Task 3: Meme Image Classification System

Objective:

Develop a classification system to distinguish meme images from non-meme images, using the provided meme dataset as the positive class and sourcing non-meme images as the negative class.

Approach:

The approach leverages transfer learning by using a pre-trained convolutional neural network (CNN) model, VGG16, as the base model. This is a common and effective technique in computer vision tasks, especially when dealing with limited training data. By utilizing the feature extraction capabilities of a pre-trained model, the system can take advantage of the knowledge learned from a large-scale dataset, enabling better generalization and potentially improved performance.

1. The code follows a typical workflow for image classification tasks, starting with data loading and preprocessing.
2. The approach utilizes data augmentation techniques (rotation, shifting, shearing, zooming, flipping) on the training data to increase data diversity and improve model robustness.
3. The pre-trained VGG16 model is used as the base model, with its layers frozen to preserve the learned features.
4. Custom layers (flatten, dense, dropout) are added on top of the pre-trained model to adapt it to the specific task of meme image classification.
5. The model is trained using the provided training and validation data generators, with the option to visualize the training history (accuracy and loss curves).
6. The trained model is evaluated on the validation data, and performance metrics (loss and accuracy) are reported.
7. The model is saved for future use, and a function is provided to make predictions on new images.

Methodology:

1. **Data Collection and Preprocessing**
   - Collect a non-meme image dataset from reliable sources, ensuring a diverse range of image types.
   - Preprocess the datasets (meme and non-meme) by resizing, normalizing, or applying necessary transformations for consistency.
2. **Feature Extraction**
   - Explore different feature extraction techniques, such as pre-trained CNNs (VGG, ResNet, EfficientNet), to extract relevant visual features from images.
   - Consider using self-supervised or transfer learning approaches to leverage pre-trained models on large-scale datasets.
3. **Model Selection and Training**
   - Choose an appropriate classification model architecture (CNN, Transformer-based, or a combination).
   - Split the datasets into train, validation, and test sets.
   - Train the chosen model on the training set, employing techniques like data augmentation, early stopping, and learning rate scheduling.
4. **Model Evaluation**
   - Evaluate the trained model's performance on the test set using appropriate metrics (accuracy, precision, recall, F1-score, confusion matrix).
   - Analyze the model's performance separately on meme and non-meme classes to identify potential biases or areas for improvement.
5. **Interpretation and Visualization**
   - Explore techniques like saliency maps, activation maps, or attention visualization to understand the model's decision-making process.
   - Visualize and interpret the model's predictions on successful and challenging examples.
6. **Hyperparameter Tuning and Ensemble Methods**

- Experiment with hyperparameter tuning techniques (grid search, random search, Bayesian optimization) to improve model performance.
- Consider ensemble methods (bagging, boosting) to combine multiple models and potentially enhance classification accuracy.

Findings and Results:

Data Exploration Results: (For a Sample of 6000 images Data)

```
print('total validation not hateful images:',
```

```
total training hateful images: 2400
total training not hateful images: 2400
total validation hateful images: 610
total validation not hateful images: 600
```

```
        shuffle=False)
```

```
Found 4800 images belonging to 2 classes.
Found 1210 images belonging to 2 classes.
```

```
from tensorflow.keras.applications.vgg16 impo
```

Defining Model Architecture: (Before Training)

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 224, 224, 3)]     0

 block1_conv1 (Conv2D)       (None, 224, 224, 64)      1792

 block1_conv2 (Conv2D)       (None, 224, 224, 64)      36928

 block1_pool (MaxPooling2D)  (None, 112, 112, 64)      0

 block2_conv1 (Conv2D)       (None, 112, 112, 128)     73856

 block2_conv2 (Conv2D)       (None, 112, 112, 128)     147584

 block2_pool (MaxPooling2D)  (None, 56, 56, 128)       0

 block3_conv1 (Conv2D)       (None, 56, 56, 256)       295168

 block3_conv2 (Conv2D)       (None, 56, 56, 256)       590080

 block3_conv3 (Conv2D)       (None, 56, 56, 256)       590080

 block3_pool (MaxPooling2D)  (None, 28, 28, 256)       0

 block4_conv1 (Conv2D)       (None, 28, 28, 512)       1180160

 block4_conv2 (Conv2D)       (None, 28, 28, 512)       2359808

 block4_conv3 (Conv2D)       (None, 28, 28, 512)       2359808

 block4_pool (MaxPooling2D)  (None, 14, 14, 512)       0

 block5_conv1 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_conv2 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_conv3 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_pool (MaxPooling2D)  (None, 7, 7, 512)         0

 flatten (Flatten)           (None, 25088)             0

 dense (Dense)               (None, 512)               12845568

 dropout (Dropout)           (None, 512)               0

 dense_1 (Dense)             (None, 1)                 513

=================================================================
Total params: 27,560,769
Trainable params: 12,846,081
Non-trainable params: 14,714,688
_________________________________________________________________
```

Training Logs: (For 10 Epochs) (For a Sample of 6000 images Data)

```
Epoch 1/10
480/480 [==============================] - 1661s 3s/step - loss: 1.2256 - ac
c: 0.4910 - val_loss: 0.6925 - val_acc: 0.5300
Epoch 2/10
480/480 [==============================] - 1527s 3s/step - loss: 0.6974 - ac
c: 0.5190 - val_loss: 0.6933 - val_acc: 0.4908
Epoch 3/10
480/480 [==============================] - 813s 2s/step - loss: 0.6960 - acc:
0.5054 - val_loss: 0.6933 - val_acc: 0.4925
Epoch 4/10
480/480 [==============================] - 737s 2s/step - loss: 0.6958 - acc:
0.5108 - val_loss: 0.6935 - val_acc: 0.4908
Epoch 5/10
480/480 [==============================] - 2495s 5s/step - loss: 0.6945 - ac
c: 0.4956 - val_loss: 0.6932 - val_acc: 0.5083
Epoch 6/10
480/480 [==============================] - 813s 2s/step - loss: 0.6935 - acc:
0.4960 - val_loss: 0.6931 - val_acc: 0.5083
Epoch 7/10
480/480 [==============================] - 836s 2s/step - loss: 0.6943 - acc:
0.5013 - val_loss: 0.6933 - val_acc: 0.4958
Epoch 8/10
480/480 [==============================] - 831s 2s/step - loss: 0.6964 - acc:
0.4985 - val_loss: 0.6929 - val_acc: 0.5083
Epoch 9/10
480/480 [==============================] - 772s 2s/step - loss: 0.6950 - acc:
0.5015 - val_loss: 0.6931 - val_acc: 0.5092
Epoch 10/10
480/480 [==============================] - 718s 1s/step - loss: 0.6948 - acc:
0.4990 - val_loss: 0.6931 - val_acc: 0.5092
```

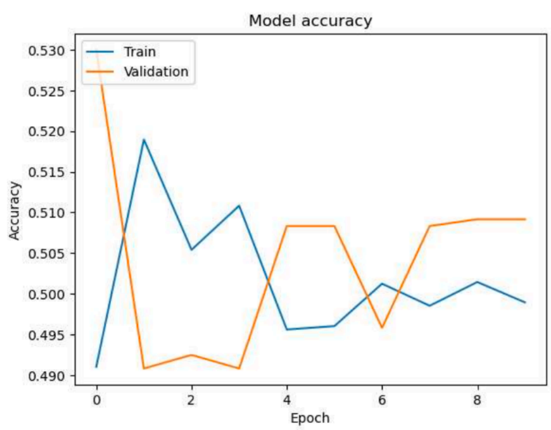
Testing with Validation Data:

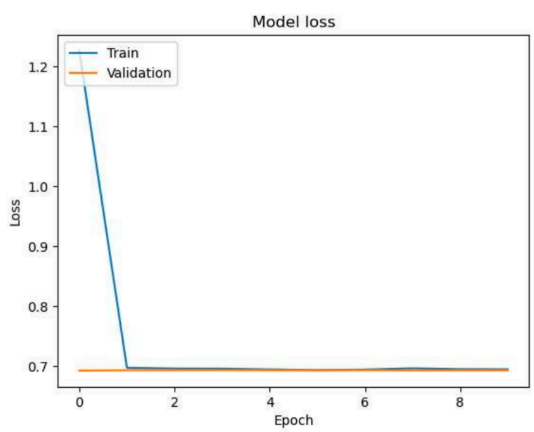
```
121/121 [==============================] - 130s 1s/step - loss: 0.6931 - acc:
0.5050
Validation Loss: 0.6931177973747253
Validation Accuracy: 0.5049586892127991
```

Inferring the trained model using the data:

```
1/1 [==============================] - 0s 125ms/step
01235.png is a Hateful Meme
1/1 [==============================] - 0s 117ms/step
01236.png is a Hateful Meme
1/1 [==============================] - 0s 134ms/step
01247.png is a Hateful Meme
1/1 [==============================] - 0s 126ms/step
01256.png is a Not Hateful Meme
1/1 [==============================] - 0s 129ms/step
01258.png is a Hateful Meme
1/1 [==============================] - 0s 122ms/step
01269.png is a Hateful Meme
1/1 [==============================] - 0s 131ms/step
01274.png is a Hateful Meme
1/1 [==============================] - 0s 138ms/step
01295.png is a Hateful Meme
1/1 [==============================] - 0s 110ms/step
01327.png is a Not Hateful Meme
1/1 [==============================] - 0s 118ms/step
```

Visualization of Performance of Model (Accuracy & Loss)





Where it can fail:

1. While transfer learning can be effective, the performance of the system may still be limited by the chosen base model (VGG16) and the amount of available training data.
2. If the provided meme image dataset or the collected non-meme image dataset contains biases or lacks diversity, the trained model may exhibit biased behavior or fail to generalize well to unseen data.
3. Training deep learning models, especially on large datasets, can be computationally intensive and may require significant computational resources (e.g., GPU acceleration).

These potential issues can be mitigated by:

1. Exploring different base models or architectures (e.g., ResNet, EfficientNet, Vision Transformers) and comparing their performance.
2. Collecting and curating a larger and more diverse dataset to improve the model's generalization capabilities.
3. Implementing more advanced data augmentation techniques or incorporating techniques like self-supervised learning or semi-supervised learning to leverage unlabeled data.
4. Considering ensemble methods or other techniques to combine multiple models and potentially improve overall performance.