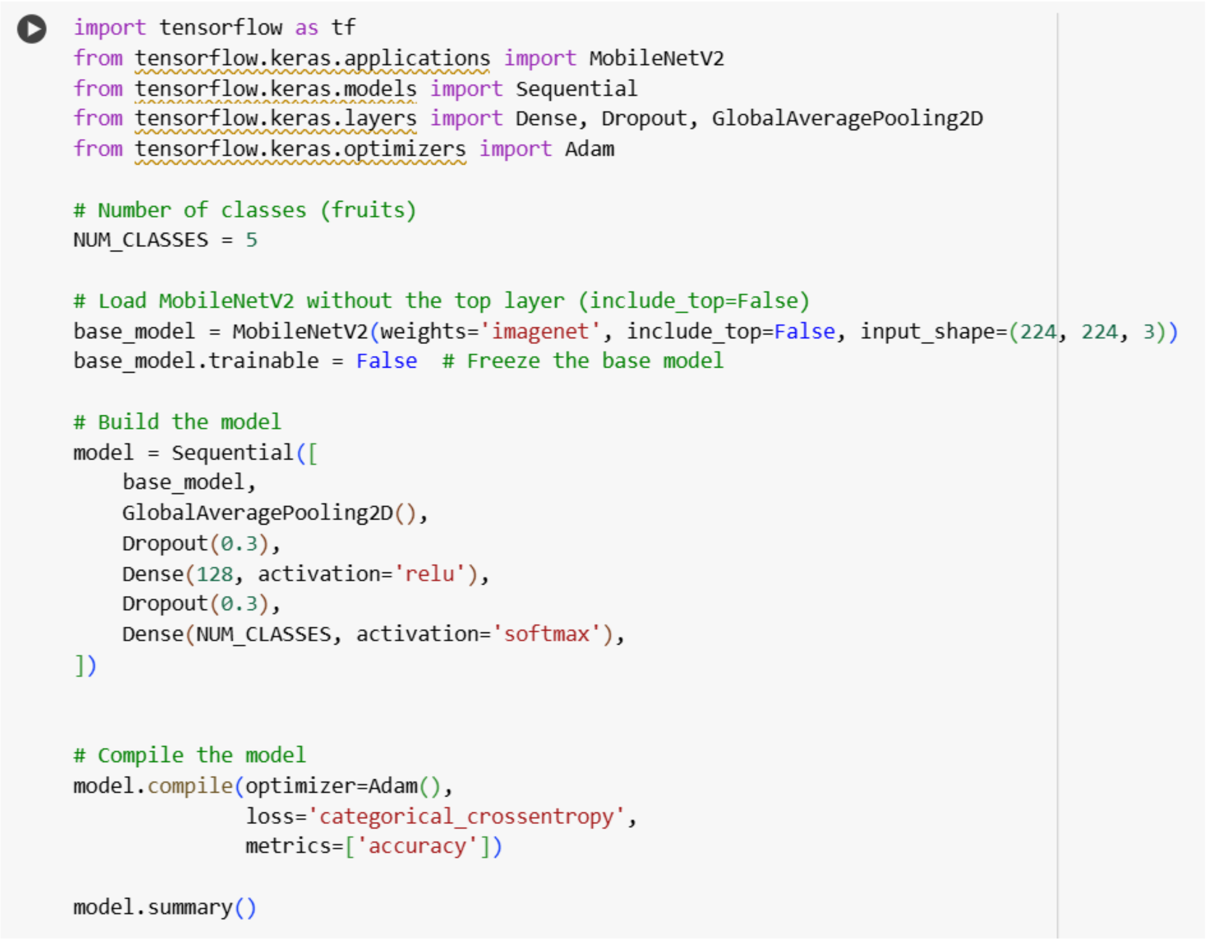
The initial model training code will be showcased in the future through a screenshot. The model validation and evaluation report will include a summary and training and validation performance metrics for multiple models, presented through respective screenshots.

Initial Model Training Code (5 marks):

MobileNetV2 Model:

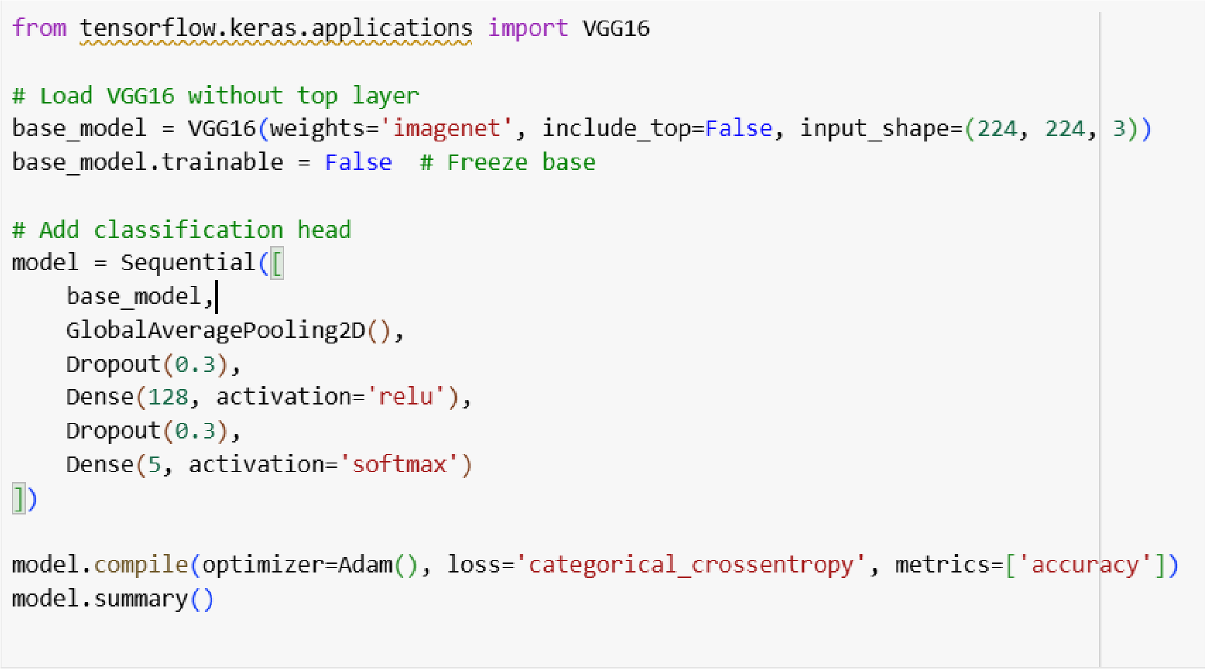


```
# Train the model
history = model.fit(
    train_generator,
    validation_data=val_generator,
    epochs=10  # You can increase to 15 or 20 later
)
```

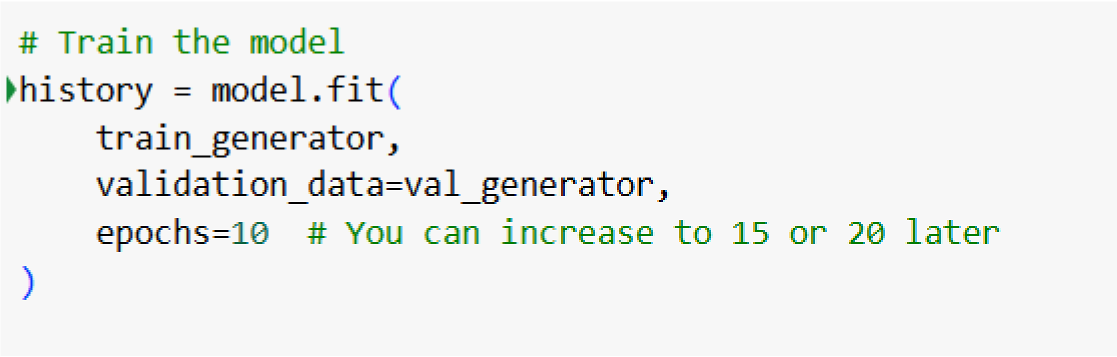


```
import tensorflow as tf
from tensorflow.keras.applications import MobileNetV2
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, GlobalAveragePooling2D
from tensorflow.keras.optimizers import Adam

# Number of classes (fruits)
NUM_CLASSES = 5

# Load MobileNetV2 without the top layer (include_top=False)
base_model = MobileNetV2(weights='imagenet', include_top=False, input_shape=(224, 224, 3))
base_model.trainable = False  # Freeze the base model

# Build the model
model = Sequential([
    base_model,
    GlobalAveragePooling2D(),
    Dropout(0.3),
    Dense(128, activation='relu'),
    Dropout(0.3),
    Dense(NUM_CLASSES, activation='softmax'),
])


# Compile the model
model.compile(optimizer=Adam(),
              loss='categorical_crossentropy',
              metrics=['accuracy'])

model.summary()
```

VGG16 Model:

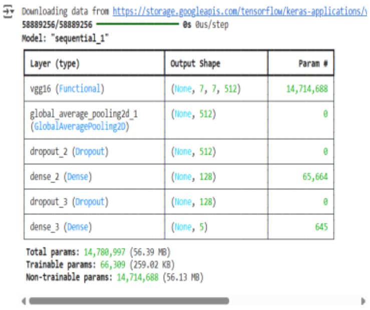
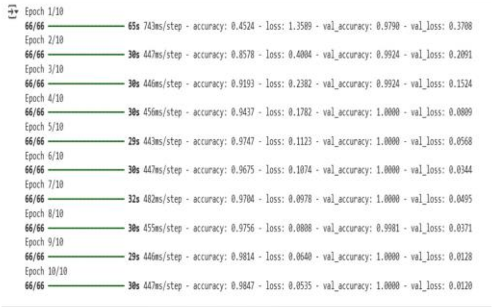
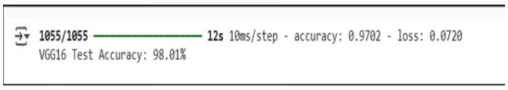
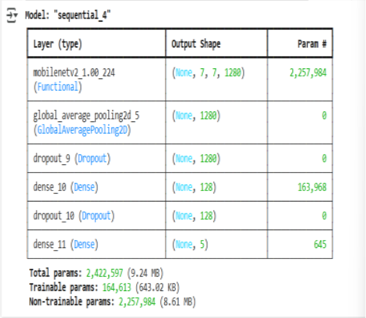
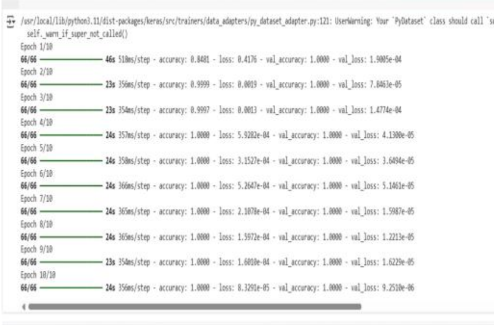
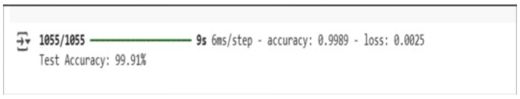


```
from tensorflow.keras.applications import VGG16

# Load VGG16 without top layer
base_model = VGG16(weights='imagenet', include_top=False, input_shape=(224, 224, 3))
base_model.trainable = False  # Freeze base

# Add classification head
model = Sequential([
    base_model,
    GlobalAveragePooling2D(),
    Dropout(0.3),
    Dense(128, activation='relu'),
    Dropout(0.3),
    Dense(5, activation='softmax')
])

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



```
# Train the model
history = model.fit(
    train_generator,
    validation_data=val_generator,
    epochs=10  # You can increase to 15 or 20 later
)
```

Model Validation and Evaluation Report (5 marks):

<table>
<tr><td>Model</td><td>Summary</td><td>Training and Validation Performance Metrics</td></tr>
<tr><td>VGG16</td><td>
<pre>
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/
58889256/58889256 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
Model: "sequential_1"
</pre>
<table>
<tr><th>Layer (type)</th><th>Output Shape</th><th>Param #</th></tr>
<tr><td>vgg16 (Functional)</td><td>(None, 7, 7, 512)</td><td>14,714,688</td></tr>
<tr><td>global_average_pooling2d_1
(GlobalAveragePooling2D)</td><td>(None, 512)</td><td>0</td></tr>
<tr><td>dropout_2 (Dropout)</td><td>(None, 512)</td><td>0</td></tr>
<tr><td>dense_2 (Dense)</td><td>(None, 128)</td><td>65,664</td></tr>
<tr><td>dropout_3 (Dropout)</td><td>(None, 128)</td><td>0</td></tr>
<tr><td>dense_3 (Dense)</td><td>(None, 5)</td><td>645</td></tr>
</table>
<pre>
Total params: 14,780,997 (56.39 MB)
Trainable params: 66,309 (259.02 KB)
Non-trainable params: 14,714,688 (56.13 MB)
</pre></td><td>
<pre>
Epoch 1/10
66/66 ━━━━━━━━━━━━━━━━━━━━ 65s 743ms/step - accuracy: 0.4524 - loss: 1.3589 - val_accuracy: 0.9790 - val_loss: 0.3708
Epoch 2/10
66/66 ━━━━━━━━━━━━━━━━━━━━ 30s 447ms/step - accuracy: 0.8578 - loss: 0.4004 - val_accuracy: 0.9924 - val_loss: 0.2091
Epoch 3/10
66/66 ━━━━━━━━━━━━━━━━━━━━ 30s 446ms/step - accuracy: 0.9193 - loss: 0.2382 - val_accuracy: 0.9924 - val_loss: 0.1524
Epoch 4/10
66/66 ━━━━━━━━━━━━━━━━━━━━ 30s 456ms/step - accuracy: 0.9437 - loss: 0.1782 - val_accuracy: 1.0000 - val_loss: 0.0809
Epoch 5/10
66/66 ━━━━━━━━━━━━━━━━━━━━ 29s 443ms/step - accuracy: 0.9747 - loss: 0.1123 - val_accuracy: 1.0000 - val_loss: 0.0568
Epoch 6/10
66/66 ━━━━━━━━━━━━━━━━━━━━ 30s 447ms/step - accuracy: 0.9675 - loss: 0.1074 - val_accuracy: 1.0000 - val_loss: 0.0344
Epoch 7/10
66/66 ━━━━━━━━━━━━━━━━━━━━ 32s 482ms/step - accuracy: 0.9704 - loss: 0.0978 - val_accuracy: 1.0000 - val_loss: 0.0495
Epoch 8/10
66/66 ━━━━━━━━━━━━━━━━━━━━ 30s 455ms/step - accuracy: 0.9756 - loss: 0.0808 - val_accuracy: 0.9981 - val_loss: 0.0371
Epoch 9/10
66/66 ━━━━━━━━━━━━━━━━━━━━ 29s 446ms/step - accuracy: 0.9814 - loss: 0.0640 - val_accuracy: 1.0000 - val_loss: 0.0128
Epoch 10/10
66/66 ━━━━━━━━━━━━━━━━━━━━ 30s 447ms/step - accuracy: 0.9847 - loss: 0.0535 - val_accuracy: 1.0000 - val_loss: 0.0120
</pre>

<pre>
1055/1055 ━━━━━━━━━━━━━━━━━━━━ 12s 10ms/step - accuracy: 0.9702 - loss: 0.0720
VGG16 Test Accuracy: 98.01%
</pre></td></tr>
<tr><td>MobileNetV2</td><td>
<pre>
Model: "sequential_4"
</pre>
<table>
<tr><th>Layer (type)</th><th>Output Shape</th><th>Param #</th></tr>
<tr><td>mobilenetv2_1.00_224
(Functional)</td><td>(None, 7, 7, 1280)</td><td>2,257,984</td></tr>
<tr><td>global_average_pooling2d_5
(GlobalAveragePooling2D)</td><td>(None, 1280)</td><td>0</td></tr>
<tr><td>dropout_9 (Dropout)</td><td>(None, 1280)</td><td>0</td></tr>
<tr><td>dense_10 (Dense)</td><td>(None, 128)</td><td>163,968</td></tr>
<tr><td>dropout_10 (Dropout)</td><td>(None, 128)</td><td>0</td></tr>
<tr><td>dense_11 (Dense)</td><td>(None, 5)</td><td>645</td></tr>
</table>
<pre>
Total params: 2,422,597 (9.24 MB)
Trainable params: 164,613 (643.02 KB)
Non-trainable params: 2,257,984 (8.61 MB)
</pre></td><td>
<pre>
/usr/local/lib/python3.11/dist-packages/keras/src/trainers/data_adapters/py_dataset_adapter.py:121: UserWarning: Your `PyDataset` class should call `s
  self._warn_if_super_not_called()
Epoch 1/10
66/66 ━━━━━━━━━━━━━━━━━━━━ 46s 518ms/step - accuracy: 0.8481 - loss: 0.4176 - val_accuracy: 1.0000 - val_loss: 1.9005e-04
Epoch 2/10
66/66 ━━━━━━━━━━━━━━━━━━━━ 23s 356ms/step - accuracy: 0.9999 - loss: 0.0019 - val_accuracy: 1.0000 - val_loss: 7.8463e-05
Epoch 3/10
66/66 ━━━━━━━━━━━━━━━━━━━━ 23s 354ms/step - accuracy: 0.9997 - loss: 0.0013 - val_accuracy: 1.0000 - val_loss: 1.4774e-04
Epoch 4/10
66/66 ━━━━━━━━━━━━━━━━━━━━ 24s 357ms/step - accuracy: 1.0000 - loss: 5.9282e-04 - val_accuracy: 1.0000 - val_loss: 4.1300e-05
Epoch 5/10
66/66 ━━━━━━━━━━━━━━━━━━━━ 24s 358ms/step - accuracy: 1.0000 - loss: 3.1527e-04 - val_accuracy: 1.0000 - val_loss: 3.6494e-05
Epoch 6/10
66/66 ━━━━━━━━━━━━━━━━━━━━ 24s 366ms/step - accuracy: 1.0000 - loss: 5.2647e-04 - val_accuracy: 1.0000 - val_loss: 5.1461e-05
Epoch 7/10
66/66 ━━━━━━━━━━━━━━━━━━━━ 24s 365ms/step - accuracy: 1.0000 - loss: 2.1078e-04 - val_accuracy: 1.0000 - val_loss: 1.5987e-05
Epoch 8/10
66/66 ━━━━━━━━━━━━━━━━━━━━ 24s 365ms/step - accuracy: 1.0000 - loss: 1.5972e-04 - val_accuracy: 1.0000 - val_loss: 1.2213e-05
Epoch 9/10
66/66 ━━━━━━━━━━━━━━━━━━━━ 23s 354ms/step - accuracy: 1.0000 - loss: 1.6010e-04 - val_accuracy: 1.0000 - val_loss: 1.6229e-05
Epoch 10/10
66/66 ━━━━━━━━━━━━━━━━━━━━ 24s 356ms/step - accuracy: 1.0000 - loss: 8.3291e-05 - val_accuracy: 1.0000 - val_loss: 9.2510e-06
</pre>

<pre>
1055/1055 ━━━━━━━━━━━━━━━━━━━━ 9s 6ms/step - accuracy: 0.9989 - loss: 0.0025
Test Accuracy: 99.91%
</pre></td></tr>
</table>