Model Pruning: Weekly Report 2

Patricia Gschoßmann

1. Weekly Progress

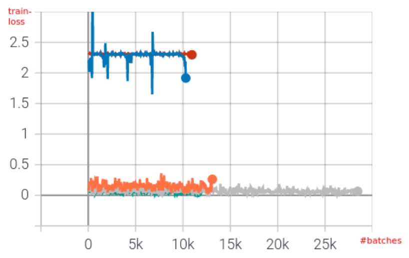
A new attempt to prune the original VGG was executed. Compared to the last approach, the goal now was to prune 35% of the parameters right away[1], since that is half of the parameters that were pruned in task 3 in total. Moreover, an additional early-stopping callback was used during the objective selection to stop the training for each α-value at the right time. Once the validation loss stopped improving for 25 epochs, α is reduced and the training is resumed with the new value.

The implementation was adapted to a more general framework. To execute the objective selection, the user now needs to specify the model type, the path to a pretained model and what percentage of the parameters should be pruned.
Furthermore different types of α-scheduler were implemented. Currently, experiments with the step-, exponential- and multiplicative scheduler are running.

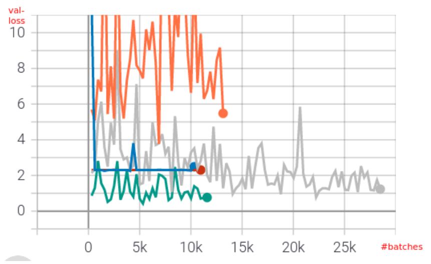
2. Results

Fig. 1 shows the training and validation loss of the model starting from $\alpha = 0.4$. Fig. 2 shows the corresponding accuracy. The model maintained a test accuracy of 90% until $\alpha = 0.4$. At $\alpha = 0.3$ the accuracy only reached 81%. After that, it dropped below 30%. The results are more promising compared to last week, since the accuracy is maintained longer while at the same time more parameters are being pruned. However, the performance is still worse than expected. The figures show, that at $\alpha = 0.3$ – where the first bigger drop in the accuracy occured – training took twice as long as before. In figure 1 one can see, that at $\alpha = 0.2$ the training loss is still very low, whereas the validation loss stays very high and does not converge - an indication that the model is overfitting to the training set. This observation is reflected in the accuracy (see fig. 2): The model achieves a high training, but a low validation accuracy. As a result, the model does not learn at all in the next iterations at $\alpha = 0.1$ and $\alpha = 0$.

[1]Instead of only 25%.



(a) Training loss



(b) Validation loss

Figure 1: Development of train- and validation loss during training for $\alpha = 0.4$ (green), $\alpha = 0.3$ (grey), $\alpha = 0.2$ (orange), $\alpha = 0.1$ (blue) and $\alpha = 0$ (red).

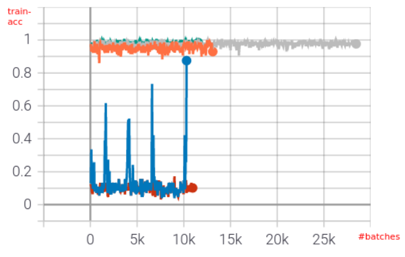
3. Plan

My aim for the following week is to maintain the original performance of the model during pruning. The main goal is to prevent the model from overfitting. I will pursue following attempts:

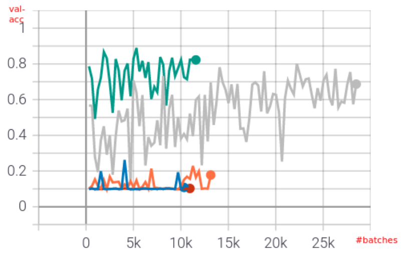
- Prevent overfitting by using a shorter patience duration for the early-stopping callback.
- Prevent overfitting via data augmentation and cross-validation.
- Execute experiments with new α-schedule.[2]

Furthermore, I want to execute the experiments by pruning a larger percentage than 35%[3], since we know from task 3, that maintaining an accuracy of $\sim$90% with in total 65% less parameters is possible.

[2]Already running.
[3]Already running.



(a) Training accuracy



(b) Validation accuracy

Figure 2: Development of train- and validation accuracy during training for $\alpha = 0.4$ (green), $\alpha = 0.3$ (grey), $\alpha = 0.2$ (orange), $\alpha = 0.1$ (blue) and $\alpha = 0$ (red).

4. Summary of new scripts

- `abstract_scheduler.py`: Abstract superclass for all schedulers including an abstract step method.
- `step_scheduler.py`: Decays α continuously by `step_size`. Subclass of `abstract_scheduler.py`.
- `exponential_scheduler.py`: Decays α exponentially by `decay_rate`. Subclass of `abstract_scheduler.py`.
- `multiplicative_scheduler.py`: Decays α by multiplying with `mu`. Subclass of `abstract_scheduler.py`.
- `reduce_on_plateau_scheduler.py`[4]: Decays α when a metric has stopped improving. Subclass of `abstract_scheduler.py`.
- `model_enum.py`: Enum including all available types of models to choose for pruning.[5]
- `scheduler_enum.py`: Enum including all available schedulers to choose for pruning.

[4]Work in progress.
[5]Currently, only VGG is implemented.