**Image Classification (George and Non\_george)**

**Atul Priyank**

1. **Approach and Rationale for Model Selection**

For the task of detecting the presence of St. George in images, we chose **EfficientNet-B0** as the base model. This decision was driven by several factors:

1. **Efficiency**: EfficientNet-B0 is known for its balance between computational efficiency and accuracy. It provides good accuracy with relatively low computational cost compared to other deeper models, making it a suitable choice for image classification tasks where both time and resource efficiency matter.
2. **Pre-trained Weights**: EfficientNet-B0 is pre-trained on the ImageNet dataset, which contains a large variety of images. By leveraging these pre-trained weights, we can fine-tune the model for our binary classification task (St. George vs. Non-St. George) and significantly reduce the training time required to achieve good performance.
3. **Transfer Learning**: The final fully connected layer was replaced to accommodate the binary classification task. Transfer learning allows us to use the learned features from a broader dataset (ImageNet) while tailoring the model to our specific problem with minimal modifications.

**Why Binary Cross-Entropy and Adam Optimizer?**

* **Loss Function**: We used CrossEntropyLoss because this is a standard loss function for multi-class and binary classification tasks in deep learning. It measures how well the model's predictions match the true labels.
* **Optimizer**: Adam was chosen for its adaptability and efficiency in handling sparse gradients. It combines the advantages of both RMSProp and Stochastic Gradient Descent (SGD) with momentum.

**Data Augmentation:**

* **Random Horizontal Flip**: We applied random horizontal flipping to the images during training to introduce more variation and help the model generalize better. This form of augmentation is simple yet effective for image classification tasks and helps mitigate overfitting.

**Early Stopping and Learning Rate Scheduler:**

* **Early Stopping**: Early stopping prevents overfitting by halting training when the validation accuracy stops improving.
* **Learning Rate Scheduler**: The learning rate scheduler adjusts the learning rate after a set number of epochs, ensuring the optimizer takes smaller steps as the training progresses, which can improve convergence.

1. **Choice of Test Metrics**

To evaluate the performance of the model, we selected several common classification metrics:

1. **Accuracy**:
   * Measures the overall correctness of the model by calculating the percentage of correctly classified instances out of the total number of instances.
   * Useful when the class distribution is balanced, as is the case in this dataset.
2. **Precision**:
   * Precision measures the proportion of positive identifications that were actually correct (e.g., how many of the predicted "George" images actually contain St. George).
   * High precision is important when minimizing false positives is crucial.
3. **Recall (Sensitivity)**:
   * Recall measures the proportion of actual positives that were correctly identified (e.g., how many of the true "George" images were correctly classified).
   * High recall is important when minimizing false negatives is crucial.
4. **F1-Score**:
   * The F1-Score is the harmonic mean of precision and recall, providing a balanced measure of the model’s accuracy when there is a trade-off between precision and recall.
5. **Confusion Matrix**:
   * Provides a detailed breakdown of correct and incorrect classifications. It shows how many true positives, true negatives, false positives, and false negatives the model has generated.

These metrics provide a holistic view of the model's performance, helping to detect if the model is biased towards one class or if there are significant false positives/negatives.

1. **Improvements and Future Steps**
2. **Additional Data Augmentation**:
   * Incorporating more complex data augmentation techniques like rotation, zoom, and color jitter could improve the model's generalization further.
3. **Hyperparameter Tuning**:
   * Fine-tuning hyperparameters such as learning rate, batch size, and dropout could yield better performance. A grid search or random search can be used to optimize these values.
4. **Different Model Architectures**:
   * Experimenting with other pre-trained architectures, such as ResNet or VGG, could help compare performance. If the task requires even faster inference, models like MobileNet or NASNet could be considered.
5. **Handling False Positives/Negatives**:
   * The confusion matrix indicates some false positives and false negatives. Focusing on misclassified images and incorporating more diverse training data or applying focal loss to handle class imbalance may help.
6. **Fine-tuning for Imbalanced Classes**:
   * If the dataset becomes imbalanced, applying techniques such as oversampling the minority class, undersampling the majority class, or using class weighting in the loss function can improve performance.
7. **Training and Evaluation**

**Training Results**

During training, the model's performance is monitored using **training loss** and **validation loss** after each epoch. The training accuracy is also calculated to see how well the model is learning from the training data.

Here are your training results:

**Training Metrics:**

* **Training Accuracy**: Starting from 84.83% at epoch 1, it steadily increased to 95.74% by epoch 5.
* **Training Loss**: The training loss started at 0.3637 and dropped to 0.1326 by epoch 5, indicating that the model was learning effectively.

**Validation Metrics:**

* **Validation Accuracy**: The validation accuracy fluctuated between 90% and 92%, indicating strong performance on unseen data during the validation phase.
* **Validation Loss**: The validation loss started at 0.2302 in the first epoch and remained around the same range (0.2416 in epoch 4), suggesting the model is not overfitting too much to the training data.

**Evaluation Results**

After the model is trained, you perform **evaluation** on the validation dataset (which simulates how the model will perform on unseen data in production). This is where the classification report and confusion matrix come into play.

**Evaluation Metrics:**

After training, you evaluated the model on the **validation dataset**, and here are the results:

* **Accuracy**: 92% on the validation set.
* **Precision**:
  + **George**: 0.93
  + **Non-George**: 0.91
* **Recall**:
  + **George**: 0.89
  + **Non-George**: 0.94
* **F1-Score**:
  + **George**: 0.91
  + **Non-George**: 0.93

These results show that the model has balanced performance across both classes, with a high overall accuracy of 92%.

**Confusion Matrix:**

* 480 True Positives (correctly classified George)
* 59 False Positives (misclassified as Non-George when it was George)
* 37 False Negatives (misclassified as George when it was Non-George)
* 634 True Negatives (correctly classified Non-George)

1. **Key Takeaways from Training and Evaluation**
2. **Training Phase**:
   * **Model learning**: The model showed consistent improvement in accuracy during training and validation, with minimal overfitting.
   * **Early Stopping**: Early stopping triggered after the model stopped improving, which helps to avoid overfitting and saves computation time.
3. **Evaluation Phase**:
   * **Balanced Performance**: The model performed well with high precision and recall for both classes. The F1-score of 0.91 (George) and 0.93 (Non-George) indicates that the model is making balanced predictions across both classes.
   * **Confusion Matrix**: While the model performed well, there were some misclassifications (37 false negatives, 59 false positives). These can be further reduced with hyperparameter tuning or additional data.
4. **Conclusion**

The task of detecting the presence of St. George in an image was successfully completed using **EfficientNet-B0** with transfer learning. By leveraging a pre-trained model, we minimized training time while achieving high accuracy (92%) on the validation set. Precision, recall, and F1-scores showed balanced performance between the two classes (George and Non-George).

The confusion matrix revealed a few false positives and negatives, which could be addressed through future improvements such as advanced data augmentation or hyperparameter tuning. Overall, EfficientNet-B0 proved to be an excellent choice for this task due to its lightweight architecture and ability to generalize well across the dataset.