Multi-layered tensor networks for image classification

Selvan, Raghavendra; Ørting, Silas; Dam, Erik B

Publication date:
2020

Document version
Peer reviewed version

Document license:
Other

Citation for published version (APA):
Multi-layered tensor networks for image classification

Raghavendra Selvan
Department of Computer Science
Department of Neuroscience
University of Copenhagen
Denmark
raghav@di.ku.dk

Silas Ørting
Department of Computer Science
University of Copenhagen
Denmark
silas@di.ku.dk

Erik B Dam
Department of Computer Science
University of Copenhagen
Denmark
erikdam@di.ku.dk

Abstract

The recently introduced locally orderless tensor network (LoTeNet) for supervised image classification uses matrix product state (MPS) operations on grids of transformed image patches. The resulting patch representations are combined back together into the image space and aggregated hierarchically using multiple MPS blocks per layer to obtain the final decision rules. In this work, we propose a non-patch based modification to LoTeNet that performs one MPS operation per layer, instead of several patch-level operations. The spatial information in the input images to MPS blocks at each layer is squeezed into the feature dimension, similar to LoTeNet, to maximise retained spatial correlation between pixels when images are flattened into 1D vectors. The proposed multi-layered tensor network (MLTN) is capable of learning linear decision boundaries in high dimensional spaces in a multi-layered setting, which results in a reduction in the computation cost compared to LoTeNet without any degradation in performance.

1 Introduction

Tensor networks are factorisations of higher order tensors into lower order tensors [15][13]. Of late, such factorised tensor representations have seen an increased interest in supervised learning since the early work that presented connections to machine learning in [11][20][2]. One reason for this interest could be attributed to the possibility of using tensor networks within the end-to-end learning settings enabled by automatic differentiation, which has driven much of current deep learning [17][9].

In this work, we focus on supervised image classification using the tensor train [3] or the matrix product states (MPS) tensor networks which factorise any order-N tensor into a chain (network) of lower order tensors [13][5][18]. In [20][5], 2-D images are flattened into 1-D vectors, lifted to exponentially high dimensions and a linear decision boundary is approximated using MPS. One approach to optimise weights of the MPS approximation is using sweeping algorithms similar to the density matrix renormalization group (DMRG) algorithm [10] as performed in [20]. Another approach is to optimise the MPS weights using gradient based optimisation using automatic differentiation implemented using the backpropagation algorithm, as performed in [5][18].

1 Source code is available at https://github.com/raghavian/mltn
2 Matrix product states (MPS) and tensor trains are used interchangeably in literature. We adhere to MPS.
2 Method

Linear decision boundaries in exponentially high dimensional spaces can be powerful; this has been the primary insight in exploring tensor networks for supervised learning [12, 20, 5]. To achieve this, low dimensional data is first lifted to a high dimensional space. In this work, the linear decision boundary in high dimensional spaces is approximated using multiple layers of MPS operations. Image information in small image regions are moved to the feature dimension using the squeeze operation. These reshaped images are flattened into 1D vectors and contracted using an MPS operation to obtain intermediate representations towards yielding the final decision boundary. This output from an MPS step is rearranged back into the image space forming one layer of the MLTN. A high level visualisation of the proposed model is shown in Figure 1. Each of these steps are described in detail next.

2.1 Linear models in high dimensional spaces

Consider a 2 dimensional image (an order-2 tensor): \( X \in \mathbb{R}^{H \times W} \), with \( N \) pixels which is then flattened into a 1-dimensional vector \( \mathbf{x} \in \mathbb{R}^{N} \). This flattened input image is lifted into a high dimensional space in two steps: first, a pixel-level local feature map is applied to increase the feature dimension. For any pixel, \( x_j \), it is given by \( \psi^j(x_j) : \mathbb{R} \rightarrow \mathbb{R}^3 \). Commonly used feature maps in literature include sinusoidal or intensity transformations \([20, 5, 16]\). In the second step, a joint feature map is obtained from the local feature maps by computing their tensor product resulting in an order-\( N \) tensor of \( d \) dimensions:

\[
\Phi^{i_1 \ldots i_N}(\mathbf{x}) = \psi^{i_1}(x_1) \otimes \psi^{i_2}(x_2) \otimes \cdots \otimes \psi^{i_N}(x_N). \tag{1}
\]

Given the high dimensional joint feature map \( \Phi(\mathbf{x}) \), the linear decision boundary is given by the following tensor inner product:

\[
f^m(\mathbf{x}) = (\Theta^m_{i_1 \ldots i_N}(\mathbf{x})) \cdot (\Phi^{i_1 \ldots i_N}(\mathbf{x})) , \tag{2}
\]

where \( \Theta \) is an order-(\( N + 1 \)) weight tensor, with output dimension, \( m \), corresponding to the number of output classes.

The weight tensor, \( \Theta \), consists of \( d^N \) tunable parameters and computing the inner product in Eq. (2) quickly becomes infeasible with increasing \( N \) \([5, 18]\). One strategy to overcome this constraint is to approximate the inner product using the MPS tensor network \([15, 13, 5, 19]\). MPS approximates an order-\( N \) tensor by factorising it into a chain of order-3 tensors. The weight tensor, \( \Theta \), can be approximated with MPS contraction as

\[
\Theta^m_{i_1 \ldots i_N}(\mathbf{x}) = \sum_{\alpha_1, \alpha_2, \ldots, \alpha_N} A^{i_1}_{\alpha_1} A^{i_2}_{\alpha_1 \alpha_2} A^{i_3}_{\alpha_2 \alpha_3} \cdots A^{i_N}_{\alpha_N} \tag{3}
\]

where \( A^{i_j} \) are the lower-order tensors. The subscripts indices \( \alpha_j \) are the virtual indices that are contracted. Dimension of the virtual indices is \( \beta \) which is the bond dimension. The MPS approximation

\[3\]The tensor indices \( i_1 \ldots i_N \) are dropped for ease of notation.

\[4\]We show the indices that are being summed over as subscripts following tensor contraction notation.
Figure 1: A: Multi-layered tensor network (MLTN) with one matrix product state (MPS) operation per layer. Notice the sequence of Squeeze – Contract – Rearrange steps forming one layer of MLTN. The output dimension of each MPS block is marked along the edges as $[N_1, N_2, m]$. B: Overview of the squeeze operation which transforms a 2D image into a vector with inflated feature dimensions. C: Four sample images from the LIDC-IDRI dataset. The first two belong to the negative class and the last two to positive class, indicating the absence and presence of tumour based on rater agreement.

reduces the number of parameters from $d^N$ to $d \cdot N \cdot \beta^2$ with $\beta$ controlling the quality of these approximations. The MPS approximation in tensor notation is indicated as the contract step in Figure 1. Tensor indices are dropped in the remainder of the manuscript for ease of notation.

2.2 Squeezed local feature maps

The loss of spatial correlation between pixels caused by flattening 2-D images into 1-D vectors has been shown to hamper performance in classification tasks when dealing with images of high spatial resolution [18]. In this work, we reshape small image regions so that the spatial information is moved to the feature dimension, which is similar to the squeeze operation in [4, 18]. The size of the squeezed regions is controlled by the kernel stride parameter $k$ to obtain square regions of size $k \times k$.

The transformation in dimensions due to the squeeze operation, parameterised by the stride $k$, is

$$
\psi(\cdot; k) : \{ X \in \mathbb{R}^{H \times W \times d}, d = 1 \} \rightarrow \{ x \in \mathbb{R}^{(N/k^2) \times d}, d = k^2 \}.
$$

(4)

Note that the squeeze operation increases the feature dimension from $d = 1$ to $d = k^2$. Different from the local feature maps used in literature [20, 5], we propose to use the squeeze operation to increase the feature dimension.

2.3 The multi-layered tensor network

Squeezing spatial information to the feature dimension helps in the retention of pixel correlation when compared to flattening in a single step. To further increase the retained pixel correlation, we propose to compute the decision rule in Eq. (2) in a hierarchical manner; this has also been attempted with different strategies in [3, 16, 18] with varying degrees of success. In this work, we use a multi-layered approach with $L$ layers resulting in the multi-layered tensor network (MLTN):

$$
f_{(l)}(x) = \Theta_{(l)} \cdot \Phi(f_{(l-1)}(x)) \text{ for } l = 1 \ldots L
$$

(5)

with $f_{(0)}(x) = x$, and $f_{(L)}(x) = f(x)$ which is the output tensor of order-1 and dimension $m$. The layer-wise weight tensor $\Theta_{(l)}$ in MLTN has an output dimension of $N/(k^2)^l$. As a result, the output at each layer, $f_{(l)}(x)$, is a vector of dimension $N/(k^2)^l$ which is rearranged back into the image space: $(H/k^l) \times (W/k^l)$ before passing to the squeeze operation for layer $(l + 1)$. Note the exponential reduction in number of pixels between successive layers. For example, given a $128 \times 128$ input image, $k = 4$ and $L = 3$ the number of pixels input to the three MPS layers are: $[32 \times 32, 8 \times 8, 2 \times 2]$ with feature dimension $d = 16$. 

3
Table 1: Performance comparison on LIDC dataset. Number of parameters, computation complexity for forward propagation, computation time per training epoch (t) and area under the receiver operating characteristics (AUROC) curve (higher is better) averaged over 5-fold cross validation for all methods are reported. All three tensor network models use $\beta = 5$.

<table>
<thead>
<tr>
<th>Models</th>
<th># Param. ($M$)</th>
<th>Complexity</th>
<th>time (s)</th>
<th>AUROC</th>
</tr>
</thead>
<tbody>
<tr>
<td>MLTN (ours)</td>
<td>0.42</td>
<td>$O \left( \frac{N \cdot L^2 \cdot k^2 \cdot d \cdot \beta^2}{(k^2)^T} \right)$</td>
<td>2.8</td>
<td>0.88 ± 0.01</td>
</tr>
<tr>
<td>LoTeNet</td>
<td>0.49</td>
<td>$O \left( \frac{N \cdot k^2 \cdot d \cdot \beta^2}{L \cdot k^2 \cdot d \cdot \beta^2} \right)$</td>
<td>18.5</td>
<td>0.87 ± 0.01</td>
</tr>
<tr>
<td>Tenet-X</td>
<td>0.82</td>
<td>$O \left( N \cdot d \cdot \beta^2 \right)$</td>
<td>30.2</td>
<td>0.82 ± 0.01</td>
</tr>
<tr>
<td>MLP (L=4)</td>
<td>0.52</td>
<td>$O \left( N \cdot d \cdot \beta^2 \right)$</td>
<td>3.2</td>
<td>0.86 ± 0.01</td>
</tr>
</tbody>
</table>

Finally, note that there are no non-linear components (including in the local feature maps) in MLTN resulting in a fully linear model. MLTN with $L = 3$ with the sequence of Squeeze – Contract – Rearrange is shown in Figure 1-A and the squeeze operation itself is elaborated in Figure 1-B.

**Optimisation:** The parameters $[\Theta_1, \ldots, \Theta_L]$ in Eq. (5) form the weights of the model which can be learned in a supervised setting. For a given labelled training set with $T$ data points, $D : \{(x_1, y_1), \ldots, (x_T, y_T)\}$, the training loss to be minimised is

$$L_{tr} = \frac{1}{T} \sum_{t=1}^{T} L(f(x_t), y_t),$$

where $L(\cdot)$ can be a suitable loss function such as cross entropy with logits for classification or mean-squared error for regression tasks.

## 3 Experiments

The proposed MLTN model can be used for supervised learning tasks. We demonstrate the capabilities of MLTN for the task of image classification and compare with relevant methods to highlight its features.

### 3.1 Data

The LIDC-IDRI dataset consists of 1018 thoracic CT images with lesions annotated by four radiologists [2]. We extracted 128x128 px 2D slices similar to [8] yielding a total of 15,096 patches. Each of these patches have annotations from four raters marking the tumour regions. These segmentation masks were converted into binary labels indicating the presence (if two or more radiologists marked a tumour) or absence of tumours (if less than two raters marked tumours), resulting in a fairly balanced dataset. All image intensities are normalised to be in $[0, 1]$.

### 3.2 Experimental set-up

The proposed method was compared with three relevant methods: LoTeNet [18], single layer tensor network in [5] (denoted Tenet-X) and a multi-layered perceptron (MLP) composed of four layers. All experiments were performed using five fold cross-validation. MLTN shown in Figure 1-A has two hyperparameters: the bond dimension $\beta$ and the initial kernel stride $k$ obtained from the validation performance on one of the folds, resulting in $\beta = 5$ and $k = 16$. The architecture for LoTeNet was the same as reported in [18] with $\beta = 5$. All models were implemented in PyTorch [14], trained on a single GTX 1080 graphics processing unit with 8GB memory using Adam optimiser [7] with a batch size of 512, batch normalization [6] between successive layers, for a maximum of 200 epochs and a patience of 10 epochs based on the validation accuracy. Learning rate for MLTN was $5 \times 10^{-6}$ to overcome exploding gradient problem at larger learning rates, whereas for other methods it was $5 \times 10^{-4}$ obtained from [19]. The development and training of all models in this work was estimated to produce 2.9 kg of CO2eq, equivalent to 23.9 km travelled by car as measured by Carbontracker[^5].

[^5]: [https://github.com/lfwa/carbontracker/](https://github.com/lfwa/carbontracker/)
3.3 Results

Performance comparison across all methods is reported in Table 1 with area under the receiver operating characteristics (AUROC) curve as the primary measure over the five folds. We noticed that MLTN and LoTeNet performed almost identically (0.87 ± 0.01) showing a large improvement over Tenet-X (0.82 ± 0.01) and a smaller improvement over MLP (0.86 ± 0.01). The average training time per epoch for MLTN (2.8s) shows a clear improvement compared to LoTeNet (18.5s) while attaining the same performance; this difference is even clearer compared to Tenet-X (30.2s). Notice that MLTN, LoTeNet and MLP have comparable number of parameters (∼0.5M) compared to Tenet-X (0.82M).

4 Discussions and Conclusion

Local feature maps such as the sinusoidal map used in [20, 18] or the intensity based ones in [5] are used to enhance pixel level features when constructing the joint feature map. In MLTN no explicit local feature maps were used. Instead, the squeeze operation was used as an implicit feature map to retain spatial information. This has similarities with the wavelet based local feature map used in [16] which reduces the number of pixels by half using wavelet transforms on the image data.

The recursive formulation in Eq. (5) for MLTN is in contrast with LoTeNet which utilises multiple MPS blocks per layer acting on small image patches. At any layer, l, the field of view of an MPS block is $k \times k$ in LoTeNet, whereas for MLTN it is increased to $(H/k^l) \times (W/k^l)$. This could possibly improve the layer-wise representations as MPS contractions act on larger field of view.

The computation complexity for the forward propagation of all methods are reported under the complexity column in Table 1 pointing to the reasons for the drastic reduction in computation time for MLTN compared to other tensor network models. MLTN reduces the exponential dependency on the number of layers ($N^L$) in LoTeNet to a linear dependence ($N \cdot L$). The scaling of number of pixels by the kernel stride $k$ reduces the computation complexity compared to Tenet-X. When compared to deep enough MLPs ($L \geq 2$), which also have an exponential dependence on the number of layers ($N^L$), the computation cost of MLTN due to squeeze operation ($k^2 \cdot d \cdot \beta^2$) can be smaller than that of the MLP.

We observed that the modifications introduced in MLTN made it susceptible to training issues such as the exploding gradient problem. We attribute this to the large field of view of MPS operations in each of the layers. This was alleviated by using smaller learning rates ($5 \times 10^{-6}$ for MLTN instead of $5 \times 10^{-4}$ for LoTeNet). Reducing the depth of MLTN ($\leq 2$), data augmentation and gradient clipping [21] also helped reduce the exploding gradient behaviour but these were not used in the reported experiments.

In conclusion, we proposed a modification to the locally orderless tensor network [18] for supervised image classification. Instead of using multiple patch level MPS operations over successive layers, we investigate the possibility of using a single MPS operation at each layer. This modification increased the field of view of MPS operation at each layer and reduced the number of MPS operations resulting in a substantial reduction in computation complexity (Table 1) without deteriorating the classification performance. The proposed formulation of MPS based tensor networks in the form of MLTN is a fully linear model and could lend itself to be applicable in more diverse settings similar to MLPs.

Acknowledgements

Authors would like to thank Jon Sporring and the Medical Image Analysis group at the Machine Learning Section, for fruitful discussions.

References


