UNIVERSITY OF COPENHAGEN

# Tensor Networks for Medical Image Classification

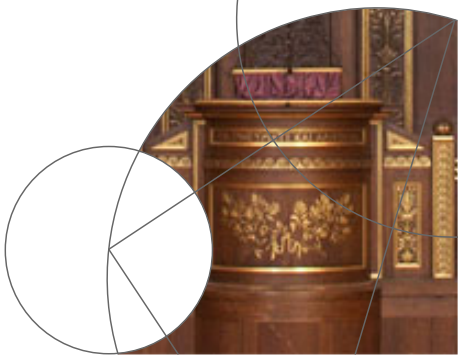
## Presented at MIDL, 2020


**Raghav**endra Selvan & Erik B. Dam

Dept. of Computer Science,
University of Copenhagen

raghav@di.ku.dk
@SuperVoxel

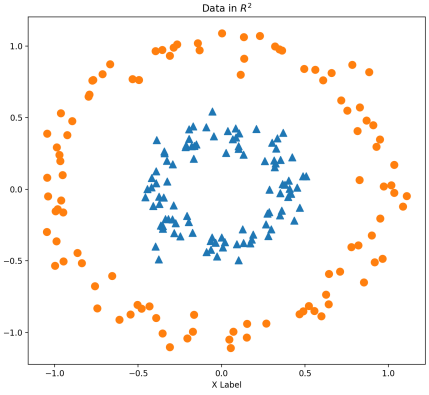

Code: `https://github.com/raghavian/lotenet_pytorch/`

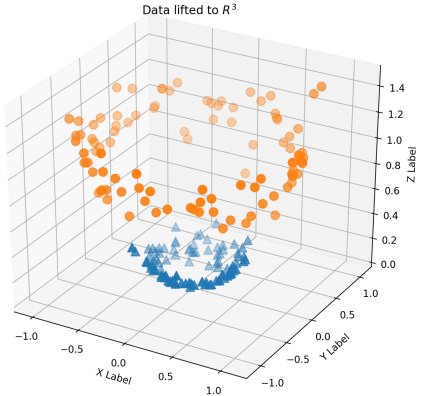


# Overview

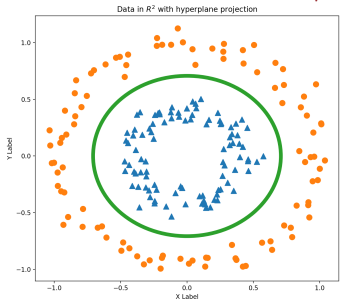
- Motivation
- Background
- Locally orderless Tensor Networks
- Experiments
- Summary & Conclusion

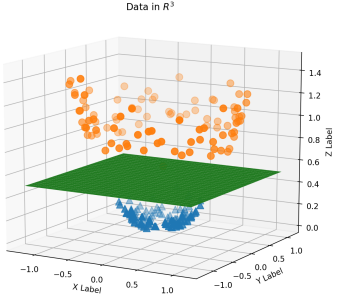
# How far can we push linear decision boundaries?




# Decision boundaries in low/high dimensions




Kernelizing to higher dimensions (SVMs)

Non-linear decisions in lower dimensions (NNs)

## Tensor networks

Linear decision boundaries in exponentially high dimensional spaces.

# One slide introduction to tensor notation

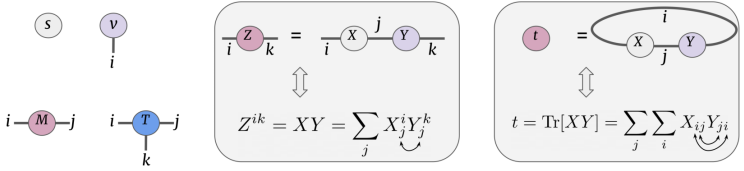



Figure 1: (left) Tensor notation depicting a scalar $s$, vector $v^i$, matrix $M^{ij}$ and a general 3-D tensor $T^{ijk}$. (center) Tensor notation for matrix multiplication or *tensor contraction*, which are used extensively in the matrix product state networks used in this work. We adhere to the convention that the contracted indices are written as subscripts. (right) Tensor notation for trace of product of two matrices.

# Linear model in high dimensions: Feature Maps

Consider input image with $N$ pixels, flattened as a vector $\mathbf{x} \in [0,1]^N$

$$\Phi^{i_1,i_2,\ldots i_N}(\mathbf{x}) = \phi^{i_1}(x_1) \otimes \phi^{i_2}(x_2) \otimes \cdots \phi^{i_N}(x_N) \tag{1}$$

$\phi^{i_j}(\cdot)$ is $d$-dimensional local feature map acting on pixel $x_j$:

$$\phi^{i_j}(x_j) = [\cos(\frac{\pi}{2}x_j), \sin(\frac{\pi}{2}x_j)]. \tag{2}$$

## High dimensional feature maps

Dimensionality of the joint feature map $\Phi(\mathbf{x})$ is $d^N$ due to tensor products i.e, an order $N$ tensor

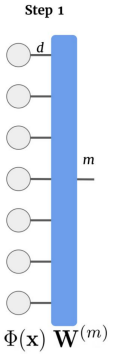
# Linear model in high dimensions: Decision Rule

Decision rule for a multi-class classification task:

$$f(\mathbf{x}) = \arg\max_m f^m(\mathbf{x}), \tag{3}$$

where $m = [0, 1, \ldots M-1]$ are the $M$ classes,

$$f^m(\mathbf{x}) = W^m \cdot \Phi(\mathbf{x}). \tag{4}$$



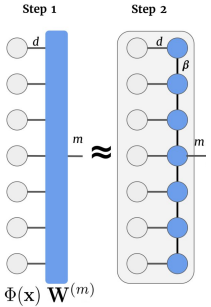

**$W$ has $M \cdot d^N$ tunable weights**

With a gray scale image of size $100 \times 100$ as input and $d = 2$
$W$ has $2 \cdot 2^{10000} \approx 10^{3010}$ parameters.

# Approximate tensor dot product with MPS

Matrix Product State (MPS) is a type of Tensor Network

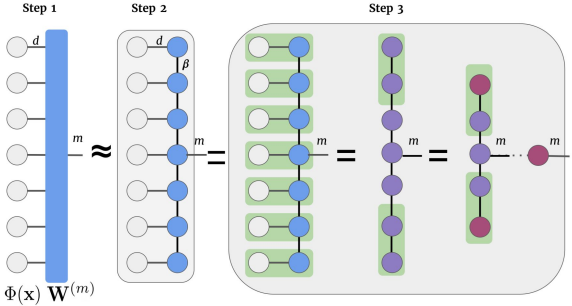
Factorisation of order $N$ tensor into chain of order 3 tensors




# Approximate tensor dot product with MPS

Matrix Product State (MPS) is a type of Tensor Network

Factorisation of order $N$ tensor into chain of order 3 tensors




Reduces computation complexity from $d^N$ to $N \cdot \beta^3 \cdot d$ (linear in $N$)

# Overview

- Motivation
- Background
- Locally orderless Tensor Networks
- Experiments
- Summary & Conclusion

# Tensor Networks for Medical Images

MPS is defined for 1-d inputs

2-d images are flattened in existing literature

Loss of spatial structure

Flattening discards useful information in medical images

## Proposed idea

Flatten small regions assuming local orderlessness.
Aggregate at multiple resolutions.

# Locally orderless Tensor Network: LoTeNet

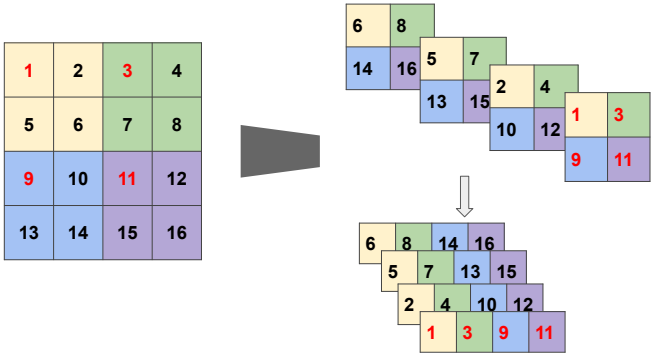
Extending Tensor Networks to medical images

1. Partition image into small patches
2. *Squeeze* patches to retain spatial information
3. Perform MPS contraction at patch level
4. Aggregate and perform squeeze + MPS at next resolution
5. Output decision boundary

# LoTeNet: Partition and Squeeze

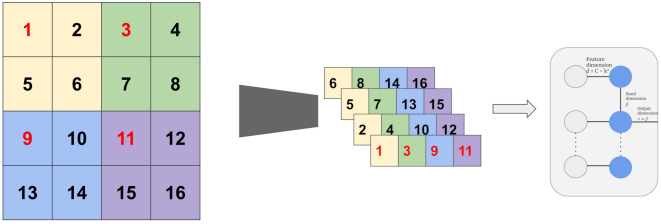
| | | | |
|---|---|---|---|
| 1 | 2 | 3 | 4 |
| 5 | 6 | 7 | 8 |
| 9 | 10 | 11 | 12 |
| 13 | 14 | 15 | 16 |

# LoTeNet: Partition and Squeeze




Squeeze operation with stride $k = 2$. A $4 \times 4 \times 1$ image patch is reshaped into $2 \times 2 \times 4$ stack which then yields a vector of size 4 with feature dimension d=4.

# LoTeNet: Patch level MPS



# LoTeNet: The Final Model

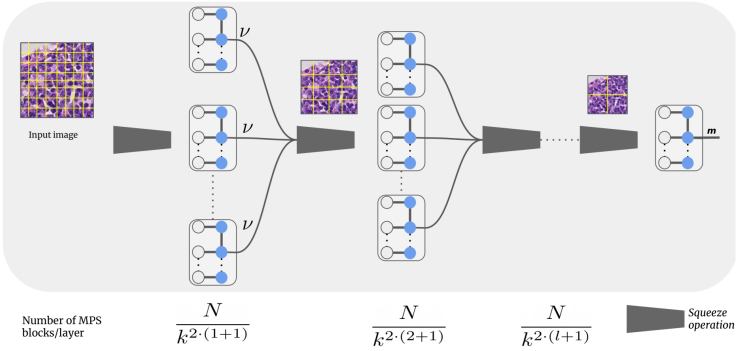


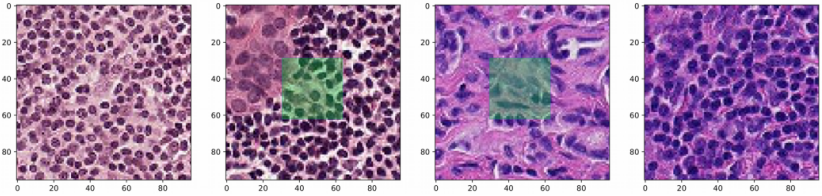

Figure 4: The proposed Locally orderless Tensor Network shown as a series of layers. Each layer consists of several MPS blocks. The squeeze operation is as described in Figure 3. The final MPS block outputs the predictions for the $M$ classes depicted as the edge with index $m$.

Optimized using backpropagation.

# Overview

- Motivation
- Background
- Locally orderless Tensor Networks
- Experiments
- Summary & Conclusion

# Model Evaluation: PCam Dataset



The PatchCamelyon (PCam) dataset

Binary classification

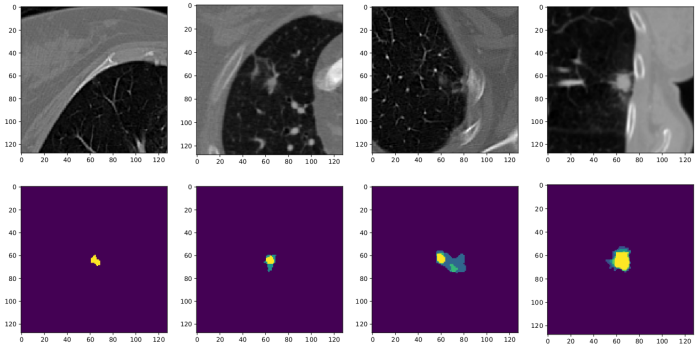
Positive label indicates $\geq$ One pixel with tumour

Image patches of size $96 \times 96$ px

$220k$ patches for training-validation $(80 : 20)$

$57.5k$ test patches

# Model Evaluation: LIDC Dataset



$128 \times 128$ px image patches

$15k$ patches. $60 : 20 : 20$ splits for training/validation/test

Annotated by 4 radiologists. Originally a segmentation dataset

# Model Evaluation: Results

Table 1: Performance comparison on PCam dataset (left) and LIDC dataset (right). For the LIDC models we also compare the GPU memory utilisation shown in gigabytes.

| PCam Models | GPU (GB) | AUC |
|---|---|---|
| Rotation Eq-CNN | | |
| DenseNet | | |
| LoTeNet (ours) | | |
| Tensor Net-X ($\beta = 10$) | | |

# Model Evaluation: Results

Table 1: Performance comparison on PCam dataset (left) and LIDC dataset (right). For the LIDC models we also compare the GPU memory utilisation shown in gigabytes.

| PCam Models | GPU (GB) | AUC |
|---|---|---|
| Rotation Eq-CNN | | 0.963 |
| DenseNet | | 0.962 |
| LoTeNet (ours) | | 0.943 |
| Tensor Net-X ($\beta = 10$) | | 0.908 |

# Model Evaluation: Results

Table 1: Performance comparison on PCam dataset (left) and LIDC dataset (right). For the LIDC models we also compare the GPU memory utilisation shown in gigabytes.

| PCam Models | GPU (GB) | AUC |
|---|---|---|
| Rotation Eq-CNN | 11.0 | 0.963 |
| DenseNet | 10.5 | 0.962 |
| LoTeNet (ours) | 0.8 | 0.943 |
| Tensor Net-X ($\beta = 10$) | 5.2 | 0.908 |

# Model Evaluation: Results

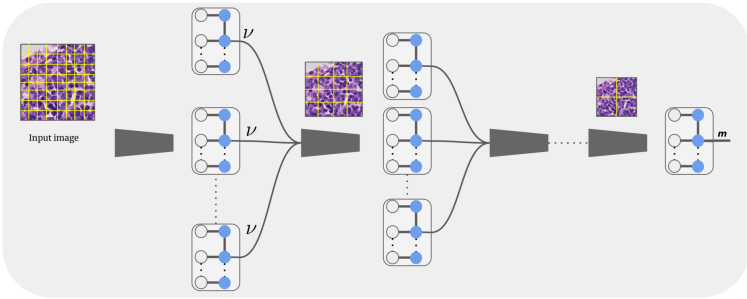
Table 1: Performance comparison on PCam dataset (left) and LIDC dataset (right). For the LIDC models we also compare the GPU memory utilisation shown in gigabytes.

| PCam Models | GPU (GB) | AUC | LIDC Models | GPU (GB) | AUC |
|---|---|---|---|---|---|
| Rotation Eq-CNN | 11.0 | 0.963 | LoTeNet (ours) | 0.7 | 0.874 |
| DenseNet | 10.5 | 0.962 | Tensor Net-X ($\beta = 10$) | 4.5 | 0.847 |
| LoTeNet (ours) | 0.8 | 0.943 | DenseNet | 10.5 | 0.829 |
| Tensor Net-X ($\beta = 10$) | 5.2 | 0.908 | Tensor Net-X ($\beta = 5$) | 1.5 | 0.823 |

# Conclusion

+ Fully linear decision boundary
+ Single model hyperparameter ($\beta$)
+ Squeeze operation helps retain structure
+ LoTeNet performs competitively
+ Massive reduction in GPU utilization
- Tendency to overfit
- Not optimized for efficiency, yet

# Summary




Proposed LoTeNet for medical image classification

Different paradigm compared to feed-forward NNs or CNNs

Low GPU memory requirement ($< 10\%$)

New and exciting applications are to be expected

# Questions

Thanks to Jacob Miller for TorchMPS [1]

Model and data are available here:
`https://github.com/raghavian/lotenet_pytorch`

raghav@di.ku.dk

[1] https://github.com/jemisjoky/TorchMPS