Review of Tensor Regression Networks

By Gabriel Samberg

October 2024

1 Introduction

In modern day data science one can often stumble upon natural data sets from fields like biology or medicine in which the data is naturally organized as high order tensors. Many Machine Learning or Deep Learning models (especially the ones that are designed for the task of classification) have what is called fully connected layers as part of the model. such layers require the flattening of the high order activation tensors and as a by product, the multi modal topology of the tensor is being lost. An additional drawback of this approach of flattening multi modal tensors and passing them through fully connected layers is due to the fact that fully connected layers require a large number of parameters.
To address these drawbacks the authors of the paper suggesting to utilize the tools introduced in Tensorial Algebra and incorporate them into a new type of end to end trainable layer that can replace the traditional components of the model that result in the loss of the multimodal structure. The new suggested layers will result in preserving the multimodal structure of the activation tensors and cost altogether in much less parameters while preserving high levels of overall accuracy of the model.

2 Main contributions of the paper

The authors are introducing two end to end trainable layers. The name of the first one is TCL (short for Tensor Contraction Layer) and the second called TRL (short for Tensor Regression Layer). These layers are designed for dimensionality reduction of the activation tensor and the replacement of traditional fully connected layer such that the multi modal structure of the activation tensors is being preserved.

2.1 Tensor Contraction Layer

Given an activation tensor $\mathcal{X}$ of size $(S_0, I_0, I_1, \ldots, I_N)$ the TCL will preform dimensionality reduction resulting with the tensor $\mathcal{X}'$ of size $(S_0, R_0, R_1, \ldots, R_N)$ following the following scheme:

$$\mathcal{X}' = \mathcal{X} \times_1 V^{(0)} \times_2 V^{(1)} \times_3 \cdots \times_{N+1} V^{(N)}$$

With the learnable factors $V^{(k)} \in \mathbb{R}^{R_k \times I_k}, k \in [0, \ldots, N]$. Notice that the mode products start only from the second mode of $\mathcal{X}$ as the first mode is corresponding to the batch size of $\mathcal{X}$. Notice also that the TCL allows preservation of the multimodal structure of $\mathcal{X}$ when reducing it's dimensionality. Finally, by using the TCL with input activation tensor of the size $(S_0, I_0, \ldots, I_N)$ and output size of $(S_0, R_0, \ldots, R_N)$ the TCL has the total of $\sum_{k=0}^{N} I_k \cdot R_k$ parameters while a fully connected layer would have $\prod_{k=0}^{N} I_k \cdot R_k$, hence we can see that for a low rank TCL the model benefits from tremendous savings in parameters.

2.2 Tensor Regression Layer

As I've stated above, the tensor regression layer was proposed in order to replace the traditional output generating layers that discard all multimodal structure of the input activation tensor when flattening it.
Before introducing the proposed layer I want to formally introduce the notion of "Generalized inner-product". Given two tensors $\mathcal{X}, \mathcal{Y} \in \mathbb{R}^{I_0 \times I_1 \times \cdots \times I_N}$ we say that thier inner product is defined as

$$\langle \mathcal{X}, \mathcal{Y} \rangle = \sum_{i_0=0}^{I_0-1} \sum_{i_1=0}^{I_1-1} \cdots \sum_{i_n=0}^{I_N-1} \mathcal{X}_{i_0,i_1,\ldots,i_n} \mathcal{Y}_{i_0,i_1,\ldots,i_n}$$

Now, having two tensors $\mathcal{X} \in \mathbb{R}^{I_x \times I_1 \times \cdots \times I_N}$ and $\mathcal{Y} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N \times I_y}$ sharing N consequent modes of same size, we define the "Generalized innner product" along the N last (and respectively first) modes of $\mathcal{X}$ (and respectively $\mathcal{Y}$) to be

$$\langle \mathcal{X}, \mathcal{Y} \rangle_N = \sum_{i_1=0}^{I_1-1} \sum_{i_2=0}^{I_2-1} \cdots \sum_{i_n=0}^{I_N-1} \mathcal{X}_{:,i_1,i_2,\ldots,i_n} \mathcal{Y}_{i_1,i_2,\ldots,i_n,:}$$

and $\langle \mathcal{X}, \mathcal{Y} \rangle_N \in \mathbb{R}^{I_x \times I_y}$.
Now, having the batched input activation tensor $\mathcal{X} \in \mathbb{R}^{S \times I_0 \times I_1 \times \cdots \times I_N}$ and the lables matrix $\mathbf{Y} \in \mathbb{R}^{S \times O}$ (Where O corresponds to the labels and S to the number of samples of data we have), we want to learn a weights tensor $\mathcal{W} \in \mathbb{R}^{I_0 \times I_1 \times \cdots \times I_N \times O}$ under some fixed rank constraints $(R_0, \ldots, R_N, R_{N+1})$ and a bias $\mathbf{b} \in \mathbb{R}^O$ s.t

$$\mathcal{W} = [\mathcal{G}; \mathbf{U}^{(0)}, \mathbf{U}^{(1)}, \ldots, \mathbf{U}^{(N)}, \mathbf{U}^{(N+1)}],$$

$$\mathbf{Y} = \langle \mathcal{X}, \mathcal{W} \rangle_{N+1} + b$$

with $\mathcal{G} \in \mathbb{R}^{R_0 \times \cdots \times R_N \times R_{N+1}}$ and $\mathbf{U}^{(k)} \in \mathbb{R}^{I_k \times R_k}$ for each k in $[0, \ldots, N]$ and $\mathbf{U}^{(N+1)} \in \mathbb{R}^{O \times R_{N+1}}$. Here notice that adding a bias vector $\mathbf{b}$ to a matrix $\langle \mathcal{X}, \mathcal{W} \rangle_{N+1}$ is done row wise.
As it can be seen, the TRL is acting directly on the input activation tensor without discarding it's multimodal structure. Now, Assume our model has a n dimensional output, then, with an input activation tensor $\mathcal{X} \in \mathbb{R}^{S \times I_0 \times \ldots I_N}$ the fully connected layer will have $n \cdot \prod_{k=0}^{N} I_k$ parameters versus a rank $(R_0, \ldots, R_N, R_{N+1})$

TRL that will have $\prod_{k=0}^{N+1} R_k + \sum_{k=0}^{N} I_k \cdot R_k + n \cdot R_{N+1}$. Here again we see that by constraining the ranks of the TRL to be low we can tremendously decrease the number of parameters used by the model.

Efficient implementation of TRL

Also worth mentioning is that the authors introduced an efficient way to implement the TRL.
Since the weight tensor $\mathcal{W}$ is given in its Tucker form, namely

$$\mathcal{W} = [\mathcal{G}; \mathbf{U}^{(0)}, \mathbf{U}^{(1)}, \ldots, \mathbf{U}^{(N)}, \mathbf{U}^{(N+1)}]$$

We see that

$$\mathbf{Y} = \langle \mathcal{X}, \mathcal{W} \rangle_{N+1} + b = \langle \mathcal{X}, \mathcal{G} \times_0 \mathbf{U}^{(0)} \times_1 \mathbf{U}^{(1)} \cdots \times_{N+1} \mathbf{U}^{(N+1)} \rangle_{N+1} + b$$

Which can be equivalently rewritten as

$$\langle \mathcal{X} \times_0 (\mathbf{U}^{(0)})^T \times_1 (\mathbf{U}^{(1)})^T \times \cdots \times_N (\mathbf{U}^{(N)})^T, \mathcal{G} \times_{N+1} \mathbf{U}^{(N+1)} \rangle_{N+1} + b$$

Which can be rewritten again as

$$\langle \mathcal{X} \times_0 \mathbf{V}^{(0)} \times_1 \mathbf{V}^{(1)} \times \cdots \times_N \mathbf{V}^{(N)}, \mathcal{G} \times_{N+1} \mathbf{U}^{(N+1)} \rangle_{N+1} + b$$

The last expression can be actually viewed as applying first TCL to the input tensor $\mathcal{X}$ and then apply TRL to $TCL(\mathcal{X})$ with the weight tensor $\mathcal{G} \times_{N+1} \mathbf{U}^{(N+1)}$. By following this approach, most of the computation is done in the lower dimensional subspace rather than directly on the dimensions of $\mathcal{X}$.

3 Experiments

In this section I will present some of the experiments conducted in the paper and my attempt to reproduce them.

3.1 Ablation studies

Experiment goal : To illustrate the effectiveness of the low-rank tensor regression in terms of learning and data efficiency and for that purpose to investigate it first in isolation.

Experiment setting : We take a fixed matrix $\mathbf{W} \in \mathbb{R}^{64 \times 64}$ and then generate 1000 data samples $\mathcal{X} \in \mathbb{R}^{8 \times 8}$ each drawn from a Gaussian distribution $\mathcal{N}(0, 3)$. Next, we generate our labels as $y = Vec(\mathcal{X})W \in \mathbb{R}^{64}$. Next we add noise $\mathcal{E}$ which is also sampled from $\mathcal{N}(0, 3)$ to each data sample $\mathcal{X}$ and result with our dataset for the experiment $\hat{\mathcal{X}} \in \mathbb{R}^{1000 \times 8 \times 8}$ such that we have $\hat{\mathcal{X}}[i, :, :] = \mathcal{X}_i + \mathcal{E}_i$. Having that, we compare training a TRL with squared loss and training a fully connected layer with a squared loss (simple Linear Regression) and plot the

weights of the model into a 64×64 image.
Here notice that since $\langle \mathcal{X}, \mathcal{W} \rangle_N = \mathcal{X}_{[0]} \mathcal{W}_{[N+1]}$, after training the TRL we take it's weights given in decomposed form, construct back $\mathcal{W}$ and then take the matrix $\mathcal{W}_{[N+1]}$ for our plot. As for the Linear Regression, we just take the weights matrix and plot it, no further manipulation is needed.

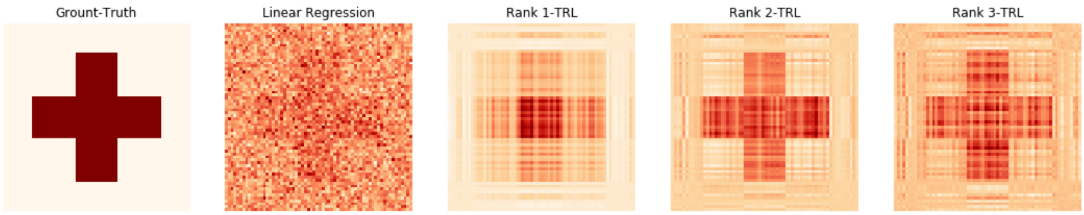

Figure 1: The visuals the authors got from experimenting.
Ground-Truth is the fixed matrix **W**
For the linear regression they have plotted the 64×64 weights matrix of the fully connected layer.
As for the TRL the plot is of the matrix $\mathcal{W}_{[2]}$

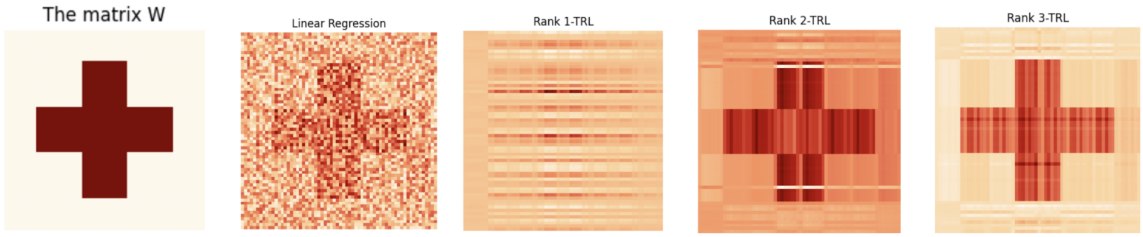

Figure 2: The visuals I have got in the attempt to recreate the visual results of the authors shown in Figure 1

Couple Important points regarding the experiment. In section 6.1 with the name "Implementation details" in the paper, the authors mention that when training the layers from scratch they recommend to add batch normalization layer before and after the TCL/TRL to avoid vanishing or exploding gradients, and to make the layers more robust to changes in the initialization of the factors. I have followed this recommendation in my attempt to recreate the authors results.
The authors omitted details regarding the hyper parameters of the models such as for how many epochs the models were trained, what learning rate was used and even what kind of optimizer was in use. Note also that the size of the dataset generated is also not mentioned.

I've come to realize through experimenting that in order for the weight matrix of the Linear Regression model to look like it looks in figure 1, the learning rate should be low and the number of epochs low as well. I've used 1 epoch with learning_rate = 0.1, the problem is that when I matched the TRL number of epochs and learning rate, I got really bad results.

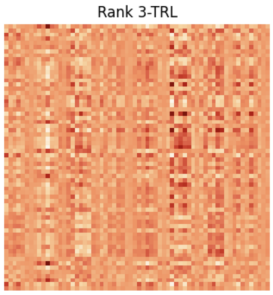

Figure 3: Rank 3-TRL when matching the low learning rate and number of epochs that were used for the Linear Regression

Interestingly enough, When fixing the number of epochs and learning rate that were used for the TRL such that it behaves close enough to the results the authors got, and matching the number of epochs and learning rate of the Linear Regression according to it, I got that the Linear Regression weight matrix looks significantly better than even higher rank TRL.

Figure 4: Linear Regression weight matrix when matching the number of epochs and learning rate used for the TRL to look as it looks in Figure 2

Another result the authors have shown in the paper is the comparison of the data efficiency between Linear regression and TRL by varying the number of training samples. Following this simple idea, I could not reproduce the results that the authors got. I could not point for the reason why is that unfortunately. I put here both of the plots.

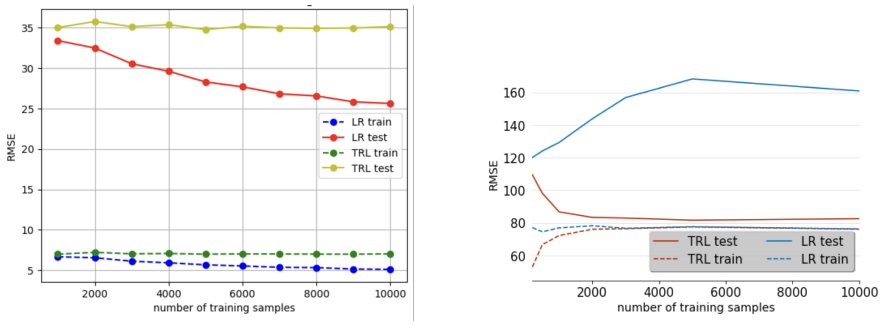

Figure 5: **On the left**: My results for the efficiency test.
On the right: Authors results for the same test.

3.2 Large scale image classification

In this experiment the authors took the ILSVRC dataset (part of the ImageNet dataset) and the ResNet model. Since I was very limited in computing power I had to compromise and use a significantly smaller dataset. For this experiment I took the MNIST data set and the pretrained ResNet50 model which I have adapted to the new dataset.

Experiment goal : To compare the baseline model against the case when the last layers of the model, namely the average pooling layer and fully connected layer are removed and replaced with the TRL with different low rank scenarios. The goal is to get a hold of space savings in parameters of the layer versus the achieved accuracy.

Table 1: Space savings versus performance in terms of accuracy

	Accuracy(%)	**Space savings(%)**
Baseline	99.05	0
(8, 1, 1, 10)TRL	98.93	19.16
(5, 1, 1, 10)TRL	98.70	49.29
(2, 1, 1, 10)TRL	98.10	79.42
(1, 1, 1, 10)TRL	21.08	89.46

As it can be seen in Table 1, the TRL achieves high accuracy while also making very decent savings in parameters when compared to the fully connected layer of the original model architecture (the Baseline).