Neural Networks and Deep Learning

Reproduction of *Dimensionality Reduction for Representing the Knowledge of Probabilistic Models*

Michał Kotlarczyk, Michał Stypułkowski

February 2019

Contents

1 Introduction

Dimensionality reduction is important task in modern world of data science. We usually handle high dimensional data and trying to extract as much information as we can from it. First issue with dimensions greater then 3 is not being able to visualize our data. Using dimensionality reduction we can lower dimension so that we can plot data in 2D or 3D. Another fact is that more dimensions may cause troubles. We can get many irrelevant information from data. Last but not least, more dimensions means more storage space needed to process data, as well as power of our hardware needs to be much higher.

In this project we are showing our results of reproduction of *Dimensionality Reduction for Representing the Knowledge of Probabilistic Models* paper which was submitted to the ICLR 2019 conference. We implemented Dimensionality Reduction for Probabilistic Representation (DRPR) algorithm to visualize MNIST and CIFAR10 dataset in two-dimensional space.

2 Algorithm

2.1 General idea

Let's now discuss general idea behind DRPR algorithm. Suppose we have trained neural network for image classification problem. Let's call it Teacher. As the output we have predicted labels, which we are getting using Softmax layer. Given probabilities from Softmax, now we are training new neural network, named Student. As the input we take the same dataset as for Teacher. Last hidden layer contains number of neurons equal to wanted dimension. We then use Softmax-like function Ψ to evaluate probabilities of belonging to each class. Our goal is to minimize distance between Teacher's Softmax output and Student's Ψ.

2.2 Details

First we train Teacher network. Its outputs $Y_i = [0,1]^k$ are probabilities from Softmax layer, where i denotes i-th example i.e. image and k is number of classes - in MNIST and CIFAR10 $k = 10$. In addidtion we define matrix $\mathbb{Y} = [Y_1, ..., Y_n]^T$ containing all outputs.
For visualization purpose, Student's last layer is build with two neurons. Let $\phi_\theta(x)$ be result of forward pass through Student, where θ are parameters of network. We also make use of Kullback-Leibler divergence

$$D_{KL}(\mathbb{P}, \mathbb{Q}) = \sum_{x \in X} \mathbb{P}(x) \log \left(\frac{\mathbb{P}(x)}{\mathbb{Q}(x)} \right),$$

where $\mathbb{P}$ and $\mathbb{Q}$ are discrete probability distributions over set X.

Dimensionality Reduction for Probabilistic Representation (DRPR)
Input: Set of training examples $\mathbb{X}$, target probabilities $\mathbb{Y}$, mapping function ϕ_θ and number of epochs t.
for epoch 1 to t **do**
for each mini-batch **do**
Create matrix $F = [\phi_\theta(x_1), ..., \phi_\theta(x_m)]^T$, where m is mini-batch size
Create matrix $M = [\mu_1, ..., \mu_k]^T = diag(Y^T \mathbf{1}_n)^{-1} Y^T F$
Create vector $\pi = \frac{1}{n} Y^T \mathbf{1}_n$
Create matrix Ψ, such as $\Psi_{ic} = \frac{\pi_c exp(-\|f_i - \mu_c\|_2^2)}{\sum_{m=1}^k \pi_m exp(-\|f_i - \mu_m\|_2^2)}$
Compute loss $\Delta(\Psi, \mathbb{Y}) = \frac{1}{m} \sum_{i=1}^m D_{KL}(y_i \| \psi_i)$
Update parameters θ using gradient descent on $\Delta(\Psi, \mathbb{Y})$
end for
end for
Output: mapping ϕ_θ

2.2.1 MNIST dataset

Teacher architecture
We used three Convolutional combined with ReLU and Max Pooling layers, followed by three linear layers.
Student CNN architecture
We used the same structure as for Teacher. The only change was in the last layer - we used 2 output neurons instead of 10.
Student Linear architecture
Simple linear structure, i.e. three Linear layers.

2.2.2 CIFAR10 dataset

Teacher architecture
We took VGG11 with batch normalization network pretrained on ImageNet dataset and fine-tuned parameters to fit CIFAR10. Change of first and last Linear layers were required to match dimensions of data.
Student architecture
We used same structure as for Teacher. Analogically as in the case of MNIST-Student, we used 2 output neurons instead of 10.

3 Results

Unfortunately none of our experiments were successful. Below we present our results. Difference between our plots and ones from the paper may come from potential difference in networks architecture.[1]

3.1 MNIST

We trained Teacher on MNIST dataset using architecture mentioned earlier. We achieved 97% accuracy on test set.

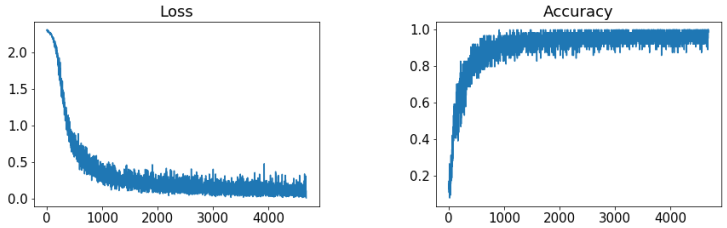

Figure 1: MNIST-Teacher loss over time (left plot) and accuracy over time (right plot).

3.1.1 Student-CNN

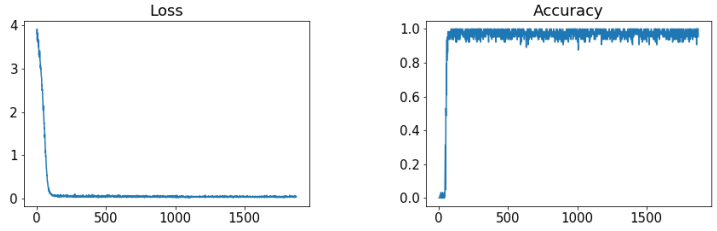

Figure 2: MNIST-Student-CNN loss over time (left plot) and accuracy over time (right plot).

[1] Our code can be found on https://github.com/MStypulkowski/DRPR.

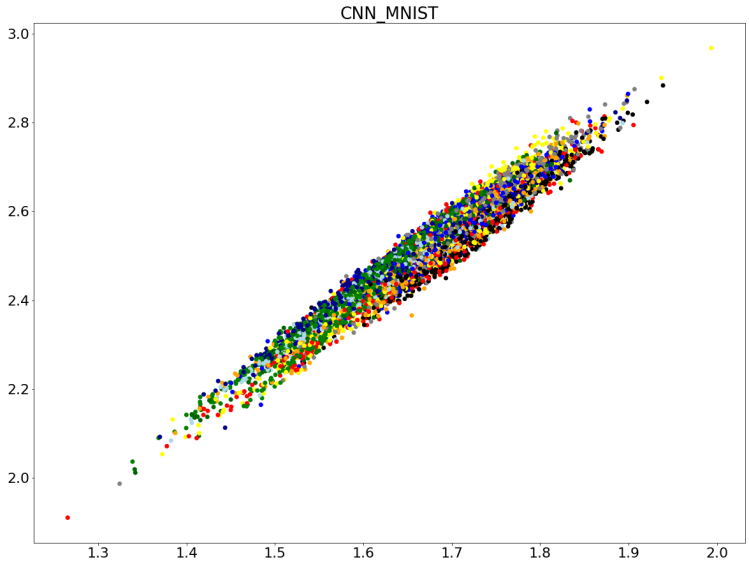

Figure 3: MNIST-CNN 2D mapping after DRPR application.

3.1.2 Student-Linear

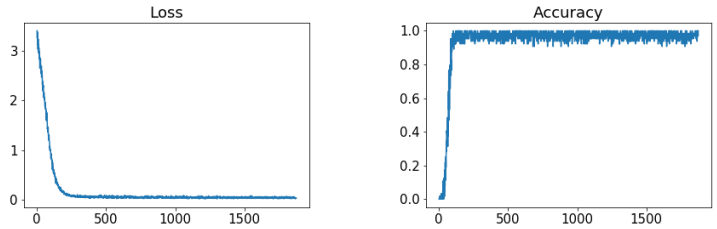

Figure 4: MNIST-Student-Linear loss over time (left plot) and accuracy over time (right plot).

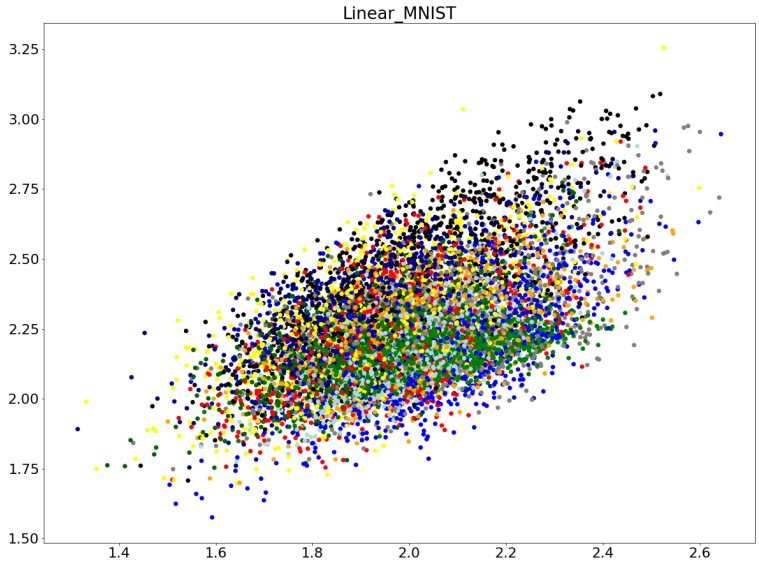

Figure 5: MNIST-Linear 2D mapping after DRPR application.

3.2 CIFAR10

We trained Teacher on CIFAR10 dataset using architecture mentioned earlier. We achieved 84% accuracy on test set.

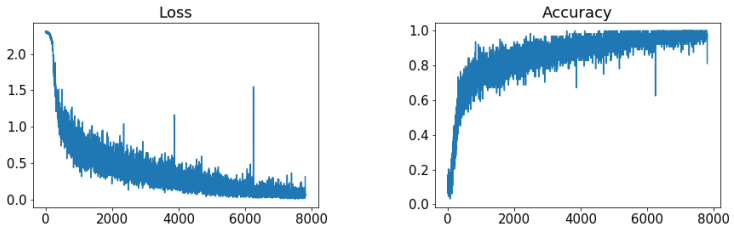

Figure 6: CIFAR10-Teacher loss over time (left plot) and accuracy over time (right plot).

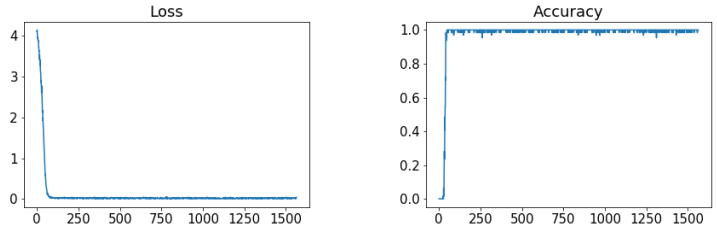

Figure 7: CIFAR10-Student loss over time (left plot) and accuracy over time (right plot).

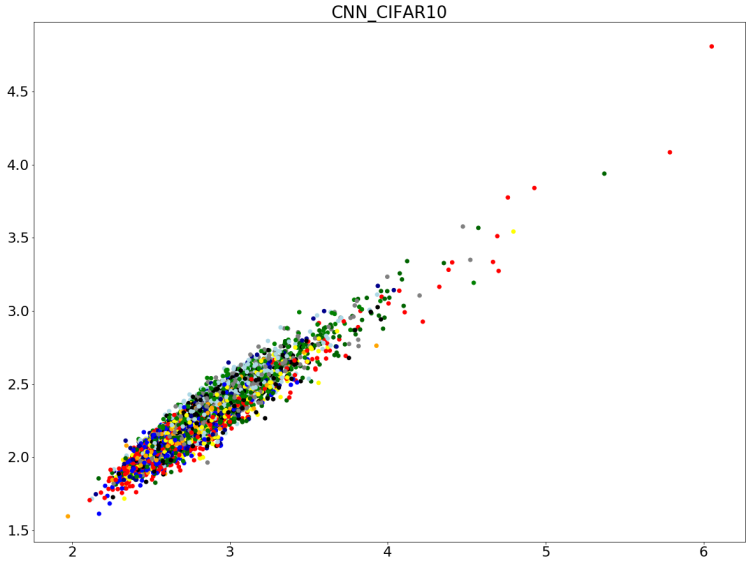

Figure 8: CIFAR10 2D mapping after DRPR application.

3.3 Conclusion

Every student is learning fine. It achieves good accuracy after just one epoch of training. Although presented visualizations are poor. In every case we can observe strong positive correlation between two outputs. We barely can determine clusters of each label.

References

[1] Marc T Law, Jake Snell, Amir-massoud Farahmand, Raquel Urtasun, Richard S Zemel *Dimensionality Reduction for Representing the Knowledge of Probabilistic Models.*
International Conference on Learning Representations (ICLR) 2019