Put Neural Networks on a Tensor Train

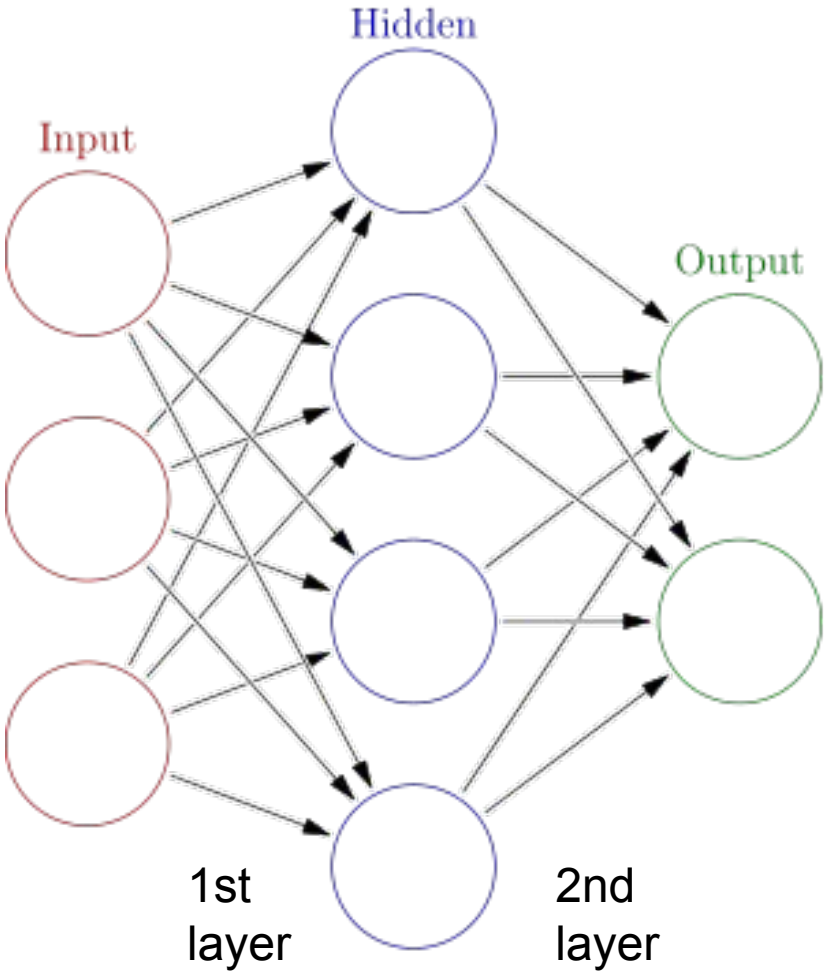
Xiongjie Yu

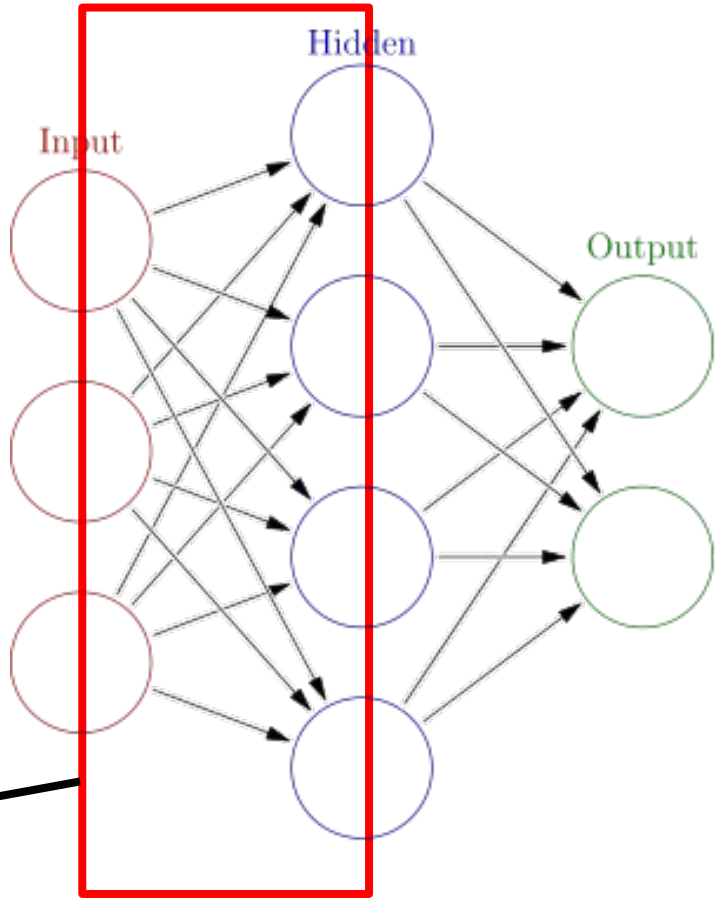
References:

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

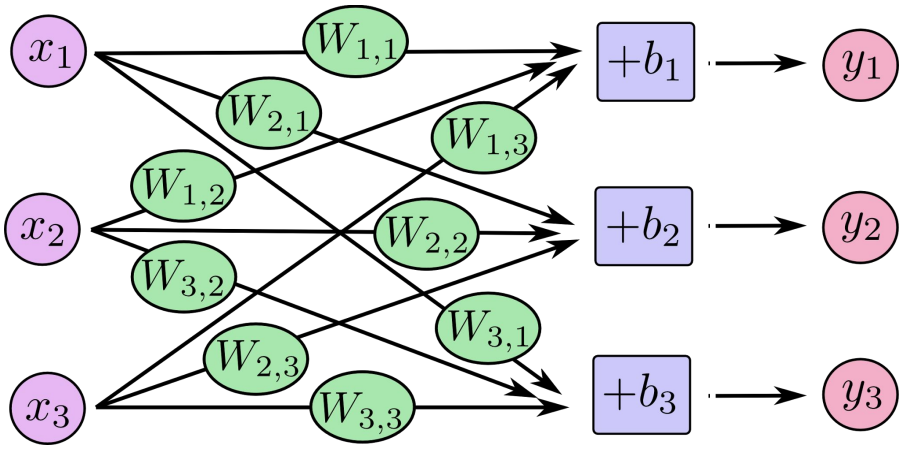
[2] *Ultimate tensorization: compressing convolutional and FC layers alike*
Timur Garipov, Dmitry Podoprikhin, Alexander Novikov, Dmitry Vetrov; *Learning with Tensors: Why Now and How?*, NIPS-2016 workshop (NIPS-2015) [arXiv].

Fully Connected, Feedforward Neural Networks





At each layer, there is a linear transformation by a weight matrix and a bias.



Problems of a Large Network

In practice, the neural network has a lot of neurons and many hidden layers.

1. Training may not be done on portable devices (memory issue).
2. The weight matrix of the fully-connected layer is huge but highly redundant.

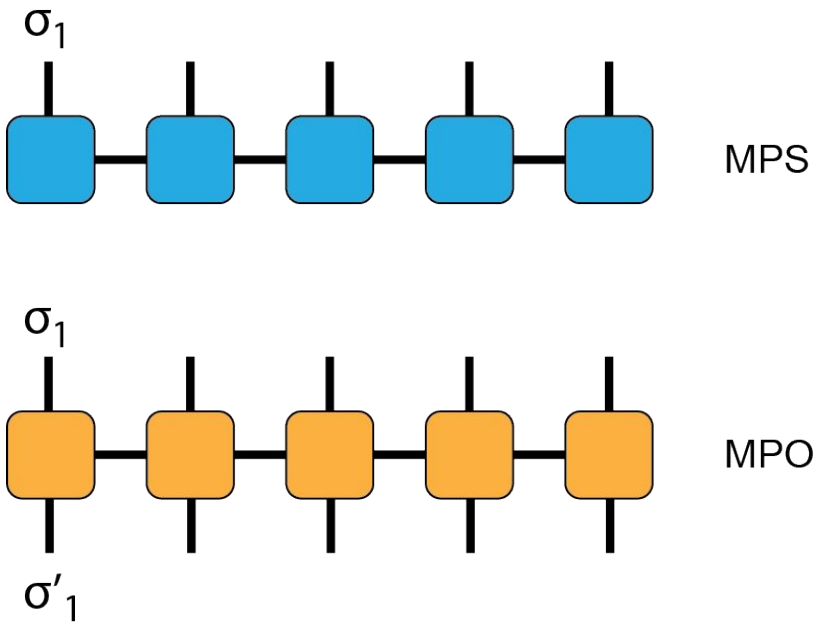
Past approaches (*with very limited compression rate*):

Low rank compression, hash net, ...

Tensor-Train (TT) Decomposition

$$\mathcal{A}(j_1, \ldots, j_d) = \sum_{\alpha_0, \ldots, \alpha_d} G_1[j_1](\alpha_0, \alpha_1) \ldots G_d[j_d](\alpha_{d-1}, \alpha_d).$$

Exactly the same idea as **matrix product states/operators** (MPS/MPO).



Tensorizing the Neural Net

For fully connected, feedforward neural networks, at each layer,

$$y = W x + b$$

Say there is a layer connecting 256 neurons to another 64 neurons.

x: vector of length 256

y: vector of length 64

b: vector of length 64

W: matrix of size (64,256)

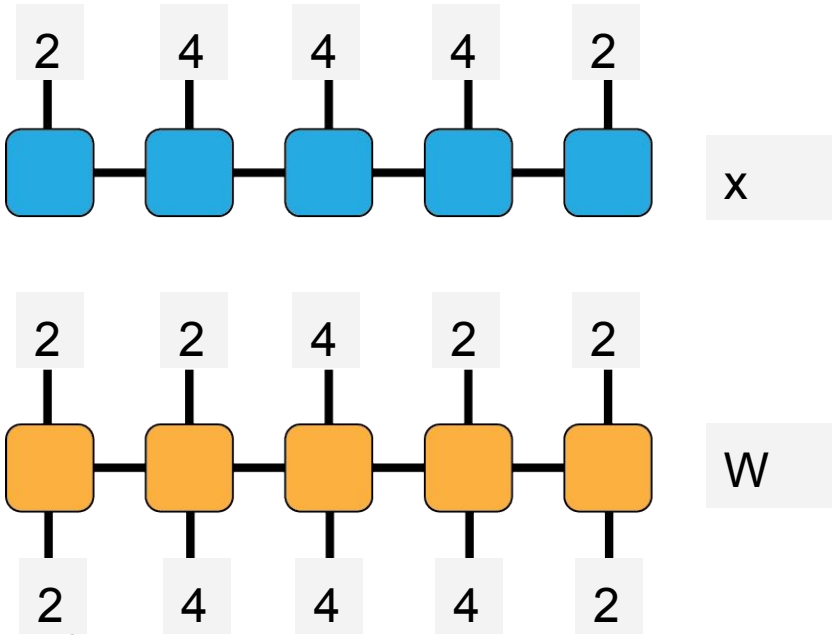
Tensorizing the Neural Net

x: tensor of size (2, 4, 4, 4, 2)

y: tensor of size (2, 2, 4, 2, 2)

b: tensor of size (2, 2, 4, 2, 2)

W: tensor of size (2*2, 2*4, 4*4, 2*4, 2*2)



Basically, x/y/b are converted to MPSs, and W is converted to an MPO.

To control accuracy, one tunes the **TT rank**, or the **max bond dimension**.

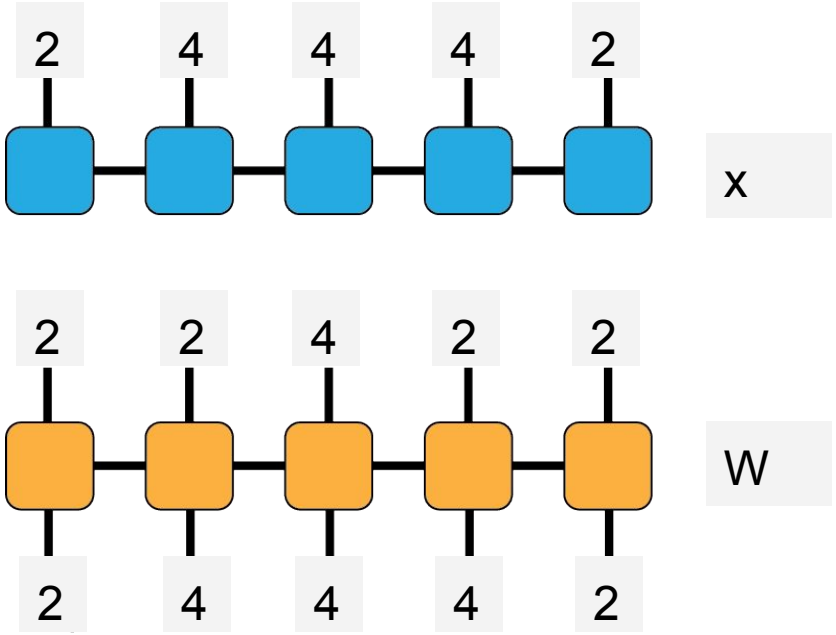
Tensorizing the Neural Net

x: tensor of size (2, 4, 4, 4, 2)

y: tensor of size (2, 2, 4, 2, 2)

b: tensor of size (2, 2, 4, 2, 2)

W: tensor of size (2*2, 2*4, 4*4, 2*4, 2*2)

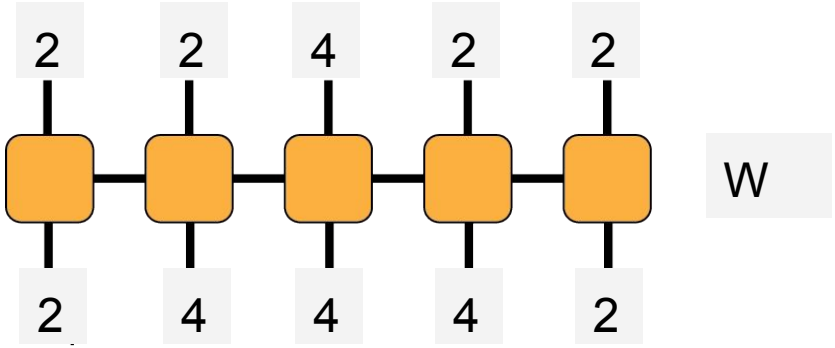


Conversion can be done by doing successive SVD or QR, or iterative fitting.

Reduction of Parameters

Let's focus on the weight matrix.

1. In the **dense matrix representation**, W has 64 x 256 matrix elements, which will be optimized during training.



2. In **TT representation**, assuming the TT rank is r, there are (2*2*r + 2*4*r*r + 4*4*r*r + 2*4*r*r + 2*2*r), or (8r+32r^2) independent parameters.

Huge win in terms of memory usage!

TT Net Has Comparable Accuracy

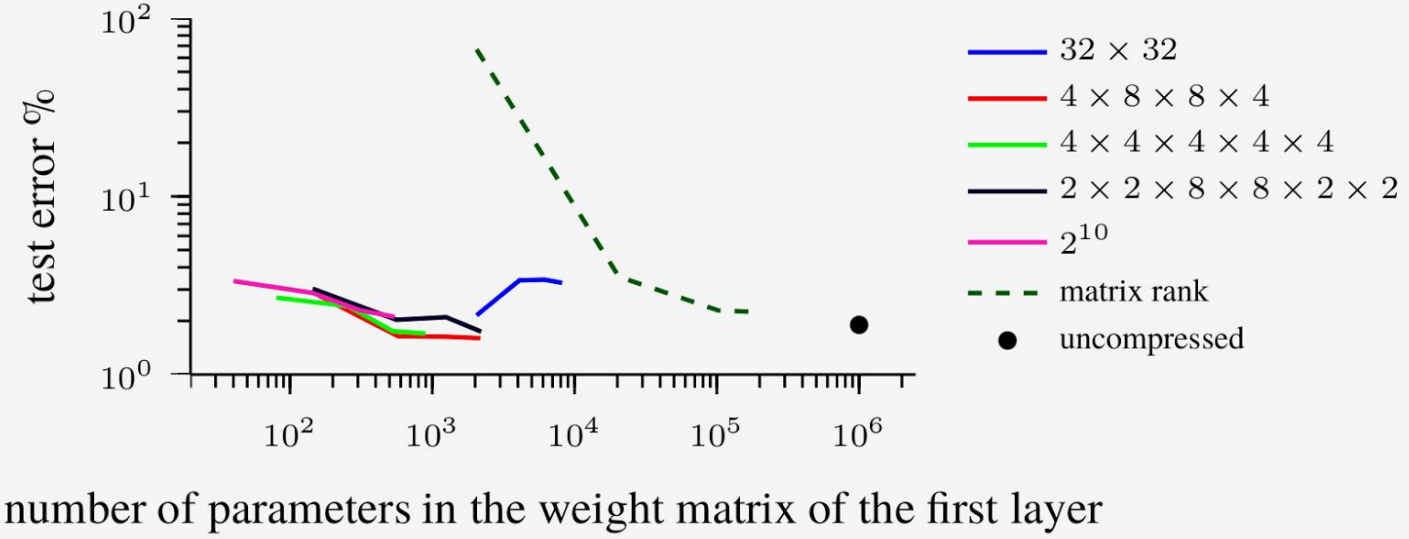


Figure 1: The experiment on the MNIST dataset. We use a two-layered neural network and substitute the first 1024×1024 fully-connected layer with the TT-layer (solid lines) and with the matrix rank decomposition based layer (dashed line). The solid lines of different colors correspond to different ways of reshaping the input and output vectors to tensors (the shapes are reported in the legend). To

Possible Pit Falls

In the MPS/MPO language, given a truncation error, the minimum bond dimension allowed is controlled by the amount of entanglement entropy across spatial cuts.

Translated into the TT language, this means that when converting vectors or matrices into TT representation, how we split the indices, and the relation between indices will dictate the minimum TT rank allowed for a given compression error.

Possible Pit Falls

In the MPS/MPO language, given a truncation error, the minimum bond dimension allowed is controlled by the amount of entanglement entropy across spatial cuts.

Translated into the TT language, this means that when converting vectors or matrices into TT representation, how we split the indices, and the relation between indices will dictate the minimum TT rank allowed for a given compression error.

This works well for images, but might fail for datasets with messy correlations.

More Resources

Jupyter notebook on the notMNIST dataset (with intro to tensorflow).

https://github.com/timgaripov/TensorNet-TF

https://github.com/Bihaqo/TensorNet

Ultimate tensorization: compressing convolutional and FC layers alike
Timur Garipov, Dmitry Podoprikhin, Alexander Novikov, Dmitry Vetrov;
Learning with Tensors: Why Now and How?, NIPS-2016 workshop (NIPS-2015) [arXiv].