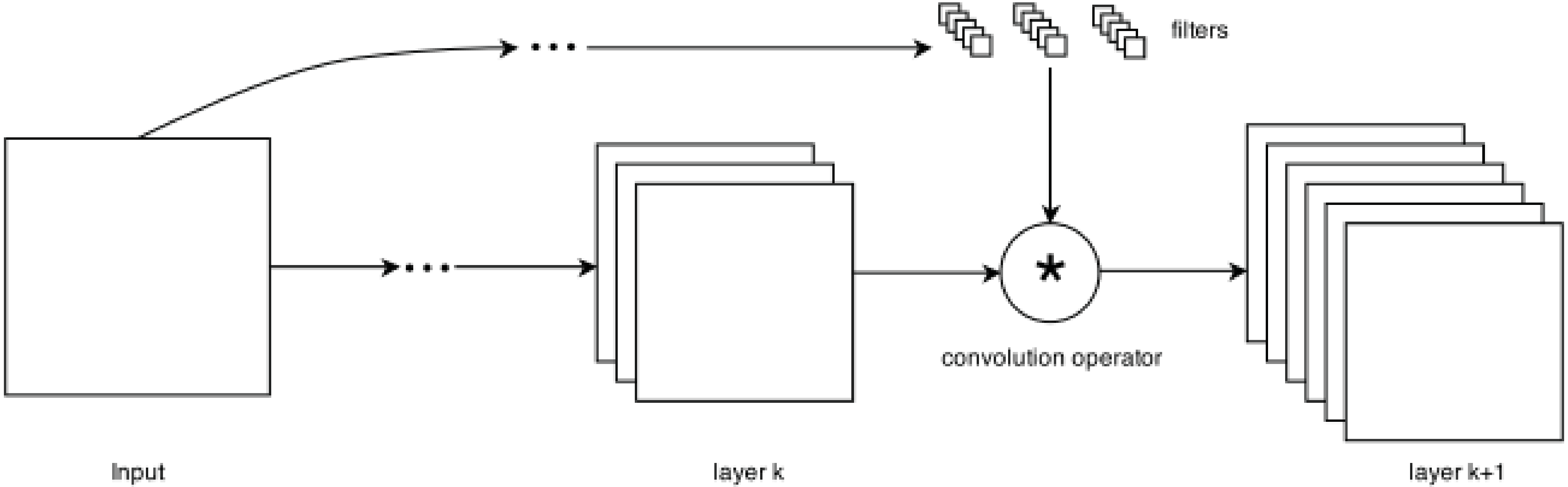
HyperNetworks

Julia Gusak

Skoltech

Klein et al. (2015), Riegler et al. (2015)

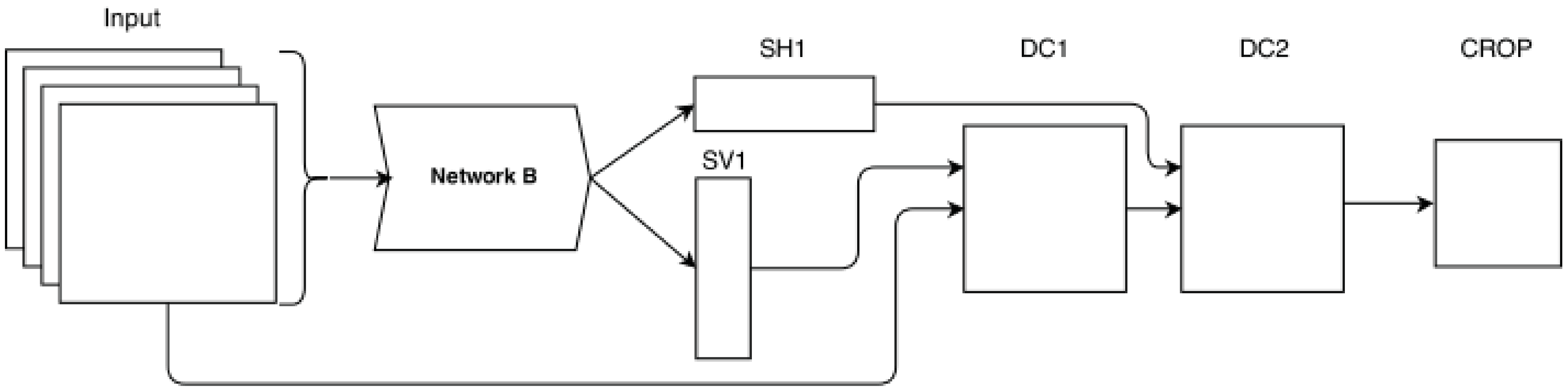
- Klein, B., Wolf, L., & Afek, Y. (2015). A dynamic convolutional layer for short range weather prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 4840-4848).
- Riegler, G., Schulter, S., Ruther, M., and Bischof, H. (2015). Conditioned regression models for non-blind single image super-resolution. In Proceedings of the IEEE International Conference on Computer Vision, pages 522–530
- **Weights vary based on the input, they are determined by a second NN**



Klein et al (2015).

- A new deep network layer called the ”Dynamic Convolutional Layer”, which generalizes the conventional convolutional layer
- Similar to the convolutional layer, the dynamic convolutional layer takes the feature maps from the previous layer and convolves them with filter
- The novelty lies in that the filters of the dynamic convolutional layer are not the param- eters of the layer, rather they are obtained as the output of a subnetwork of arbitrary depth that maps the input to a set of filters

Klein et al. (2015)



- The architecture of the network.
- Network B is a sub-network which computes the filters (H1 and V1) used by the dynamic convolution layers.
- SH1 is the result of applying a softmax function on H1 and SV1 is the result of applying a softmax function on V1.
- DC1 is a dynamic convolution layer that takes the last image in the sequence and convolves it with SV1. DC2 is a dynamic convolution layer that is takes DC1 and convolves it with SH1.

Klein et al. (2015)

- Application: task of short range weather prediction.
- It is shown that by using the new layer, they gain improvement in performance compared to the other baselines, including the conventional CNN.
- Comparison of methods
  - The patch based dynamic CNN provides the lowest error rates.
  - The next best performing method is the patch based conventional CNN
  - The following best performing method is the whole image dynamic CNN.

Method	Tel Aviv Dataset	Davenport Dataset	Kansas City Dataset
Last Frame	20.059±0.536	258.818±2.552	241.392±2.975
Global Motion Estimator	16.837±0.496	173.402±1.547	179.953±2.065
Patch Based Linear Regression	13.002±0.435	164.854±1.377	160.489±1.682
Patch Based CNN	11.480±0.431	105.242±0.839	101.880±1.042
Whole Image Dynamic Convolution Network	12.340±0.461	117.316±0.929	118.402±1.174
Patch Based Dynamic Convolution Network	11.114±0.412	101.983±0.802	98.790±0.995

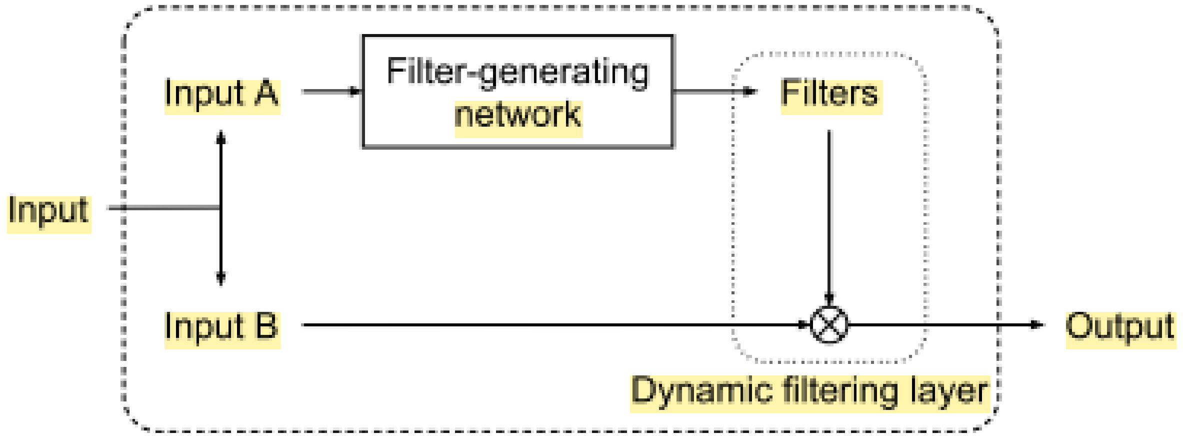
Jia et al. (2016)

- Employ hypernetworks across multiple layers
- For video frame synthesis and stereo prediction

- Jia, X., De Brabandere, B., Tuytelaars, T., & Gool, L. V. (2016). Dynamic filter networks. In *Advances in Neural Information Processing Systems* (pp. 667-675).

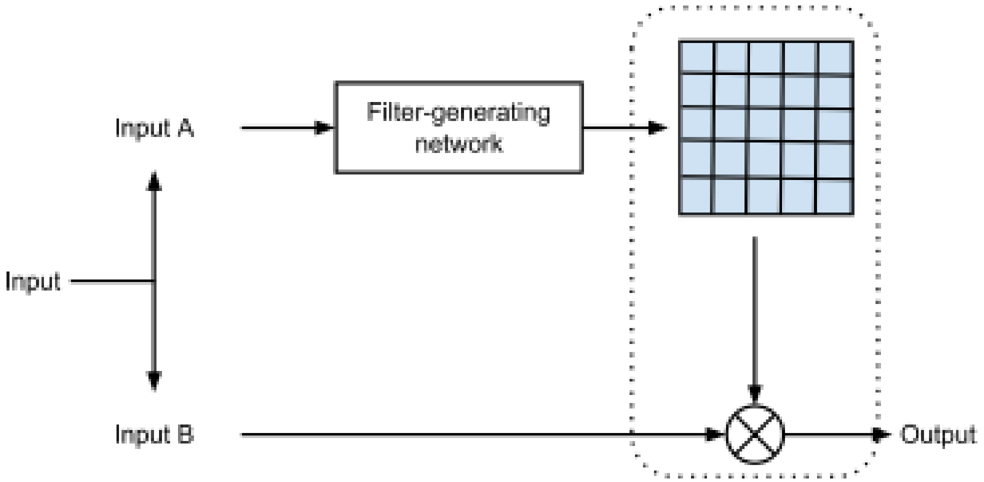
Jia et al. (2016)

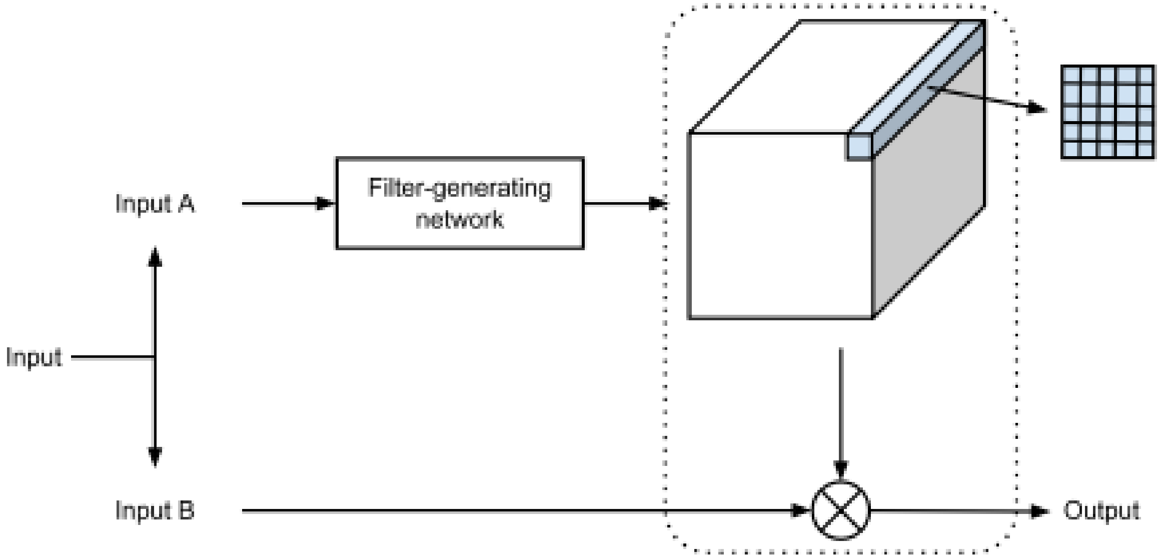
- Poposed **dynamic filter module** consists of two parts: a **filter-generating network** and a **dynamic filtering layer**
- The filter-generating network dynamically generates sample-specific filter parameters conditioned on the network's input
- The dynamic filtering layer then applies those sample-specific filters to the input
- The filters can be convolutional, but other options are possible.
- In particular, they propose a special kind of dynamic filtering layer, **dynamic local filtering layer,** which is not only sample-specific but also position-specific
- The work **differs from Klein et al (2015), Riegler et al (2015) in that it is more genera**l: dynamic filter networks are not limited to translation-invariant convolutions, but also allow position-specific filtering using a dynamic locally connected layer



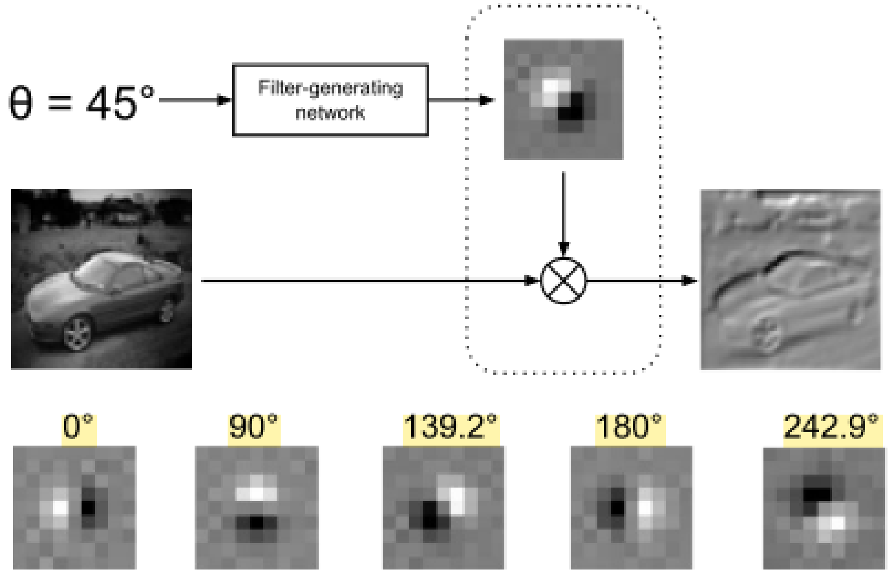
Jia et al. (2015)

- Left: Dynamic convolution: the filter-generating network produces a single filter that is applied convolutionally on IB
- Right: Dynamic local filtering: each location is filtered with a location-specific dynamically generated filte



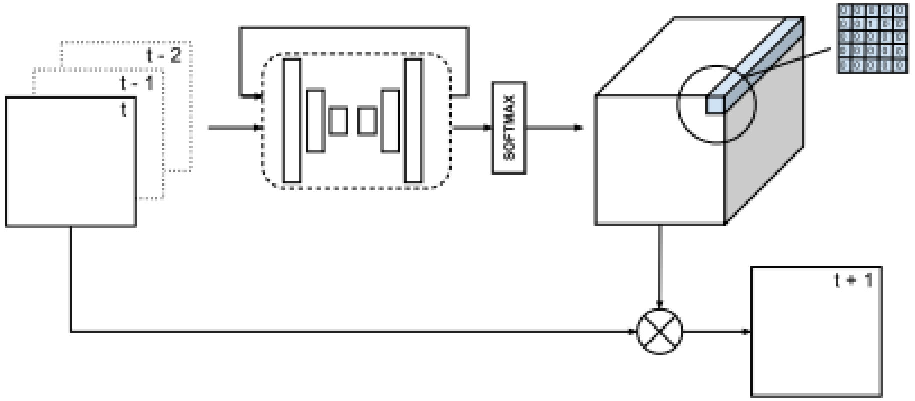


Jia et al (2016)



- Learning steerable filters
  - A simple use case of a dynamic filter network which uses a dynamic convolutional layer with two different types of inputs
  - The task is to filter an input image with a steerable filter of a given orientation θ. The task of the filter-generating network here is to transform an angle into a filter, which is then applied to the input image to generate the final output.

Jia et al (2016)



- Video prediction
  - Shows that we can integrate the dynamic filter module with a dynamic local filtering layer in a recurrent network to predict a sequence of frames
  - Given a sequence of frames, the task is to predict the sequence of future frames that directly follow the input frames.
  - The convolutional encoder-decoder as the filter-generating network.
  - A softmax layer is applied to each generated filter such that each filter is encouraged to have only a few non-zero elements

Jia et al (2016)

- Stereo prediction
  - Shows its use case when there is only one kind of input
  - Predicting the right view given the left view of two horizontal-disparity cameras
  - This task is a variant of video prediction, where the goal is to predict a new view in space rather than in time, and from a single image rather than multiple ones

Ha et al. (2016)

- RNNs, in which the weights are determined by another RNN
- The weight generating RNN receives both previous hidden state and the next token as its input
- Two networks are disjointed, their input vary over time

- Ha, D., Dai, A., and Le, Q. V. (2016). Hypernetworks. arXiv preprint arXiv:1609.09106

Krueger et al. (2017)

- Bayesian formulation, i.e. variational inference that involves a parameter generating network and a primary network

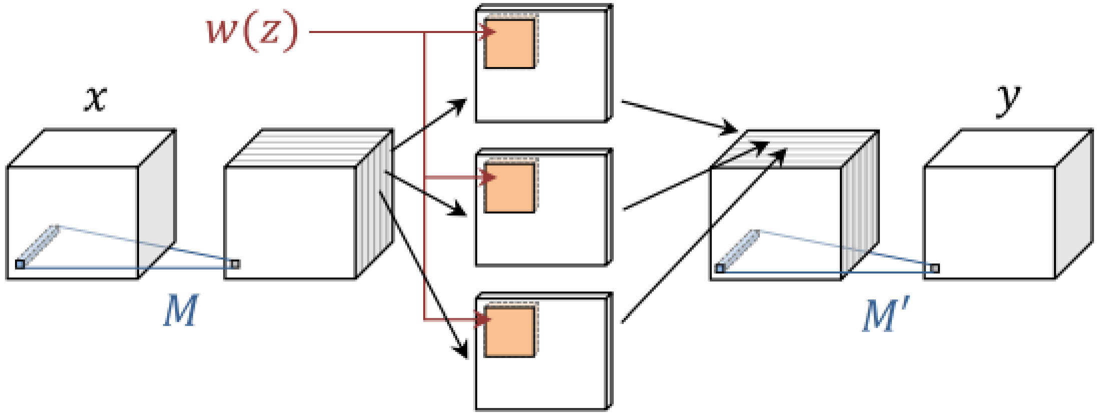
- Krueger, D., Huang, C.-W., Islam, R., Turner, R., Lacoste, A., and Courville, A. (2017). Bayesian hypernetworks. arXiv preprint arXiv:1710.04759.

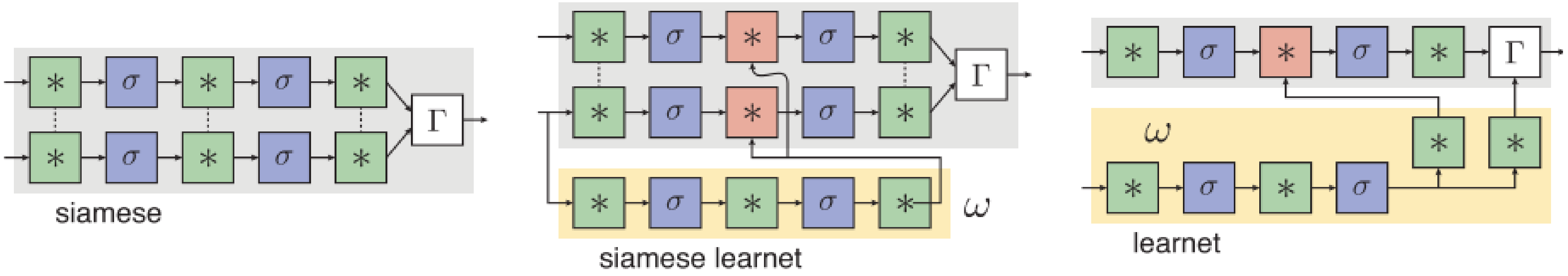
Bertinetto et al. (2016)

- Hypernetworks for few-shot learning tasks
- Weight generating network is used to adapt to the current task and the ability to share knowledge different tasks

- Bertinetto, L., Henriques, J. F., Valmadre, J., Torr, P., and Vedaldi, A. (2016). Learning feed-forward one-shot learners. In Advances in Neural Information Processing Systems, pages 523–531

Bertinetto et al. (2016)





Bertinetto et al. (2016)

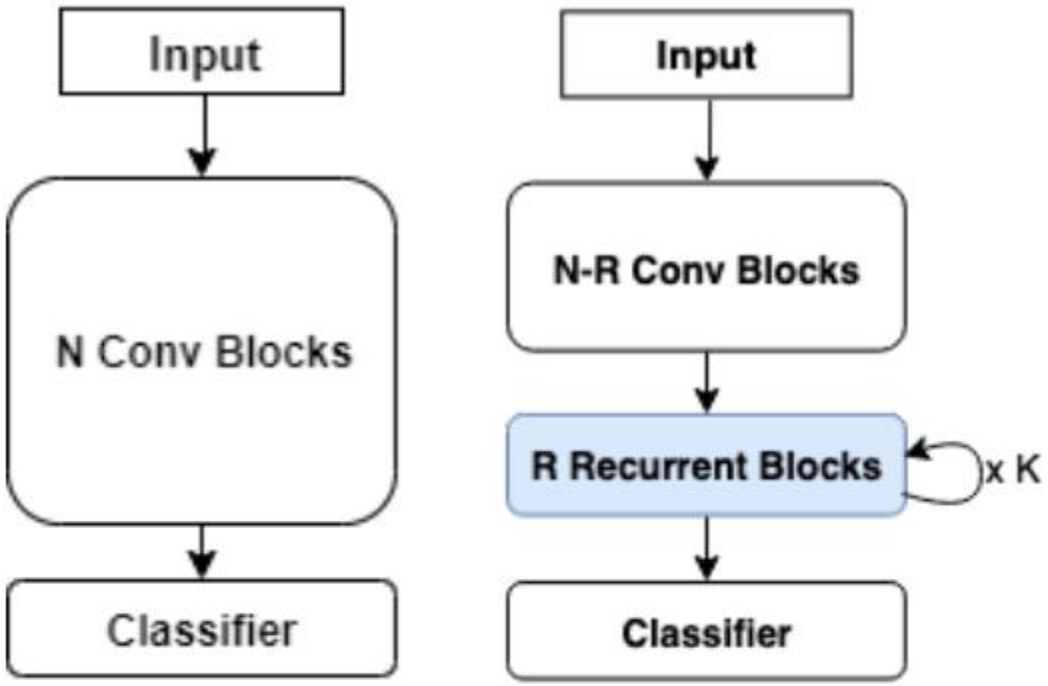
	Inner-product (%)	Euclidean dist. (%)	Weighted ℓ^1 dist. (%)
Siamese (shared)	48.5	37.3	41.8
Siamese (unshared)	47.0	41.0	34.6
Siamese (unshared, factorized)	48.4	–	33.6
Siamese learnet (shared)	51.0	39.8	31.4
Learnet	43.7	36.7	**28.6**

Table 1: Error rate for character recognition in foreign alphabets (chance is 95%).

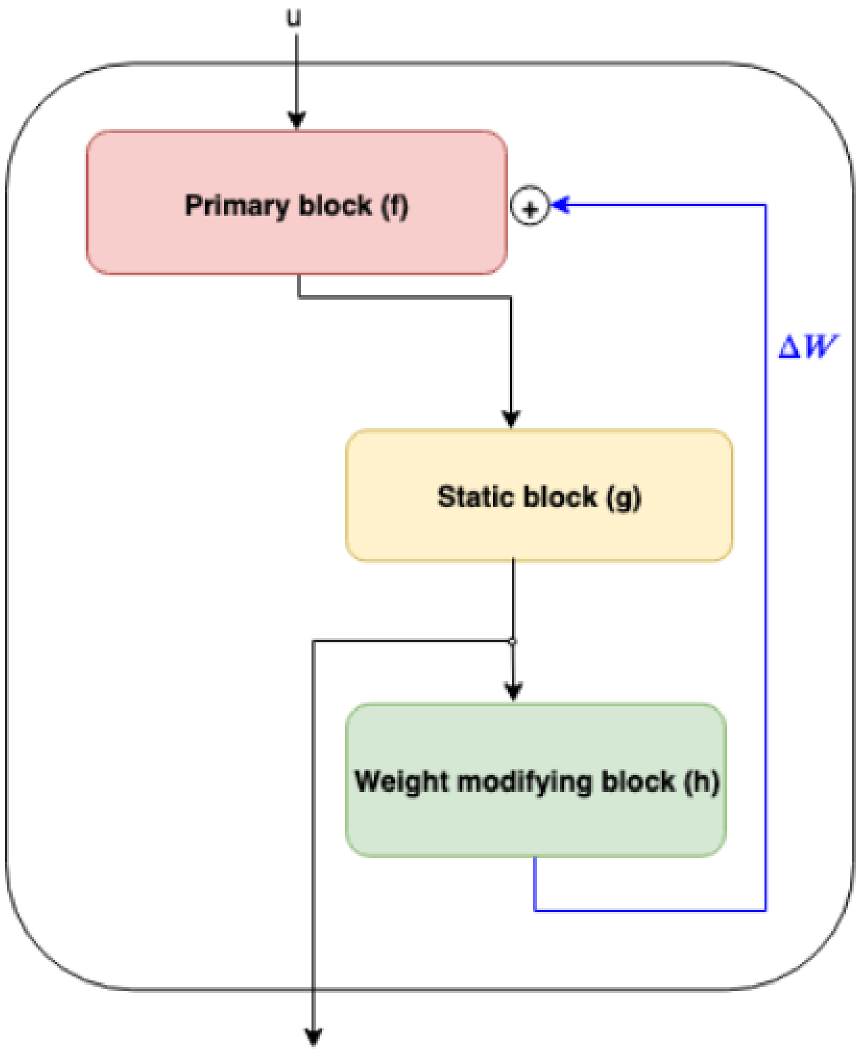
Brock et al. (2018), Zhang et al. (2019)

- The ability of hypernetworks to relpace backpropagation-based training by prediction of weights was exploited
- For performing architecture search

(Battash et al, 2019) Adaptive and Iteratively Improving Recurrent Lateral Connections



(a) (b)



(Battash et al, 2019) Adaptive and Iteratively Improving Recurrent Lateral Connections

Table 1: Results on the MNIST dataset. No recurrent iterations (-) implies one iteration through each block.

Method	Recurrent iterations	Top-1 accuracy	Number of parameters
Baseline (phase one of our method)	-	96.50	900
Baseline-big	-	98.07	5687
Our recurrent connections, no kaizen loss	2	98.03	5691
Our recurrent connections, no kaizen loss	3	98.15	5691
Our recurrent connections, no kaizen loss	4	98.16	5691
Our full method	2	98.16	5691
Our full method	3	98.32	5691
Our full method	4	98.43	5691

Tensor methods meet Deep learning

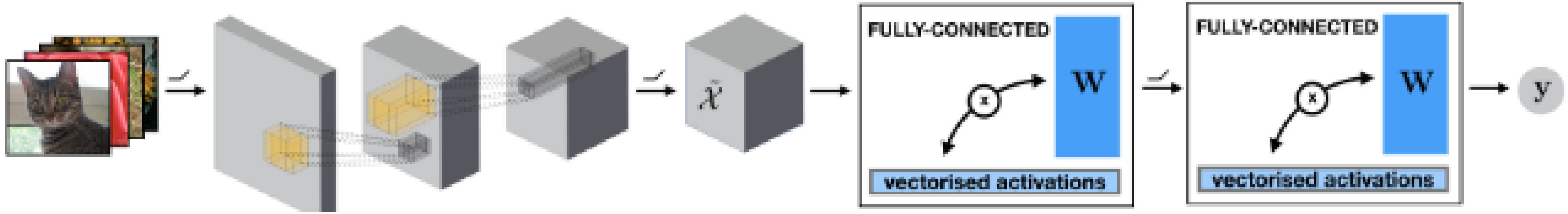
Julia Gusak, Skoltech

Tensor contraction + Tensor regression layer

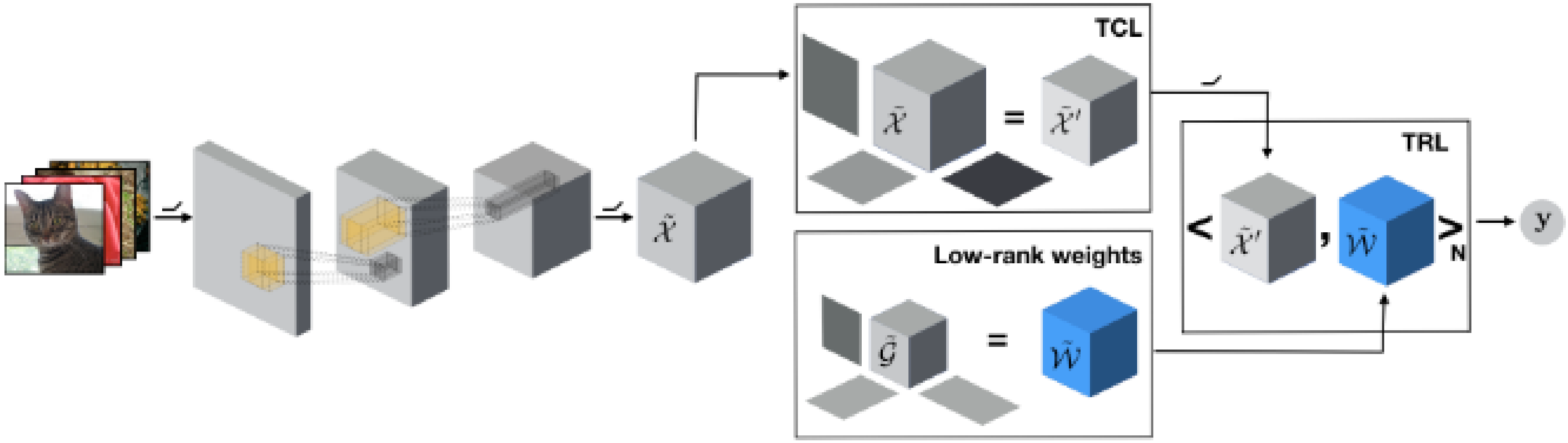
- J. Kossaifi, A. Khanna, Z. Lipton, T. Furlanello, and A. Anandkumar Tensor contraction layers for parsimonious deep nets, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, 2017, pp. 26–32.
- J. Kossaifi, Z. C. Lipton, A. Khanna, T. Furlanello, and A. Anandkumar, Tensor contraction & regression networks, arXiv preprint arXiv:1707.08308, (2017)
- A. Kolbeinsson, J. Kossaifi, Y. Panagakis, A. Bulat, A. Anandkumar, I. Tzoulaki, and P. Matthews, Robust deep networks with randomized tensor regression layers, arXiv, (2019)

Tensor contraction + Tensor regression layer

- Fully connected layer



- Tensor contraction + TRL



Tensor contraction + Tensor regression layer

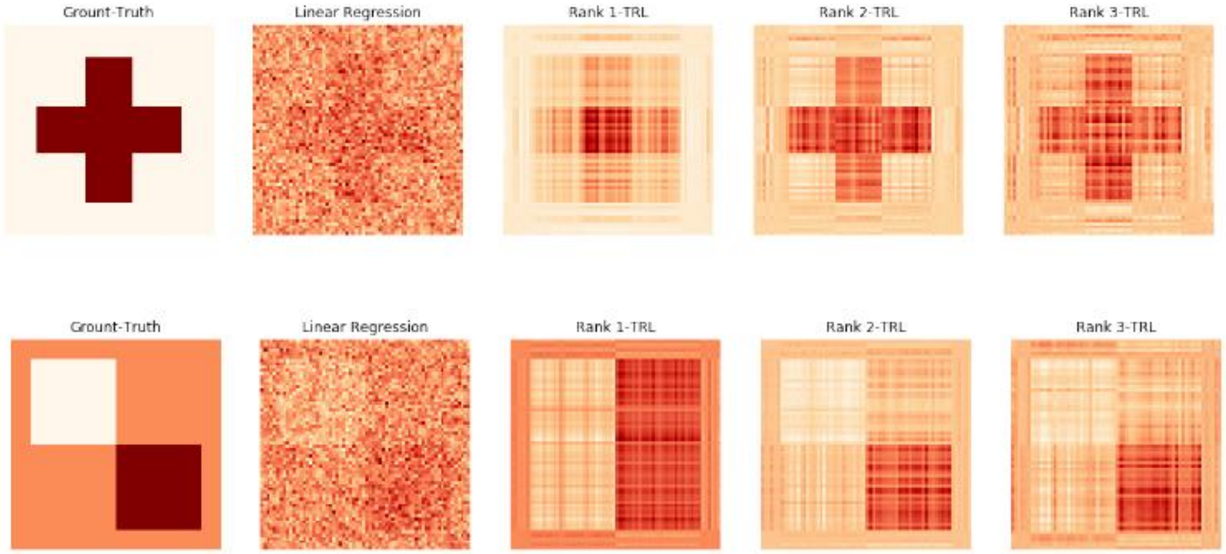


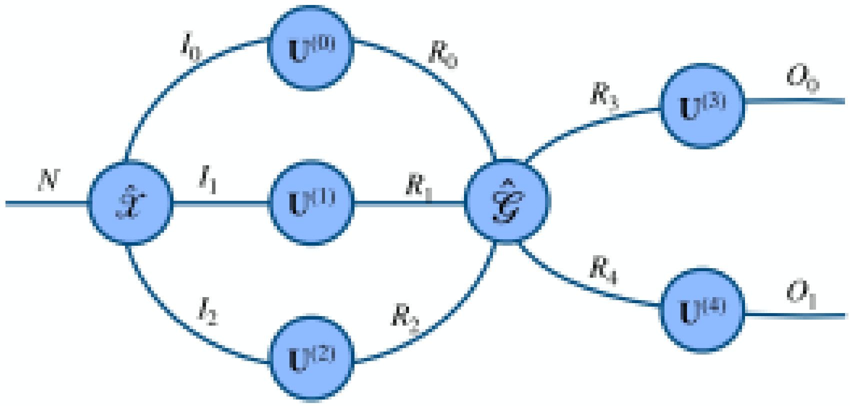
Figure 4: Empirical comparison (4) of the TRL against linear regression with a fully-connected layer. We plot the weight matrix of a TRL and a fully-connected layer. Due to its low-rank weights, the TRL better captures the structure in the weights and is more robust to noise.

Tensor contraction + Tensor regression layer

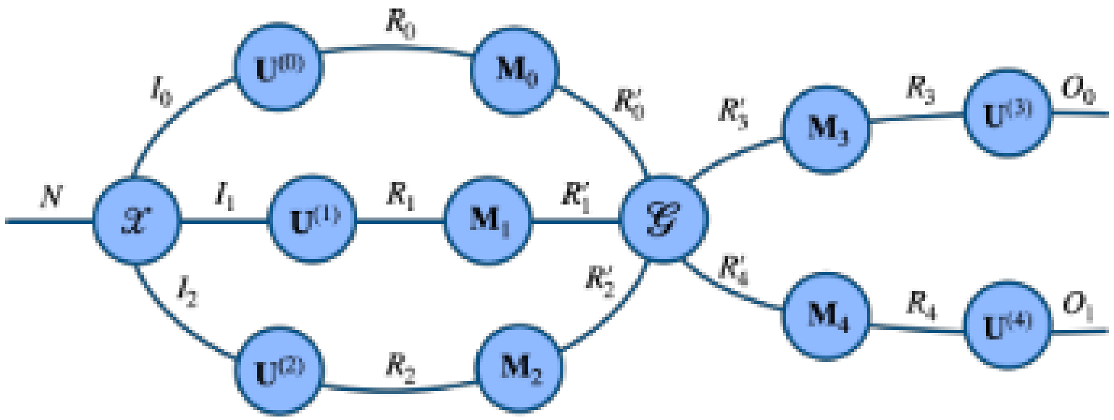
Table 3: Results obtained on ImageNet by adding a TCL to a VGG-19 architecture. We reduce the number of hidden units proportionally to the reduction in size of the activation tensor following the tensor contraction. Doing so allows more than 65% space savings over all three fully-connected layers (i.e. 99.8% space saving over the fully-connected layer replaced by the TCL) with no corresponding decrease in performance (comparing to the standard VGG network as a baseline).

Method		Accuracy		Space Savings
TCL–size	Hidden Units	Top-1 (%)	Top-5 (%)	(%)
baseline	4096	68.7	88	0
(512, 7, 7)	4096	**69.4**	**88.3**	-0.21
(384, 5, 5)	3072	68.3	87.8	**65.87**

Robust NN with randomized TRL



(a) Tensor diagram of a TRL



(b) Tensor diagram of a SRR-TRL

T-Net: Parametrizing Fully Convolutional Nets with a Single High-Order Tensor

- Kossaifi, J., Bulat, A., Tzimiropoulos, G., & Pantic, M. (2019). T-Net: Parametrizing Fully Convolutional Nets with a Single High-Order Tensor. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 7822-7831).

T-Net: Parametrizing Fully Convolutional Nets with a Single High-Order Tensor

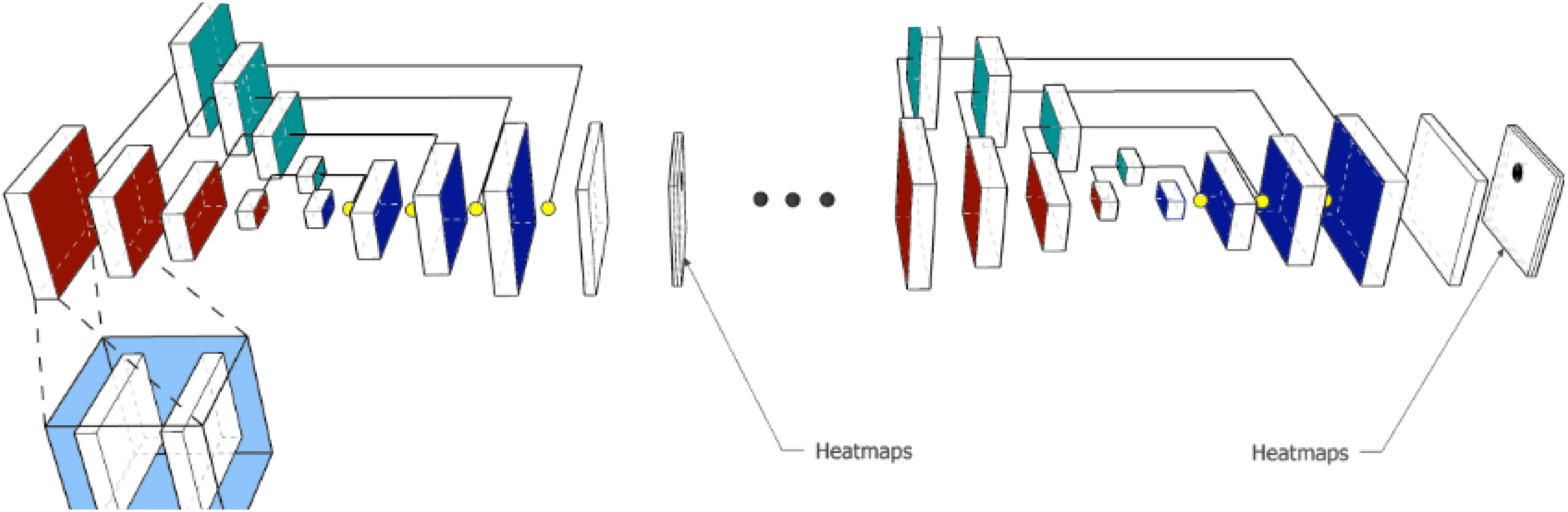
- Convolutional neural network (CNNs) is fully parameterized with a single high-order, low-rank tensor
- The modes of such tensor represent each of the architecture design parameters of the network (e.g. number of conv blocks, depth, number of statcks, imput features, etc.)
- The model is end-to-end trainable (low-rank structure acts as implicit regularization)

T-Net: Parametrizing Fully Convolutional Nets with a Single High-Order Tensor

- Proposed approach allows for learning correlations between the different tensor dimensions and hence to fully capture the structure of the network.
- Considered application:
  - human-pose estimation (single pose datasets, MPII; accuracy in terms of PCKh)
  - Facial part segmentation (accuracy using the mean accuracy and mIOU metrics)
- Achives higher accuracy, especially for high compression rate

T-Net: Parametrizing Fully Convolutional Nets with a Single High-Order Tensor

- Each block in the fully convolutional network is a basic-block module (blue insert), containing b_depth (by default 2) convolutional layers with 3 × 3 kernels followed by BatchNorm and ReLU.
- For all experiments, stack of 4 sub-networks is used, with 3 pathways each: downsampling/encoder (red blocks), upsampling/decoder (dark blue) and skip connection (cyan). Yellow dots are element-wise sums.



T-Net: Parametrizing Fully Convolutional Nets with a Single High-Order Tensor

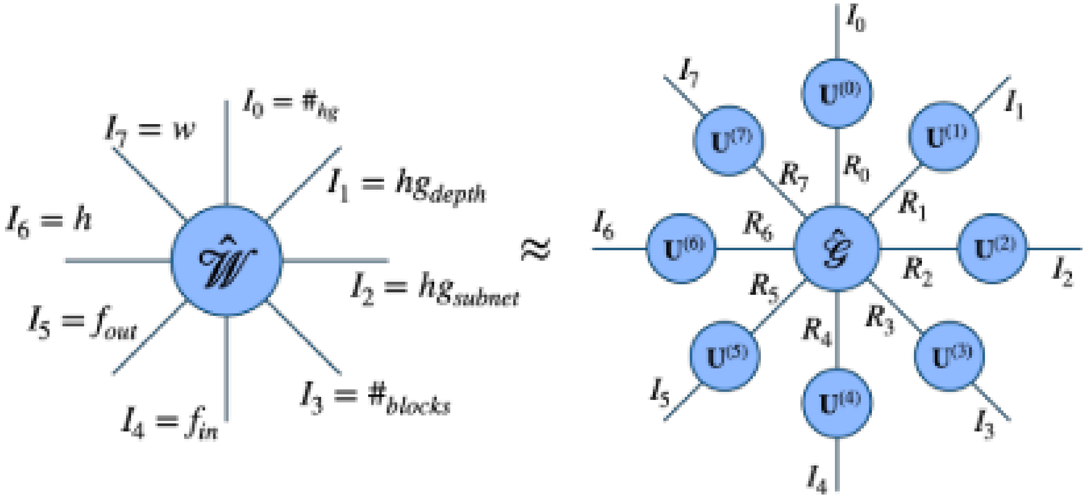


Figure 2: **Tensor diagram** of the Tucker form of the weight tensor $\mathcal{W}$ parametrizing our model.

- all weights of the network are parametrized by a single 8th–order tensor W of shape I0×I1×···×I7, the modes of which correspond to the
- number of HGs (I0 = #hg),
- the depth of each HG (I1 = hg_depth),
- the three signal pathways (I2 =hg_subnet),
- the number of convolutional layers per block (I3 = b_depth),
- the number of input features (I4 = fin),
- the number of output features (I5 = fout),
- the height (I6 = h) and width (I7 = w) of each of convolutional kernels.

T-Net: Parametrizing Fully Convolutional Nets with a Single High-Order Tensor

- T-Net

$$\begin{aligned}\mathcal{W} &= \mathcal{G} \times_0 \mathbf{U}^{(0)} \times_1 \mathbf{U}^{(2)} \times \cdots \times_7 \mathbf{U}^{(7)} \\ &= [\![\mathcal{G}; \mathbf{U}^{(0)}, \cdots, \mathbf{U}^{(7)}]\!]\end{aligned}$$

- MPS T-Net

$$\mathcal{W}(i_0, i_1, \cdots, i_7) = \underbrace{\mathcal{G}_0[i_0]}_{1\times R_1} \times \underbrace{\mathcal{G}_1[i_1]}_{R_1\times R_2} \times \cdots \times \underbrace{\mathcal{G}_7[i_7]}_{R_7\times 1}$$

T-Net: Parametrizing Fully Convolutional Nets with a Single High-Order Tensor

Method	Parameters	Compression ratio	Accuracy
Uncompressed Baseline	full, f_{in}=f_{out}=128	1x	87%
Trimmed Baseline	f_{in}=f_{out}=112	1.3x	86.9%
Trimmed Baseline	f_{in}=f_{out}=92	2x	85.9%
Trimmed Baseline	f_{in}=f_{out}=64	4x	84.5%
Trimmed Baseline	hg_depth=3	1.3x	86.79%
Trimmed Baseline	hg_depth=2	1.8x	86.82%
Trimmed Baseline	hg_depth=1	3.0x	85.30%
MobileNet-[16]	f_{in}=f_{out}=194	3.6x	84.3%
MobileNet-[16]	f_{in}=f_{out}=160	5.4x	82.7%
[17]	rank–(128, 128, 2, 2)	1.4x	84.9%
[17]	rank–(96, 96, 3, 3)	1.3x	86.8%
[17]	rank–(64, 64, 3, 3)	2.3x	86.4%
[17]	rank–(32, 32, 3, 3)	4.7x	85.3%
[17]	rank–(16, 16, 3, 3)	6.9x	83.7%
Tucker T-Net [Ours]	rank–(4, 3, 3, 2, 110, 110, 3, 3)	1.7x	**87.5%**
Tucker T-Net [Ours]	rank–(4, 4, 2, 2, 110, 110, 3, 3)	1.8x	**87.4%**
Tucker T-Net [Ours]	rank–(3, 3, 3, 2, 110, 110, 2, 2)	3.7x	**87.1%**
Tucker T-Net [Ours]	rank–(3, 2, 3, 2, 96, 96, 3, 3)	3.4x	**86.7%**
Tucker T-Net [Ours]	rank–(3, 3, 2, 2, 80, 80, 3, 3)	4.2x	**86.3%**
Tucker T-Net [Ours]	rank–(2, 2, 2, 2, 96, 96, 3, 3)	5.2x	**86.0%**
MPS T-Net [Ours]	rank–(1, 4, 4, 12, 24, 110, 9, 3, 1)	7.4x	**85.5%**

Table 3: **Human pose estimation task**. Comparison between T-Net and various baselines and state-of-the-art methods. Accuracy is reported in terms of PCKh. For the tensor decomposition-based methods, we report the rank, and for the others, the number of channels in the convolutional layers.

T-Net: Parametrizing Fully Convolutional Nets with a Single High-Order Tensor

Tucker-rank								Accuracy	Compression
#hg	hg_{depth}	hg_{subnet}	b_{depth}	f_{in}	f_{out}	h	w	(PCKh)	ratio
Original								86.99%	1.0x
3	4	3	2	128	128	3	3	87.42%	1.28x
2	4	3	2	128	128	3	3	86.95%	1.82x
1	4	3	2	128	128	3	3	86.05%	3.03x
4	**3**	3	2	128	128	3	3	87.71%	1.28x
4	**2**	3	2	128	128	3	3	87.59%	1.82x
4	**1**	3	2	128	128	3	3	86.89%	3.03x
4	4	**2**	2	128	128	3	3	87.53%	1.43x
4	4	**1**	2	128	128	3	3	86.19%	2.50x
4	4	3	**1**	128	128	3	3	82.59%	1.82x
4	4	3	2	**96**	**96**	3	3	87.43%	1.64x
4	4	3	2	**64**	**64**	3	3	86.13%	3.03x
4	4	3	2	**32**	**32**	3	3	83.10%	6.25x
4	4	3	2	128	128	**2**	**2**	87.30%	1.98x

Table 2: **Human pose estimation task.** Study of the redundancy of each of the modes of the 8^{th}–order weight tensor. We compress one dimension at a time by reducing its corresponding rank in the Tucker tensor. Reported accuracy is in terms of PCKh.

XnorNet

- Rastegari, M., Ordonez, V., Redmon, J., & Farhadi, A. (2016, October). Xnor-net: Imagenet classification using binary convolutional neural networks. In *European Conference on Computer Vision* (pp. 525-542). Springer, Cham.

XnorNet

- Binary weights

$$\mathbf{I} * \mathbf{W} \approx (\mathbf{I} \oplus \mathbf{B})\,\alpha$$

- Binary weights and binary activations

$$\mathbf{I} * \mathbf{W} \approx (\operatorname{sign}(\mathbf{I}) \circledast \operatorname{sign}(\mathbf{W})) \odot \mathbf{K}\alpha$$

Matrix and tensor decompositions for training binary neural networks

- Bulat, A., Kossaifi, J., Tzimiropoulos, G., & Pantic, M. (2019). Matrix and tensor decompositions for training binary neural networks. *arXiv preprint arXiv:1904.07852*.

Matrix and tensor decompositions for training binary neural networks

- The paper is on improving the training of binary neural networks in which both activations and weights are binary.
- The weight tensor of each layer is parametrized using matrix or tensor decomposition.
- The binarization process is then per-formed using this latent parametrization, via a quantization function (e.g. sign function) applied to the reconstructed weights
- Note: While the reconstruction is binarized, the computation in the latent factorized space is done in the real domain.
- Applications: human-pose estimation (MPII), large-scale image classification (ImageNet)

Matrix and tensor decompositions for training binary neural networks

- A common limitation in prior work is that each filter **Wi** of shape C×w×h **(a slice of W)** of a given convolutional layer is binarized independently as follows:
  - **Bi = sign(Wi)**
- A key idea in the proposed work is to model the filters jointly by reparametrizing them in a shared subspace using a matrix or tensor decomposition, and then binarizing the weights:
  - **W = UV, Bi = sign(Wi)**
  - This allows us to introduce an **inter-dependency between the to-be-binarized weights through the shared factor U** either at a layer level or even more globally at a network level.
  - **Decomposition factors (i.e U,V) are kept real during training.** This allows to introduce additional redundancy which facilitates learning.
  - **During inference, the method uses only the reconstructed weights**, which have been binarized using the sign function (the decomposition factors are neither used nor stored)

Matrix and tensor decompositions for training binary neural networks

- Explored decompositions: SVD and Tucker
- Ways two apply decompositions: **layer-wise** and **holistically**
- **Layer-wise**: a weigth tensor for each layer is modeled separetely (i.e. different decompositions for each layer)
  - SVD

$$\mathrm{sign}(\mathbf{W}) = \mathrm{sign}(\mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^{\mathrm{T}}) \qquad \mathbf{I} * \mathbf{W} = (\mathrm{sign}(\mathbf{I}) \circledast \mathrm{sign}(\mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^{\mathrm{T}})) \odot \alpha.$$

  - Tucker

$$\mathrm{sign}(\mathcal{W}) = \mathrm{sign}(\mathcal{G} \times_0 \mathbf{U}^{(0)} \times_1 \mathbf{U}^{(1)} \times_2 \mathbf{U}^{(2)} \times_3 \mathbf{U}^{(3)})$$

- **Holistically**: whole network is tensorized with one tensor
  - They propose to group together identically shaped weights inside the network in a higher-order tensor in order to exploit the inter relation between them holistically
  - For example, they use **three 5-th order tensors for ResNet-18**, the individual weights of a given layer k can be obtained from **W = W'(l, :, :, :, :)**, where

$$\mathrm{sign}(\mathcal{W}') = \mathrm{sign}(\mathcal{G}' \times_0 \mathbf{U}^{(0)} \times_1 \mathbf{U}^{(1)} \times \cdots \times_4 \mathbf{U}^{(4)}).$$

Matrix and tensor decompositions for training binary neural networks

- Pose estimation (MPII)

Method	#parameters	PCKh
HBC [4]	6.2M	78.1%
Ours	6.0M	**82.5%**
Real valued	6.0M	85.8%

Decomposition	Holistic	Learn. alpha	PCKh
None	-	✗	78.4%
None	-	✓	79.3%
SVD	✗	✗	78.7%
SVD	✗	✓	79.0%
Tucker	✗	✗	79.3%
Tucker	✗	✓	79.9%
Tucker	✓	✗	82.0%
Tucker	✓	✓	**82.5%**

- Large-scale image classification (ImageNet)

Method	Top-1 accuracy	Top-5 accuracy
BNN [8]	42.2%	69.2%
XNOR-Net [31]	51.2%	73.2%
Ours	**55.6%**	**78.5%**
Real valued [12]	69.3%	89.2%

Decomposition	Holistic	Learn. alpha	Top-1	Top-5
None	-	✗	52.3%	74.1%
None	-	✓	53.0%	74.7%
SVD	✗	✓	52.5%	74.2%
Tucker	✗	✗	54.0%	76.9%
Tucker	✗	✓	54.7%	77.4%
Tucker	✓	✗	55.2%	78.2%
Tucker	✓	✓	**55.6%**	**78.5%**

Matrix and tensor decompositions for training binary neural networks

- One of the key ingredients of the **recent success of binarized neural network** was the introduction of the **α** weight scaling factor, computed analytically as the average of absolute weight values
  - This estimation generally performs well, but it **attempts to minimize the difference between the real weights and the binary ones W ≈ α sign(W)** and does not explicitly decrease the overall network loss
- **This work proposes to learn the scaling factor by minimizing its value with respect to the networks cost function**, learning it discriminatively via back-propagation.
  - a more spread out distribution that can take both positive and negative values
  - has significantly higher magnitude, thus leading to a faster and more stable training.

Matrix and tensor decompositions for training binary neural networks

Comments and further directions

- Can these techique be improved by learning binary decompositions directly through back-prop?
- Can intoduction of smothing improve smth?
- Is there any sence to consider other decompositions?
- How to handle grouped-wise convolutions?

XnorNet++

- A. Bulat and G. Tzimiropoulos, Xnor-net++: Improved binary neural networks, arXiv preprint arXiv:1909.13863, (2019).

XnorNet++

$$\mathcal{I} * \mathcal{W} \approx (\mathrm{sign}(\mathcal{I}) \circledast \mathrm{sign}(\mathcal{W})) \odot \Gamma$$

Case 1:

$$\Gamma = \alpha, \quad \alpha \in \mathbb{R}^{o \times 1 \times 1}$$

Case 2:

$$\Gamma = \alpha, \quad \alpha \in \mathbb{R}^{o \times h_{out} \times w_{out}}$$

Case 3:

$$\Gamma = \alpha \otimes \beta, \quad \alpha \in \mathbb{R}^{o}, \beta \in \mathbb{R}^{h_{out} \times w_{out}}$$

Case 4:

$$\Gamma = \alpha \otimes \beta \otimes \gamma, \quad \alpha \in \mathbb{R}^{o}, \beta \in \mathbb{R}^{h_{out}}, \gamma \in \mathbb{R}^{w_{out}}$$

Incremental multi-domain learning with network latent tensor factorization

- A. Bulat, J. Kossaifi, G. Tzimiropoulos, and M. Pantic, Incremental multi-domain learning with network latent tensor factorization, arXiv preprint arXiv:1904.06345, (2019).

Incremental multi-domain learning with network latent tensor factorization