Artificial neural networks and deep learning

Homework 1: Image classification

Motta Dennis Ostrovan Eugenio

November 22, 2020

1 Initial approach

Our first step in approaching the problem was to carefully inspect the requirements and the provided data. We took note of the number of available images, the split between train and test set, the size of the images and we inspected some of them (around 100) to get an idea of what we were working with.

2 Building a prototype

The next step was to create a working prototype able to produce a valid submission using a very simple model. In this step the goal was to resolve any issues there might arise with the kaggle platform and other tools we were using or with the submission file format. The performance in this step was very close to random, as we expected.

3 First model

Once we had a working notebook, we went back to the model. We attempted various forms of the standard architecture which combines multiple groups of convolutions, activations and pooling layers followed by a fully connected layer and an output layer. We enabled data augmentation while also experimenting with different values of batch size, number of convolution filters, number of neurons in the fully connected layer, learning rate, training duration. We achieved only slightly better than random performance, this because we initially thought that a model with a number of parameters in the order of ~100K would have been enough. We discovered later that we were wrong.

4 Transfer learning

Next we used transfer learning from the VGG16 model. In this phase we hoped to achieve better performance and also get a better idea on how a well performing model is made. We split the data in training-set and validation-set to have a better understanding of the behaviour of the model. We also introduced callbacks for Tensorboard visualization, checkpoint saving, early stopping on the validation loss and a callback for reducing the learning rate on plateaus of validation loss. In this phase we achieved better test accuracy, around 0.75.

5 2 phase model

We had the intuition that among the three classes (nobody, all, somebody wearing a mask) the "nobody" class is more different than the other two, so we made a new architecture with two models: the first model classifies images into "nobody" and "some or all", then if the image is classified as "some or all" we use a second model to distinguish between the two. We adapted the training data accordingly. This improved the performance, but only a little (0.84). We also tried all possible configurations of this approach (selecting "all" first and "some" first). In all cases the class "some" proved the most difficult to distinguish.

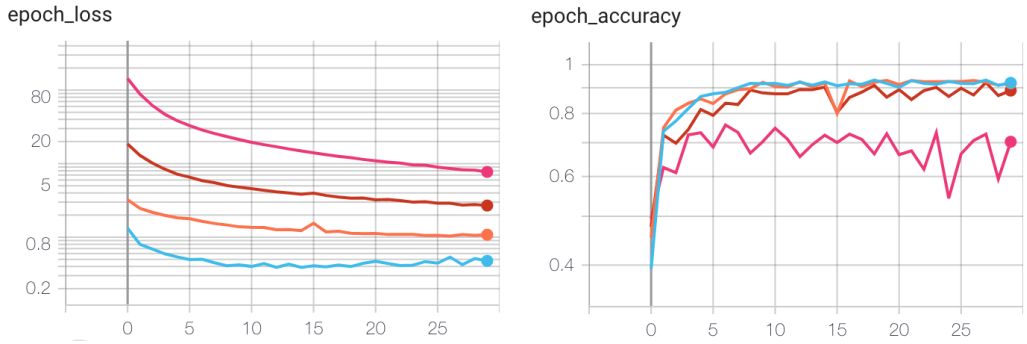
Provided code
The code for this model is provided in the *2phaseSolution.ipynb* file.

6 Training on all the data

Then while studying the notes of the lectures we remembered a suggestion that the professor made: we could use weight decay with an L2 regularizer to avoid overfitting, find a balanced hyperparameter for the decay using the validation set and then train on all the data available. This allowed us to reach a performance of around 0.92

Provided code.
The code for finding the right hyperparameter for the L2 regularizer is provided in the *homework1-L2-regularizer.ipynb* file. While the actual model used for the Kaggle Final Score is provided in the *homework1* file.

Figure 1: Loss and accuracy over epochs for evaluating regularization hyper-parameter(log scale)



Blue - 0.0001; Orange - 0.001; Brown - 0.01; Pink - 0.1

7 Additional analysis

While developing the various models we developed a few solutions to understand better both the input that we gave to the model and the output that it gave us. We displayed the images transformed by the data augmentation part to understand visually the changes done. But also, when the prediction was made on the test-set, we displayed some samples of the images with their respective prediction. We noticed that the most common errors are between the "somebody wearing a mask" class and the "all" class, that's understandable since it's easy to misinterpret other parts of images (like facial hair or a covering perspective) as a mask.

Figure 2: Display one image from the test set and the model's prediction on it

8 Conclusion

This was a very instructive experience, we learned many practical aspects of deep learning, not only on the use of specific tools, but also on the methodology behind solving a problem using deep learning. We learned to debug neural networks and to experiment in an organized manner in order to find appropriate parameters.