Model Pruning: Weekly Report 3

Patricia Gschoßmann

1. Weekly Progress

This week's experiments were all executed to prune 65% of the parameters at once. Multiple approaches were pursued:

- Test with different α-schedules
- Add gaussian noise on half of the training data
- Train as regression task on logits of pretrained model

Apart from this, an important error in the code was found: The parallel branch for convolutional layers should have 1×1 kernels, however, 3×3 sized kernels were used originally. This error was fixed before all of the following experiments.

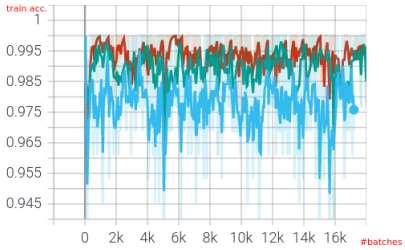
2. Results

2.1. Different α-schedules

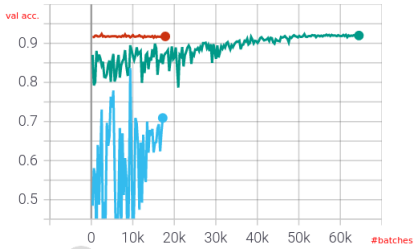
Last week's α-schedule was able to maintain the model's original training accuracy until $\alpha = 0.1$ and the original validation accuracy until $\alpha = 0.4$ (with minor performance drops). In order to reach better results, different α-schedules were tested. One interesting result can be seen in fig. 1, where the model at $\alpha = 0.43$ reached the original accuracy without performance drops after a longer training period. This indicates that the patience variable should be increased, i.e. the model should be trained longer in each iteration.
The experiment is still running, which is why it is not yet possible to say whether the training performance will collapse as in the week before.

2.2. Data Augmentation

Last week's results have shown, that the model starts overfitting during the α-decay. In order to maintain not only the training accuracy, but also the validation accuracy, I decided to add additional data augmentation techniques to the training set. For the following results, gaussian noise was randomly applied to half of the training data.
Figure 2 shows the training and validation accuracy for selected α values between 1 and 0.2.[1] It is not a surprise, that



(a) Training accuracy



(b) Validation accuracy

Figure 1: Development of train- and validation accuracy during training for $\alpha = 0.95$ (red), $\alpha = 0.43$ (turquoise), $\alpha = 0.26$ (light blue). An exponential schedule with a decay rate of 0.05 was used.

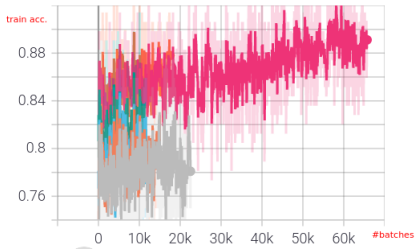
the model could not reach the original model's training accuracy of 92%. Unfortunately, the best validation loss saturated just above 80%. However, what is interesting is, that the model around $\alpha = 0.57$ reached a higher accuracy than the one where α is aroung 0.9. As before, this indicates that the patience variable could be increased. Nevertheless, the results do not look very promising, since the accuracy drops in both cases for lower α values.

2.3. Regression

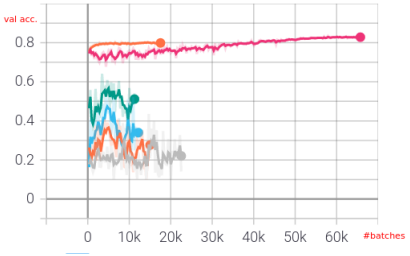
Another approach I pursued to maintain the validation accuracy is training the model as in a regression task. In the following, the logits of the pretrained model were used as target values. Until now, this set up is only tested with the exponential α-scheduler.
Figure 3 shows the training and validation loss for selected α values between 1 and 0.3[2]. The results look promising at

[1] The experiment was aborted because of its development.

[2] The experiment is still running.

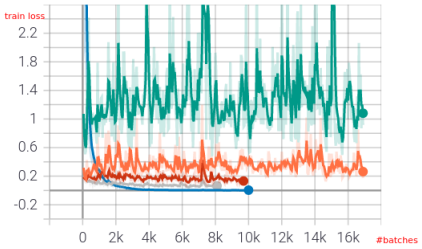


(a) Training accuracy

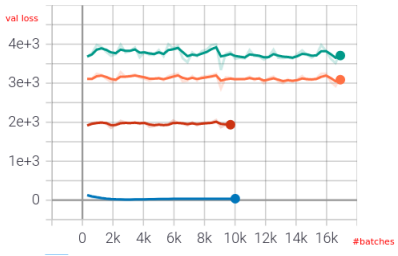


(b) Validation accuracy

Figure 2: Train- and validation accuracy during training with data augmentation for α values between 1 and 0.2: $\alpha = 0.95$ (orange, better performance), $\alpha = 0.57$ (pink), $\alpha = 0.39$ (turquoise), $\alpha = 0.3$ (light blue), $\alpha = 0.25$ (orange, worse performance). An exponential schedule with a decay rate of 0.05 was used.



(a) Training loss



(b) Validation loss

Figure 3: Train- and validation loss during training as regression task for α values between 1 and 0.3: $\alpha = 0.95$ (dark blue), $\alpha = 0.67$ (red), $\alpha = 0.52$ (orange), $\alpha = 0.35$ (turquoise). An exponential schedule with a decay rate of 0.05 was used.

first, since both losses decreased as desired. Unfortunately, this development does not last especially for the validation loss. However, because this approach has only been tried once and is still being pursued, the results cannot yet be interpreted precisely. The results could be caused by an unsuitable approach or through an error in the implementation, which needs further investigation.

3. Plan

For the next week I want to investigate following approaches in order to improve my results:

- Debug regression implementation and continue experiments.
- Initialize the weights of the parallel branches appropriately before training.
- Allow weight updates for the pretrained model.
- For the basic set up: Reduce the α-decay rate to train with smaller steps.
- Apply gaussian noise on a smaller percentage of the data.

4. Summary of new scripts

- `regression_model.py`: Subclass of `pruned_model.py`. Implements classification task as regression, i.e. computes the loss differently: Total loss is the sum over the mean squared errors between each output value and corresponding logit of the pretrained model.