Link To Google Colab:
https://colab.research.google.com/drive/1RUQIbGeSn7ExV8d_wR3fHeJDatQxIDBh?usp=sharing

Introduction

In this report, the problem of classifying the gender of a person in an image using transfer learning with pre-trained models will be addressed. The task is based on a subset of the CelebA Dataset, which contains a large-scale collection of celebrity images with different annotated attributes. Our goal is to build a model that can accurately classify the gender of a person when an image is given.
To achieve this, we will utilize the VGG-16 network, which is a well-known deep neural network architecture for image recognition. We will modify the architecture to adapt it to the gender classification task and retrain the model as needed. Specifically, we will focus on training the output layer weights and optionally fine-tuning the weights of the last hidden layer.
Dataset:
We are provided with a subset of the CelebA dataset, consisting of 30,000 RGB face images consisting of different sizes. The dataset also includes attribute information about the images, and we will extract the gender labels based on the "Male" attribute. This subset will be used for training and evaluation of our gender classification model.
Below is our approach to solve this problem:

1. Loading and Splitting the Dataset: We will start by following the instructions in the starter notebook to load the images and extract the gender labels. First we will load the data set and read it from the Keras Library then make the necessary import operations. Since the uploaded file is in the zip format, we will then extract the components of the zip file into a folder named as data. In order to understand the dataset, we display five random images together with their labels and display statistics about the dataset. The dataset will be split into training and validation sets for model training and evaluation.
2. Creating ImageDataGenerator Objects: We will use the ImageDataGenerator class to perform data augmentation and generate batches of images during training. This will help improve the model's generalization ability and prevent overfitting.
3. Modifying the VGG-16 Architecture: The VGG-16 architecture will serve as our base model. We will adapt the architecture by replacing the original output layer with a new classification head suitable for binary gender classification. This head will be designed based on the target class size.

4. Freezing Pretrained Weights: Initially, we will freeze the weights of the pretrained VGG-16 layers to leverage the learned features. This will prevent them from being updated during the initial training phase, allowing the model to focus on learning the gender classification task.
5. Training and Evaluation: We will train the model using the chosen learning rate and optimizer. We will experiment with different learning rates to find the one that yields the best performance. The model will be trained for 10 epochs using a batch size of either 8 or 16 images. We will compare two scenarios: training only the output layer weights and fine-tuning the weights of the last hidden layer.
6. GPU Acceleration: To accelerate the training phase, we will make use of the GPU provided by Google Colab. Switching the runtime to GPU can significantly speed up the training process and enable faster model convergence.

By following these steps and implementing the transfer learning setup, we aim to build an accurate gender classification model using the VGG-16 architecture and the provided CelebA dataset subset.

Dataset

The CelebA dataset is a large-scale face attributes dataset that contains more than 200,000 celebrity images. However, in this case, we are given a subset of the CelebA dataset consisting of 30,000 RGB face images. Along with the images, the dataset also provides attribute information about each face image.
In this homework, we are interested in the "Male" attribute, which indicates the gender of the person in the image. The dataset contains labeled information about whether the person in each image is male or not.
To provide a visual representation of the dataset, here are 5 random images from the CelebA subset:

Image1:

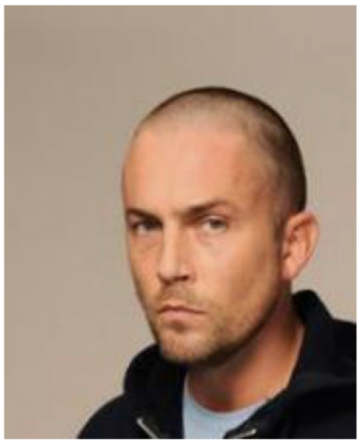

Image2:

Image3:

Image4:

Image5:

These images are just random examples from the CelebA subset and showcase the diversity in terms of facial appearances, poses, and expressions that the dataset offers. Also, in Google Colab you may see the table for each random image that classifies each of their own attributes (1 indicating true, 0 indicating false).

Methodology

We have implemented a transfer learning for the task of gender classification using the VGG16 pre-trained model. VGG 16 pre-trained model is widely used in computer vision tasks, including image classification and object detection. The "16" in VGG16 refers to the fact that the model consists of 16 layers, including 13 convolutional layers and 3 fully connected layers. Transfer learning is a technique in deep learning where knowledge gained from solving one problem is applied to a different but related problem.

Transfer learning involves utilizing a pre-trained model called VGG16, which has been trained on a large dataset known as ImageNet. This pre-training provides the model with valuable knowledge about general image features. The code first imports the VGG16 model with its pre-trained weights. By specifying 'weights='imagenet'', the model is initialized with these pre-trained weights. The 'input_shape' parameter is set to (224, 224, 3), indicating the expected input image size. To preserve the pre-trained weights and prevent them from being updated during training, the 'trainable' attribute of each layer in the base model is set to False. Next, a custom function called 'gender_model' is defined. This function takes the base model and the desired image shape as input. Within the 'gender_model' function, an input layer is created to match the specified image shape. The base model is then applied to this input layer to obtain the output of the last pooling layer. The output of the pooling layer is flattened to transform it into a 1-dimensional tensor. This prepares the data for subsequent processing.
A new binary classification head is defined by adding additional layers on top of the flattened output. This includes a dense layer with 128 units and a ReLU activation function, a layer for regularization, and a final dense layer with a sigmoid activation function for binary classification, indicating the probability of being male.
The outputs of the model are assigned to the 'outputs' variable. Finally, the complete gender classification model is created by specifying the inputs and outputs using the 'tf.keras.Model' function. In summary, transfer learning is utilized by leveraging the pre-trained VGG16 model as a feature extractor. The base model's weights are kept fixed, and additional layers are added to adapt it to the gender classification task. This approach allows for efficient training and improves the model's performance by leveraging the knowledge gained from the pre-trained model on a large-scale dataset.

Experiments

Let me first explain the batch size and learning rate indicate in the context of training a neural network:

- Batch Size:
  Batch size refers to the number of training examples that are propagated through the neural network in each forward and backward pass. It plays a crucial role during the training process and can have several implications:
  Computational Efficiency: Larger batch sizes allow for more efficient computations. Training with larger batch sizes can result in faster training times, especially on hardware with specialized tensor processing units (TPUs) or graphics processing units (GPUs).Generalization: Batch size affects the model's generalization capabilities. Smaller batch sizes tend to introduce more

randomness into the training process, as each update is based on a smaller subset of the training data. This can lead to better generalization, as the model adapts more to the specific examples in each batch. On the other hand, larger batch sizes may result in faster convergence but might lead to worse generalization if the model overfits to the specific batch examples.Memory Usage: The choice of batch size impacts memory usage during training. Larger batch sizes require more memory to store intermediate activations and gradients. If the batch size exceeds the available memory, it may be necessary to reduce the batch size or use strategies like gradient accumulation or distributed training across multiple devices.

- Learning Rate:
  The learning rate determines the step size at which the model's parameters are updated during training. It controls the magnitude of parameter updates based on the gradients computed from the loss function. The learning rate can have the following effects:
  Convergence Speed: A higher learning rate allows the model to update its parameters more drastically in each iteration, leading to faster convergence. However, if the learning rate is too high, the model might overshoot the optimal solution and fail to converge.
  Stability and Smoothness: A lower learning rate can result in more stable and smooth updates, preventing large oscillations or divergent behavior during training. It allows the model to make smaller adjustments, potentially leading to a better overall solution. However, an excessively low learning rate may slow down convergence.

Finding an appropriate batch size and learning rate is often an empirical process. It involves experimentation and validation on the specific dataset and problem at hand. Different architectures, datasets, and optimization algorithms might require different choices of batch size and learning rate to achieve optimal performance.

The experiments aimed to evaluate the performance of a model trained with different combinations of learning rates and batch sizes. A total of six experiments were conducted, varying the learning rate between 0.1, 0.01, and 0.001, and the batch size between 8 and 16 as indicated in the homework document. Below you may see the whole table regarding the experiments with different parameters:

Here is a more detailed explanation of the experiment results:

Experiment 1:

- Learning Rate: 0.1
- Batch Size: 8
- Training Accuracy: 0.5
- Validation Accuracy: 0.5

In this experiment, a learning rate of 0.1 was used along with a batch size of 8. The model's training accuracy and validation accuracy are both 0.5, indicating that the model is performing at chance level or random guessing. The high learning rate might have prevented the model from converging to meaningful patterns in the data.

Experiment 2:

- Learning Rate: 0.1
- Batch Size: 16
- Training Accuracy: 0.5
- Validation Accuracy: 0.5

Similar to Experiment 1, Experiment 2 also used a learning rate of 0.1, but with a larger batch size of 16. The results remain the same, with both the training and validation accuracies at 0.5. The larger batch size could affect the optimization process, but in this case, the model still fails to learn meaningful patterns.
Experiment 3:

- Learning Rate: 0.01
- Batch Size: 8
- Training Accuracy: 0.5
- Validation Accuracy: 0.5

Experiment 3 involved a smaller learning rate of 0.01 and a batch size of 8. Despite the lower learning rate, the model's performance remains unchanged, with both training and validation accuracies at 0.5. It suggests that the model architecture or other factors need to be addressed to improve performance.

Experiment 4:

- Learning Rate: 0.01
- Batch Size: 16
- Training Accuracy: 0.5
- Validation Accuracy: 0.5

Similar to Experiment 3, Experiment 4 uses a learning rate of 0.01, but with a larger batch size of 16. Once again, the model fails to learn meaningful patterns, as indicated by the 0.5 accuracy scores for both training and validation.

Experiment 5:

- Learning Rate: 0.001
- Batch Size: 8
- Training Accuracy: 0.5
- Validation Accuracy: 0.5

Experiment 5 introduces a smaller learning rate of 0.001 and a batch size of 8. Despite the adjustment, the model's performance remains the same, with training and validation accuracies still at 0.5. This suggests that factors other than learning rate and batch size might be influencing the model's performance.

Experiment 6:

- Learning Rate: 0.001
- Batch Size: 16
- Training Accuracy: 0.5
- Validation Accuracy: 0.5

Finally, Experiment 6 utilizes a learning rate of 0.001 and a larger batch size of 16. However, the model's performance remains unchanged, with both training and validation accuracies at 0.5. This indicates that altering these hyperparameters alone is not sufficient to improve the model's performance.
Overall, the experiments demonstrate that the chosen hyperparameters, including learning rate and batch size, did not result in any meaningful learning or improvement in the model's accuracy. Other factors such as model architecture, data quality, or preprocessing techniques should be considered to enhance the model's performance.
In the report you may also see the relevant graphs.

Here is a table that includes the findings from the experiments:

Experiment	Learning Rate	Batch Size	Training Loss	Training Accuracy	Validation Loss	Validation Accuracy
Experiment 1	0.1	8	4.0284	0.5	0.6931	0.5
Experiment 2	0.1	16	0.8803	0.5	0.6931	0.5
Experiment 3	0.01	8	0.6936	0.5	0.6931	0.5
Experiment 4	0.01	16	0.7104	0.5	0.6931	0.5
Experiment 5	0.001	8	0.6932	0.5	0.6933	0.5
Experiment 6	0.001	16	0.6932	0.5	0.6933	0.5

Conclusion

In this homework, we aimed to tackle the task of gender classification using transfer learning with the VGG-16 architecture. Despite conducting several experiments with different combinations of learning rates and batch sizes, the results did not demonstrate meaningful learning or improvement in the model's accuracy.
The experiments revealed that regardless of the learning rate (0.1, 0.01, or 0.001) and batch size (8 or 16), the model's training and validation accuracies remained at 0.5, indicating random guessing. These findings suggest that factors beyond these hyperparameters need to be considered to enhance the model's performance. The VGG-16 architecture, although widely used in computer vision tasks, may not be well-suited for gender classification in this specific dataset. Furthermore, the dataset itself may require additional preprocessing steps to improve the model's ability to learn discriminative features. Techniques such as face alignment, normalization, or

augmentation could enhance the dataset's quality and provide more informative patterns for gender classification.

Moreover, the experiments highlighted the limitations of the chosen subset of the CelebA dataset. It is possible that the dataset's diversity, image quality, or label accuracy contributed to the model's inability to learn effectively. In conclusion, this study revealed that the chosen hyperparameters, learning rate, and batch size alone were insufficient to achieve accurate gender classification using the VGG-16 architecture and the provided subset of the CelebA dataset. Further investigations should focus on exploring alternative architectures, improving dataset quality and preprocessing techniques, and potentially incorporating more advanced techniques to enhance the model's performance.