# TensorNet on TensorFlow

Timur Garipov

April 15, 2016

Section 1

# Introduction

# TensorNet

- TensorNet approach was introduced by A. Novikov et al. and the paper [1] was published on the NIPS-2015.
- The core idea of the TensorNet is to use Tensor-Train (TT) decomposition for matrices in fully-connected layers of neural networks.
- TT decomposition [2, I. Oseledets (2011)] is the way to store multidimensional arrays compactly which doesn't suffer from the "curse of dimensionality" and allows to efficiently implement basic operations.

# Matrices in the TT-format

Consider a matrix $\boldsymbol{W} \in \mathbb{R}^{M \times N}$ reshaped in a $2 \times d$ dimensional tensor:

$$\boldsymbol{W}(i,j) = \boldsymbol{W}(i_1, i_2, \ldots, i_d;\ j_1, j_2, \ldots, j_d),$$

where:

$$i \in \{1, \ldots, M\}, \quad j \in \{1, \ldots, N\}$$
$$i_k \in \{1, \ldots, m_k\}, \quad j_k \in \{1, \ldots, n_k\}$$
$$M = \prod_{k=1}^{d} m_k, \quad N = \prod_{k=1}^{d} n_k$$

For example a $192 \times 120$ can be reshaped in 6-dimensional tensor with shape $(8 \times 8 \times 3) \times (4 \times 5 \times 6)$. In this case:

$$W(4,5) = W(1,2,1;1,1,5)$$

# Matrices in the TT-format

The representation of the matrix $\boldsymbol{W}$ in TT-format is following:

$$\boldsymbol{W}(i_1, i_2, \ldots, i_d;\ j_1, j_2, \ldots, j_d) = \boldsymbol{G}_1[i_1, j_1]\boldsymbol{G}_2[i_2, j_2]\ \ldots\ \boldsymbol{G}_d[i_d, j_d],$$

where $\boldsymbol{G}_k[i_k, j_k] \in \mathbb{R}^{r_k \times r_{k+1}}$. $\boldsymbol{G}_k$ is a collection of $m_k n_k$ matrices with size $r_k \times r_{k+1}$. Note that $r_1 = r_{d+1} = 1$.

We will use the following terminology:

- $\{m_k\}_{k=1}^d, \{n_k\}_{k=1}^d$ — TT-modes;
- $\{\boldsymbol{G}_k\}_{k=1}^d$ — TT-cores;
- $\{r_k\}_{k=1}^d$ — TT-ranks.

TT-decomposition exists for any matrix $\boldsymbol{W}$ and uses $\mathcal{O}(dmnr^2)$ memory to store $\mathcal{O}(m^d n^d)$ elements. Efficient only with small ranks.

# TT-layer

- One of the most commonly used layers in the neural nets are the fully-connected (FC) layers.
- It perform a linear transformation of an input vector $x \in \mathbb{R}^N$ to the output vector $y \in \mathbb{R}^M$:

$$y = \boldsymbol{W} x + b,$$

where $\boldsymbol{W} \in \mathbb{R}^{M \times N}$, $b \in \mathbb{R}^M$.

# TT-layer

Now let's reshape $x$, $y$ and $b$ to $d$-dimensional tensors:

$$y(i) = y(i_1, i_2, \ldots, i_d), \quad x(j) = x(j_1, j_2, \ldots, j_d), \quad b(i) = b(i_1, i_2, \ldots, i_d);$$

and represent the $\boldsymbol{W}$ matrix in the TT-format. So we have the TT-layer:

$$y(i_1, \ldots, i_d) = \sum_{j_1, \ldots, j_d} \left[ \underbrace{(\boldsymbol{G}_1[i_1, j_1] \ \ldots \ \boldsymbol{G}_d[i_d, j_d])}_{\boldsymbol{W}(i_1, \ldots, i_d;\ j_1, \ldots, j_d)} \cdot x(j_1, \ldots, j_d) \right] + b(i_1, \ldots, i_d)$$

During the training we use fixed ranks and modes of the $\boldsymbol{W}$ TT-representation and optimize with respect to TT-cores.

# TT-layer's properties

- The TT-layer uses less memory than an ordinary FC-layer.
- Computation of mat-vec product for TT-layer is reduced to a sequence of $d$ "small" matrix products and can be done efficiently on GPUs.
- TT-format allows to use "wide" layers with huge amounts of neurons.
- TT-layer can be considered as a regularized FC-layer.

# TensorNet: Issues

There are some issues that make TensorNet restricted in use:

- it is quite hard to train several TT-layers together;
- in convolutional neural networks convolutional layers have to be trained separately from TT-layers;
- TensorNet is only implemented under `MatConvNet` framework for `MATLAB`, which is not very handy for several reasons.

Section 2

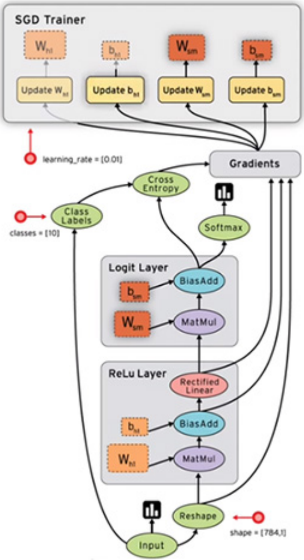
# TensorNet on TensorFlow

# TensorFlow implementation

TensorFlow [4] is an open-source software library developed by Google for numerical computation using data flow graphs.

Main features of TensorFlow:

- deep flexibility;
- `Python` and `C++` API;
- multidimensional tensors processing;
- support of GPU and multi-device computations;
- auto-differentiaion;
- modern stochastic optimization methods are implemented.



# Batch Normalization

To simplify learning of the TensorNet and to learn deep TT-models the Batch-Normalization has been used.

Batch normalization (BN) is a recently proposed [3] technique which accelerates and simplifies deep network training.

Why is deep neural networks training complicated?

- Distribution of each layer's inputs changes as the parameters of previous layers change.
- Careful tuning of learning rate and initialization parameters required.
- Exploding and vanishing gradients.

# BN-layer

BN-layer normalize its inputs using statistics gathered from mini-batch.

Let's denote:

- $\{x_i\}_{i=1}^n$ — input mini-batch ($x_i \in \mathbb{R}^d$);
- $\gamma, \beta \in \mathbb{R}^d$ — scale and shift parameters of the BN-layer;
- $\{y_i\}_{i=1}^n$ — output mini-batch ($y_i \in \mathbb{R}^d$).

BN-layer workflow:

$$\mu \leftarrow \frac{1}{n}\sum_{i=1}^{n} x_i, \qquad \text{(mini-batch mean)}$$

$$\sigma^2 \leftarrow \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu) * (x_i - \mu) \qquad \text{(mini-batch variance)}$$

$$\hat{x}_i \leftarrow \frac{x_i - \mu}{\sqrt{\sigma^2 + \varepsilon}} \qquad \text{(normalize)}$$

$$y_i \leftarrow \gamma * \hat{x}_i + \beta \qquad \text{(scale and shift)}$$

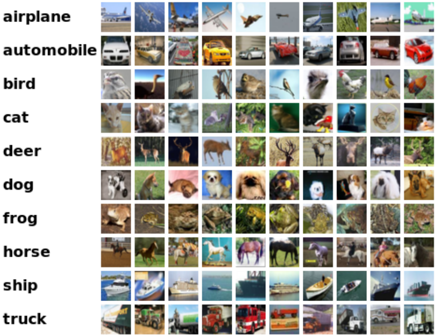
Section 3

# Experimental results

# CIFAR-10 dataset

All considered models were trained on the CIFAR-10 dataset:

- $32 \times 32$ RGB images;
- 10 classes;
- 50 000 train examples;
- 10, 000 validation examples.



# Experiments: non-convolutional networks

| **5-FC-net** | |
|---|---|
| **Layers** | $5 \times \text{FC}$ |
| **Sizes** | $3\,072 \times 4\,096 \times \cdots \times 4\,096 \times 10$ |

| **2-TT-net** | |
|---|---|
| **Layers** | $2 \times \text{TT} + \text{FC}$ |
| **Sizes** | $3\,072 \times 262\,144 \times 4\,096 \times 10 \quad \text{(6-D)}$ |

| **4-TT-net** | |
|---|---|
| **Layers** | $4 \times \text{TT} + \text{FC}$ |
| **Sizes** | $3\,072 \times 64\,000 \times \ldots 64\,000 \times 8\,192 \times 10 \quad \text{(5-D)}$ |

| **10-TT-net** | |
|---|---|
| **Layers** | $10 \times \text{TT} + \text{FC}$ |
| **Sizes** | $3\,072 \times 32\,768 \times \ldots 32\,768 \times 3\,125 \times 10 \quad \text{(5-D)}$ |

# Experiments: non-convolutional networks

| Net | Precision (%) | | Comments |
|---|---|---|---|
| | Train | Validation | |
| 5-FC-net | 94.91 | 59.25 | |
| 2-TT-net | — | 68.53 | previous result |
| 2-TT-net | 73.64 | 68.29 | low ranks |
| 2-TT-net | 80.80 | 71.34 | high ranks |
| 4-TT-net | 84.63 | **74.92** | low ranks |
| 4-TT-net | 98.34 | 72.36 | high ranks |
| 10-TT-net | 80.18 | 69.55 | low ranks |
| 10-TT-net | 88.37 | 69.68 | high ranks |

Best known non-convolutional network result improved.

# Experiments: convolutional networks

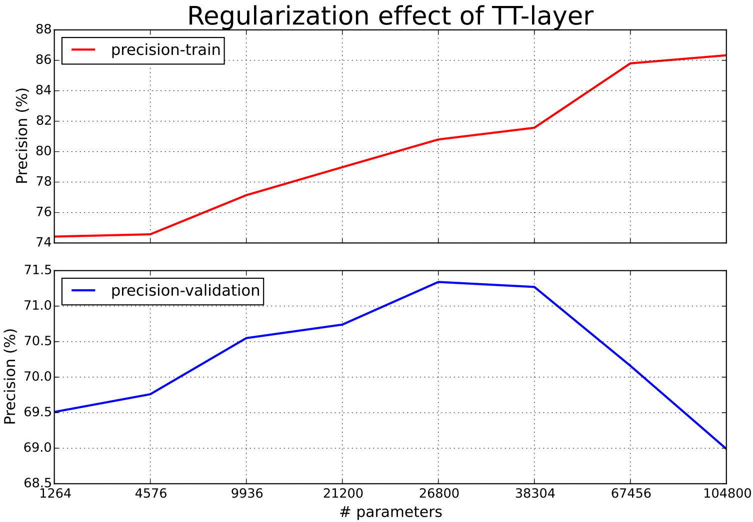
| conv-FC-net | |
|---|---|
| **Layers** | $2 \times$ (CONV + MAX-POOL) + $3 \times$ FC |
| **Sizes** | $[32, 32, 3] \times [5, 5, 64] \times [5, 5, 64] \times 384 \times 192 \times 10$ |

| conv-TT-net | |
|---|---|
| **Layers** | $2 \times$ (CONV + MAX-POOL) + $2 \times$ TT + FC |
| **Sizes** | $[32, 32, 3] \times [5, 5, 64] \times [5, 5, 64] \times 32\,768 \times 1\,024 \times 10$ |

# Experiments: non-convolutional networks

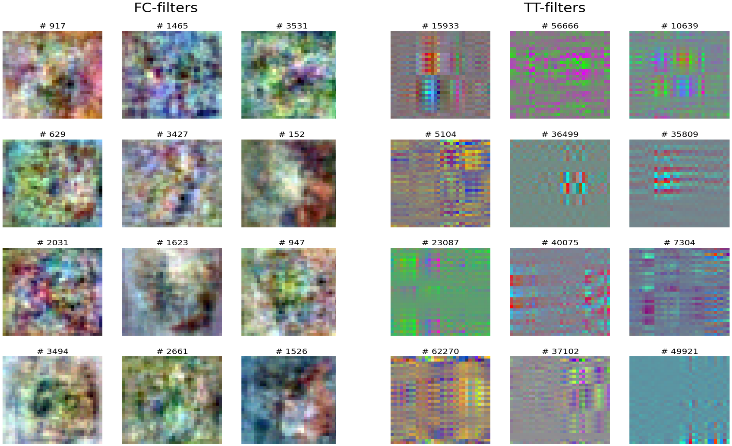
| Net | Precision (%) | | Comments |
|---|---|---|---|
| | Train | Validation | |
| conv-FC-net | 97.58 | 78.96 | |
| conv-FC-net | 99.34 | 79.63 | pretrained convolutions |
| conv-TT-net | 94.49 | 80.10 | |
| conv-TT-net | 96.10 | **80.68** | pretrained convolutions |

# Regularization effect



# Filters visualization

Examples of filters learned on the first layer:



# TensorNet on TensorFlow

What has been done to imporve the TensorNet?

- Python API for learning TensorNet has been implemented under TensorFlow framework.
- Recently proposed batch normalization (BN) technique has been applied. This technique allowed to combine several TT-layers and train convolutional layers alongside with TT ones.
- In experiments with deep TT networks the quality of the classification the `CIFAR-10` dataset by non-convolutional network has been improved. (From 68.53% to 74.92%)
- It has been shown that convolutional network results can be enhanced by using TT-layers.

# References I

[1] Alexander Novikov, Dmitry Podoprikhin, Anton Osokin, Dmitry Vetrov, "Tensorizing Neural Networks", in *Advances in Neural Information Processing Systems 28 (NIPS)*, 2015

[2] I. V. Oseledets, "Tensor-Train decomposition", *SIAM J. Scientific Computing* , vol. 33, no. 5, pp. 2295-2317, 2011.

[3] Ioffe, S., Szegedy, C. "Batch normalization: Accelerating deep network training by reducing internal covariate shift", *arXiv preprint* arXiv:1502.03167, 2015.

[4] Martin Abadi, Ashish Agarwal, Paul Barham, et al. "TensorFlow: Large-scale machine learning on heterogeneous systems", 2015. Software available from tensorflow.org