Pelinsu Sarac, 28820 Sabancı University

CS 412 HW-4 SUMMARY REPORT

Link to Colab Notebook:
https://colab.research.google.com/drive/1mY3wW0GBZYgLBSSiH3QsFCaSVieJoixT?usp=sharing

Introduction

In this homework, the main problem is to create a gender classifier for images utilizing a pre-trained deep neural network model, namely VGG-16. In other words, the main theme of the homework is applying transfer learning for a binary classification task. To achieve this, followed path is as follows: Loading the dataset to be used and narrowing it down according to the main task (gender classification), splitting the dataset into train, validation, and test sets as well as augmenting them, modifying the VGG-16 model for classification by changing the top layer to a new binary classification head, fine-tuning the new model and finally testing it on the test set.

Dataset

The dataset used in this homework is a subset of the *CelebA* Dataset, which covers many celebrities' faces with multiple attributes about them. The subset that was worked on has 30.000 RGB images along with the attribute information. However, only image ID and "Male" attributes were used in this homework, as the task is to classify according to gender.

Some example images from the set are as follows: (3 Females above, 2 Males below)

Methodology

1. Loading Dataset and Narrowing It Down

The dataset was provided in two pieces: a compressed file of images and attributes as a CSV file. First, these two pieces were uploaded to Google Drive. Then, the image file was decompressed, and attribute data was narrowed down to “image ID” and “Male” attributes, as the main objective of the homework is to do a gender classification. After that, a brief statistical summary was given to see the class distribution. The count of the two genders is not equal, but the imbalance is tolerable.

2. Splitting the Dataset into Train-Validation-Test and Augmentation

The dataset was split into 3 sets: train, validation, and test. The sizes of the sets as proportion are 80%, 10%, and 10% respectively. These proportions correspond to 24.000 images for training, 3000 for validation, and 3000 for testing. After the split, the first training and validation sets were augmented by using *ImageDataGenerator* from the Keras library. The class mode parameter was given as binary, and the batch size was chosen as 8 for the sake of speed. Augmentation of the test set was done right before the model evaluation.

3. Transfer Learning with VGG-16

To be able to utilize the pre-trained VGG-16 model, it was first imported from the Keras library. This model was previously trained with the *ImageNet* dataset, which consisted of over a million images with 1000 classes. For the sake of benefitting from the pre-trained model, the vast majority of network layers were left untouched. To achieve this, while importing the model, the top layer was discarded.

When the model was completely imported, there are 14,714,688 trainable parameters over all layers. Because the aim is not to modify the hidden layers, i.e., these parameters, they had to be “frozen”. This was achieved by traversing over all layers and turning off their “trainable” attributes.

Then, we must be able to add a new top layer that can classify genders, in other words, a layer that can do binary classification. To be able to use VGG-16 and the new layer as a whole, a function is created. This function accepts a base model, in our case it will be VGG-16, as a parameter. Inside the function, it was made sure that the inputs first go through the VGG-16 layers, flattened, and only then go through the final layer. The final layer was decided as a **single output node with sigmoid activation** as it was advised in the lecture slides.

Experiments

This section is mainly about the fine-tuning of the model described previously. Among the hyperparameters, epoch size was given in the homework description as 10, and batch size was chosen as 8 for the sake of speed and to avoid memory-related errors. Thus, fine-tuning was mainly about tuning the learning rate, as stochastic gradient descent was chosen as the optimizer. It should also be noted that binary cross entropy was chosen to measure loss.

3 learning rates were used: 0.1, 0.01, and 0.001.

- Learning rate = 0.1

EPOCH	LOSS	ACCURACY	VALIDATION-LOSS	VALIDATION-ACCURACY
1	900.0242	0.9118	876.5392	0.9247
2	526.2103	0.9450	926.5124	0.9303
3	347.1241	0.9584	870.2325	0.9260
			.	
			.	
			.	
10	39.2536	0.9900	830.1307	0.9377

- Learning rate = 0.01

EPOCH	LOSS	ACCURACY	VALIDATION-LOSS	VALIDATION-ACCURACY
1	93.8577	0.9102	86.4102	0.9263
2	53.4120	0.9441	107.4083	0.9243
3	33.2743	0.9601	76.0810	0.93
			.	
			.	
			.	
10	4.6087	0.9891	84.7145	0.941

- Learning rate = 0.001

EPOCH	LOSS	ACCURACY	VALIDATION-LOSS	VALIDATION-ACCURACY
1	9.4303	0.9095	12.3402	0.915
2	5.3433	0.9442	7.5232	0.9357
3	3.2822	0.9589	7.8167	0.9373
			.	
			.	
			.	
10	0.3887	0.9900	8.5336	0.9307

Among these trials, validation accuracy does not differ significantly from model to model. However, validation loss, on the other hand, shows significant changes. According to the tables above, validation loss is minimized with a learning rate of 0.001.

Before deciding on the final model, one last experiment was done. The learning rate was again chosen as 0.001. However, in this experiment, the last hidden (convolutional) layer was also trainable. Here are the results:

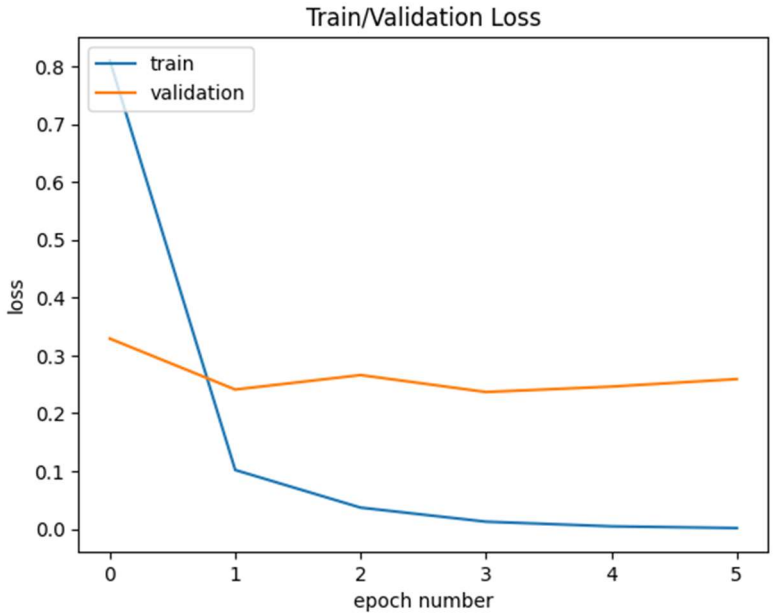
EPOCH	LOSS	ACCURACY	VALIDATION-LOSS	VALIDATION-ACCURACY
1	0.8366	0.8871	0.3187	0.8940
2	0.1093	0.9599	0.2527	0.9283
3	0.0421	0.9853	0.2416	0.9293
			.	
			.	
			.	
10	3.9263e-04	1	0.2909	0.9437

As can be seen from the table, while there is a slight increase in validation accuracy, there is a significant decrease in validation loss. If a decision must be made according to the validation loss, this model is the far best. However, one must also note that training set accuracy reaches 1 at the end of 10 epochs, which signals that the model is highly prone to overfitting. Thus, while training the final model, there should be a precaution for overfitting. It was achieved by a callback mechanism called "Early Stopping" which stops the training if there is no improvement in the monitored metric. In our case, early stopping stops the training if there is no decrease in validation loss.

Early stopping can take many parameters. One that it takes here is patience, which corresponds to how many recent epochs it should look to check whether there is an improvement or not. It was roughly decided to be 2 by observing the progress of validation loss without early stopping. The early stopping mechanism stopped the training after 6 epochs. Here are the results:

EPOCH	LOSS	ACCURACY	VALIDATION-LOSS	VALIDATION-ACCURACY
1	0.8097	0.8870	0.3290	0.9110
2	0.1019	0.9634	0.2410	0.9273
			.	
			.	
			.	
6	0.0016	0.9999	0.2590	0.9413

Here is the learning curve obtained from the final model:



It can be noted that the curve suggests that the model is learning fast, thus early stopping is indeed a good idea for it.

To conclude the section, here is a summary table for the experiments:

EXPERIMENT NAME	VALIDATION ACCURACY AT LAST EPOCH
LEARNING RATE = 0.1	0.9377
LEARNING RATE = 0.01	0.9410
LEARNING RATE = 0.001	0.9307
LR = 0.001 + LAST HIDDEN LAYER IS TRAINABLE	0.9437
LR = 0.001 + LAST HIDDEN LAYER IS TRAINABLE + EARLY STOPPING	0.9413

Conclusion

To utilize transfer learning for gender classification, as a pre-trained model VGG-16 was used. After discarding the original classification layer (the final layer), a new binary classification layer is added, which consists of a single node with sigmoid activation. During fine-tuning, 0.001 as the learning rate has given the best results when the last hidden layer was also trainable. A final model was trained with early stopping to prevent overfitting. Finally, this model was tested with the test set, which was first augmented, and then the results are evaluated. According to the *evaluate* function provided by Keras, the **loss is 0.2779** and the **accuracy is 0.9453**.