Making Networks Prune-Ready

Stefan Wijnja, Ellis Wierstra, Thomas Hamburger

January 23, 2020

Abstract

TODO

Contents

Problem Statement

Having neural networks trained to carry out tasks like object detection or shape recognition can be very useful in everyday life, like for face recognition or self-driving cars. But getting a high performance on these tasks typically comes with a pretty high price. High performance often requires a high number of parameters within the neural network, and a high number of parameters demands great computational and memory resources.

For big companies getting these resources is not that much of a problem, but for researchers and developers these are generally not readily available. For this last group, it is important to find effective ways to reduce the amount of computation needed for their neural network whilst still maintaining a high performance. A very promising method to do so is called pruning.

Quite recently, research showed that big neural networks are made up of sub-networks that seem to have high degrees of sparsity. This means that these sub-networks contain certain connections in their network that are insignificant in the broad view of the neural network. [1,3] Pruning is a way to reduce the size of the neural network by removing these non-essential weights. Pruned networks have been shown to be trained to equal or better accuracy than an unpruned network, in the same amount of steps.

To make sure the neural network works and no outputs will be lost, neural networks follow a certain initialisation schema when it comes to initialising weights. Pruning has effect on these initialisations, like explained briefly before, meaning it leaves out the 'less important' weights in such an initialisation. This poses the next interesting question: What is the effect of using different initialisation schemas on the neural network's robustness to pruning, meaning the trade-off between accuracy and sparsity level?

So the purpose of this project is finding out how different initialisation schemas respond to pruning, if there are dissimilarities in the accuracy and sparsity levels between the different schemas. In this project, the initialisation schemas used are the Xavier normal and uniform distribution, and the Kaimin normal and uniform distribution. Both schemas will be used in training a feedforward neural network and a convolutional neural network. And as for the pruning, here magnitude based pruning with resetting will be implemented.

Results of the question above will afterwards be used to study whether or not it is possible to construct a custom initialisation schema that will improve robustness even further and if this will work as well for other pruning methods. If so, using that schema with pruning could prove to be even more beneficial for training neural networks.

Datasets Used

The first dataset used for this project is the Modified National Institute of Standards and Technology database, MNIST (LeCun, Cortes & Burges, „,). The MNIST database is a large database containing labelled images of handwritten digits, so the digit written in the image corresponds with the label of said image. The data contains 60.000 examples in its training set and 10.000 examples in its test set. The data is as mentioned already labelled and immediately usable, it did not need cleaning. For an example of the data, see fig. . .

The second dataset used in this project is Canadian Institute For Advanced Research-10, CIFAR-10 (Krizhevsky, Nair & Hinton, . . .). The dataset consists of 60.000 colour images, spread out evenly over 10 classes. The dataset is divided into five training sets and one test set, all containing 10.000 images. The dataset is, just like MNIST, already labelled with the classes corresponding with what is to be seen on the image, and did not need any cleaning either. For an example of the data, see fig. . .

Theoretic Definitions

Neural Networks In general Feedforward neural network Convolutional neural network what the 6 stand for in combination with ff functions differing in the neural network, ReLU, Pooling

Pruning types magnitude pruning

Initialisation schemas In general Xavier Kaiming

Trade off sparsity and accuracy Sparsity Accuracy Trade-off

Seeds In general

Pytorch In general

Tensorboard In general

Approach / Method

-possible approaches and actual approach with motivation -of misschien achteraf dus andere methoden geven

There were two types of neural networks build for this project, a feedforward neural network and a convolutional neural network. These networks were chosen from the 6 neural networks used in Frankle and Carbin, 2019. Both networks were created using pytorch in github. The feedforward neural network was trained on the MNIST dataset, and the convolutional network got trained on the CIFAR-10 dataset. In both of the networks a pruning mechanism was implemented (provided by the stakeholder?). The pruning mechanism used is magnitude based pruning with resetting.

(specific spec of networks? tabelletje?)

In order to answer the question central to this project, it is necessary to experiment and train the neural networks with different initialisation schemas. The schemas used are the Kaiming and the Xavier schemas. Both come in two variations in pytorch, namely, distributed and normalized.

To find out how different initialisation schemas respond to pruning, both neural networks will run 3-5 times with both initialisation schemas. These experiments will have to run with the seed to filter out a certain randomness in the results (little more explanation). But, to still get reliable results, 3 different seeds will be used so results can be compared and analysed. See fig. . . . for a visualised manner of the experiments.

Fig . . . Visualisation of experimenting graphviz

When all experiments are completed, the results will be analysed to see whether or not the initialisation schema has responded to the pruning. Focus will be on finding dissimilarities in the trade off of accuracy and sparsity between the different schemas. It will be determined which schema works best for which network and why that is the case. (sparsity levels?)

After all this is done, the results will be studied to see if there is a possible custom initialisation schema that will improve performance even further and if it is possible for that schema to work as well for other pruning methods.

Implement a feedforward neural network and train it on the MNIST dataset. Implement a convolutional neural network and train it on the CIFAR10 dataset.

Implement the pruning mechanism. Magnitude based pruning with resetting will suffice. Experiment with different initialization schemas built-in into PyTorch. The question we are trying to answer is: Does the initialization schema of a network affect its robustness to pruning? Note that since initializations are sampled from probability distributions, multiple runs with different seeds are required. Determine, for both of the above networks, and at varying degrees of sparsity levels, which schema works best and why. Based on the above results, is it possible, then, to construct a custom initialization schema that improves the robustness further? Check whether the schema built on (6) generalizes to other pruning methods.

aantal epochs? vervolgonderzoek higher sparsity nog een metric om graphs te krijgen, niet alleen de accuracy

Results Cleaned / labelled data Procedure to construct prediction/clustering model The findings The evaluation to show: which one is the best for different measurements, or how the results can be different if given different parameters, or why your findings are the best Possibly: What didn't work and why

Have two working models, and at least one pruning method implemented. Have baselines implemented: models initialized via different schemes, no pruning performed Experiment with various initialization schemes. Compare against each other and against the unpruned baseline in terms of performance/sparsity tradeoff. Determine which initialization scheme(s) offer(s) the best accuracy-sparsity tradeoff and assert why. Present results via a powerpoint.

Conclusion

Usability / constraints of solution Generalizability of solution Possible improvements

GENERAL:

In English Refer the criteria for report Use checklist for more detailed rules

CHECKLIST:

All sections Provide clear structure, one subject per section, clear storyline to solution Use literature and references Table and Graph give all tables and figures numbers and a name make sure text refers to tables and figures, discuss tables with results; explain what can be seen for graphs, make sure what the axes are and what lines mean etc Use spell checker No ´colloquial' language Avoid recording program code in raw form in text, use pseudocode or friendly format, important code or data in appendix Use formulas only when needed and explain well

For operating different readers, use extra sections to 'explain' e.g. neural networks