# Tangent Space Separability in Feedforward Neural Networks

**Bàlint Daróczy**
Institute for Computer Science and Control (SZTAKI)
H-1111, Budapest, Hungary
daroczyb@ilab.sztaki.hu

**Rita Aleksziev**
Institute for Computer Science and Control (SZTAKI)
H-1111, Budapest, Hungary
alexievr@ilab.sztaki.hu

**András Benczúr**
Institute for Computer Science and Control (SZTAKI)
H-1111, Budapest, Hungary
benczur@ilab.sztaki.hu

## Abstract

Hierarchical neural networks are exponentially more efficient than their corresponding "shallow" counterpart with the same expressive power, but involve huge number of parameters and require tedious amounts of training. By approximating the tangent subspace, we suggest a sparse representation that enables switching to shallow networks, GradNet after a very early training stage. Our experiments show that the proposed approximation of the metric improves and sometimes even surpasses the achievable performance of the original network significantly even after a few epochs of training the original feedforward network.

## 1 Introduction

Recent empirical results of deep hierarchical models go beyond traditional bounds in generalization theory [29, 2], algorithmic complexity [20] and local generalization measures [11, 22, 24, 16, 15]. Even simple changes in the models or optimization eventuate significant increase or decrease in performance. Beside exciting empirical phenomenon (e.g. larger networks generalize better, different optimization with zero training error may generalize differently [23]) and theoretical developments (e.g. flatness of the minimum can be changed arbitrarily under some meaningful conditions via exploiting symmetries [7]) our understanding of deep neural networks still remain incomplete [23]. There are several promising ideas and approaches inspired by statistical physics [28], tensor networks [30] or in our case particularly by differential geometry [1, 25, 14].

We investigate the connection between the structure of a neural network and Riemannian manifolds to utilize more of their potential. In a way, many of the existing machine learning problems can be investigated as statistical learning problems. Although information geometry [1] plays an important role in statistical learning, the geometrical properties of target functions both widely used and recently discovered, along with those of the models themselves, are not well studied.

Over the parameter space and the error function we can often determine a smooth manifold [25]. In this paper we investigate the tangent bundle of this manifold in order to take advantage of specific

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

Riemannian metrics having unique invariance properties [31, 3]. We use the partial derivatives in the tangent space as representation of data points. The inner products in the tangent space are quadratic, therefore if we separate the samples with a second order polynomial, then the actual metric will be irrelevant.

The geometrical property of the underlying manifold was used for optimizing generative models [27] and as a general framework for optimization in [25, 32], neither of them utilize the tangent space as representation. The closest to our method are [13] where the authors used the diagonal of the Fisher information matrix of some generative probability density functions in a kernel function for classification. Martens and Grosse [21] approximated the Amari's natural gradient [1] with block-partinioning the Fisher information matrix. The authors show that, under some assumptions the proposed Kronecker-factored Approximate Curvature (K-FAC) method still preserves some invariance results related to Fisher information. Closed formula for Gaussian Mixtures was proposed in [26]. Our contributions are the following:

- We suggest an approximation algorithm for the inner product space, in case of weekly trained networks of various sparsities.
- We give heuristics for classification tasks where the coefficients of the monomials in the inner product are not necessarily determined.

Our experiments were done on the CIFAR-10 [17] and MNIST [19] data sets. We show that if we determine the metric of the underlying feedforward neural network in an early stage of learning (after only a few epochs), we can outperform the fully trained network by passing the linearized inner product space to a shallow network.

## 2 Tangent space and neural networks

Feed-forward networks with parameters $\theta$ solve the optimization problem $\min_\theta f(\theta) = \mathbb{E}_{\mathcal{X}}[l(x;\theta)]$ where $l(x;\theta)$ is usually a non-convex function of configuration parameters $\theta$. In case of discriminative models the loss function depends on the target variable as well: $l(x;c,\theta)$.

First, we define a differential manifold $(\mathcal{M})$ based on $l(x;\theta)$ by assigning a tangent subspace and a particular vector to each configuration point, a parameter sample pair $(x,\theta)$. In case of Riemann the metric induced via an inner product $g_{x,\theta} : T_{x,\theta}M \times T_{x,\theta}M \to \mathbb{R}$ where $\{x,\theta\} \in \{\mathcal{X},\Theta\} \subset \mathcal{M}$. If we minimize over a finite set of known examples, then the problem is closely related to the empirical risk minimization and loglikelihood maximization.

The parameter space of continuously differentiable feedforward neural networks (CDFNN) usually has a Riemannian metric structure [25]. Formally, let $X = \{x_1,..,x_T\}$ be a finite set of known observations with or without a set of target variables $Y = \{y_1,..,y_T\}$ and a directed graph $N = \{V,E\}$ where $V$ is the set of nodes with their activation functions and $E$ is the set of weighted, directed edges between the nodes. Let the loss function $l$ be additive over $X$. Now, in case of generative models, the optimization problem has the form $\min_{f\in\mathcal{F}_N} l(X;f) = \min_{f\in\mathcal{F}_N} \frac{1}{T}\sum_i^T l(x_i;f)$ where $\mathcal{F}_N$ is the class of neural networks with structure $N$. Optimization can be interpreted as a "random walk" on the manifold with finite steps defined by some transition function between the points and their tangent subspaces.

The general constraint about Riemannian metrics is that the metric tensor should be symmetric, positive definite and the inner product in the tangent space assigned to any point on a finite dimensional manifold has to be able to be formalized as $< x,x >_\theta = dx^T G_\theta dx = \sum_{i,j} g^\theta_{i,j} dx^i dx^j$. The metric $g_\theta$ varies smoothly by $\theta$ on the manifold and is arbitrary given the conditions.

### 2.1 GradNet

Let our loss function $l(x;\theta)$ be a smooth, positive, parametric real function where $x \in \mathbb{R}^d$ and $\theta \in \mathbb{R}^n$. We define a class of $n \times n$ positive semi-definite matrices as

$$h_\theta(x) = \nabla_\theta l(x;\theta) \otimes \nabla_\theta l(x;\theta) \tag{1}$$

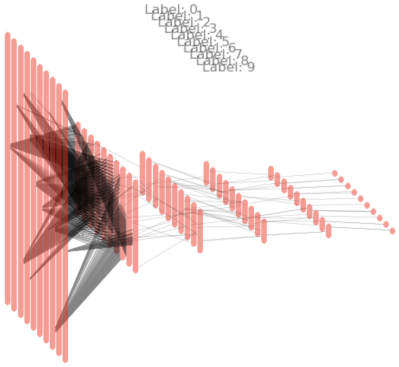


Figure 1: Important edges in the gradient graph of the MNIST network.

where $\nabla_\theta l(x;\theta) = \{\frac{\partial l(x;\theta)}{\partial \theta_1}, .., \frac{\partial l(x;\theta)}{\partial \theta_n}\}$. Using eq. (1) we can determine a class of Riemannian metrics

$$G = g_X(h_\theta(x)) \tag{2}$$

where $g_X$ is a quasi arithmetic mean over $X$. For example, if $g_X$ is the arithmetic mean, then the metric is $G_\theta = \mathbf{AM}_X[\nabla_\theta l(x_i|\theta)\nabla_\theta l(x_i|\theta)^T]$ and we can approximate it with a finite sum as $G^{kl}_\theta \approx \sum_i \omega_i \left(\frac{\partial}{\partial \theta_k} l(x_i|\theta)\right)\left(\frac{\partial}{\partial \theta_l} l(x_i|\theta)\right)$ with some importance $\omega_i$ assigned to each sample. Through $G$, the tangent bundle of the Riemannian manifold induces a normalized inner product (kernel) at any configuration of the parameters formalized for two samples $x_i$ and $x_j$ as

$$< x_i, x_j >_\theta = \nabla_\theta l(x_i;\theta)^T G_\theta^{-1} \nabla_\theta l(x_j;\theta) \tag{3}$$

where the inverse of $G_\theta$ is positive semi-definite since $G_\theta$ is positive semi-definite.

The quadratic nature of the Riemannian metrics is a serious concern due to high dimensionality of the tangent space. There are several ways to determine a linear inner product: decomposition or diagonal approximation of the metric, or quadratic flattening. Due to high dimensionality, both decomposition and flattening can be highly inefficient, although flattening can be highly sparse in case of sparsified gradients. Our main idea is to combine per sample sparsification with quasi-blockdiagonal approximation, shown in Fig. 2.

But before we trade on our new sparse representation, we have to handle another problem. Since our models are discriminative and not generative, the loss surfaces are not known in absence of the labels. Hence we define GradNet, Fig. 3, a multi-layer network over the tangent space, as $h_{\text{GradNet}}(x; l(x, \hat{c}; \theta))$ with the assumption that the final output of the network after training is $\text{argmax}_c \sum_{\hat{c}} h_{\text{GradNet}}(x; l(x; \hat{c}, \theta)$.

Results in [5, 6, 4] indicate high over-parametrization and redundancy in the parameter space, especially in deeper feedforward networks, therefore the outer product structure is highly blocked particularly in case of ReLU networks and sparsified gradients.

Let us consider a multi-layer perceptron with rectified linear units and a gradient graph with sparsity factor $\alpha$ corresponding to the proportion of the most important edges in the gradient graph derived by the Hessian for a particular sample. The nodes are corresponding to the parameters and we connect two nodes with a weighted edge if their value in the Hessian matrix is nonzero and their layers are neighbors. We sparsify the gradient graph layer-by-layer by the absolute weight of the edges. As shown in Fig. 1, the resulting sparse graph describes the structure of the MINST task well: for each label 0–9, a few nodes are selected in the first layer and only a few paths leave the second layer.

In order to avoid having too much parameters to train, we chose the hidden layer of our network to have a block-like structure demonstrated in Figure 3. This model is capable of capturing connections between gradients from adjacent layers of the base network.

The final algorithm is

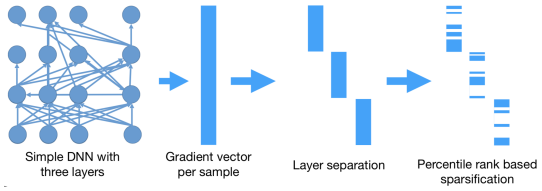


Figure 2: Layer-by-layer per sample sparsification by percentile ranking

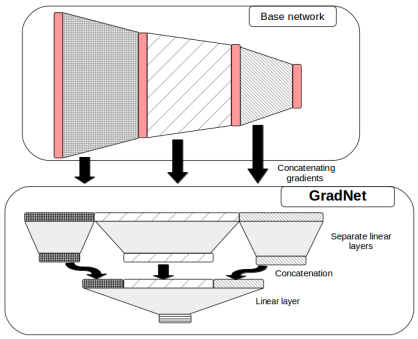


Figure 3: GradNet

**Algorithm 1** Training procedure of GradNet

**Input:** Pre-trained model with parameter $\theta$, dataset $D$, GradNet $N$, normalization method $n$, number of epochs $t$ and sparsification coefficient $\alpha$
**Output:** Trained GradNet

**procedure** TRAIN($M, D, N, n, t$)
  **for** `epoch from 1 to t` **do**
    **for** $batch$ `in` $D$ **do**
      $X \leftarrow augmentation(batch)$
      $c \leftarrow$ real labels for each data point in $batch$
      $\hat{c} \leftarrow$ random labels for each data point in $batch$
      $X_g \leftarrow \nabla_\theta(l(x; \hat{c}, \theta))$ for each data point in the $batch$
      $\hat{X_g} \leftarrow n(X_g, \alpha)$ ▷ normalization and sparsification
      $N \leftarrow update(N, \hat{X_g}, c)$ ▷ update network with normalized gradients
  **return** $N$ ▷ Return trained $N$
  Prediction for data point $x$: $\text{argmax}_c \sum_{\hat{c}} N(n(\nabla_\theta l(x; \hat{c}, \theta)))$

## 3 Experiments

In our first experiment we set Markov Random Fields (MRF), particularly Restricted Boltzmann Machines (RBM)[10] with 16 and 64 hidden units on the first half of the MNIST [18] training set and calculated the normalized gradient vectors based on the Hammersley-Clifford theorem [8]. We used the RBMs output and the normalized gradient vectors within a linear model. The results (see Appendix) show that the normalized gradient vector space performed very similar after some initialization with $1k$ sample and after training while the original latent space performed poorly immediately after initialization.

We trained several traditional Convolutional Neural Networks (CNN) and descendants such as residual [9] and dense networks [12] over the CIFAR-10 dataset [17] as the base model for GradNet. In order to find the optimal architecture, the right normalization process and regularization technique, and the best optimization method, we experimented with a large number of setups (see Appendix). Interestingly, the GradNet surprassed the performance of the underlying CNN, at some settings even after only one epoch. Source codes are available at `https://github.com/daroczyb/gradnet`.

## 4 Conclusions

By approximation of the inner product, we showed promising results with our GradNet network in the sparsified gradient space. GradNet outperformed the original network even if built from a few epochs of the original network. In the future, we would like to extend our method to Hessian metrics and further investigate sparsity and possible transitions to less complex manifolds via pushforward and random orthogonal transformations.

## 5 Acknowledgement

The publication was supported by the Hungarian Government project 2018-1.2.1-NKP-00008: Exploring the Mathematical Foundations of Artificial Intelligence, by the Higher Education Institutional Excellence Program, and by the Momentum Grant of the Hungarian Academy of Sciences. B.D. was supported by an MTA Premium Postdoctoral Grant 2018.

## References

[1] Amari, S.-i. (1996). Neural learning in structured parameter spaces-natural Riemannian gradient. In *NIPS*, pages 127–133. Citeseer.

[2] Bartlett, P. L. and Maass, W. (2003). Vapnik-Chervonenkis dimension of neural nets. *The handbook of brain theory and neural networks*, pages 1188–1192.

[3] Campbell, L. (1986). An extended Čencov characterization of the information metric. *Proceedings of the American Mathematical Society*, 98(1):135–141.

[4] Choromanska, A., Henaff, M., Mathieu, M., Arous, G. B., and LeCun, Y. (2015). The loss surfaces of multilayer networks. In *Artificial Intelligence and Statistics*, pages 192–204.

[5] Denil, M., Shakibi, B., Dinh, L., De Freitas, N., et al. (2013). Predicting parameters in deep learning. In *Advances in neural information processing systems*, pages 2148–2156.

[6] Denton, E. L., Zaremba, W., Bruna, J., LeCun, Y., and Fergus, R. (2014). Exploiting linear structure within convolutional networks for efficient evaluation. In *Advances in neural information processing systems*, pages 1269–1277.

[7] Dinh, L., Pascanu, R., Bengio, S., and Bengio, Y. (2017). Sharp minima can generalize for deep nets. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1019–1028. JMLR. org.

[8] Hammersley, J. M. and Clifford, P. (1971). Markov fields on finite graphs and lattices. *seminar, unpublished*.

[9] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778.

[10] Hinton, G. E. (2002). Training products of experts by minimizing contrastive divergence. *Neural computation*, 14(8):1771–1800.

[11] Hochreiter, S. and Schmidhuber, J. (1997). Flat minima. *Neural Computation*, 9(1):1–42.

[12] Iandola, F., Moskewicz, M., Karayev, S., Girshick, R., Darrell, T., and Keutzer, K. (2014). Densenet: Implementing efficient convnet descriptor pyramids. *arXiv preprint arXiv:1404.1869*.

[13] Jaakkola, T. S., Haussler, D., et al. (1999). Exploiting generative models in discriminative classifiers. *Advances in neural information processing systems*, pages 487–493.

[14] Kanwal, M., Grochow, J., and Ay, N. (2017). Comparing information-theoretic measures of complexity in boltzmann machines. *Entropy*, 19(7):310.

[15] Kawaguchi, K., Kaelbling, L. P., and Bengio, Y. (2017). Generalization in deep learning. *arXiv preprint arXiv:1710.05468*.

[16] Keskar, N. S., Mudigere, D., Nocedal, J., Smelyanskiy, M., and Tang, P. T. P. (2016). On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*.

[17] Krizhevsky, A. (2009). Learning multiple layers of features from tiny images.

[18] LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

[19] LeCun, Y. and Cortes, C. (2010). MNIST handwritten digit database.

[20] Liu, T., Lugosi, G., Neu, G., and Tao, D. (2017). Algorithmic stability and hypothesis complexity. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2159–2167. JMLR. org.

[21] Martens, J. and Grosse, R. (2015). Optimizing neural networks with kronecker-factored approximate curvature. In *International conference on machine learning*, pages 2408–2417.

[22] Neyshabur, B., Bhojanapalli, S., McAllester, D., and Srebro, N. (2017). Exploring generalization in deep learning. In *Advances in Neural Information Processing Systems*, pages 5947–5956.

[23] Neyshabur, B., Li, Z., Bhojanapalli, S., LeCun, Y., and Srebro, N. (2018). Towards understanding the role of over-parametrization in generalization of neural networks. *arXiv preprint arXiv:1805.12076*.

[24] Novak, R., Bahri, Y., Abolafia, D. A., Pennington, J., and Sohl-Dickstein, J. (2018). Sensitivity and generalization in neural networks: an empirical study. *arXiv preprint arXiv:1802.08760*.

[25] Ollivier, Y. (2015). Riemannian metrics for neural networks i: feedforward networks. *Information and Inference: A Journal of the IMA*, 4(2):108–153.

[26] Perronnin, F. and Dance, C. (2007). Fisher kernels on visual vocabularies for image categorization. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–8. IEEE.

[27] Rifai, S., Dauphin, Y., Vincent, P., Bengio, Y., and Muller, X. (2011). The manifold tangent classifier. In *NIPS*, volume 271, page 523.

[28] Rolnick, D. and Tegmark, M. (2017). The power of deeper networks for expressing natural functions. *arXiv preprint arXiv:1705.05502*.

[29] Sontag, E. D. (1998). VC dimension of neural networks. *NATO ASI Series F Computer and Systems Sciences*, 168:69–96.

[30] Stoudenmire, E. and Schwab, D. J. (2016). Supervised learning with tensor networks. In *Advances in Neural Information Processing Systems*, pages 4799–4807.

[31] Čencov, N. N. (1982). Statistical decision rules and optimal inference. *American Mathematical Society*, 53.

[32] Zhang, H., Reddi, S. J., and Sra, S. (2016). Riemannian svrg: fast stochastic optimization on riemannian manifolds. In *Advances in Neural Information Processing Systems*, pages 4592–4600.

## Appendix

We measured the performance of various GradNet models on the gradient space of a CNN trained on the first half of the CIFAR-10 [17] training dataset. We used the other half of the dataset and random labels to generate gradient vectors to be used as training input for the GradNet. In the testing phase we use all of the gradient vectors for every data point in the test set, we give them all to the network as inputs, and we define the prediction as the index of the maximal element in the sum of the outputs.

During our experiments, as a starting point we stopped the underlying original CNN at $0.72$ accuracy and compared the following settings.

Table 1: Performance measure of the normalized gradient based on RBM.

| MNIST | | |
|---|---|---|
| #hidden | Base RBM | GradNet |
| 16 | 0.6834 | 0.9675 |
| 16 | 0.8997 | 0.9734 |
| 64 | 0.872 | 0.9822 |
| 64 | 0.9134 | 0.9876 |

Table 2: Performance measure of the improved networks.

| CIFAR | | |
|---|---|---|
| Base NN | GradNet | Gain |
| 0.79 | 0.8289 | +4.9% |
| 0.76 | 0.8201 | +7.9% |
| 0.74 | 0.8066 | +9% |
| 0.72 | 0.7936 | +10.2% |
| 0.68 | 0.7649 | +12.5% |
| 0.65 | 0.7511 | +15.5% |
| 0.62 | 0.7274 | +17.3% |
| 0.55 | 0.7016 | +27.5% |
| 0.51 | 0.6856 | +34.4% |
| 0.49 | 0.678 | +38.3% |

| MNIST | | |
|---|---|---|
| Base NN | GradNet | Gain |
| 0.92 | 0.98 | +6.5% |
| 0.96 | 0.9857 | +2.7% |
| 0.9894 | 0.9914 | +0.2% |

- Regarding regularization, we considered using dropout, batch normalization, both of them together, or none.
- We experimented with SGD and Adam optimization methods.
- Since we suspected that not all coordinates of the gradients are equally important, we only used the elements of large absolute value making the process computationally less expensive. We kept the elements of absolute value greater than the $q$-th percentile of the absolute value vector, and we tested our model setting this $q$ value for 99, 95, 90, 85, 80 and 70. We also tried a method where we pre-computed the indices of the most important $10\%$ of the values for each label, and used this together with the above technique.
- In order to determine the exact structure of the GradNet, we tried layers and blocks of different sizes. These models differ only in the size and partition of the middle layer, which were the following in our tests: 5+25+10; 20+100+40; 10+50+20; 5+100+25; 10+200+50; 20+400+100.
- In terms of normalization, we ran tests using standard norm with and without L2-norm following it; scale norm; power norm with exponents $\frac{1}{8}$, $\frac{1}{4}$, $\frac{1}{2}$, and 2; and scale norm followed by power norm with exponent $\frac{1}{2}$.

Learning curves for the different networks are presented in Figures 4 - 9. We observed that SGD gives a better performance than Adam (Fig. 4), and that regularization is not needed (Fig. 5). We also found that it is sufficient to use the elements of each gradient vector that are greater than the 85-th percentile of all of the absolute values in the vector (Fig. 6). Regarding structure, the best-performing GradNet was the one with hidden layer of size 130 partitioned into sublayers of sizes 5, 100 and 25 (Fig. 7). Out of all the considered normalization methods, the scale norm and the power norm together gave the most satisfactory outcome (Fig. 8,9).

To show the performance of the GradNet with these particular settings, we took snapshots of a CNN at progressively increasing levels of pre-training, and we trained the GradNet on the gradient sets of these networks. We ran these tests using a CNN trained on half of the CIFAR dataset and with one trained on half of MNIST. Table 2 shows the accuracies of all the base networks together with the accuracies of the corresponding GradNets.

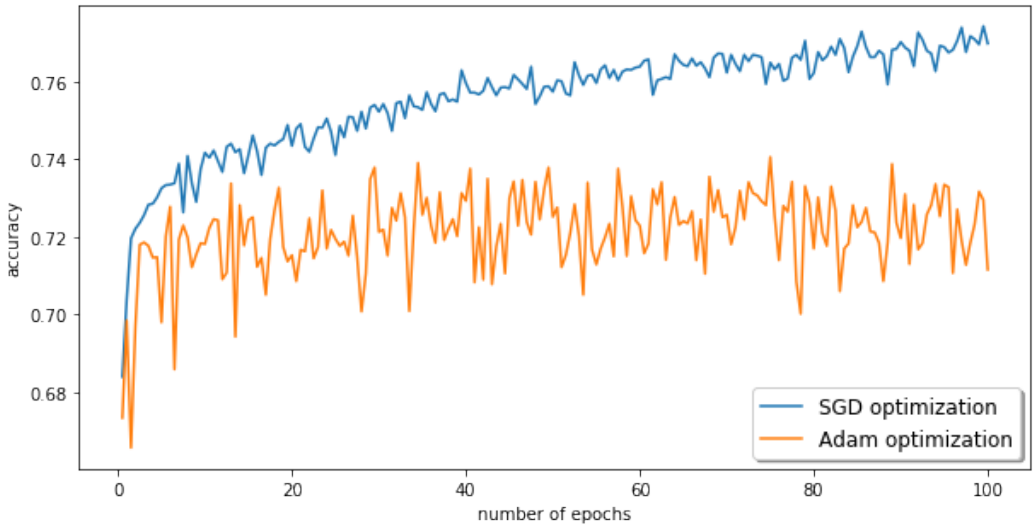


Figure 4: Optimization methods

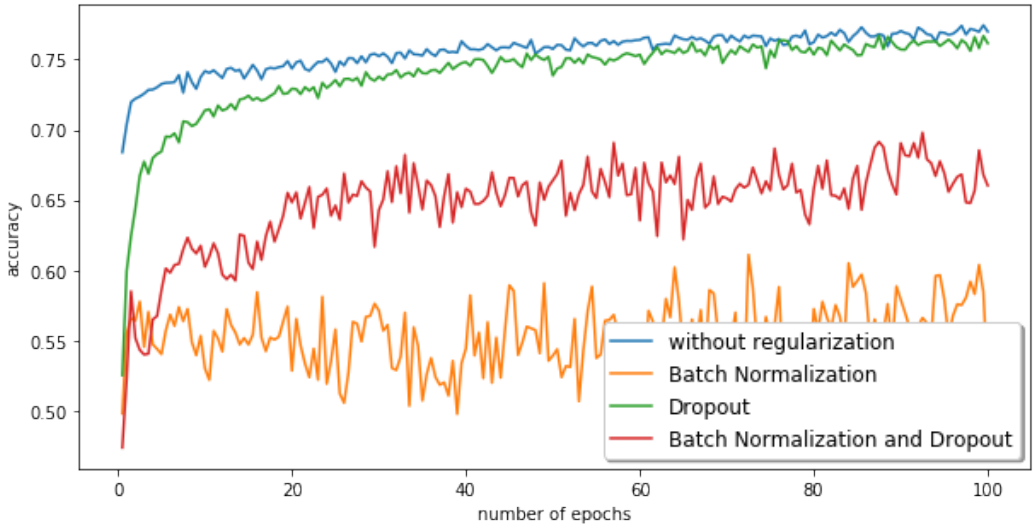


Figure 5: Regularization methods

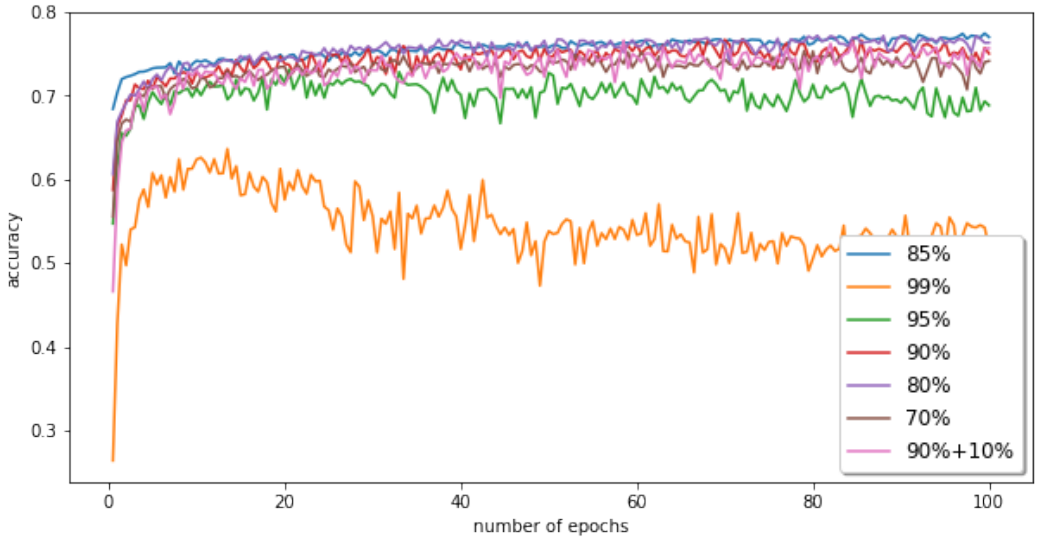


Figure 6: Selection percentile

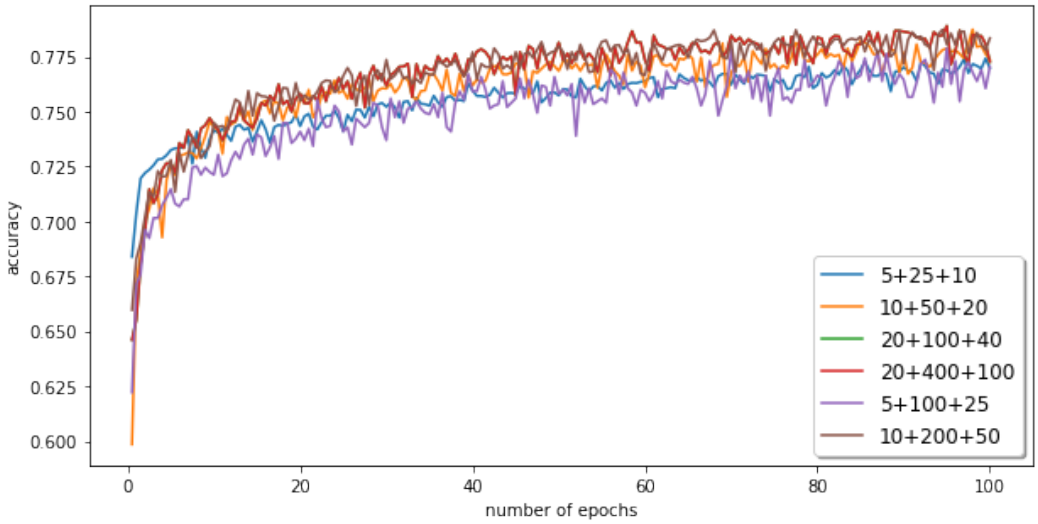


Figure 7: Structure

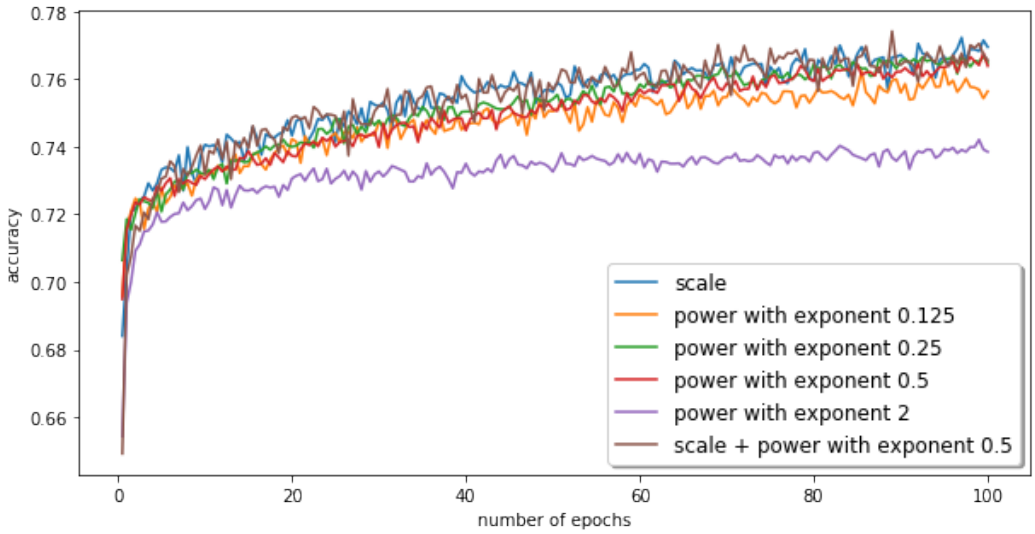


Figure 8: Normalization with structure 5+25+10

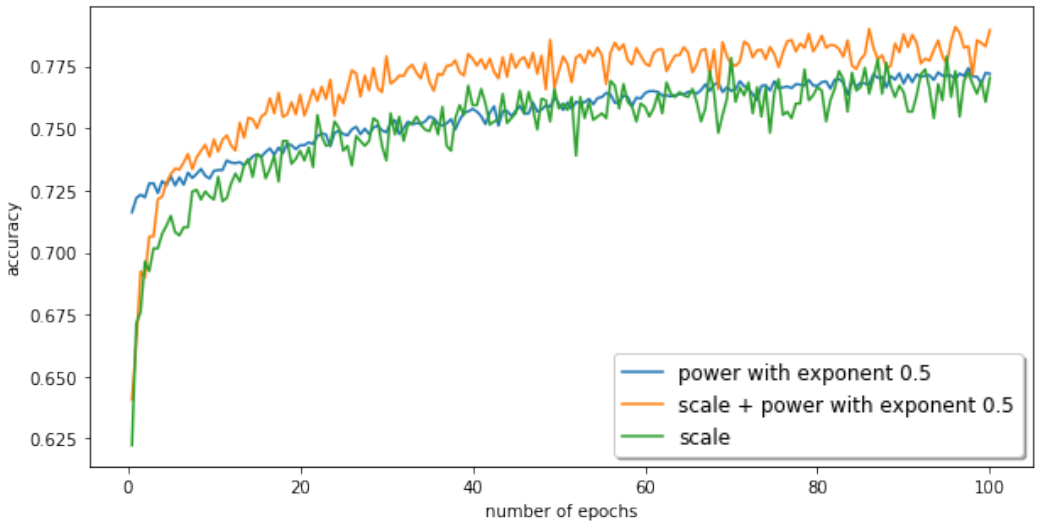


Figure 9: Normalization with structure 5+100+25