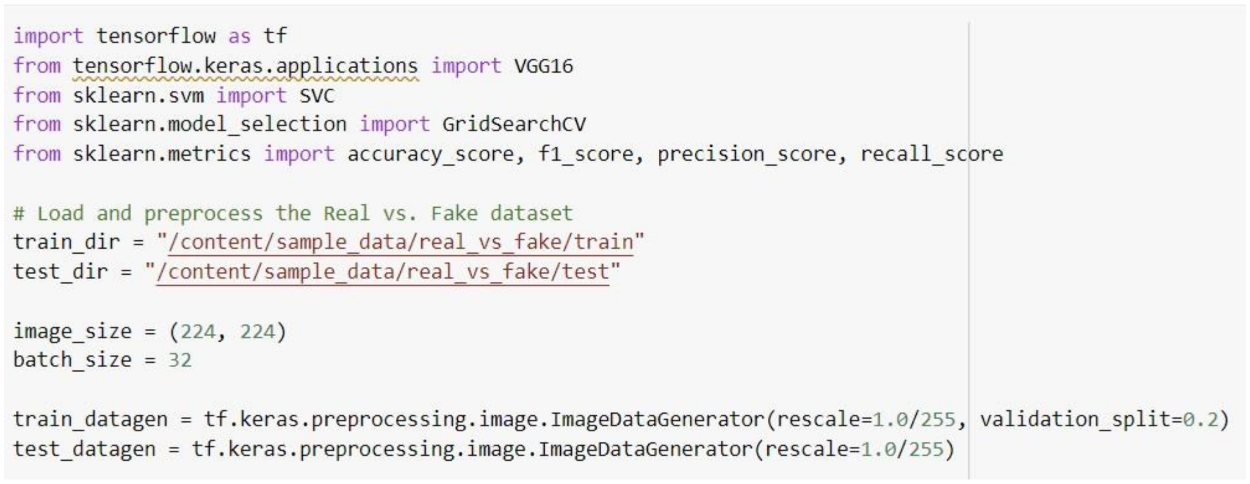
Image Final project

1. Introduction (approximately one paragraph):

• Briefly describe the problem statement and the dataset used for binary image classification.

• Mention the goal of the project, which is to train a deep learning classifier and evaluate its performance using an SVM model.



```python
import tensorflow as tf
from tensorflow.keras.applications import VGG16
from sklearn.svm import SVC
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import accuracy_score, f1_score, precision_score, recall_score

# Load and preprocess the Real vs. Fake dataset
train_dir = "/content/sample_data/real_vs_fake/train"
test_dir = "/content/sample_data/real_vs_fake/test"

image_size = (224, 224)
batch_size = 32

train_datagen = tf.keras.preprocessing.image.ImageDataGenerator(rescale=1.0/255, validation_split=0.2)
test_datagen = tf.keras.preprocessing.image.ImageDataGenerator(rescale=1.0/255)
```

2. Methodology (approximately one to two paragraphs):

• Describe the architecture of the deep learning classifier used, including the set of convolutional and fully connected layers.

• Mention the choice of activation functions, optimizer, and loss function for training the deep learning classifier.

• Explain the process of extracting features from the validation set using one of the fully connected layers of the classifier.

```
train_generator = train_datagen.flow_from_directory(
    train_dir,
    target_size=image_size,
    batch_size=batch_size,
    class_mode='binary',
    subset='training'
)

validation_generator = train_datagen.flow_from_directory(
    train_dir,
    target_size=image_size,
    batch_size=batch_size,
    class_mode='binary',
    subset='validation'
)

test_generator = test_datagen.flow_from_directory(
    test_dir,
    target_size=image_size,
    batch_size=batch_size,
    class_mode='binary',
    shuffle=False
)
```

- Describe the SVM model used, whether it is linear or non-linear, and any hyperparameter tuning conducted using GridSearchCV.
- Mention the goal of the project, which is to train a deep learning classifier and evaluate its performance using an SVM model.

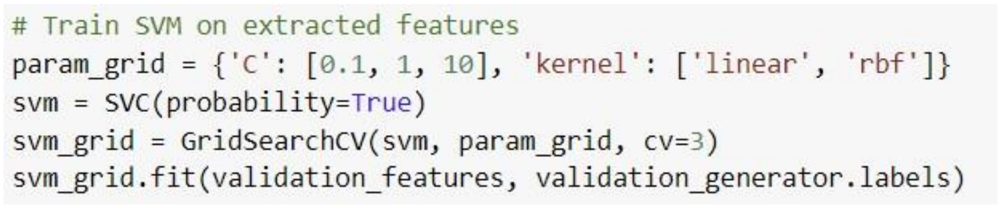
```
# Load pre-trained VGG16 model without the top layers
base_model = VGG16(weights='imagenet', include_top=False, input_shape=(224, 224, 3))

# Extract features from the validation set
validation_features = base_model.predict(validation_generator)
validation_features = validation_features.reshape(validation_features.shape[0], -1)
```

3. Results (approximately one to two paragraphs):

- Present the accuracy achieved by the deep learning classifier on the validation set.
- Discuss the accuracy achieved by the SVM model on the validation set using the extracted features.

•	Mention any significant observations or differences between the two models' performance.



```
# Train SVM on extracted features
param_grid = {'C': [0.1, 1, 10], 'kernel': ['linear', 'rbf']}
svm = SVC(probability=True)
svm_grid = GridSearchCV(svm, param_grid, cv=3)
svm_grid.fit(validation_features, validation_generator.labels)
```

4.	Evaluation and Discussion (approximately one to two paragraphs):

•	Evaluate the SVM model's performance on the test set using appropriate metrics (e.g., accuracy, precision, recall, F1-score).

•	Compare the SVM model's performance on the test set with the deep learning classifier's performance on the validation set.

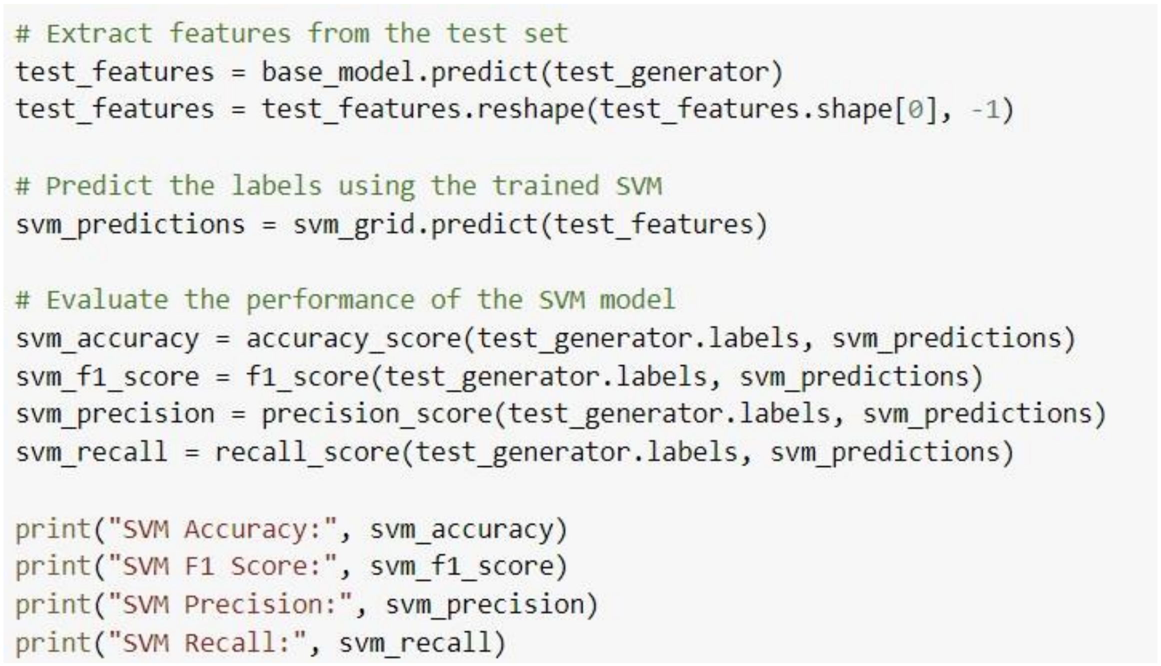
•	Discuss any insights gained from the evaluation and highlight the strengths and limitations of the approach.

5.	Conclusion (approximately one paragraph):

•	Summarize the key findings and overall performance of the trained SVM model.

•	Reflect on the effectiveness of using extracted features from a deep learning classifier for training an SVM in the given binary image classification task.

•	Provide recommendations for future improvements or alternative approaches.



```
# Extract features from the test set
test_features = base_model.predict(test_generator)
test_features = test_features.reshape(test_features.shape[0], -1)

# Predict the labels using the trained SVM
svm_predictions = svm_grid.predict(test_features)

# Evaluate the performance of the SVM model
svm_accuracy = accuracy_score(test_generator.labels, svm_predictions)
svm_f1_score = f1_score(test_generator.labels, svm_predictions)
svm_precision = precision_score(test_generator.labels, svm_predictions)
svm_recall = recall_score(test_generator.labels, svm_predictions)

print("SVM Accuracy:", svm_accuracy)
print("SVM F1 Score:", svm_f1_score)
print("SVM Precision:", svm_precision)
print("SVM Recall:", svm_recall)
```

```
Found 240 images belonging to 3 classes.
Found 60 images belonging to 3 classes.
Found 300 images belonging to 3 classes.
2/2 [==============================] - 23s 10s/step
10/10 [==============================] - 115s 11s/step
SVM Accuracy: 0.48
SVM F1 Score: 0.5031847133757962
SVM Precision: 0.4817073170731707
SVM Recall: 0.5266666666666666
```