Assignment 2 - Hacking

Inés Broto Clemente

December 2021

1 Deliverables

- **Error Metric:** Accuracy
- **Target Value for Error Metric:**
  - **Tiny ImageNet**: 50% on Training and 30% on Test
  - **ImageNet**: 95% on Training and 75% on Test
- **Achieved Value for Error Metric:**
  - **Tiny ImageNet**: 51.093% on Training and 7% on Test
  - **ImageNet**: -

2 Project Development

2.1 Translation

Although it seemed an easy task taking into account that I was planning to do it with static map or translator, the huge size of the data was difficult to handle through my computer. Moreover, when trying to use some translating libraries to directly translate the labels the `too many requests` error was raised. In the end, I managed to translate the Tiny Image Net labels using the `GoogleTranslator` function from the `deep_translator` library. When scaling up, it will also work for the ImageNet dataset even though I haven't performed experiments with it in the end.

2.2 Data

After finishing the first pipeline approach and getting some acceptable values for the training accuracy using the Tiny ImageNet Dataset I tried to scale up my model and improve Test accuracy training it with the whole ImageNet Dataset. However, after lots of tries I had to change my strategy since I don't have enough computational recources as well as not enough storage available and I was running out of time. Since the main goal of my project was to build a model but not that much the data used to train it (altough I know it's also very important), I decided to move on and focus on buliding a new model based on the Efficient Net arquitecture.

2.3 First Approach: Resnet 18 arquitecture

This CNN architecture adds residual connections between layers to relieve very deep networks training [HZRS15]. After creating the first pipeline, I wanted to see how learning rate and dropout could influence the network's performance and also see how Train and Test errors where doing so far, that's why I performed some experiments. Since I have a limited computational power I only performed 10 epochs to see how these values affected the first part of the training and be able to narrow down the possible values for further experiments. A **learning rate** equal to **0.1** and a **dropout probability** equal to **0.2** turned out to be the optimal values.

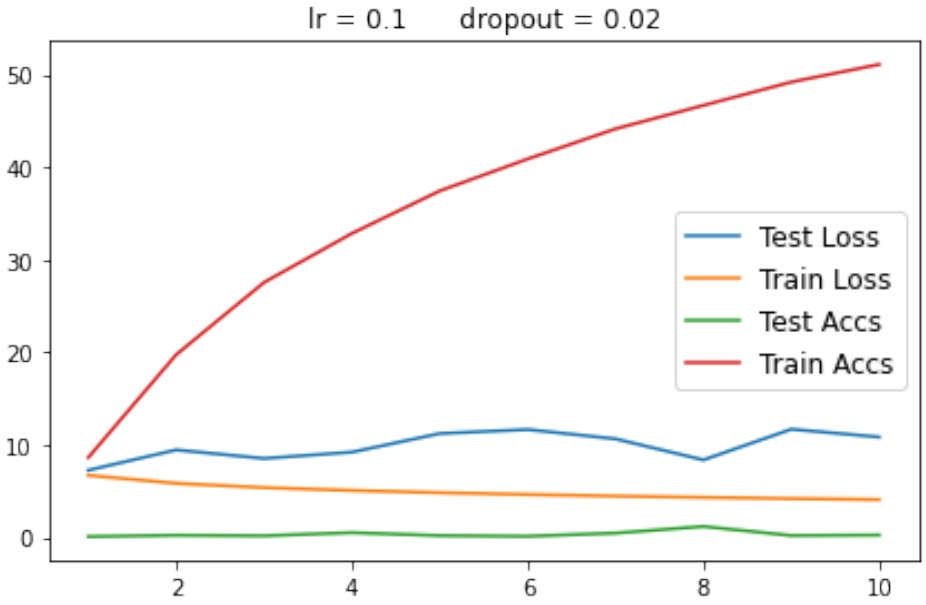

Figure 1: Resnet Performance for the first 10 epochs

2.4 Second Approach: Efficient Net

As suggested, I tried to improve the ResNet performance rethinking the convolution scaling up method. As is suggested in the paper [TL20], I tryied to scale up all the dimensions of the convolution at once, so that the network doesn't just gains insight on depth, width or resolution separately but on all of them at a time. Altough the whole pipeline worked after some rough days of coding, the Test error seems not to improve. I'm pretty sure that's because of the lack of data but since I don't have neither enough computing resources, time to download the dataset or storage to work with it I could not perform any test with more data.

3 Running the application

In my Github reportory you will find all the scripts. Inside the data folder there are the scripts used to prepare the data for training and to translate the labels and, in the same way, the models are stored in the model folder. Finally, the `main.py` trains the model and with the `predict.py` file you can perform inference using the `last_model.pth` model.

4 Updated Work break-down structure

It follows a work break-down structure I've followed so far in order to have a proper planning for the work. Although trying to be as realistic as possible inmy first assignment, this part has taken me more time than expected even if I haven't achieved good results.

Image Classification App

1. Data set collection: Tiny ImageNet 200 - 5h
   (a) Downloading or accessing the data - 2h
   (b) Understanding, translating, cleaning and organizing the data - 3h
2. Network designing and building: ResNet 18 - 9h
3. Network training and fine-tuning - 16h

4. Data set upgrade: ImageNet - 20h (FAILED)
5. Network upgrade: Efficient Net - 23h
6. Network training and fine-tuning - 20h
7. Hacking deliverable and report- 6h
8. Final application -
9. Final report -
10. Final presentation -

Total amount of working hours so far: 99h

5 Next Steps

From now on, I'm planning to wrap up the inference part of the project to build up an app able to perform the main pipeline: get an image and answer with a label for the content on that image. As mentioned before, the accuracy values even on the test sample are not as high as one could desire so that the results will be wrong most of the times. However, this could improve in a future if I manage to train the model with a higher amount of data and also when I learn more about deep learning and get more used to the programming libraries.

References

[HZRS15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[RDS+15] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge, 2015.

[SLJ+14] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions, 2014.

[SZ15] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2015.

[TL20] Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks, 2020.

[YM15] Leon Yao and John Miller. Tiny imagenet classification with convolutional neural networks. *CS 231N*, 2(5):8, 2015.