Deep Hallucination Classification

Gavrila Alexandru

1. The task

The task for this project is classifying deep hallucinated images generated by deep degenerative models.

2. Dataset

The dataset consists of 30,000 image files used for training, 5,000 used for validation and 5,000 used for testing. Each image is a 32x32x3 hallucinated image. These images are split between 8 classes. All images are labeled with their class in a .txt file. The dataset is already split into a training set, a validation set and a test set. The .txt files for the training and validation sets contain the names of the image files on the first column, followed by the label on the second column, while the test set only contains the names of the image files.

3. Data Processing

The images were read and kept in a numpy array for each of the sets and normalized (dividing by max value). The labels were extracted from the .txt files and added to another numpy array (for the training and validation sets). I tried using data augmentation but it yielded lower accuracy. I suspect this is because the data was generated and not photographed, meaning flipping, rotation and translation would tamper with the features based on which the data was generated. The images were used in the RGB format.

4. Model

For the model I used Keras and TensorFlow API to create a MLP. I did the calculations on the GPU and CPU using NVIDIA's CUDA computation because the training part of this task would have taken too long running only on CPU. In hindsight, the process of successfully installing TensorFlow GPU took a third of the time allocated for this project.

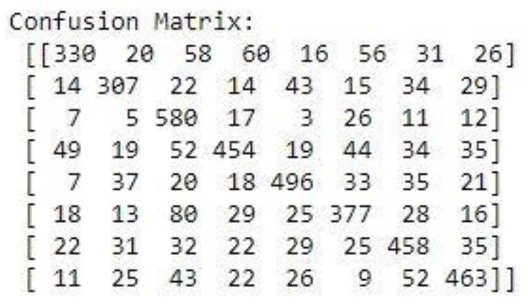
I used a sequential model consisting of mainly convolutional layers. The model layout is as follows: Input, Conv2D, Conv2D, Conv2D, BatchNormalization, MaxPooling, Dropout, Conv2D, Conv2D, Conv2D, BatchNormalization, MaxPooling, Dropout, Conv2D, BatchNormalization, MaxPooling, Dropout, Dense, BatchNormalization, Dropout, Dense (Output). I tried multiple combinations of layers and I settled on this layout which resembles AlexNet. The Batch Normalization and Dropout were used to prevent overfitting. The optimizer and loss function I found to work best after trying multiple combinations are Adam for the optimizer and Categorical Crossentropy for the loss function.

5. Training and Hyperparameters

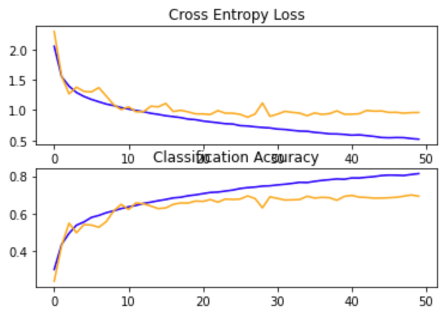
The model was trained for 40 to 50 epochs in batches of 64. The batch size was finetuned to this value. The optimizer and loss function parameters that worked the best were the default ones. The number of filters on the convolutional layers were: 32, 32, 64, 64, 128, 256, 128. The kernel size that worked the best is 3 as a bigger size (5,7, etc.) would be too large. I tried using vertical and horizontal kernels but they yielded worse results. The padding used was one matching the kernel size and the activation function was ReLu (I also tried tanh but it worked worse). The dropout rate was the maximum advisable (0.5) as to prevent overfitting. I tried using multiple combination of layer weight regularizers using l1, l2 and l1_l2 norms but none of them proved to bring any improvement to the results.

6. Accuracy and Confusion Matrix

The validation accuracy ranged from 62% to 71% during the tuning of the model and the hyperparameters. Based on the graph there was very little difference between the training accuracy and the validation accuracy. The images below are the values for the highest scoring version of the model.



```
Confusion Matrix:
 [[330  20  58  60  16  56  31  26]
 [ 14 307  22  14  43  15  34  29]
 [  7   5 580  17   3  26  11  12]
 [ 49  19  52 454  19  44  34  35]
 [  7  37  20  18 496  33  35  21]
 [ 18  13  80  29  25 377  28  16]
 [ 22  31  32  22  29  25 458  35]
 [ 11  25  43  22  26   9  52 463]]
```



7. Conclusion

Considering the number of layers (~21) and the number of trainable parameters (1,200,000) this can be considered a deep CNN. I think it performed well for the task, given that the average validation and test accuracies are of about 70%. Given the nature of the dataset and the other scores obtained in the challenge I think a higher accuracy model for this task would be very to achieve.