Synthetic gradient in residual networks

Lukáš HANINČÍK*

Slovak University of Technology in Bratislava
Faculty of Informatics and Information Technologies
Ilkovičova 2, 842 16 Bratislava, Slovakia
lukas.hanincik@gmail.com

Abstract. Phases of forward-propagation and backpropagation in training of neural networks are processed by synchronous approach. All these phases consist of sequential computation steps of forward transformation and backward propagation of the error gradient. It causes all layers to be dependent on the neighboring layers. We show a method, how to remove the phase of backpropagation from the training process. This idea is based on the synthetic gradient which is an approximated version of the real gradient and is provided by modules of synthetic gradient. This method replaces the sequential approach of the training process by an parallel approach. Based on performed experiments, synthetic gradient increase a speed of training process of residual networks and it reaches as similary results as an original method, backpropagation. We implemented this method in a residual neural network in the DeepCYTOF algorithm which is trained for classification of cytometric data.

1 Introduction

The training process in ordinary neural networks consists of the forward-propagation and backpropagation phases. In forward-propagation, input data is propagated through the layers along the whole neural network. Each layer performs some transformation of received data and sends it to the following layer. When data reaches the output of the neural network, the error value is calculated (basically as difference of output of neural network and expected value). Forward-propagation ends by quantifying the error value [2].

After forward-propagation follows backpropagation. During backpropagation, the error value propagates backward through the whole network from the output to the input. Each layer uses the propagated error for updating its parameters (as known as weights) [2, 4]. A layer which obtains an error value calculates the gradient $\frac{\partial L}{\partial h_i}$ where L is an error value and h_i are transformed data of layer i [2].

Each layer which updated its parameters is going to propagate a gradient to the previous layer, which uses it for calculating its own gradient. The backpropagation algorithm does not allow to update a layer's parameters parallel with the updating parameters of the other layers [4]. Each layer which is ready to update its parameters must wait for the gradient propagated from the following layer and it means that each layer is dependent on the neighboring layers. Based on this limitation, a neural network is able to train just by synchronous approach, layer by layer [1, 4].

The synchronous approach of training neural networks causes three types of network lockings to occur during the training process [1, 4]:

1. *Forward locking* - no module can process its incoming data before the previous nodes in the directed forward graph have executed.
2. *Update Locking* - no module can be updated before all dependent modules have executed in forwards mode.
3. *Backwards Locking* - no module can be updated before all dependent modules have executed in both forwards mode and backwards mode.

In this paper we show a method called *Decoupled neural interface*, which removes *Update locking* and *Backwards locking* from the training process of neural networks. Decoupled neural interface (DNI) is used as provider of a synthetic gradient and allows to update a layer's parameters in parallel. We implemented DNI to a residual network (ResNet) which is trained for the blood cells calibration. Given Resnet is one of the components of the DeepCyTOF algorithm. Calibrated blood cells are subsequently classified to discrete groups which allows to predict a wide spectrum of diseases.

---

* Bachelor study programme in field: Informatics
Supervisor: Mgr. Tomáš Jarábek, Institute of Informatics, Information Systems and Software Engineering, Faculty of Informatics and Information Technologies STU in Bratislava

IIT.SRC 2019, Bratislava, April 17, 2019, pp. 1–8.

2 Decoupled neural interface and synthetic gradient

As we introduced above, common backpropagation of error value limits neural network on synchronous approach. Each layer, which transformed input data is ready to update its parameters but it must wait for gradients propagated from following layer. Based on this fact, all layers are trained sequentially (as shown on Fig. 1 a).

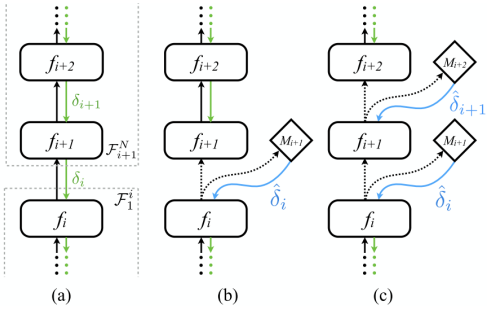

Figure 1. (a) A section of a vanilla feed-forward neural network F_1^N. (b) Incorporating one synthetic gradient model for the output of layer i. This results in two sub-networks F_1^i and F_{i+1}^N which can be updated independently. (c) Incorporating multiple synthetic gradient models after every layer results in N independently [1, 4].

Decoupled neural interface presents an interface for communication with the *module of synthetic gradient*. Module of synthetic gradient is a plain neural network with the one dense layer which is trained for asymptotic approximation of real gradient. Generally, the module of synthetic gradient predicts a real gradient backpropagated from the output of neural network [1, 4].

The module of synthetic gradient receives the transformed data of accompanied layer (output of accompanied layer). As shown on Fig. 1 b, layer f_i sends transformed data to layer f_{i+1} and so on to the module of synthetic gradient M_{i+1}. Module of synthetic gradient M_{i+1} receives transformed data of layer f_i and uses it for predicting a gradient. The predicted gradient is sent back to layer f_i which shall update its parameters immediately [4].

Since layer f_i is able to update its parameters immediately without the need of additional informations from other layers, it breaks a dependence on gradient propagated from layer f_{i+1}. Each layer which directly communicates with the module of synthetic gradient is not dependent on the other layers in phase of backpropagation anymore (as shown on Fig. 1 c). Each layer which is ready to update its parameters, doesn't have to wait for the gradient backpropagated from following layer because it can updates its layers using the synthetic gradient provided by accompanied modules of synthetic gradient. It completely removes *Update locking* and *Backward locking* from the neural network training process [4].

2.1 Training of modules of synthetic gradient

As we introduced above, modules of synthetic gradient are plain neural networks trained for asymptotic approximation of the real gradient. Performance of modules of synthetic gradient is increased by training the model of synthetic gradient. As shown on Fig 2, the module of synthetic gradient M_B receives a gradient by layer f_B. That is the same gradient as which was used for updating parameters of layer f_B. Module of synthetic gradient M_B considers the obtained gradient δ_A as ground-truth gradient and updates its parameter in order to minimize the L_2 distance $||\delta_A - \hat{\delta}_A||$, where δ_A is real gradient propagated by layer f_B and $\hat{\delta}_A$ is synthetic gradient used for updating parameters of layer f_A [1,4]. The feedback model M_B can also be conditioned on any privileged information or context, c, available during training such as a label [4].

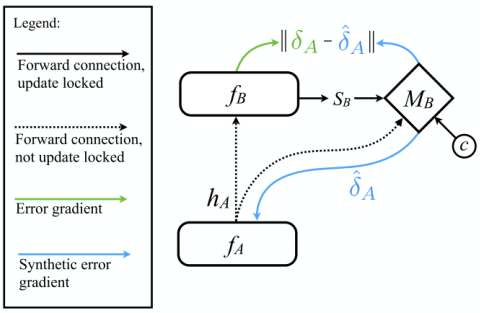

Figure 2. General communication protocol between f_A and f_B [4].

The key idea of DNI is to approximate the gradient of the error $\frac{\partial L}{\partial h_i}$, using the synthetic gradient $\hat{\delta}_i$ as $\hat{\delta}_i \approx \frac{\partial L}{\partial h_i}$, where L is the prediction error and h_i is the output of layer i (data transformed by layer i). Synthetic gradient is the output of activation function $\hat{\delta}_i = \hat{\delta}(h_i, \phi_i)$, where ϕ_i are parameters of the module of synthetic gradient i [4].

As we mentioned, the synthetic gradient must be trained to approximate the true gradient. For updating the parameters of the module of synthetic gradient i, we use the synthetic gradient provided by module of synthetic gradient $i + 1$, $\hat{\delta}_{i+1}$. We "unroll" our synthetic gradient just one step,

$$\hat{\delta}_i \approx \frac{\partial L}{\partial h_i} = \frac{\partial L}{\partial h_{i+1}} \frac{\partial h_{i+1}}{\partial h_i} \approx \hat{\delta}_{i+1} \frac{\partial h_{i+1}}{\partial h_i}$$

Unrolled synthetic gradient $z_i = \hat{\delta}_{i+1} \frac{\partial h_{i+1}}{\partial h_i}$ is considered as constant training target for the synthetic gradient $\hat{\delta}_i$ [4].

We update parameters ϕ_i of module of synthetic gradient i so as to minimize the mean-squared error of these one-step unrolled training targets z_i. For updating parameters ϕ_i stochastic gradient descent [2] is used [4].

3 Residual network

Very deep neural networks have a problem called *vanishing gradient* [3]. A way to solve the problem of vanishing gradient is by replacing the original nonlinear layers [2] by residual blocks. A residual block consists of a few nonlinear layers (as known as *stacked layers*) and a identity shortcut. The identity shortcut is a specific interface which provides identity mapping, received on the input of the residual block, to the output of the residual block (as shown on Fig. 3)

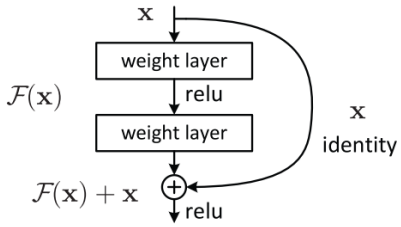

Figure 3. An illustration of a residual block. [3].

As introduced [3, 7], it is easier to optimize the residual mapping than to optimize the original, unreferenced mapping. Even thought the identity shortcut, a residual network can still be trained end-to-end by stochastic gradient descent with backpropagation [3]. [4] introduced, there is nothing to restrict the use of DNI for arbitrary network graph. We think, an implementation of DNI to residual network should not cause a significant degradation of network performace. Deeper functionality of residual networks are explained in [3, 7].

4 DeepCyTOF algorithm

DeepCyTOF algorithm is a algorithm designed for cytometric analysis. Output of DeepCyTOF algorithm are gated blood samples. In cytometry science, gating represents a classification of blood cells to discrete groups. It helps to know and understand different human diseases, predict occurrence of disease and diagnose a disease in early stage [5, 6].

The DeepCyTOF algorithm consist of three separate neural networks. First neural network is denoising autoencoder (DAE). DAE is the neural network trained for replacing zero values for real non-zero values. This process is called denoising. Denoised data follows to MMD-ResNet. MMD-ResNet is specific residual network trained for calibration input samples to a reference target sample. This residual network uses maximum mean discrepancy (MMD) as loss function [2]. Training of MMD-ResNet leads to decreasing of gap between two distribitions. Denoised and calibrated data are finally gated using feed-forward neural network [2]. Deeper functionality of DeepCyTOF algorithm is explaned in [6].

5 Experiment and results

As we introduce DeepCyTOF algorithm which is constructed by three neural networks. Subject of our research is observation of influence of synthetic gradient on the accuracy of residual network.

We implemented DNI to MMD-ResNet which is trained for calibration of input samples. We used DNI as provider of synthetic gradient for each layer (except of input and output layers). Each stacked layer of MMD-ResNet includes a batch normalization and ReLU non-linear activation function [2, 8]. Residual blocks consist of two stacked layers. Input stacked layer of residual block has weight matrix of size 25×8 and output stacked layer has weight matrix of size 8×25. Each layer updates its parameter using ***RMS Prop*** optimizer [2]. Mini-batch size for the MMD-ResNet is 1000. In the MMD-ResNet, a penalty of 10^{-2} on the l_2 norm of the network weights is added to the loss for regularisation. We used the same data as [6] which was available on its GitHub repository.

The kernel used for MMD-ResNet is a sum of three Gaussian kernels

$$k(x, y) = \sum_{i=1}^{3} exp\left(- \frac{||x - y||^2}{\sigma_i} \right)$$

we set the $\sigma_i \in \{\frac{m}{2}, m, 2m\}$, where m is the median of the average distance between a point in the target sample to its 20 nearest neighbours.

First runs give a promising results. We observed a progress of MMD between calibrated source sample and reference target sample. First results indicated almost double increase of time spent for training MMD-ResNet with using an original backpropagation against a MMD-ResNet trained using synthetic gradient. This measurement was performed on 400 iterations.

As shown on Fig. 4, during the training of the MMD-ResNet, the MMD between source sample and target sample is decreasing. Decreasing the MMD

means, that the distributions of values in source sample is getting similar to distribution of values in target sample, but not equal. Based on this experiment, we can say, that implementation of synthetic gradient in residual network do not have a significantly negative influence to performance of given residual network. As we expected, MMD of ResNet trained by synthetical gradient decresed slowly but in last iterations, it reached the MMD of ResNet trained by original backpropagation.

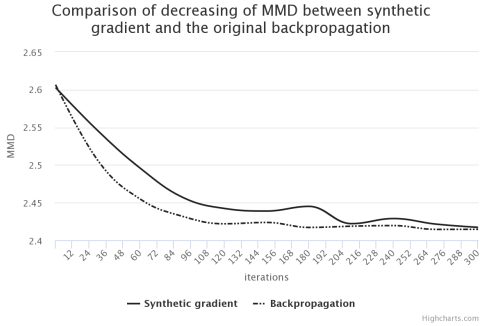

Figure 4. Progress of decreasing of MMD between source sample and target sample. On the plot above we can see, that ResNet trained by backpropagation stops learning after a hundred iterations of learning. ResNet trained by synthetic gradient is able to reach a similar results after another hundred iterations. Afterwards, both of them begin to overfit.

MMD metrics of evaluation of ResNet is based on stochastical approach. A MMD value is computed between a random batch from source sample and random batch from target sample. Size of these batches is 1000. Stochastic selection of batch from target sample caused a variability of error value (MMD). With respect to this fact, we smoothed a MMD progress line in plot.

Table 1. Table displays a basic comparison of MMD ResNet trained by synthetic gradient and original backpropagation. All values are measured for models to reach an optimal value of MMD, 2.419*.* $Iterations$ *shows a number of iterations needed for a reaching an optimal values.* $Time$ *row shows a time needed for a reaching an optimal value of MMD.* $F1score$ *presents a* $F1$ *score of final classifications of data calibrated by MMD-ResNet.*

MMD = 2.419	**Backprop.**	**Synth. grad.**
Iterations	150	240
Time	107s	90s
$F1$ score	81.71	76.29

As shown in table 1, MMD-ResNet trained by synthetic gradient reachs a bit lower $F1$ score in final classification of calibrated data. Difference is prominent, but it can be resolved by upgrading an implementation of modules of synthetic gradient. Training of MMD-ResNet by backpropagation spent more time for reaching an optimal value of MMD. With respect to this fact, we believe, that synthetic gradient has significantly better time results in models, which requires more iterations of learning for reaching a better performance.

6 Conclusion

We mentioned above, that output layer of our MMD-ResNet does not have a accompanied module of synthetic gradient. This is not needed because the output layer receives ground-truth gradient calculated on the real error value of MMD function.

We used residual network for implementation of synthetic gradient because of its depth. Deeper networks which use synthetic gradients in the training process, usually reach a higher performance than the shallower networks [4]. From that point of view, residual networks are a great choice for observing the influence of synthetic gradients, because we can better understand, how the neural network behaves by using synthetic gradient to update its layer parameters. For the evaluation of MMD-ResNet, we used the $F1$ score statistical test. $F1$ score was applied on the final classification of data, which was calibrated by MMD-ResNet trained by synthetic gradient.

Another reason, why we used residual networks for this experiment is, that we haven't noticed any documented experiment where a residual network with implemented DNI is used.

References

[1] Czarnecki, W.M., et al.: Understanding Synthetic Gradients and Decoupled Neural Interfaces, 2017.

[2] Goodfellow, I., et al.: *Deep learning*. Volume 1. MIT press Cambridge, 2016.

[3] He, K., et al.: Deep residual learning for image recognition. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[4] Jaderberg, M., et al.: Decoupled Neural Interfaces using Synthetic Gradients, 2016, vol. 1.

[5] Li, H., et al.: DeepCyTOF: Automated Cell Classification of Mass Cytometry Data by Deep Learning and Domain Adaptation. *bioRxiv*, 2016, p. 054411.

[6] Li, H., et al.: Gating mass cytometry data by deep learning. *Bioinformatics (Oxford, England)*, 2017, vol. 33, no. 21, pp. 3423–3430.

[7] Targ, S., et al.: Resnet in Resnet: Generalizing Residual Architectures, 2016, pp. 1–7.

[8] Xu, B., et al.: Empirical Evaluation of Rectified Activations in Convolutional Network, 2015.