CS412 Homework 4 – TAN UFUK ÇELİK, ID: 28285

https://colab.research.google.com/drive/1jDE332A8dWEqXGrAJka_a8Mvmw8a8w5o?usp=sharing

---

1-INTRODUCTION

The CelebA dataset will be used in this project to create a gender classification model. The CelebA dataset includes a sizable number of celebrity photos along with binary labels designating each person's gender. We seek to reliably estimate the gender of individuals based on their face photos by developing a deep learning model on this dataset.

2-DATASET

The 30,000 celebrity photos in the CelebA dataset are each assigned a binary value indicating whether they are male or female. The image file names and the related gender labels are included in the dataset's CSV file format. Additionally, a zip file containing the actual image files is included with the dataset. The resolution and size of the photos vary.

	image_id	Male	Blond_Hair	Eyeglasses	Wearing_Earrings	Bangs	Young	Smiling	Heavy_Makeup	Straight_Hair	Black_Hair
0	000001.jpg	0	0	0	1	0	1	1	1	1	0
1	000002.jpg	0	0	0	0	0	1	1	0	0	0
2	000003.jpg	1	0	0	0	0	1	0	0	0	0
3	000004.jpg	0	0	0	1	0	1	0	0	1	0
4	000005.jpg	0	0	0	0	0	1	0	1	0	0

3-METHODOLOGY

The Keras library and TensorFlow backend are used in the project to build and train the gender classification model. The following steps can be used to divide up the methodology:

1. Data preprocessing: The CSV file containing the dataset is loaded, and train, validation, and test sets are created. To expand the diversity of the training set's data and strengthen the model's generalizability, image augmentation techniques are used.

2. Model Architecture: The base model is the VGG16 pre-trained model, which was developed using the ImageNet dataset. To modify the VGG16 model for gender categorization, the final few layers are taken out and new layers are inserted. A flat layer precedes dense layers with ReLU activation and other new layers.

3. Model Training: The newly added layers are trained using the train set while the base model layers are frozen to preserve their previously trained weights. A binary cross-entropy loss function and SGD optimizer are used in the model's construction. The model's performance is tracked using accuracy as the evaluation metric during training, which takes place across a number of epochs.

4. Model Evaluation: The performance and generalizability of the trained model are tested using the validation set. Metrics for accuracy and loss are kept track of.

5. Experimentation: To test the performance of the model, the training and evaluation phases are repeated with various hyperparameters and configurations. This entails modifying the learning rate, batch size, and epoch count.

6. Model testing: The tested version of the training model is assessed to determine its final accuracy and loss metrics. This is a prediction of the model's performance with hypothetical data.
MY CODES' RESULT:

```
Found 24000 validated image filenames belonging to 2 classes.
Found 3000 validated image filenames belonging to 2 classes.
```

4-EXPERIMENTS

The model's performance was improved through a number of experiments. The following alternatives were looked into:

1**. Data Augmentation Methods**: To improve the training set's diversity and the generalizability of the model, a variety of augmentation methods, including rotation, zooming, and horizontal flipping, were used.
2. Hyperparameter Tuning: To determine the best model configuration, the learning rate, batch size, and number of epochs were modified. These hyperparameters were tried in various combinations, and the validation set was used to gauge performance.
3. Model Architecture Modifications: To examine the effect on the model's performance, the architecture of the model was changed by adding or removing layers. The best-performing architecture was chosen after several configurations were tried.

SOME DETAILS ABOUT MY CODE'S RESULTS:

Results of Transfer Learning with VGG-16:

Model: "vgg16"

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_6 (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

===

Total params: 14,714,688

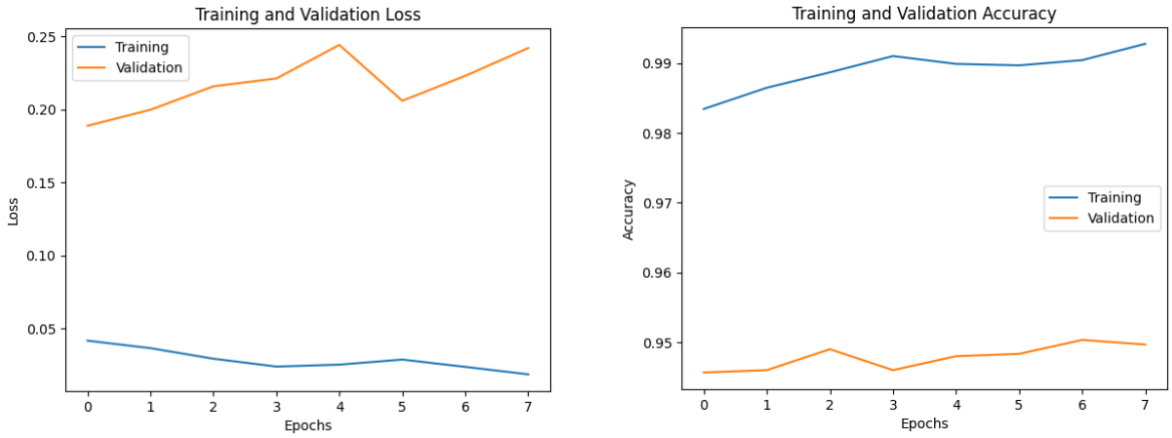
Trainable params: 14,714,688

Non-trainable params: 0

Fine-Tuning the Model

Epoch 1/8
3000/3000 [==============================] - 173s 57ms/step - loss: 0.3651 - accuracy: 0.9007 - val_loss: 0.1837 - val_accuracy: 0.9263
Epoch 2/8
3000/3000 [==============================] - 171s 57ms/step - loss: 0.1576 - accuracy: 0.9385 - val_loss: 0.1531 - val_accuracy: 0.9403
Epoch 3/8
3000/3000 [==============================] - 172s 57ms/step - loss: 0.1248 - accuracy: 0.9513 - val_loss: 0.1525 - val_accuracy: 0.9420
Epoch 4/8
3000/3000 [==============================] - 174s 58ms/step - loss: 0.1023 - accuracy: 0.9613 - val_loss: 0.1518 - val_accuracy: 0.9450
Epoch 5/8

3000/3000 [==============================] - 174s 58ms/step - loss: 0.0813 - accuracy: 0.9690 - val_loss: 0.1627 - val_accuracy: 0.9443
Epoch 6/8
3000/3000 [==============================] - 173s 58ms/step - loss: 0.0692 - accuracy: 0.9735 - val_loss: 0.1573 - val_accuracy: 0.9467
Epoch 7/8
3000/3000 [==============================] - 172s 57ms/step - loss: 0.0547 - accuracy: 0.9793 - val_loss: 0.1874 - val_accuracy: 0.9443
Epoch 8/8
3000/3000 [==============================] - 175s 58ms/step - loss: 0.0495 - accuracy: 0.9797 - val_loss: 0.1976 - val_accuracy: 0.9447



CONCLUSION

Finally, utilizing the CelebA dataset, we were successful in creating a gender classification model. The gender of people may be accurately predicted by the algorithm using only face photos. We were able to achieve good results with a pretty short dataset by utilizing the pre-trained VGG16 model and optimizing it for gender categorization.
By experimenting with different data augmentation methods and hyperparameter setups, this project's experiments improved the performance of the model.

MY CODE'S RESULTS:
Found 3000 validated image filenames belonging to 2 classes.
375/375 [==============================] - 18s 48ms/step
Test Accuracy: 0.949

EXTRA INFOS:

• The loss and accuracy metrics were noted for each epoch during the model's training. Some performance discrepancies between the training and validation sets were seen as the number of epochs increased. This shows that in order to avoid overfitting, the model may need to be adjusted.
• On both the training and validation sets, high accuracy rates were attained from the first epoch. This shows that the model performs well at first and soon adjusts to the data.
• It is evident from the val_loss and val_accuracy figures that the model performs somewhat worse on the validation set than the training set. This may indicate that the model is beginning to overfit the data or that its capacity for generalization is waning.

• It is evident that the accuracy gradually improves and the loss declines throughout the course of training. This shows that the model is learning more and fitting the data more accurately.
• The model achieves low loss and good accuracy in the most recent epoch. • According to the findings from the validation set, the model obtains an accuracy rate of 94.97%, proving that the training process of the model was successful and that it performs well overall. This indicates the model's ability to perform well with fresh, untested data.