Deep Learning: Assignment 3

by

Palaskar Adarsh Mahesh

B20EE087

April 15, 2023

1 Question 1

In this question we have performed multitask learning on the CelebA dataset. The dataset is divided into a training set of 162,770 images, a validation set of 19,867 images, and a testing set of 19,962 images. The 40 attributes in the dataset include binary attributes such as "male" and "smiling", as well as continuous attributes such as "facial hair" and "eyeglasses", which take on values between -1 and 1. The attribute annotations are provided in a text file with one row per image and one column per attribute.

The model is trained on 8 of the 40 attributes using a ResNet18 backbone, described in the following sections.

1.1 Choice of attributes, Backbone, and other parameters and the associated reasons:

1.1.1 Choice of Attributes:

For this task, I have chosen to predict the following 8 attributes from the CelebA dataset:

1. Male/Female
2. Smiling
3. Young
4. Wearing Glasses
5. Wearing Hat
6. Wearing Necklace
7. Wearing Earrings
8. Wearing Lipstick

I have chosen these attributes because they have a good balance of positive and negative examples in the dataset and are also visually distinct from each other.

1.1.2 Choice of Backbone:

As for the backbone architecture, I have chosen to use the ResNet18 architecture. This is because ResNet has been shown to have better performance than VGG on image classification tasks while also having a smaller number of parameters, which makes it faster to train.

I will be using transfer learning to train the model, which involves taking a pre-trained ResNet18 model and fine-tuning it on the CelebA dataset. Specifically, I will be using a ResNet18 model pre-trained on the ImageNet dataset, which is a large dataset of natural images. By using a pre-trained model, I can leverage the knowledge learned from the ImageNet dataset to improve the performance of my model on the CelebA dataset.

1.1.3 Choice of other parameters:

The CelebA dataset contains a lot of background information that is not relevant to the task of attribute classification. Therefore, centre cropping is used to extract a smaller region of interest from the larger image, focusing on the face of the celebrity. Normalization is applied to standardize the pixel values across all images, making it easier for the model to learn meaningful patterns in the data. The Adam optimizer is used due to its adaptive adjustment of the learning rate based on the gradient magnitude, which enables the model to converge faster and more reliably than traditional optimization techniques like SGD. The cross-entropy loss criterion is used to measure the difference between the predicted probabilities and the true class labels, encouraging the model to output high probabilities for the correct class and low probabilities for the incorrect class. A learning rate of 0.001 is chosen as a commonly used value that strikes a balance between fast convergence and stable optimization, allowing the model to converge quickly and produce accurate results while minimizing the risk of instability.

1.2 Model Architecture:

The MultiTaskResNet18 model architecture is a ResNet18-based multi-task learning model for the CelebA dataset, which contains images of celebrities annotated with various attributes such as gender, age, and presence of accessories. Here is a detailed explanation of the model architecture:

The first layer of the model is the ResNet18 backbone network, which has four blocks of convolutional layers followed by a fully connected layer. The ResNet18 model is pre-trained on the ImageNet dataset and has learned to extract useful features from images. The last fully connected layer of the ResNet18 model is replaced with an identity layer, which passes the output of the previous layer unchanged. The output of the ResNet18 model is then passed through eight separate linear layers for the eight attributes. Each linear layer predicts one of the eight attributes in the CelebA dataset.

During training, the model learns to predict all eight attributes simultaneously using a multi-task loss function. The loss function takes into account the errors in all eight tasks and adjusts the model's weights accordingly to minimize the total loss. By sharing the ResNet18 backbone network across all tasks, the model can leverage the shared features to improve the performance of each task.

1.3 Task-Wise Accuracy:

After Training the model for 5 epochs and testing it on the test split of the dataset, the following Task-Wise Accuracy numbers were observed.

Attribute (Task)	**Accuracy**
Gender	0.9688
Smiling	0.9115
Young	0.8638
Wearing Glasses	0.9795
Wearing Hat	0.7972
Wearing Necklace	0.6718
Wearing Earrings	0.8152
Wearing Lipstick	0.9877

1.4 Overall Accuracy:

The overall accuracy for all the 8 attributes on which the model was trained was found to be equal to 0.8173.

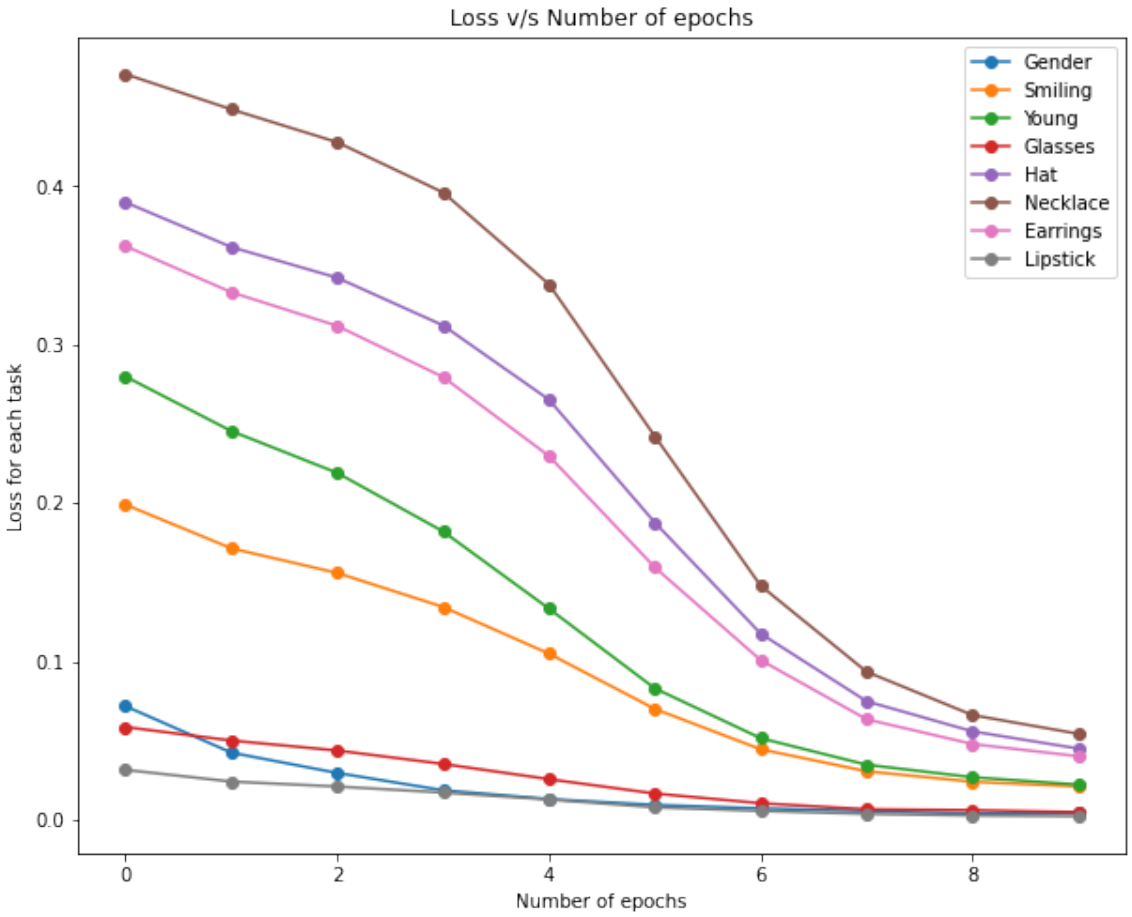

Figure 1: Losses for each task for each Epoch

2 Question 2 (Bonus Question)

2.1 Drop rate for each task using metrics given in the paper:

I will be calculating drop rates using two metrics of the given 4 metrics in the paper.

2.1.1 Network Depth:

Tasks that can be learned using a shallow network (fewer layers) are less likely to interfere with the learning of other tasks, whereas tasks that require a deeper network (more layers) are more likely to cause the negative transfer. Therefore, the drop rate for each task is calculated based on the ratio of its required depth to the total depth of the network. The drop rate is then used to determine the probability that the task will be dropped during training, with higher drop rates indicating a higher probability of dropping the task. By dynamically dropping tasks during training, the proposed approach can mitigate negative transfer and improve the overall performance of the model.

Now, since for all tasks, our model uses all the layers, the drop rate for all the tasks will be equal to 1.

```
Drop Rate based on Network Depth for Gender: 1.0
Drop Rate based on Network Depth for Smiling: 1.0
Drop Rate based on Network Depth for Young: 1.0
Drop Rate based on Network Depth for Wearing Glasses: 1.0
Drop Rate based on Network Depth for Wearing Hat: 1.0
Drop Rate based on Network Depth for Wearing Necklace: 1.0
Drop Rate based on Network Depth for Wearing Earrings: 1.0
Drop Rate based on Network Depth for Wearing Lipstick: 1.0
```

2.1.2 Task Incompleteness:

The task incompleteness metric measures the percentage of incomplete or partially labelled instances for each task. The drop rate for each task is then calculated based on the ratio of the task's incompleteness to the total incompleteness across all tasks. The idea behind this metric is that tasks with a higher percentage of incomplete instances are more likely to cause negative transfer, as incomplete labels may confuse the model and affect the learning of other tasks.

```
Relative Incompleteness for Gender: 0.6260592080230313
Relative Incompleteness for Smiling: 1
Relative Incompleteness for Young: 0.8617230068700827
Relative Incompleteness for Wearing Glasses: 0.889436204908604
Relative Incompleteness for Wearing Hat: 1
Relative Incompleteness for Wearing Necklace: 1
Relative Incompleteness for Wearing Earrings: 1
Relative Incompleteness for Wearing Lipstick: 0.765159548862277
```

Figure 2: Relative Incompleteness when k = 10

2.2 Step-by-step explanation on how to calculate the above drop rates:

2.2.1 Network Depth:

1. In our model, all the tasks are present at the same depth, which is the depth of the entire network.
2. To calculate the network depth, we calculate the depth of the ResNet18 backbone.
3. We then added one for the last linear layer we used for different tasks.
4. Now, since all the task depths are equal to the network depths, the drop rates corresponding to all the tasks will be equal to one, as reported in the previous section.

2.2.2 Task Incompleteness:

1. Using the loss of 1^{st} epoch and k^{th} epoch of each task, a ratio of each task's current loss to its initial loss is calculated.
2. It shows how much the task has learned.
3. Then the expected value across all the loss ratios is calculated during each epoch.

4. The Relative Incompleteness is then given by the minimum of 1 and loss ratio by expected value.

2.3 Observations:

Since we have implemented the Task Incompleteness dropout in the DST algorithm, the observations will be based on this metric:

Task incompleteness can negatively impact the performance of multi-task learning models, as incomplete labels may confuse the model and affect the learning of other tasks. The impact of task incompleteness on multi-task learning performance depends on the severity and distribution of the incomplete labels across tasks. Task incompleteness can be mitigated by adapting the drop rate for each task during training, using a metric based on the percentage of incomplete or partially labelled instances for each task. The proposed approach of using dynamic task dropping based on task incompleteness can improve the performance of multi-task learning models. In this case, we observe that after the tenth epoch, Gender, Young, Wearing Glasses, and Wearing Lipstick have a relative incompleteness of less than 1, and thus they will be dropped according to a probability distribution when the model is trained using DST, and thus reducing their negative impact and improving accuracy. The observed improvements are reported in the next section.

2.4 DST Algorithm Implementation:

The DST algorithm is implemented using the task incompleteness metric, which was implemented according to the explanation in section 2.2 and integrated into the train function to calculate the relative incompleteness for each epoch.

The following improvements were observed:

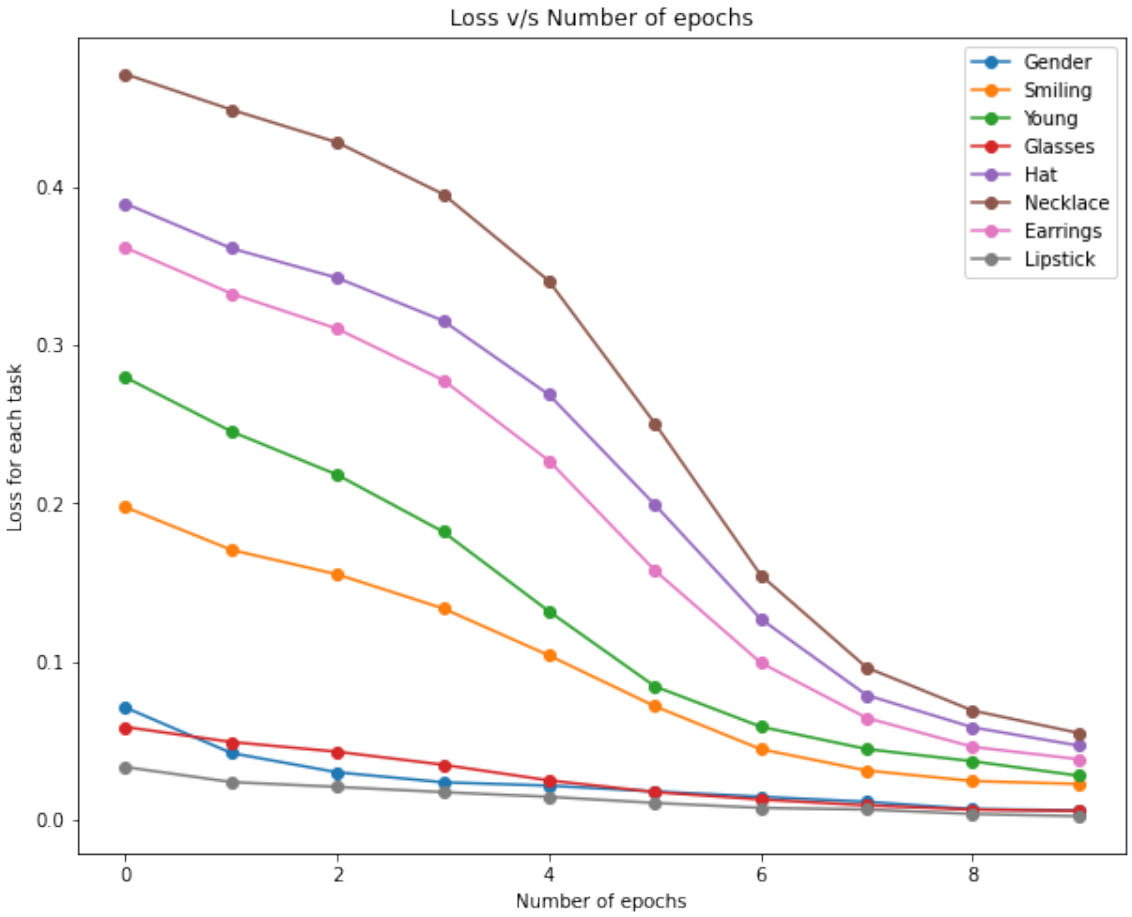

Figure 3: Losses for each task for each Epoch

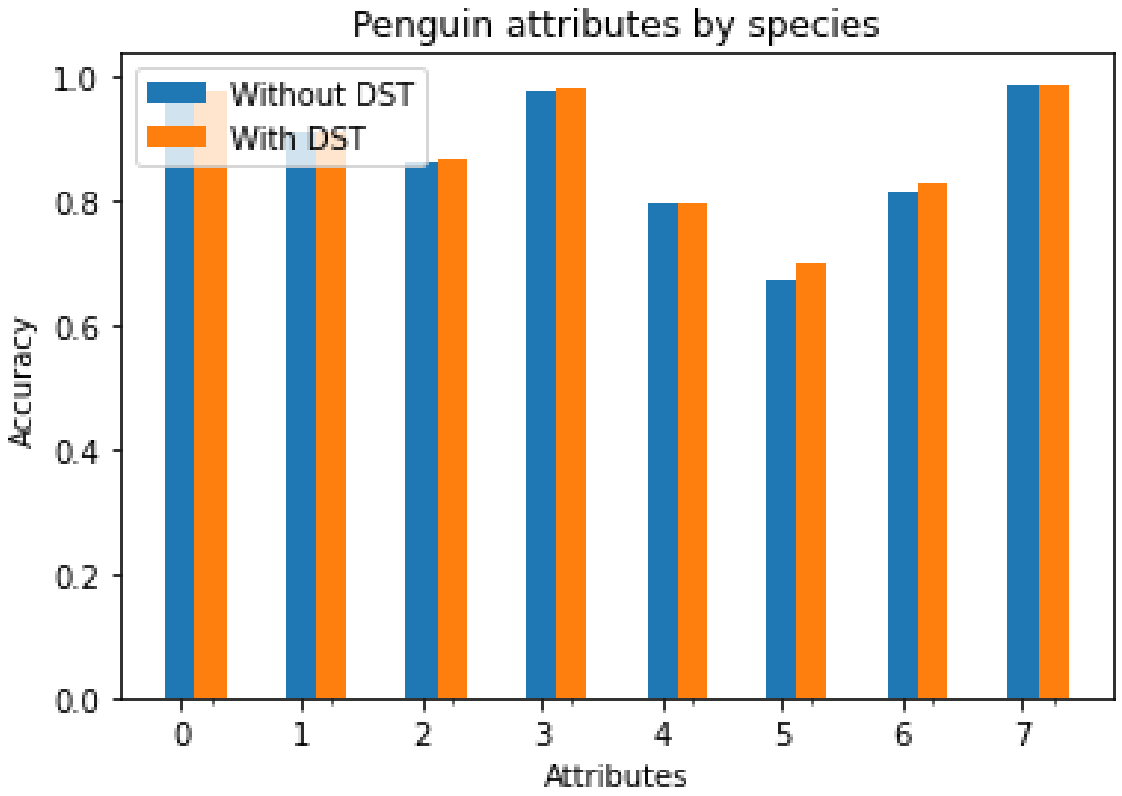

Figure 4: Comparing Task Wise Accuracy

The overall accuracy also increases from 0.8744 to 0.8820, which is roughly an increase of 1 percent, only using one metric for dropout. This can be enhanced by using all the metrics and increasing the number of training epochs, which was constrained due to limited computation available on Google Colab.