LOCAL LEARNING WITH NEURON GROUPS

Adeetya Patel[1,2] **Michael Eickenberg**[3] **Eugene Belilovsky**[1,2]
[1] Concordia University [2] Mila – Quebec AI Institute [3] Flatiron Institute
{adeetya.patel, eugene.belilovsky}@concordia.ca,
meickenberg@flatironinstitute.org

ABSTRACT

Traditional deep network training methods optimize a monolithic objective function jointly for all the components. This can lead to various inefficiencies in terms of potential parallelization. Local learning is an approach to model-parallelism that removes the standard end-to-end learning setup and utilizes local objective functions to permit parallel learning amongst model components in a deep network. Recent works have demonstrated that variants of local learning can lead to efficient training of modern deep networks. However, in terms of how much computation can be distributed, these approaches are typically limited by the number of layers in a network. In this work we propose to study how local learning can be applied by splitting layers or modules into sub-components, introducing a notion of width-wise modularity to the existing depth-wise modularity associated with local learning. We investigate local-learning penalties that permit such models to be trained efficiently. Our experiments on the CIFAR-10 dataset demonstrate that introducing width-level modularity can lead to computational advantages over existing methods based on local learning and opens potential opportunities for improved model-parallel training. This type of approach increases the potential of distribution and could be used as a backbone when conceiving collaborative learning frameworks.

1 INTRODUCTION

Neural networks are typically trained by stochastic gradient descent in combination with the back-propagation algorithm. This learning algorithm allows joint adaptation of all layers and neuron connections in a network based on a global objective function. It is typically believed that this joint adaptation is essential to obtain a high performance for large scale data sets as joint adaptation towards an overarching objective allows the individual layers and neurons to adapt function efficiently. On the other hand, several recent works have studied the idea to use a purely local loss function, where there is no feedback between the independent layers. These approaches are based on the classic sequential greedy learning procedure Ivakhnenko & Lapa (1965); Bengio et al. (2007) but allow the different model components to simultaneously learn their model parameters. Surprisingly, relying purely on this forward communication, high-performance models can be constructed Belilovsky et al. (2020); Nøkland & Eidnes (2019). These observations have implications on both the functional properties of high-performance deep networks as well as practical implications for distributed training of neural networks. If layerwise learning can still bring high performance, this raises the question: to what degree do neural network components need to be trained jointly?

Biological neural systems rely on extremely localized synaptic updates, often modeled as Hebbian learning (Hebb, 1949) or adaptations thereof. The only known way for a biological neural system to incorporate global information into learning is by way of feedback/recurrent neural connections, which can bring information that was computed at a later stage back to earlier neurons and hence be incorporated by local learning. However, for many definitions and practical purposes, these recurrent connections are part of the forward model. Since even within a biological neural network ”layer” (e.g. a retinotopic map), any local learning would never use information from the whole layer, there is enough biological inspiration to ask whether convolutional network training can be decoupled within modules that have already been sequentially decoupled.

In this work we investigate the limits of using local loss functions, moving them from the layer-level to the “neuron group” level. Specifically, we consider the case where non-overlapping groups of neurons

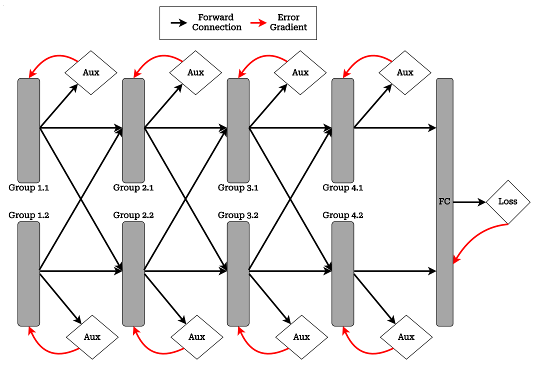

Figure 1: Grouped Neuron DGL, each layer and neuron group has a local objective. All local groups are learned in parallel, feeding their output directly to the next layer's groups.

within a network layer each have their own local objective function and optimize their parameters in isolation. Our investigation reveals that networks can still demonstrate high performance despite this modification. We also investigate how to encourage these decoupled networks to learn diverse behavior (i.e. learn different representations from each other) based solely on forward communication. We illustrate the practical applications of this by comparing the proposed method to existing local learning approaches in the context of total training time, inference time, and model performance.

2 Related Work

Sequential local learning has been revisited recently by Belilovsky et al. (2019) which showed that this approach can yield high-performance models on large datasets and architectures. Belilovsky et al. (2020) further demonstrated that this can be performed in the parallel local learning setting, where layers are learned simultaneously. It can even be done in an asynchronous manner if each layer is allowed to maintain a memory. Nøkland & Eidnes (2019) demonstrated an alternative loss function which yields improved performance in the local learning setting. Wang et al. (2021) further extended this idea, illustrating a novel regularization term that combats the collapsing of the representation towards the target supervised task through the use of an autoencoder.

Related to our work Veness et al. (2019) studied a local objective where each neuron solves a binary classification problem. Results were illustrated in an online-learning setting.

Choromanska et al. (2019); Lee et al. (2015) considers local objective functions with globally generated targets. These, however, require feedback communication between the various layers and model components.

3 Methods

In this section, we describe the background, our approach and methodology.

3.1 Notation and Background

We describe the local learning framework of Belilovsky et al. (2020); Nøkland & Eidnes (2019) and introduce notation. Consider an input to a neural network x_0, we denote operation of layer j as $f_{\theta_j}(x_{j-1})$, where θ_j correspond to the parameters of the network and $\mathcal{L}(y, x_j; \gamma_j, \theta_j)$ corresponds to local loss function applied to the representation x_j, where γ_j are parameters of an auxiliary network. In Belilovsky et al. (2020); Nøkland & Eidnes (2019), it is proposed to learn the parameters θ_j jointly and in parallel. In the subsequent section, we consider further dividing the objective to introduce a group-wise local loss $\mathcal{L}(y, x_j^i; \gamma_j^i, \theta_j^i)$, where x_j^i is a subset of x_j.

3.2 Grouped Neuron DGL (GN-DGL)

Building on layer-wise local learning we consider further splitting each layer into groups with isolated losses. We propose to learn each of these modules in parallel online (as done in Belilovsky et al. (2020); Nøkland & Eidnes (2019) instead of sequentially as done in Belilovsky et al. (2019)). In this method, we try to learn a greedy objective for each of the neuron groups as shown in Algorithm 1. Here, each neuron group has approximately the same number of neurons. It is important to note here that neuron group receives output representations from all groups of the previous layer during a forward pass as shown in the Figure 1. We refer to this approach as Grouped Neuron DGL (GN-DGL).

3.3 Stop-Gradient Grouped Neuron DGL

We can loosen the communication restrictions between layer groups, permitting them to send their outputs to each other's auxiliary networks. In this method, depicted in Algorithm 2, the auxiliary networks of each neuron group collect the gradient-decoupled output representations from other neuron groups of the same layer and uses it as part of their final prediction. We note the communication cost can be decreased by pooling the output before sending to auxiliary networks. Importantly, this is only a forward connection: no information is sent back to the origin nodes.

Algorithm 1: Grouped-Neuron DGL

```
Input: Mini-batches $\mathcal{S} \triangleq \{(x_0^t, y^t)\}_{t\leq T}$
1  Initialize Parameters $\{\theta_j^i, \gamma_j^i\}_{j\leq J, i\leq G}$.
2  for $(x_0, y) \in \mathcal{S}$ do
3      for $j \in 1, ..., J$ do
4          for $i \in 1, ..., G$ do
5              $x_j^i \leftarrow f_{\theta_j^i}(x_{j-1})$.
6              Compute $\nabla_{(\gamma_j^i, \theta_j^i)} \hat{\mathcal{L}}(y, x_j^i; \gamma_j, \theta_j^i)$.
7              $(\theta_j^i, \gamma_j^i) \leftarrow$Update params $(\theta_j^i, \gamma_j^i)$.
8          end
9          $x_j \leftarrow \|_{i=1}^G x_j^i$.
10     end
11 end
```

Algorithm 2: Stop-Gradient GN-DGL

```
Input: Mini-batches $\mathcal{S} \triangleq \{(x_0^t, y^t)\}_{t\leq T}$
1  Initialize Parameters $\{\theta_j^i, \gamma_j^i\}_{j\leq J, i\leq G}$.
2  for $(x_0, y) \in \mathcal{S}$ do
3      for $j \in 1, ..., J$ do
4          for $i \in 1, ..., G$ do
5              $x_j^i \leftarrow f_{\theta_j^i}(x_{j-1})$.
6              $z_j^i = sg[\|_{k\neq i} x_j^k]$
7              Compute$\nabla_{(\gamma_j^i, \theta_j^i)} \hat{\mathcal{L}}(y, x_j^i, z_j^i; \gamma_j^i, \theta_j^i)$.
8              $(\theta_j^i, \gamma_j^i) \leftarrow$Update params $(\theta_j^i, \gamma_j^i)$.
9          end
10         $x_j \leftarrow \|_{i=1}^G x_j^i$.
11     end
12 end
```

3.4 Grouped Neuron DGL with diversity-promoting penalty

An expected weakness of local group losses is that representations learned in different groups will be correlated and redundant to each other, not permitting optimal aggregation of predictive power. We attempt to address this by allowing auxiliary models to send their softmax output distributions to each other. We add the diversity promoting term in the loss function which encourages the diversity locally based solely on communication of softmax outputs, between auxiliary modules.

For the diversity-promoting penalty we utilize a variant of the penalty studied in Dvornik et al. (2019). Each neuron group of the network is parameterized by θ_j^i, where i and j indicates the group index and layer index respectively. Each group leads to the class probabilities $p_j^i = \mathrm{softmax}(f_{\theta_j^i}(x_{j-1}))$. Now consider $\hat{p}_j^i = \frac{p_j^i \odot m_y}{\|p_j^i \odot m_y\|_1}$ where m_y is a vector with zero at position y and 1 otherwise. Now, for a group a in the layer j, we measure diversity-promoting penalty with each remaining group b in the same layer as shown in Equation 1,

$$\phi(\hat{p}_j^a, \hat{p}_j^b) = \cos(\hat{p}_j^a, sg[\hat{p}_j^b]) \tag{1}$$

Here, sg denotes the stop-gradient operation. It is important to note here that to estimate the diversity-promoting penalty for group a with all remaining groups b of the same layer, we use a gradient-decoupled version of $\hat{p}_j^b$. This penalty encourages diversity by pushing the non-target labels away from each other.

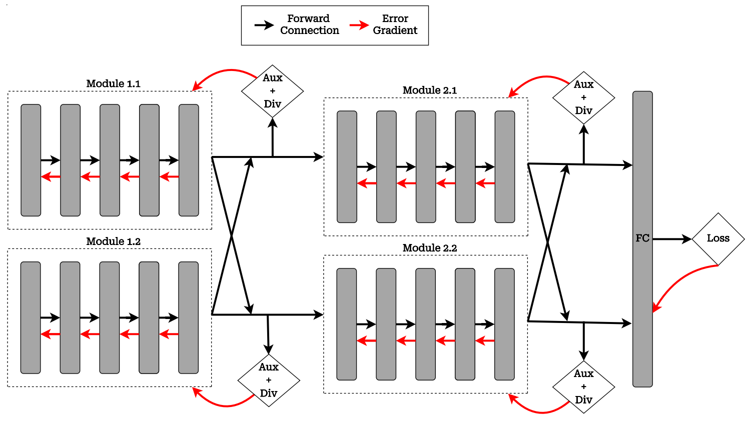

Figure 2: Illustration of our approach. Multi-layer modules are used in combination with a gradient decoupled output and diversity-promoting penalty

4 Experiments and Results

We perform two sets of experiments focused on the popular CIFAR-10 dataset used in many prior works Belilovsky et al. (2020); Huo et al. (2018). First, we study a simple VGG6a model using layerwise and group neuron level local loss functions. This allows us to establish basic performance characteristics of this approach compared to layerwise training. In the second set of experiments, we focus on an application to model-parallel training, comparing our method in the case of multi-layer modules to the recent methods such as Decoupled Greedy Learning (DGL) Belilovsky et al. (2020) and local learning with Information Propagation (InfoPro) Wang et al. (2021).

4.1 Layerwise and Grouped Neuron DGL

The VGGNet, denoted VGG6a, used in the experiments consists of six layers (four convolutional and two fully connected) and is taken from Nøkland & Eidnes (2019). The convolutional layers have 128, 256, 512, 512 channels respectively, and 8192, 1024 features in the last two fully connected layers. For DGL, we train each layer with its own auxiliary network as described in Belilovsky et al. (2020). For GN-DGL, we further split the layers width-wise into 2 to 10 neuron groups. The neuron groups within each layer and their auxiliary networks are trained locally by the auxiliary loss. The details about width-wise splitting are further elaborated in Appendix A.2. Each neuron group has its own auxiliary loss except the last fully connected layer as shown in Figure 1. We use the same auxiliary network design and loss (termed *predsim*) as Nøkland & Eidnes (2019).

Method	DGL	2-Group	4-Group	6-Group	8-Group	10-Group
Test Accuracy	92.25	91.63	90.55	89.8	89.07	88.91

Table 1: Grouping of neurons in each layer of VGG6a. We observe with increased number of groups, there is performance degradation, however a surprisingly high overall accuracy can be maintained despite a lack of communication across neuron groups. We note that the presented results do not utilize stop-gradient techniques, nor a local diversity-promoting penalty.

Method	DGL	6-Group GN-DGL	6-Group GN-DGL with diversity	6-Group GN-DGL Stop Gradient + diversity
Test Accuracy	92.25	89.8	90.57	91.9

Table 2: Ablating the effect of local diversity-promoting penalty and use of stop-gradient communication in GN-DGL.

We define naive splitting where each neuron group is trained locally by corresponding auxiliary networks and there is no encouragement of diversity among local modules from the same layer.

	VGG6a	**2xVGG6a**	**3xVGG6a**
DGL	92.25	93.27	93.27
6-Group GN-DGL	89.8	92.6	93.28

Table 3: Ablating the effect of width in GN-DGL. We observe that as width increases, using GN-DGL begins to yield performance comparable to DGL, with increased parallelization.

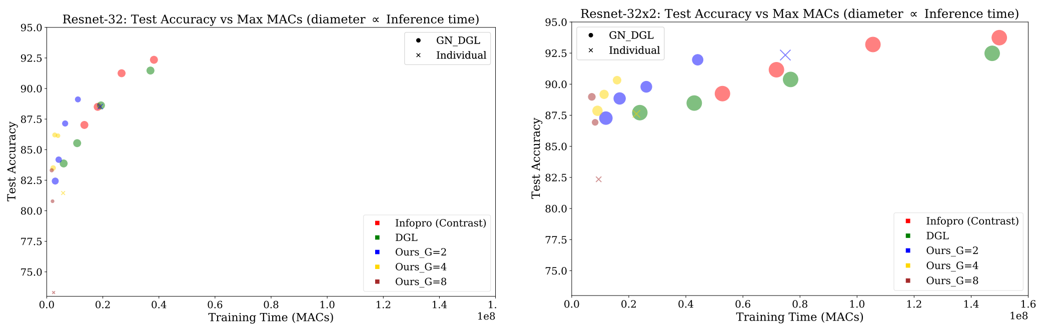

Figure 3: Accuracy versus maximum training time in MACs for a given node in a theoretical distributed scenario for various local learning methods. We observe that for both ResNet-32 (left) and a wider 2xResNet-32 (right), the GN-DGL leads to better tradeoffs in training time versus accuracy. The size of the bubbles is proportional to the inference time of the models.

Results of naive splitting experiments on VGG6a is presented in Table 1. Here the increasing number of group size is indicated with G-Group, where G is the minimum number of groups in any layer. We also report the DGL Belilovsky et al. (2020) test accuracy for VGG6a, where the network is only divided depth-wise and each layer is a local-module in itself with their own auxiliary network and loss. We observe that as we increase G, the test accuracy slightly decreases (which is expected). We believe the decrease in accuracy is mainly due to each neuron group being trained in isolation, resulting in correlated features.

Diversity of features and Stop-Gradient In order to improve performance of the overall model, we permit forward communication between within-layer modules. Specifically we allow softmax activations and gradient-decoupled pre-auxiliary module features to be sent across local modules. This corresponds to the stop-gradient and diversity-based approaches discussed in Sec 3.3 and 3.4. In Table 2, we show the ablations which demonstrate that these additions consistently improve the performance of the model allowing it to nearly recover the DGL performance.

Increasing Width We also investigate the effect of increasing width in Table 3, we observe that increasing the width of a 6-group GN-DGL can bring the performance closer to the DGL performance.

4.2 Multi-layer Grouped Neuron DGL

We now follow the extension to modules (collections of layers) and sub-modules (sub-networks that can be grouped together to form a module). Unlike the previous section, for multi-layer versions of GN-DGL, we divide the network in such a way that we get K and G local modules depth-wise and width-wise respectively, resulting in a total of $K \times G$ local modules. Here, K corresponds to depth-wise splitting used in Wang et al. (2021), on whose experimental setup we base this experiment set. Each local module consists of approximately same number of layers and same number of channels in a particular layer and they are trained in isolation to each other as shown in Figure 2. When performing multi-layer splits in the context of the CNN, the inference time of the overall network is decreased (fewer cross-connections lead to fewer computations).

Comparison with other local learning methods: Here, we compare the proposed approach with other local-learning methods such as Decoupled Greedy Learning (DGL) Belilovsky et al. (2020) and local learning with Information Propagation (InfoPro) Wang et al. (2021). The other local-learning approaches only split the network across depth, unlike our approach of splitting across depth as well as width. Such splitting across both directions gives several advantages in performance and computation speed. We consider other local-learning methods as a special case of G = 1 for comparison. We note an inefficiency of the InfoPro method is its decoder model that tries to reconstruct the input to the network at each layer, leading to high computational cost at deeper layers (due to spatially upsampling a high number of channels).

Results on CIFAR-10 image classification benchmarks are presented in Figure 3. In our setup, we consider the three factors for comparison, total training time, inference time, and final performance as these yield a number of trade-offs. For a fair consideration of the training time, we consider the maximum computation time of any sub-component for a method. It can be observed that various configurations of the Grouped Neuron DGL method have better tradeoffs in terms of performance, total training time and inference speed when compared to DGL and InfoPro with ResNet-32. We also report the performance of the model corresponding to G with K=1, which serves as a baseline of running individual sub-networks trained independently and recombined by a fully connected layer. This naive baseline underperforms in terms of training time and accuracy tradeoffs.

5 Conclusions

We have investigated the idea of training independent local neuron groups for standard image classification, focusing on the convolutional neural network. Our results suggest this approach can allow for increased parallelization of local learning methods. Future investigation will study more efficient auxiliary models and communication schemes between layers.

Acknowledgements

This research was funded by internal funding Concordia. We acknowledge resources provided by Compute Canada and Calcul Quebec.

References

Belilovsky, E., Eickenberg, M., and Oyallon, E. Greedy layerwise learning can scale to imagenet. *Proceedings of the 36th International Conference on Machine Learning (ICML)*, 2019.

Belilovsky, E., Eickenberg, M., and Oyallon, E. Decoupled greedy learning of CNNs. In III, H. D. and Singh, A. (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 736–745. PMLR, 13–18 Jul 2020. URL `https://proceedings.mlr.press/v119/belilovsky20a.html`.

Bengio, Y., Lamblin, P., Popovici, D., and Larochelle, H. Greedy layer-wise training of deep networks. In *Advances in neural information processing systems*, pp. 153–160, 2007.

Choromanska, A., Cowen, B., Kumaravel, S., Luss, R., Rigotti, M., Rish, I., Diachille, P., Gurev, V., Kingsbury, B., Tejwani, R., et al. Beyond backprop: Online alternating minimization with auxiliary variables. In *International Conference on Machine Learning*, pp. 1193–1202. PMLR, 2019.

Dvornik, N., Schmid, C., and Mairal, J. Diversity with cooperation: Ensemble methods for few-shot classification. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 3723–3731, 2019.

Hebb, D. The organization of behavior. emphnew york, 1949.

Huo, Z., Gu, B., and Huang, H. Training neural networks using features replay. *Advances in Neural Information Processing Systems*, 2018.

Ivakhnenko, A. G. and Lapa, V. G. *Cybernetic Predicting Devices. CCM Information Corporation.*, 1965.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Lee, D.-H., Zhang, S., Fischer, A., and Bengio, Y. Difference target propagation. In *Joint european conference on machine learning and knowledge discovery in databases*, pp. 498–515. Springer, 2015.

Nøkland, A. and Eidnes, L. H. Training neural networks with local error signals. *arXiv preprint arXiv:1901.06656*, 2019.

Veness, J., Lattimore, T., Budden, D., Bhoopchand, A., Mattern, C., Grabska-Barwinska, A., Sezener, E., Wang, J., Toth, P., Schmitt, S., et al. Gated linear networks. *arXiv preprint arXiv:1910.01526*, 2019.

Wang, Y., Ni, Z., Song, S., Yang, L., and Huang, G. Revisiting locally supervised learning: an alternative to end-to-end training. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=fAbkE6ant2`.

A Appendix

A.1 Training hyper-parameters

For experiments in Sec 4.1, the networks are trained using Adam optimizer Kingma & Ba (2014) and a batch size of 128 for total 150 epochs. We use the initial learning rate of 0.001 with decay at epochs 50 and 100 to 0.0005 and 0.0001 respectively with the dropout rate of 0.01.

For Sec 4.2 experiments, we use the hyper-parameters from Wang et al. (2021), on which we base our experiments in this section. We train the networks using SGD optimizer with a Nesterov momentum of 0.9 for 160 epochs. We use initial learning rate of 0.1 with cosine learning rate annealing. The batch size is set to 1024 and L2 weight decay ratio of 1e-4 is adopted.

A.2 Width-wise splitting of VGG6a

The neuron groups within each layer has approximately same number of neurons in each group. For 2-Group configuration, we split first three convolutional layers in (2,4,2) pattern respectively, meaning first convolutional layer is divided in 2 neuron groups and so on. Furthermore, we introduce G-Group configurations for $G \in \{4, 6, 8, 10\}$, where the layers still follow the splitting pattern of (2,4,2) multiplied by $G/2$. i.e., for 6-Group configuration, the first convolutional layer is split in 6 neuron groups and so on.

A.3 Results of Multi Layer GN-DGL

We further demonstrate numerical tables corresponding to the results in Figure 3.

Speed up comparison with end-to-end training time (Higher is better)											
KxG	**Infopro (Softmax)**	**Infopro (Contrast)**		**DGL**		**Multi-layer GN-DGL (with Stop-Gradient & Diversity)**					
						G=2		G=4		G=8	
	Test Acc.	Test Acc.	Speed up	Test Acc.	Speed up	Test Acc.	Speed up	Test Acc.	Speed up	Test Acc.	Speed up
1	93.01	93.01	1	93.01	1						
2	91.87	92.35	1.833	91.47	1.894	88.52	3.674				
4	91.36	91.25	2.625	88.62	3.622	89.11	6.304	81.45	11.982		
8	88.6	88.5	3.856	85.53	6.456	87.14	10.667	86.14	17.445	73.31	28.531
16	85.77	87.02	5.218	83.87	11.609	84.18	16.426	86.2	24.522	80.78	33.736
32						82.43	23.072	83.5	30.47	83.31	39.333

Table 4: ResNet-32 on CIFAR-10 (Training speed up comparison)

Speed up comparison with end-to-end inference time (Higher is better)											
KxG	**Infopro (Softmax)**	**Infopro (Contrast)**		**DGL**		**Multi-layer GN-DGL (with Stop-Gradient & Diversity)**					
						G=2		G=4		G=8	
	Test Acc.	Test Acc.	Speed up	Test Acc.	Speed up	Test Acc.	Speed up	Test Acc.	Speed up	Test Acc.	Speed up
1	93.01	93.01	1	93.01	1						
2	91.87	92.35	1	91.47	1	88.52	1.961				
4	91.36	91.25	1	88.62	1	89.11	1.899	81.45	3.777		
8	88.6	88.5	1	85.53	1	87.14	1.785	86.14	3.448	73.31	7.03
16	85.77	87.02	1	83.87	1	84.18	1.612	86.2	2.936	80.78	5.824
32						82.43	1.338	83.5	2.324	83.31	4.336

Table 5: ResNet-32 on CIFAR-10 (Inference speed up comparison)