# Exploring neural representations in deep neural networks trained with local error signals

Fabian Bergmann, Georg Kruse *Technical University of Berlin*, Raphael Leuner *Freie University of Berlin*

***Abstract*—In the recent boom of Deep Learning, Backpropagation of error has become the dominant method for optimizing Deep Neural Networks and is the only optimization method that is widely applied in the field. It has been argued however, that the brain, which has initially inspired some of the development of DNNs, is biologically unable to optimize networks in such a global way. Therefore, many training methods have been suggested, in which a loss and the subsequent weight optimization is calculated at the level of each individual layer. This raises the question, of whether these networks learn representations in their layers with similar properties to Backpropagation trained networks.**

**In this work, two of these local loss methods, the Prediction Similarity loss and the Direct Difference Target Propagation loss have been used to train simple convolutional neural networks and these networks have been compared to their Backpropagation trained counterparts in terms of accuracies and layer-wise data representation.**

**Our results give some initial evidence towards the assumption that the Prediction Similarity loss might be a training method that is able to approximate Backpropagation both in terms of performance as well as layer-wise data representations. Direct Difference Target Propagation however did not perform well with convolutional models and therefore no statements about its similarity to Backpropagation can be made.**

## I. Introduction

CURRENT Deep Neural Networks trained by the Backpropagation algorithm were once thought to mimic layer-wise learning in the brain, however more recent research indicates that the brain is biologically not capable to propagate error information to a great extent back via the rules of calculus [10]. Instead of propagating back error information from one source through the whole network, alternative architectures are proposed that employ multiple local-error signals, which is potentially more similar to learning mechanisms found in the brain. The question arises, whether the networks trained by local-error methods learn how to extract information from the input differently than conventional Backpropagation networks, for example by exhibiting a differently structured progression of the learned representations throughout the layers.

Local-error signals can be considered as heightened supervision for each layer that tries to enforce the desired optimization with regard to the input it receives from the prior layer. Therefore, the layers are forced to build upon the representation of the previous layer. Networks trained by Backpropagation in contrast take all parameters of the network at once into account, and thus there is no regard towards the manner in which the optimization goal is accomplished. However, should Backpropagation nonetheless also exhibit sequential layer-by-layer building of representations and therefore a similar pattern of representations to locally trained networks, it could indicate that Backpropagation might to a certain degree be a sufficient surrogate learning method for more biologically valid methods.

We choose two local-error algorithms that have displayed to be capable of achieving satisfactory performance and are not intractable to calculate for substantial network sizes. We chose the Prediction Similarity Local Loss (PredSim) [16] architecture where each layer possesses its own loss function and optimizer and another trained with Direct Difference Target Propagation (DDTP) [13]. The trained models are investigated with methods that give insight on how the representations of each layer behave. We look at the Intrinsic Dimension (ID) of the representation at each layer and compare the activations between layers with Representational Similarity Analysis (RSA) and Centered Kernel Alignment (CKA).

## II. Related Work/Background

Backpropagation is biologically unrealistic for several reasons [20]: it requires weight symmetry in the forward and backward passes, propagates an global error across the entire network and adjusts all weights simultaneously. Nevertheless, current deep convolutional networks are typically trained using global Backpropagation.

Recent work tried to overcome these issues in order to create a biologically more plausible alternative to standard Backpropagation. An alternative represents Target Propagation (TP), as it uses target activations instead of errors to update the weights of each layer locally [11], [12]. TP also motivated other lines of more brain-like algorithms like dendritic microcircuits [19]. Other approaches use local loss functions to pre-train the hidden layers of networks resulting in better performance [24]. Instead of layer-wise pre-training, several recent approaches train the whole network using such layer wise error terms [21], [22]. The synthetic gradient mechanism introduced by Jaderberg et al. also uses local error signals in each layer, which are only based on the layer's activations and the labels. [23].

A popular hypothesis for the success of DNN's is their ability to learn good data representations [18]. These representations have therefore been subject to intensive research. A popular metric to analyse such representations is RSA, which originates form the field of neuroscience [3]. It measures the correlation between layer-wise activations [2]. Other metrics to analyse the similarity of layers and networks are CKA and [7] Canonical Correlation Analysis (CCA) [17] and new methods like the neuron activation subspace match model have been developed for better investigation [18].

## III. Methods

### A. Intrinsic Dimension (ID)

The ID refers to the minimal number of variables required to represent a data set. It is an important geometric property to characterize and analyze the data representations in neural networks [4]. In order to evaluate the ID of data manifolds in neural networks, in this work the TwoNN method [6] for global ID estimation is used. This method can not only be applied to curved and topologically complex data manifolds but is also computationally efficient.

TwoNN leverages the insight, that the ratio of distances between the first and the second nearest neighbour $\mu_i = r_i^{(2)}/r_i^{(1)}$ for a given data point $i$ (and under the assumption that the probability density around each point is constant) yields an explicit cumulative distribution function (cdf) $F(\mu_i)$ that depends on the ID $d$. Thus, with the following equation

$$\frac{log(1 - F(\mu_i))}{log(\mu_i)} = d. \tag{1}$$

an estimate of the intrinsic dimension $d$ can be recovered.

In order to generalize over all data points and to receive a robust estimate, an empirical cdf $F^{emp}(\mu)$ can be computed by sorting the values of $\mu_i$ in an ascending order through a permutation $\sigma$, so that it can be defined as $F^{emp}(\mu_{\sigma(i)}) = \frac{i}{N}$. Then, using the insight from (1), the points given by the coordinates

$$\{(log((\mu_i),\ -log(1 - F^{emp}(\mu_i))) \,|\, i = 1, ..., N\} \tag{2}$$

are fitted on a plane with a straight regression line that passes through the origin. The slope of the line gives the ID $d$.

The ID of convolutional layers in deep neural networks has been widely analyzed and therefore is a good metric to compare locally and globally trained CNN's. In CNN's trained with global Backpropagation, the ID throughout the layers shows a hunchback shape: At the input layers it increases with each layer, reaches a maximum at the middle layers and then continues to drop until it reaches its minimum at the output layer. The work of A. Ansuini et. al [4] indicates that the higher the maximum ID and the steeper the following implosion of ID, the better the generality.

### B. Representational Similarity Analyses (RSA)

In the field of neuroscience many models to characterize and analyze representations in biological neural networks have been introduced. There, an activity pattern is referred to as a "representation" and especially in neuroscientific research the similarity of such representations is compared to gain insights into the similarity of different biological neural networks. This approach can be transferred to artificial neural networks [3]. To evaluate the representational similarity of deep neural networks:

1) In a first step the layer wise activations for the same inputs are measured and normalized and the correlation between these representations is calculated using a correlation measure. In that manner the layer-wise similarity can be calculated.
2) In a second step this similarity or dissimilarity is aggregated in a representation dissimilarity matrix (RDM).

The RDM of RSA can offer an intuitive visual overview of how strongly correlated each layer is with every other and moreover whether general overall similarity patterns are discernible. The resulting matrix is moreover symmetric and zero on its diagonal. [2] Therefore, the lower the correlation between the representations in the layers, the higher the dissimilarity.

### C. Centered Kernel Alignment (CKA)

Another method to measure the similarity is centered kernel alignment. It shares some similarity with RSA, but in contrast, CKA does not normalize the activations of the layers and therefore results are expected to be more pronounced compared to RSA.

Equation (3) relates dot products between examples to dot products between features. The left hand side measures the similarity between the features X and Y, while the right hand side corresponds to the dot product between the inter-example similarity matrices.

$$\|X^T Y\|_F^2 = \langle vec(XX^T), vec(YY^T)\rangle_F \tag{3}$$

where $\langle\cdot,\cdot\rangle_F$ denotes the Frobenius inner product and $\|\cdot\|_F$ the Frobenius norm. Let now $K_{i,j} = k(x_i, x_j)$ and $L_{i,j} = l(y_i, y_j)$ be two linear kernels $k(x, y) = l(x, y) = x^T y$. Now the right hand side of equation 3 can be kernelized

$$\langle vec(XX^T), vec(YY^T)\rangle_F \rightarrow \langle vec(HKH), vec(HKH)\rangle_F \tag{4}$$

where $H$ denotes the centering matrix. By normalizing equation (4) we arrive at the expression known as centered kernel alignment [7]:

$$CKA(K, L) = \frac{\langle vec(HKH), vec(HKH)\rangle_F}{\|HKH\|_F \|HLH\|_F} \tag{5}$$

Equation (5) can be used, similar to RSA, as a similarity measure. The results of CKA can also be plotted in RDM's for better interpretation analog to RSA.

## IV. Biological Learning Methods

We choose to investigate two variants of training neural networks, where learning information is passed to each individual layer. They are potentially more akin to biological learning processes in the brain than networks trained by global Backpropagation.

### A. Prediction Similarity Local Loss

Each layer of the network is trained individually via SGD from its own dedicated loss function. For the last layer, the standard cross-entropy loss is used. The remaining layers receive a weighted combination of cross-entropy and similarity-matching loss. Given the input $X = [x_1, \ldots, x_n]$ of a layer with parameters $\theta_1$ the similarity-matching loss is given by:

$$\min_{\Theta} \|S\left(NeuralNet\left(X; \Theta\right)\right) - S(Y)\|_F^2 \tag{6}$$

In the assistance neural network $NeuralNet$ the input $X$ first passes through the original layer with parameters $\theta_1$ followed by further layers parametrized by $\theta_2$. Thus, the similarity matching loss intends to optimize over the parameters of $NeuralNet$ given by $\Theta = [\theta_1, \theta_2]$. Usually $\theta_2$ is just composed of the parameters of a single layer. A self similarity metric $S$ indicates how similar each entry is from one another. With euclidean distance as metric, similarity corresponds to distance.

Therefore, the term $S(Y)$, where $Y = [y_1, \ldots, y_n]$ are one-hot encoded target values, represents the desired similarity that data points mapped by $NeuralNet$ should exhibit. As a result $NeuralNetwork$ intends to find a representation where data points of the same class are clustered together, while data points from different classes lie apart from each other. In the optimal case, the data points lie in clear cut clusters of their respective classes.

The architecture is taken from [16] and a visual representation of it can be found in Figure 1.

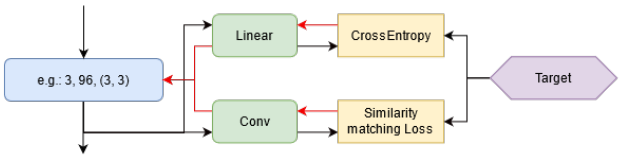


Fig. 1: Visual representation of the training of one layer using the Prediction Similarity loss, based on [16]

### B. Target Propagation

Target Propagation is built on the idea of reconstructing targets for each individual layer using inverse functions. Based on the concept of autoencoders, this optimizes the previous layers towards the activations it should have had in order to produce the correct output in the next layer [1][11]. Here, we will use Direct Difference Target Propagation (DDTP) proposed by [13] to train a network with independent losses at each individual layer. The forward activations for a given minibatch $b$ are defined by

$$h_i^{(b)} = f_i(h_{i-1}^{(b)}) \tag{7}$$

where $f_i$ in our case is the $i$-th convolutional layer combined with a relu activation function. In contrast to Vanilla Target Propagation, in DDTP the targets are not propagated backwards layer by layer; instead the output targets

$$\hat{h}_L^{(b)} = h_L^{(b)} - \hat{\eta}\frac{\partial \mathcal{L}(h_L, y)}{\partial h_L^{(b)}} \tag{8}$$

are directly propagated to every previous layer. Here, $\mathcal{L}$ is the cross entropy loss between the last activations and the targets.

This calculation to create the local targets $\hat{h}_i$ is done with direct linear mappings $g_i$ using the "difference target propagation" formula proposed by [12] in order to increase stability.

$$\hat{h}_i^{(b)} = g_i(\hat{h}_L^{(b)}) + h_i^{(b)} - g_i(h_L^{(b)}) \tag{9}$$

The convolutional layers $f_i$ are optimized by gradient descent using the mean squared error $\mathcal{L}_i$ between activations and targets

$$\mathcal{L}_i = \frac{1}{B}\sum_b ||\hat{h}_i^{(b)} - h_i^{(b)}||_2^2 \tag{10}$$

A visual representation of the training can be found in Figure 2

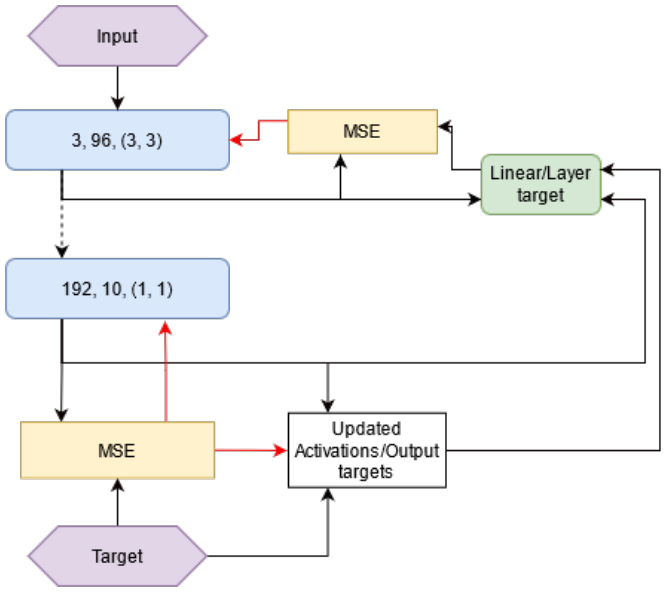


Fig. 2: Visual representation of the training of one layer in the DDTP network

*1) Train Feedback Layers:* To train the feedback mappings $g_i$ for each layer but the last, the activities are being corrupted by adding Gaussian noise

$$\tilde{h}_i^{(b)} = h_i^{(b)} + \sigma\epsilon, \epsilon \in \mathcal{N}(0, 1) \tag{11}$$

These corrupted activities are then passed forward through the following layers creating separate corrupted final output activities for every output layer

$$\tilde{h}_k^{(b)} = f_i(\tilde{h}_{k-1}^{(b)} \tag{12}$$

The corrupted output activities are then propagated backwards using the feedback layers (reconstruction)

$$h_i^{rec(b)} = g_i(\tilde{h}_L^{(b)}) + h_i^{(b)} - g_i(h_L^{(b)}) \tag{13}$$

The feedback layers $g_i$ are then optimized by gradient descent using the mean squared error $\mathcal{L}_i^{diff,rec}$ between corrupted activities and reconstructed corrupted activities

$$\mathcal{L}_i^{diff,rec} = \frac{1}{B}\sum_b ||\hat{h}_i^{(rec(b)} - \tilde{h}_i^{(b)}||_2^2 \tag{14}$$

## V. Experiments

In our experiments we tried to improve comparability by using an abbreviation of the straightforward, but nonetheless adequately performing, "All Convolutional Neural Network" (AllCNN-C) architecture [15]. Furthermore, we trained and tested the algorithms on the sufficiently complex CIFAR-10 image classification benchmark dataset [14].

### A. AllCNN-C architecture

The AllCNN-C architecture was chosen because it consists only of convolutional layers, apart from a single average pooling layer at the end. Furthermore, we removed the dropouts to reduce the types of layers to a minimum while still yielding comparable performance. This also makes it relatively inexpensive to train with all three methods. The properties of its nine convolutional layers are described in table I. Every convolutional layer is followed by a relu activation, the last layer by a softmax activation.

For the model trained with DDTP, a shortened architecture made out of only four convolutional layers was chosen, as this showed a slightly higher accuracy and much faster training.

A Visualization of both architectures can be found in the Appendix figures 8 and 9

| AllCNN-C | shortAllCNN-C |
|---|---|
| 3 x 3 conv. 96, stride 1 | 5 x 5 conv. 32, stride 2) |
| 3 x 3 conv. 96, stride 1 | |
| 3 x 3 conv. 96, stride 2 | |
| 3 x 3 conv. 192, stride 1 | 5 x 5 conv. 64, stride 2) |
| 3 x 3 conv. 192, stride 1 | |
| 3 x 3 conv. 192, stride 2 | |
| 3 x 3 conv. 192, stride 1 | 8 x 8 conv. 64, stride 2) |
| 1 x 1 conv. 192, stride 1 | |
| 1 x 1 conv. 10, stride 1 | 1 x 1 conv. 10, stride 1) |
| global averaging | global averaging |

TABLE I: Architectures of AllCNN-C and shortened AllCNN-C network, based on [15]

### B. Training of the PredSim Local Loss Network

The PredSim Local-Loss Network was trained at every layer with an Adam optimizer and a learning rate of $0.0005$. The comparable conventional backprop network was also trained with an Adam optimizer and the same learning rate. Both were trained for $50$ epochs. These hyperparameters were chosen, as they yielded satisfactory performance for both networks. The networks where trained three times with different seeds. The PredSim Local Loss Network achieved an average test set accuracy of $0.83\%$ while the Backpropagation network reached $0.82\%$.

### C. Training of the DDTP Network

The shorter network architecture with increased kernel sizes compared to the standard AllCNN-C architecture was chosen as this showed a faster conversion rate and reached higher ac-curacies on the CIFAR-10 dataset. Both the DDTP trained network and the network trained with Backpropagation of error shared the same layer-wise architecture as well as the same hyperparameters. They were trained using an Adam optimizer with a learning rate of $0.001$, with a batchsize of $128$. The learning rate of the linear feedback layers was $0.005$, the feedback layers were trained after each epoch. Three networks with different seeds were trained for $100$ epochs and the results were averaged over these three runs for each training method. Despite the superior performance of the shortened network compared to the original AllCNN-C architecture, the DDTP loss function did not show a great performance with an average accuracy of $0.52\%$, compared to $0.71\%$ for the Backpropagation trained network.

### D. Results

The comparison of the ID in Figure 3 shows that the AllCNN-C trained with global Backpropagation and the PredSim loss AllCNN-C show a similar development of ID and both show the so called "hunchback-shape" as described previously. This indicates that both methods to a certain degree might have learnt a representation sequence of similar quality and therefore might generalize over the data in a similar manner.

Nonetheless, some subtle differences are observable. First, the PredSim loss network reaches a higher maximum followed by a steeper drop compared to the conventional Backpropagation network. Second, the globally trained network shows a bump before the last layer. The experiments are however not substantial enough to draw any strong conclusions from these results.

The shape of the ID curve of the short AllCNN-C network trained with global Backpropagation and DDTP do not completely show the expected "hunchback-shape". However, since the networks are very shallow, it can be argued that the shape seen in Figure 3 are a version of the expected shape, especially when it comes to the decrease of intrinsic dimensions towards the end of the network. Here, both training architectures show a remarkable similarity.

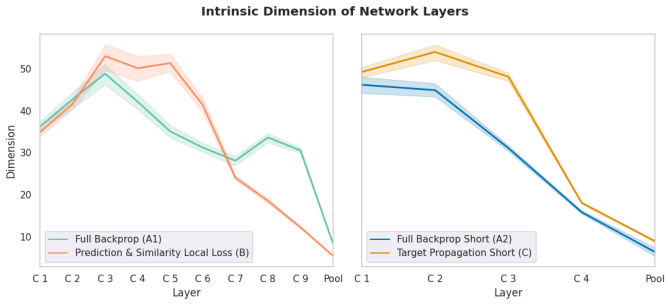


Fig. 3: Intrinsic dimensions with 95% confidence interval comparing locally trained networks with Backpropagation trained networks. Left: PredSim loss on original AllCNNC; Right: DDTP loss on shortened AllCNNC

The RSA and CKA matrices between Backpropagation and PredSim loss networks in figure 4 and 5 show a similar pattern of similarity of the individual networks respectively. Both individual networks show the traditional network structure, where neighbouring layers are the most similar to one another. This similarity decreases along the off-diagonal with

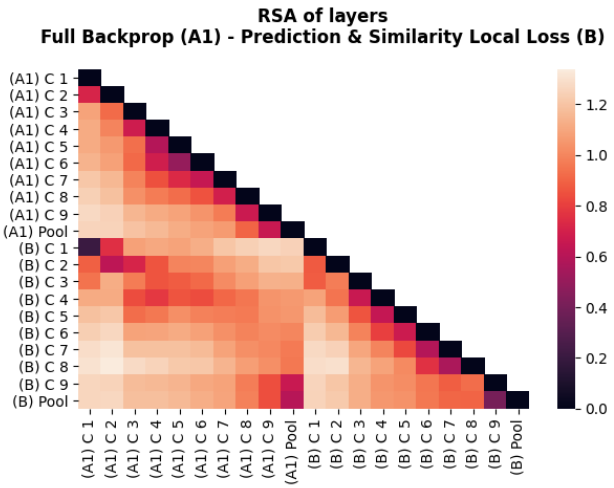


Fig. 4: Representational Dissimilarity Matrix from RSA calculation between backprobagation (A1) and PredSim (B) loss

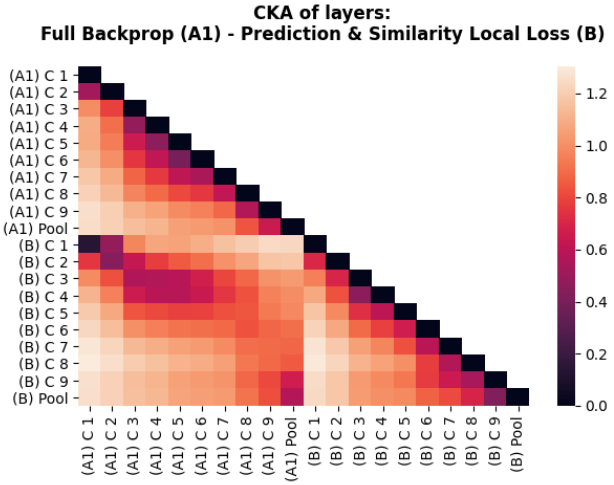


Fig. 5: Representational Dissimilarity Matrix from CKA calculation between Backpropagation (A1) and PredSim (B) loss

further distant layers. This indicates, that the representations in the layers build on one another in both cases, locally trained networks as well as networks trained with global Backpropagation. Thus, this also indicates that the nature of the learned data representations does not differ greatly between the networks. Both training algorithms seem to learn layer wise data representations which build upon each other, hinting towards similar learned structure in the sequence of representations.

When compared to the other network respectively, a strong correlation between the input and output layers can be seen in the RSA and the CKA matrix. This is due to the fact that both networks are trained with the same dataset as input and both reach high classification accuracies. Therefore, the data representations in the input and the output are quite similar. While the RSA matrix shows only a vague slightly shifted correlation trend on the diagonal between the anchor points of the input and output layers, the CKA matrix on the other hand displays a more pronounced diagonal. The difference between the RSA and the CKA matrix could be due to the fact that the in the CKA analysis no normalisation along the axes is performed. Due to this limitation of RSA, the results of the CKA matrix are taken into stronger consideration. The heightened correlation on the diagonal, although slightly shifted, indicates similar learned representation between layers that lie at similar positions in the networks. Suggesting again some similar structure in the sequence of learned representations. The diagonal fades out towards the middle, this is though to be expected since the degrees of freedoms for possible representations accumulate in the middle of the networks. The reason for the offset on the diagonal can only be hypothesized about (maybe similar representations are learned later for one of the networks).

Both the RSA as well as the CKA matrices between Backpropagation and DDTP networks (figure 6 and 7 respectively) show a constantly decreasing correlation on the diagonal between the layers of the networks. The dissimilarity remains highest at the very last layer, where in well performing networks a similar representation between the two final layers would be expected irrespectively of the prior data representation, since both networks have been trained on the same dataset. The low correlation between the DDTP and the Backpropagation trained network is in line with the previously described lower performance of the DDTP trained network compared to the Backpropagation network. This low performance makes it difficult to produce meaningful conclusions from the comparison of these two methods.

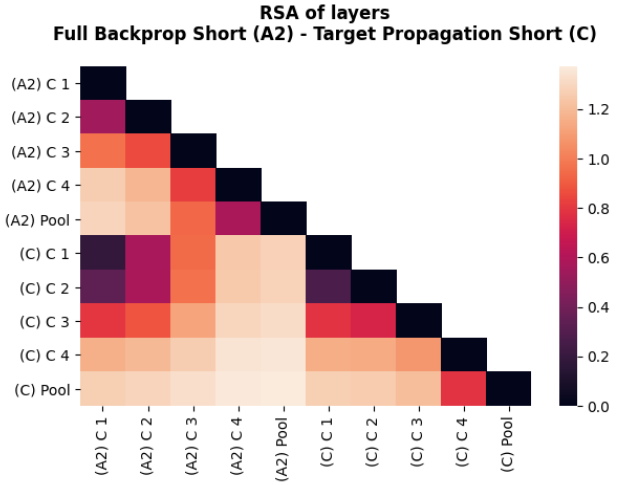


Fig. 6: Representational Dissimilarity Matrix from RSA calculation between Backpropagation (A2) and DDTP (C) loss

## VI. Discussion

The experiments revealed that the structure in the sequence of representations between Backpropagation networks and local loss networks might possess similarities. However, some subtle differences especially for the PredSim network are discernible like a different ”hunchback-shape” for the ID (fig. 3) or an offset of the correlation diagonal in the CKA plot 5.

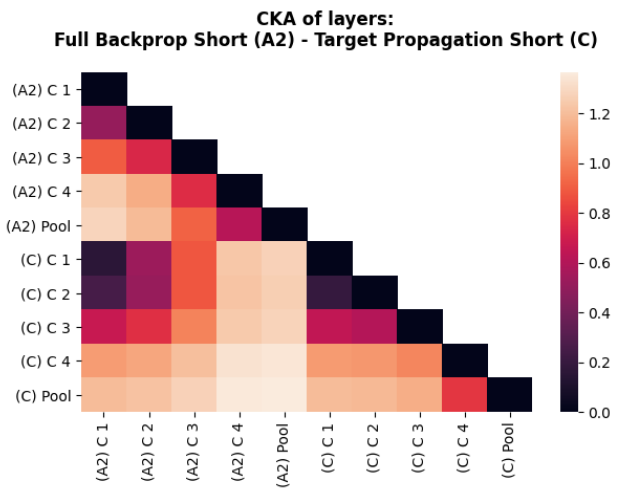


Fig. 7: Representational Dissimilarity Matrix from CKA calculation between Backpropagation (A2) and DDTP (C) loss

These should be investigated further by first of all substantiating the results with more runs that include different network architectures and data sets and then maybe adding metrics that evaluate the representations of the networks at a more fine grained scale, evaluate different geometric properties, etc.

Overall, the experiments are not extensive enough to draw any final conclusions, in particular for the DDTP network. They merely offer arguments for a substantial degree of similarity between Backpropagation and more biological local loss networks. Moreover, the results should be understood as pointers towards possible future work that could enhance our understanding on how comparable biological networks are to Backpropagation networks, and even on how artificial neural networks learn in general.

Furthermore, many other architectures exist that are thought to also be more biologically valid than Backpropagation. These other biological architectures should therefore also be investigated and might reveal a stronger divergence in their learned representations than the local loss networks. Or a common similarity can be deduced that might reveal something about learning representations in general.

### A. *Challenges of the DDTP Network*

The training of an all convolutional DDTP Network proved to be quite difficult and in our view did not reach a sufficient performance to get meaningful insights for either the CKA or ID analysis. The small correlations between the output layers of the Backpropagation and DDTP trained networks in both CKA and RSA support that. In its current implementation, DDTP does not seem to be a well suited method to train extensive convolutional neural networks.

The reasons of why DDTP does not perform well with only convolutional layers remains to be analyzed further. While the original paper [13] did briefly discuss results for convolutional networks, in the original architectures the convolutional layers were always followed by at least two dense layers. Due to that architectural difference, the performances reached in their analysis can not be carried over to the all convolutional architectures we used in our work.

## VII. Conclusion

In this paper, networks trained with global Backpropagation were compared to locally trained networks in terms of representational similarity and intrinsic dimensions of layer activations. This work was inspired by the bigger question of whether the brain could be able to learn representations that are structurally similar to the ones learned by biologically illogical Backpropagation algorithms. We chose to investigate the PredSim and DDTP architectures that are supposedly more biologically valid because of their optimization choices at the local level.

However, the (DDTP) did not perform sufficiently well on the CIFAR-10 dataset to make any meaningful statements about its similarity or difference to Backpropagation. Our implementation attempts suggest that in its current form it might not be a well suited loss architecture for all convolutional neural networks.

For the PredSim architecture, our experiments have shown that there is initial evidence for it to create similarly behaving networks on the level of individual layers, both for the analysis of intrinsic dimensions and the representational similarity loss. Also the networks' overall behaviours are similar, as the local loss architecture did perform on a comparably well level to the same network trained with Backpropagation of error. This suggests that a network that optimizes each layer individually in order to create optimal classification predictions as well as being a good representation of the similarity between targets, could be one way of attaining similar learned representation results as Backpropagation networks without actually backpropagating errors across several layers.

Since these initial findings are only based on one AllCNN-C network architecture and a single data set, further and more extensive experiments are necessary to strength the evidence. Without improvements, it is probably not warranted to further analyze DDTP loss networks; but there are numerous other local error implementations that could be compared to Backpropagation and that could be ranked based on their ability to approximate Backpropagation in future work.

## References

[1] T. P. Lillicrap, A. Santoro, L. Marris, C. J. Akerman, and G. Hinton, "Backpropagation and the brain," Nature Reviews Neuroscience, vol. 21, no. 6, Art. no. 6, Jun. 2020, doi: 10.1038/s41583-020-0277-3.

[2] N. Kriegeskorte, M. Mur, and P. Bandettini, "Representational Similarity Analysis – Connecting the Branches of Systems Neuroscience," Front Syst Neurosci, vol. 2, Nov. 2008, doi: 10.3389/neuro.06.004.2008.

[3] Nili, H., Wingfield, C., Walther, A., Su, L., Marslen-Wilson, W., Kriegeskorte, N. (2014). A toolbox for representational similarity analysis. PLoS computational biology, 10(4), e1003553.

[4] A. Ansuini, A. Laio, J. H. Macke, and D. Zoccolan, "Intrinsic dimension of data representations in deep neural networks," in Advances in neural information processing systems, 2019, vol. 32

[5] V. Erba, M. Gherardi, and P. Rotondo, "Intrinsic dimension estimation for locally undersampled data," Scientific Reports, vol. 9, no. 1, Art. no. 1, Nov. 2019, doi: 10.1038/s41598-019-53549-9.

[6] E. Facco, M. d'Errico, A. Rodriguez, and A. Laio, "Estimating the intrinsic dimension ofdatasets by a minimal neighborhood information,"Scientific reports, vol. 7, no. 1, p. 12140,2017.

[7] Kornblith, S., Norouzi, M., Lee, H., Hinton, G. (2019, May). Similarity of neural network representations revisited. In International Conference on Machine Learning (pp. 3519-3529). PMLR.
[8] L. Ardizzone, J. Kruse, C. Rother, and U. Köthe, “Analyzing Inverse Problems with Invertible Neural Networks,” presented at the International Conference on Learning Representations, Sep. 2018, Accessed: Mar. 05, 2021. [Online]. Available: https://openreview.net/forum?id=rJed6j0cKX.
[9] L. Dinh, J. Sohl-Dickstein, and S. Bengio, “Density estimation using Real NVP,” arXiv:1605.08803 [cs, stat], Feb. 2017, Accessed: Mar. 05, 2021. [Online]. Available: http://arxiv.org/abs/1605.08803.
Cortes, C., Mohri, M., and Rostamizadeh, A. Algorithms for learning kernels based on centered alignment.Journalof Machine Learning Research, 13(Mar):795–828, 2012
[10] Bengio, Y., Lee, D. H., Bornschein, J., Mesnard, T., Lin, Z. (2015). Towards biologically plausible deep learning. arXiv preprint arXiv:1502.04156.
[11] Bengio, Y. (2014). How Auto-Encoders Could Provide Credit Assignment in Deep Networks via Target Propagation. arXiv preprint arXiv:1407.7906
[12] Lee, D., Zhang, S., Fischer, A., Bengio, Y. (2015). Difference Target Propagation. ECML/PKDD.
[13] Meulemans, A., Carzaniga, F.S., Suykens, J., Sacramento, J., Grewe, B.F. (2020). A Theoretical Framework for Target Propagation. arXiv preprint arXiv:2006.14331.
[14] Krizhevsky, A. (2009). Learning Multiple Layers of Features from Tiny Images.
[15] Springenberg, J.T., Dosovitskiy, A., Brox, T., Riedmiller, M.A. (2015). Striving for Simplicity: The All Convolutional Net. CoRR, abs/1412.6806.
[16] Nøkland, Arild, and Lars Hiller Eidnes. ”Training neural networks with local error signals.” International Conference on Machine Learning. PMLR, 2019.
[17] Morcos, Ari S., Maithra Raghu, and Samy Bengio. ”Insights on representational similarity in neural networks with canonical correlation.” arXiv preprint arXiv:1806.05759 (2018).
[18] Wang, Liwei, et al. ”Towards understanding learning representations: To what extent do different neural networks learn the same representation.” arXiv preprint arXiv:1810.11750 (2018).
[19] Guerguiev, Jordan, Timothy P. Lillicrap, and Blake A. Richards. ”Towards deep learning with segregated dendrites.” Elife 6 (2017): e22901.
[20] Mostafa, Hesham, Vishwajith Ramesh, and Gert Cauwenberghs. ”Deep supervised learning using local errors.” Frontiers in neuroscience 12 (2018): 608.
[21] Zhao, Junbo, et al. ”Stacked what-where auto-encoders.” arXiv preprint arXiv:1506.02351 (2015).
[22] Zhang, Yuting, Kibok Lee, and Honglak Lee. ”Augmenting supervised neural networks with unsupervised objectives for large-scale image classification.” International conference on machine learning. PMLR, 2016.
[23] Jaderberg, Max, et al. ”Decoupled neural interfaces using synthetic gradients.” International Conference on Machine Learning. PMLR, 2017.
[24] Dong, Xuanyi, et al. ”Supervision-by-registration: An unsupervised approach to improve the precision of facial landmark detectors.” Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018.

## APPENDIX A
## ARCHITECTURE OF ACNNC NETWORKS

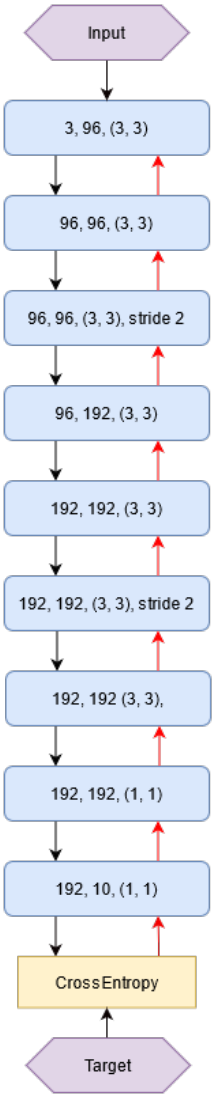


Fig. 8: AllCNNC architecture, used for comparison between Backpropagation and Local Error Signal trained networks

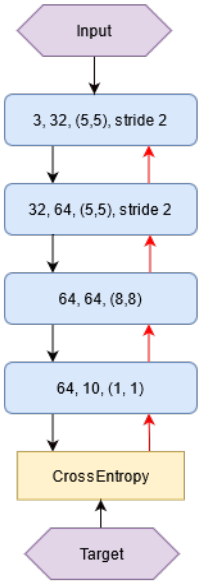


Fig. 9: Shortened AllCNNC architecture, used for comparison between Backpropagation and DDTP trained networks