Lifelong Learning With Dynamically Expandable Networks – Reproducibility Report

Błażej Sowa, Łukasz Siudek

Motivation

Reproducibility of results in machine learning is crucial in order to make the experiment believable. However, many researchers state that there exists a reproducibility crisis on the science field. This problem seems to concern engineering studies, and among others – machine learning. This might be a major problem, considering the fact that machine learning is a very fast growing field of research. To make projects reproducible, several points have to be taken into consideration, some of which are listed in this report.

Introduction

In this report we are going to show our results of attempted reproduction of a recent conference paper from ICLR, called "Lifelong Learning With Dynamically Expandable Networks" which proposes a novel deep neural network for lifelong learning, called "Dynamically Expandable Network". It performs partial retraining of the network trained on old tasks by exploiting task relatedness, while increasing its capacity when necessary to account for new knowledge required to account for new tasks, to find the optimal capacity for itself, while also effectively preventing semantic drift.

Reproducibility

Available Information Overview

In this section, we list important reproducibility metrics.

Dataset

We use the same datasets as those introduced in the paper.

1. CIFAR-10
2. MNIST-Variation

Data preprocessing

Authors of the paper say, that in the MNIST-Variation the handwritten digits are rotated to arbitrary angles and have noise in the background, which makes the prediction task more challenging.

We use the same preprocessing methods on the MNIST dataset. At first, we aplly random rotation to the images. Then, we add Gaussian noise with our custom parameters $\mu = 0$ and $\sigma = 0.2$.

Dataset Partitions

For the MNIST dataset, the authors of DEN paper use 1,000/200/5,000 images for train/val/test split for each class. They form each task to be one-versus-rest binary classification.

For the CIFAR-10 dataset, for each class, $5,000$ out of $6,000$ images are used for training and the remainder is used for test.

Model training

Out of two specified datasets, the MNIST dataset is trained by Feedforward networks and Convolutional neural networks are used in order to train the CIFAR-10 dataset.

Out of several models introduced in the paper, we have implemented the following Feedforward networks:

1. DNN-STL. Base deep neural network, each task has its own network and exactly one output.

2. DNN-MTL. Base DNN trained for all tasks at once. One network with many outputs.

3. DNN-L2. Base DNN, where at each task t, W^t is initialized as W^{t-1} and continously trained with SGD, with l_2-regularization between W^t and W^{t-1}. For this purpose, we use the equation:

$$\underset{W_l^{\mathcal{N}}}{\text{minimize}}\ \mathcal{L}(W^t; \mathcal{D}_t) + \lambda \left\| W^t - W^{t-1} \right\|_2^2,$$

   where $\lambda = 0.005$.

4. DNN. Same as previous, but no l_2-regularization is used.

5. DEN. Dynamically Expandable Network, but only with Selective Training algorithm used.

We use two-layer network with 312-128 neurons with ReLU activation. When it comes to the Convolutional networks, we use LeNet, with two convolutional layers and three fully-connected layers.

In model training, we used SGD with momentum 0.9 and weight decay parameter equal to 10^{-4}. These parameters were not stated in the original paper, so we came up with our own.

For both datasets, we set the fixed values: every batch size is 256, learning rate $\eta = 0.1$ and we iterate over 100 epochs for each task.

Randomization control

Authors of the paper do not state how they set seed values. On the other hand, we use constant seed value.

Software and Hardware Environment

All models and algorithms in the DEN paper were implemented using the Tensorflow library. Package version is not specified.

In order to implement our version of the network, we used PyTorch (version 0.3.0.post4).

Our hardware setup (GCloud virtual machine):

- 4 x vCPU
- 16GB RAM
- NVIDIA Tesla K80

No hardware information is given about the machine used in the DEN paper.

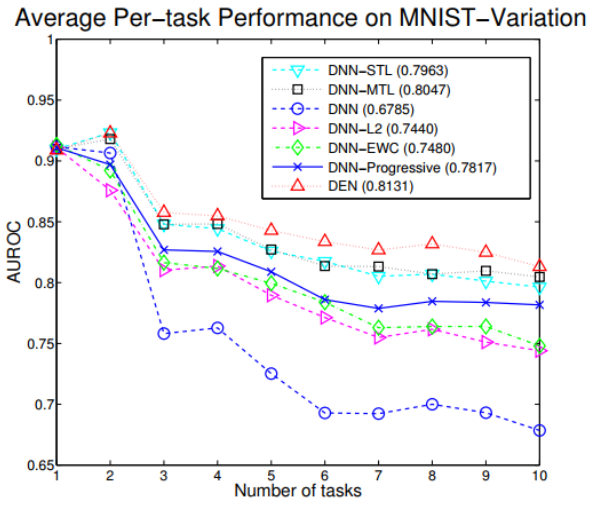
Results

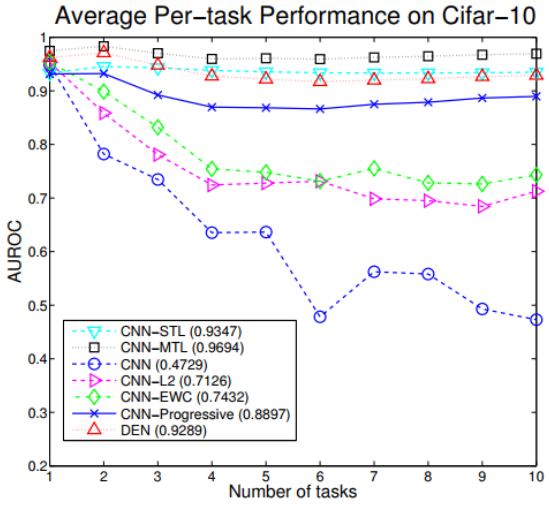
In this section, we are going to show our results and explain encountered issues.

Performance evaluation

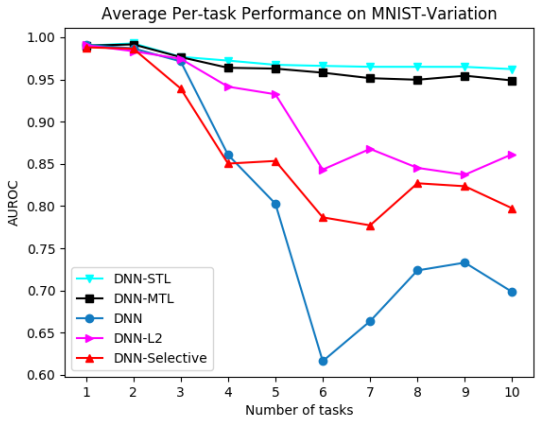
In order to evaluate performance of the networks we had to implement the AUROC (The Area Under an ROC Curve) function, where ROC curve is created by plotting the true positive rate against the false positive rate at various thresholds. The larger the computed area, the better our binary classification works.

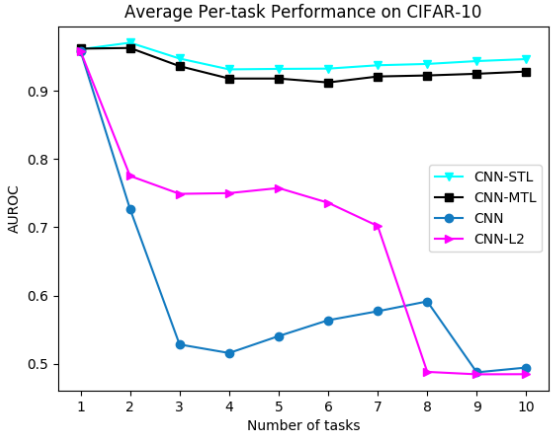
Original graph





Our graph





Comparing the two pairs of graphs, our performance for the baseline models does not differ much from the original. When it comes to the MNIST dataset, results are slightly better on our model. This can be a result of different dataset preprocessing. Other potential factors, that could have affected our model:

- Different neural network learning algorithms.
- Different tasks order.
- Differently initialized model parameters.

In case of the CNN-L2 network on the CIFAR-10 dataset, we could have not set the regularization parameter correctly, in order to stop catastrophic forgetting.

Issues

Unfortunately, we could not finish our implementation of DEN on course time. Due to various implementation problems we managed to write only the Selective Training algorithm. Solutions that we achieved using this model are called DNN-Selective.

Other issues we encountered:

- During the $l1$-regularization we noticed that weights do not exactly equal zero, but oscillate around it. The solution was to set a certain threshold, that when the value is less than that threshold, then we set the value to zero.
- Implementation using PyTorch turned out to be problematic when we had to train only some subset of neurons. The solution was to set *hooks* that set certain gradients to zero during each backpropagation.
- A lot of model parameters were missing in the DEN paper. We found that the best score is achieved when $l1$-regularization parameter equals 10^{-5}. When that parameter was too small, then too many neurons were selected. Likewise, when the parameter was too large – the network could not achieve good scores for the new task.
- We could not reproduce results for the CIFAR-100 dataset while using modified AlexNet described in the paper. Instead, we used CIFAR-10 and LeNet model, the same as the one in the older version of the paper. Our implementation of the AlexNet is available in the source code.

Conclusion

To sum up, in this report we introduced important reproducibility metrics and listed some of the missing information. One might conclude, that information given in the paper is insufficient to exactly reproduce presented results, but we think that the final results are accurate enough and can be easily improved by presenting the missing data.

For now, we do not have plans to complete DEN implementation, however if this report finds any interest, we might do it.

Acknowledgments

The authors thank Google for GCE Credits awarded through Google Cloud Platform Education Grants to the Neural Networks and Deep Learning course and to this project.

References

[1] Jaehong Yoon, Eunho Yang, Sung Ju Hwang, *Lifelong Learning with Dynamically Expandable Networks*, arXiv:1708.01547v1.

[2] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, *Gradient-based Learning Applied to Document Recognition*, Proceedings of the IEEE, 86(11):2278–2324, 1998.

[3] Babatunde K. Olorisade, Pearl Brereton, Peter Andras, *Reproducibility in Machine Learning-Based Studies: An Example of Text Mining*, 16 Jun 2017, ICML 2017 RML Submission.

[4] Monya Baker, *1,500 scientists lift the lid on reproducibility*, 25 May 2016, corrected 28 July 2016.