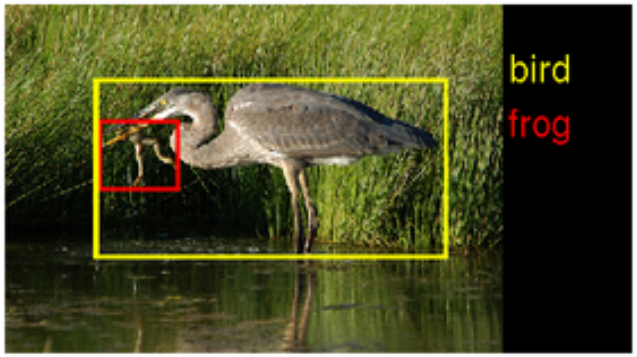
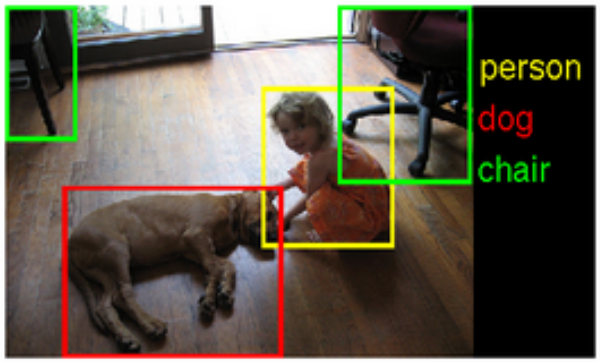
Tensorizing Neural Networks

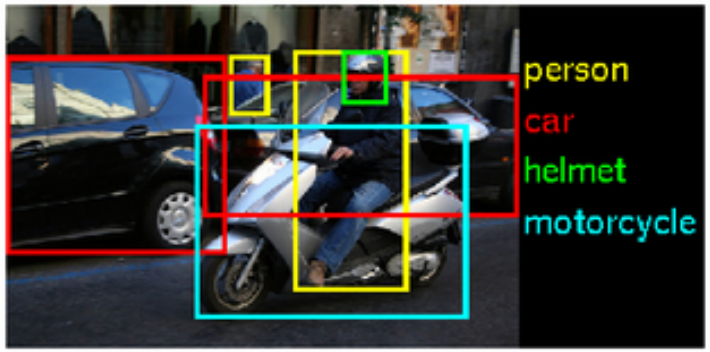
Alexander Novikov Dmitry Podoprikhin Anton Osokin
Dmitry Vetrov

August 24, 2015

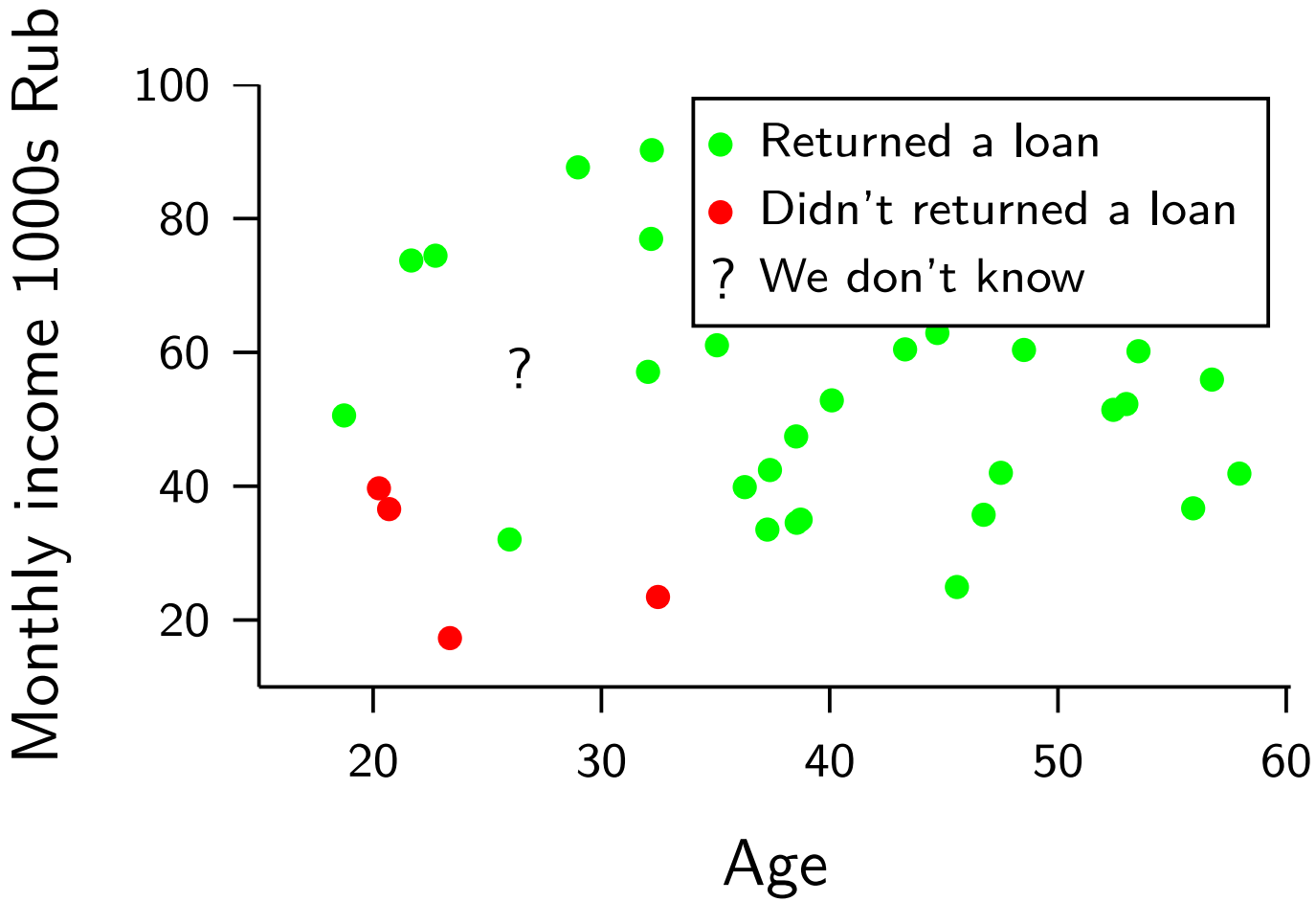
Machine learning





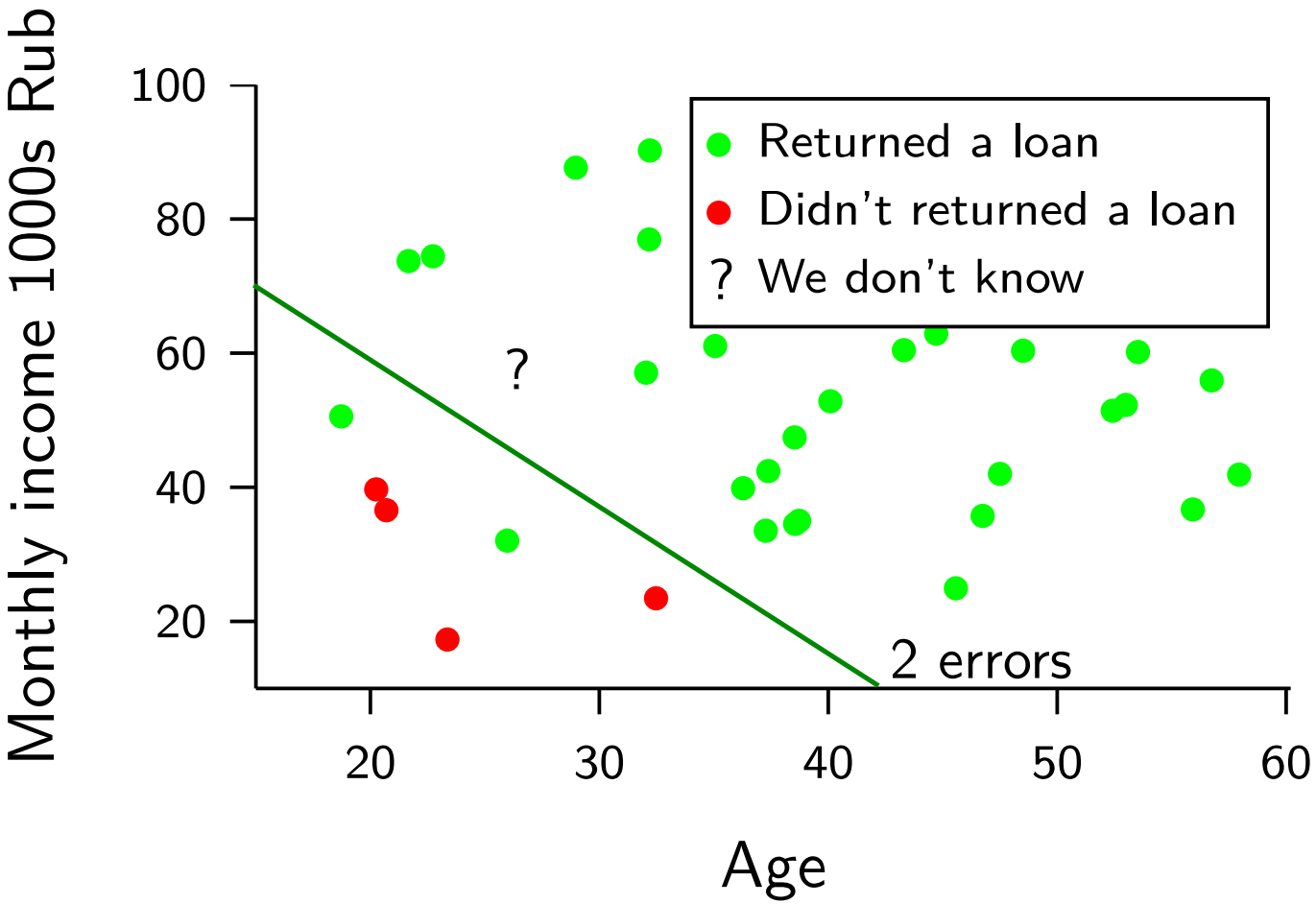


Machine learning



Machine learning

Machine learning



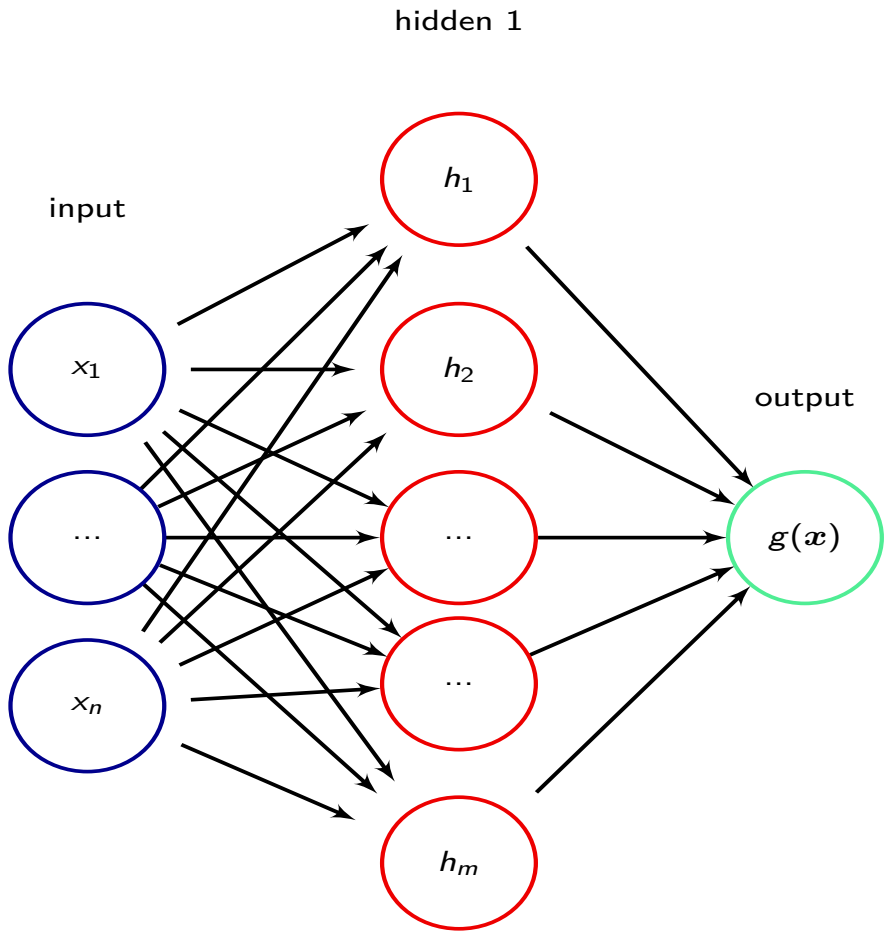
Neural networks

Lets use a composite function:

$$g(\boldsymbol{x}) = f(\boldsymbol{W}_2 \cdot \underbrace{f(\boldsymbol{W}_1 \boldsymbol{x} + \boldsymbol{b}_1)}_{h \in \mathbb{R}^m} + \boldsymbol{b}_2)$$

$$h_k = f\left((\boldsymbol{W}_1 \boldsymbol{x} + \boldsymbol{b}_1)_k\right)$$

$$f(x) = \frac{1}{1 + \exp(-x)}$$



Neural networks learning

$$g(\boldsymbol{x}) = f(\boldsymbol{W}_2 \cdot f(\boldsymbol{W}_1 \boldsymbol{x} + \boldsymbol{b}_1) + \boldsymbol{b}_2)$$

To teach the neural network minimize its' error on the training set:

$$\boldsymbol{W}_1^*, \boldsymbol{b}_1^*, \boldsymbol{W}_2^*, \boldsymbol{b}_2^* = \underset{\boldsymbol{W}_1, \boldsymbol{b}_1, \boldsymbol{W}_2, \boldsymbol{b}_2}{\arg\min} \sum_q (y_q - g(\boldsymbol{x}_q))^2$$

Limitations

You always want to use larger layers and to use more of them.

1. It's hard to train too many layers (too deep networks);
2. For large layers matrix $\boldsymbol{W}$ doesn't fit to memory (especially on mobile devices).

1 Neural networks

2 Matrix formats

3 TensorNet

4 Experiments

Matrix rank decomposition

Lets consider an $M \times N$ matrix $\boldsymbol{W}$ with the rank equals r. We can use $(M + N)r$ parameters instead of MN:

$$\underbrace{\boldsymbol{W}}_{M \times N} = \underbrace{\boldsymbol{A}}_{M \times \mathrm{r}} \underbrace{\boldsymbol{B}}_{\mathrm{r} \times N}$$

It works, but we want even more.

Tensor Train summary

Tensor Train (TT) decomposition (Oseledets 2011):

- A compact representation for vectors, matrices and tensors;
- Allows for efficient application of linear algebra operations.

Mapping example: a vector

Build a mapping from a vector $\boldsymbol{b}$ indices to tensor's elements: $t \leftrightarrow \boldsymbol{i} = (i_1, \ldots, i_d)$

Example (Matlab `reshape`):

$$\begin{aligned} \boldsymbol{B}(1,1,1) &= \boldsymbol{b}(t(1,1,1)) &&= \boldsymbol{b}(1), \\ \boldsymbol{B}(2,1,1) &= \boldsymbol{b}(t(2,1,1)) &&= \boldsymbol{b}(2), \\ &\ldots, \\ \boldsymbol{B}(2,3,3) &= \boldsymbol{b}(t(2,3,3)) &&= \boldsymbol{b}(18). \end{aligned}$$

Matrices in the TT-format

Build a mapping from row / column indices of matrix $\boldsymbol{W} = [\boldsymbol{W}(t, \ell)]$ to vectors $\boldsymbol{i}$ and $\boldsymbol{j}$: $t \leftrightarrow \boldsymbol{i} = (i_1, \ldots, i_d)$ and $\ell \leftrightarrow \boldsymbol{j} = (j_1, \ldots, j_d)$.

TT-format for matrix $\boldsymbol{W}$:

$$\boldsymbol{W}(i_1, \ldots, i_d; j_1, \ldots, j_d) = \boldsymbol{W}(t(\boldsymbol{i}), \ell(\boldsymbol{j})) = \underbrace{\boldsymbol{G}_1[i_1, j_1]}_{1 \times r} \underbrace{\boldsymbol{G}_2[i_2, j_2]}_{r \times r} \ldots \underbrace{\boldsymbol{G}_d[i_d, j_d]}_{r \times 1}$$

Notation & terminology:

- $\boldsymbol{W} \in \mathbb{R}^{M \times N}$, $M = m^d$, $N = n^d$;
- $i_k \in \{1, \ldots, m\}$, $j_k \in \{1, \ldots, n\}$;
- $\boldsymbol{G}_k$ — TT-cores;
- r — TT-rank;

The TT-format exists for any matrix $\boldsymbol{W}$ and uses $O(dmnr^2)$ memory to store $m^d n^d$ elements. Efficient only if the TT-rank is small.

1 Neural networks

2 Matrix formats

3 TensorNet

4 Experiments

Tensor Train layer

Input is a N-dimensional vector $\boldsymbol{x}$, output is a M-dimensional vector $\boldsymbol{h}$:

$$\boldsymbol{h} = \boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}.$$

$\boldsymbol{W}$ is represented in the TT-format:

$$\boldsymbol{h}(i_1, \ldots, i_d) = \sum_{j_1, \ldots, j_d} \boldsymbol{G}_1[i_1, j_1] \ldots \boldsymbol{G}_d[i_d, j_d]\, \boldsymbol{x}(j_1, \ldots, j_d) + \boldsymbol{b}(i_1, \ldots, i_d)$$

The parameters are the vector $\boldsymbol{b}$ and the TT-cores $\{\boldsymbol{G}_k\}_{k=1}^{d}$

Tensor Train layer learning

$$g(\boldsymbol{x}) = \sum_{j_1,\ldots,j_d} \boldsymbol{G}_1[i_1, j_1] \ldots \boldsymbol{G}_d[i_d, j_d]\, \boldsymbol{x}(j_1, \ldots, j_d) + \boldsymbol{b}(i_1, \ldots, i_d)$$

Instead of optimizing over all possible matrices $\boldsymbol{W}$:

$$\boldsymbol{W}^*, \boldsymbol{b}^* = \arg\min_{\boldsymbol{W},\boldsymbol{b}} \sum_q (y_q - g(\boldsymbol{x}_q))^2,$$

we optimize over the matrices representable in the TT-format with rank r:

$$\boldsymbol{G}_1^*, \ldots, \boldsymbol{G}_d^*, \boldsymbol{b}^* = \arg\min_{\boldsymbol{G}_1,\ldots,\boldsymbol{G}_d,\boldsymbol{b}} \sum_q (y_q - g(\boldsymbol{x}_q))^2.$$

Tensor Train layer: the Jacobian

The Jacobian of the linear transformation:

$$\boldsymbol{h}(\boldsymbol{i}) = \sum_{\boldsymbol{j}} \boldsymbol{G}_1[i_1, j_1] \dots \boldsymbol{G}_k[i_k, j_k] \dots \boldsymbol{G}_d[i_d, j_d]\, \boldsymbol{x}(\boldsymbol{j}) + \boldsymbol{b}(\boldsymbol{i}).$$

$$\frac{\partial \boldsymbol{h}(\boldsymbol{i})}{\partial \boldsymbol{G}_k[i_k, j_k]} = \sum_{\boldsymbol{j}^{\setminus k}} \overbrace{\boldsymbol{G}_1[i_1, j_1] \dots}^{\mathsf{r} \times 1} \cancel{\boldsymbol{G}_k[i_k, j_k]} \overbrace{\dots \boldsymbol{G}_d[i_d, j_d]}^{1 \times \mathsf{r}}\, \boldsymbol{x}(\boldsymbol{j}) =$$

$$\sum_{\boldsymbol{j}^{\setminus k,d}} \boldsymbol{G}_1[i_1, j_1] \dots \cancel{\boldsymbol{G}_k[i_k, j_k]} \dots \boldsymbol{G}_{d-1}[i_{d-1}, j_{d-1}]$$

$$\underbrace{\sum_{j_d} \boldsymbol{G}_d[i_d, j_d]\, \boldsymbol{x}(\boldsymbol{j})}_{\mathsf{r} \times m n^{d-1}}$$

1 Neural networks

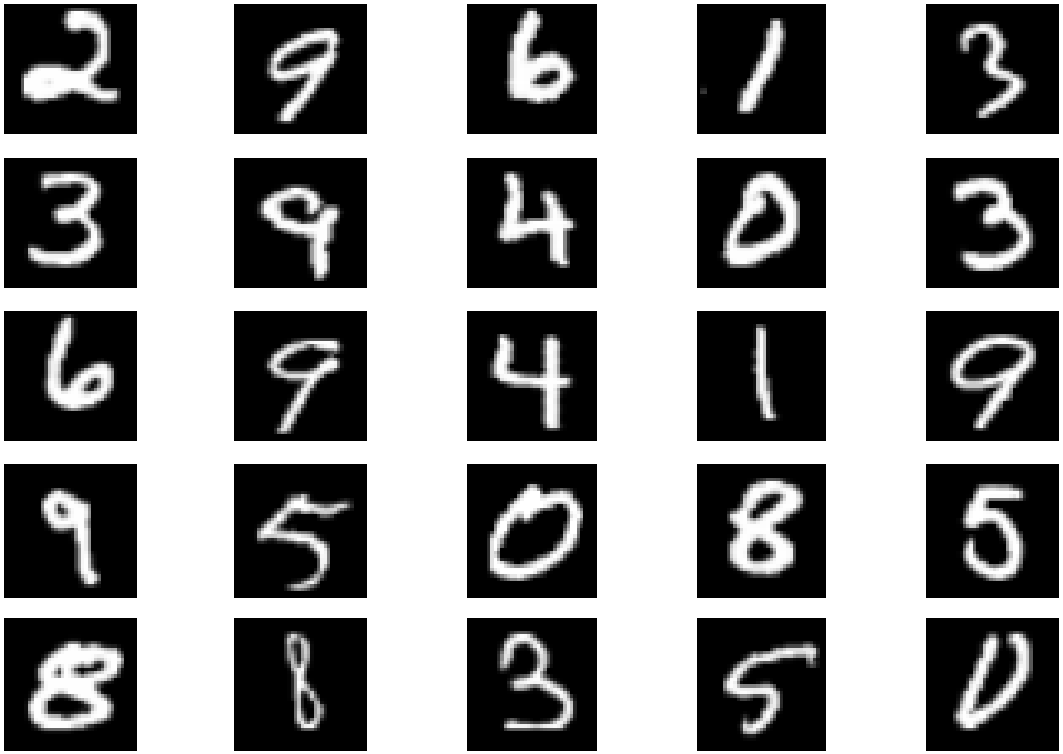
2 Matrix formats

3 TensorNet

4 Experiments

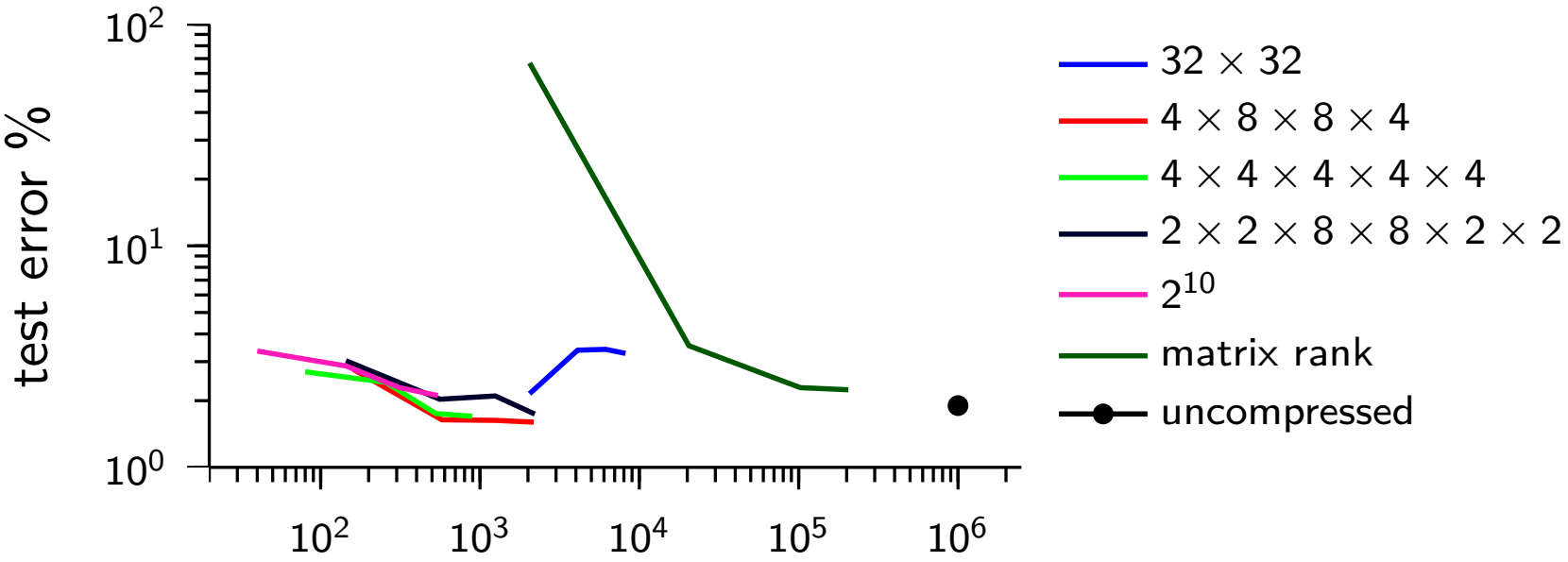
Mnist

Hadwritten digits recognition



Mnist cont'd

A two layered neural network with 1024×1024 and 1024×10 matrices.



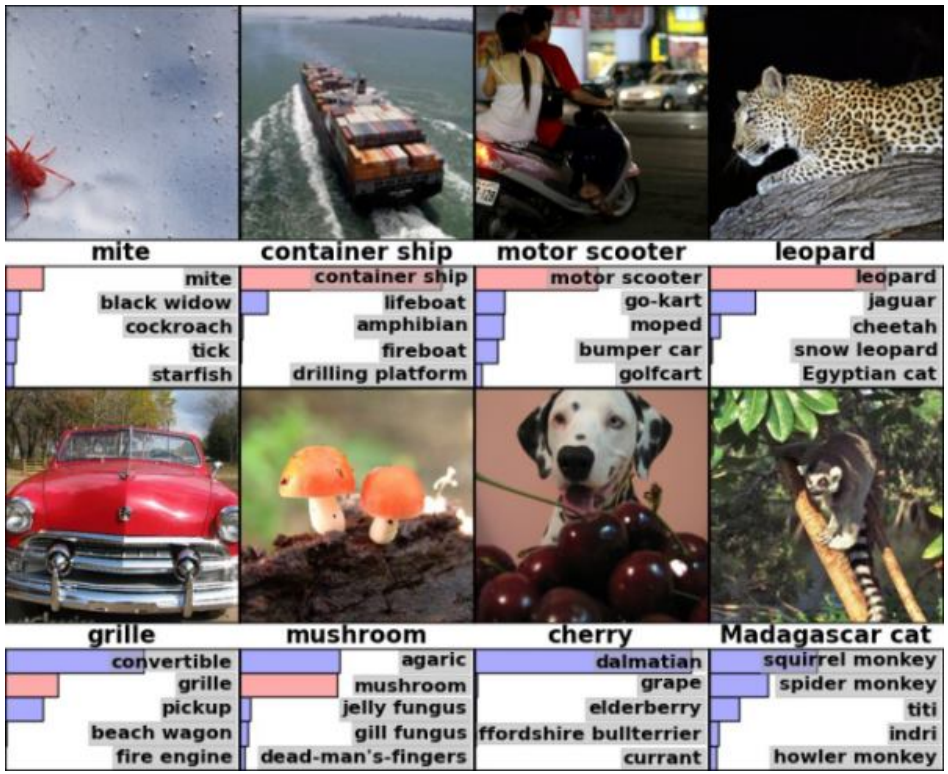
Speed

Type	1 im. time (ms)	100 im. time (ms)
CPU fully-connected layer	16.09	97.2
CPU TT-layer	1.24	94.7
GPU fully-connected layer	2.7	33
GPU TT-layer	1.92	12.86

$25\,088 \times 4\,096$ layer, the fully-connected layer uses 392MB and the TT-layer uses 0.766MB.

Image classification

Cifar & ImageNet



We used a $262\,144 \times 4\,096$ TT-matrix to outperform other plain (non-convolutional) neural networks on Cifar.

ImageNet

Architecture	Matrices compr.	Network compr.	Error
FC FC FC	1	1	11.2
TT4 FC FC	50 972	3.9	11.2
TT2 FC FC	194 622	3.9	11.5
TT1 FC FC	713 614	3.9	12.8
TT4 TT4 FC	37 732	7.4	12.3
LR1 FC FC	3 521	3.9	97.6
LR5 FC FC	704	3.9	53.9
LR50 FC FC	70	3.7	14.9

A 3-layered network, FC stands for the traditional layer; TT□ stands for the TT-layer with all the TT-ranks equal "□"; LR□ stands for the rank decomposition with the rank equal "□".

References I

Oseledets, I. V. (2011). "Tensor-Train Decomposition". In: *SIAM J. Scientific Computing* 33.5, pp. 2295–2317.