



MNIST Training for BNN

Jack Diep, Florian Köhler, Yannick Naumann

September 6, 2021

Design Your Own CPU - Design of Embedded Systems

Content

1. Neural Networks

- What is a neural network?
- Training
- Our Goal

2. BNN Design

- The Network
- Layers
- Binarization of Input data

3. BNN Training Analysis

- Layer Analysis
- Parameter Analysis

What is a neural network?

What is a neural network?

- The heart of deep learning

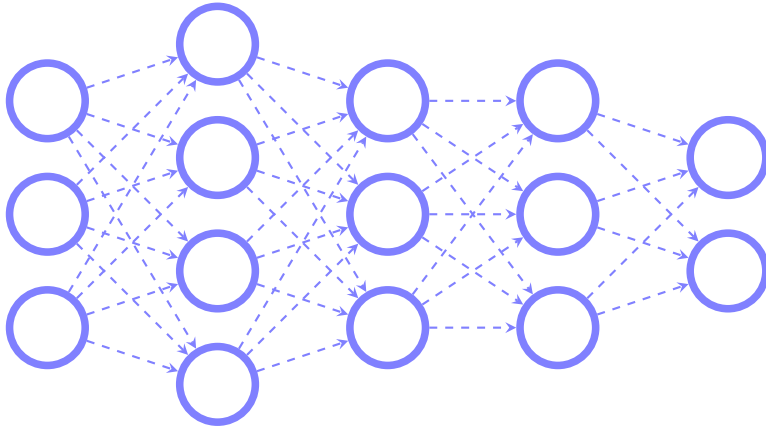
What is a neural network?

- The heart of deep learning
- Classify given data
e.g. speech or image recognition

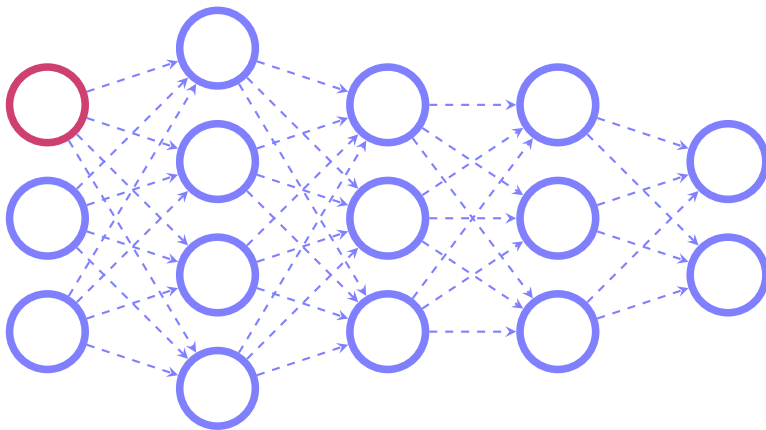
What is a neural network?

- The heart of deep learning
- Classify given data
e.g. speech or image recognition
- Rely on training data

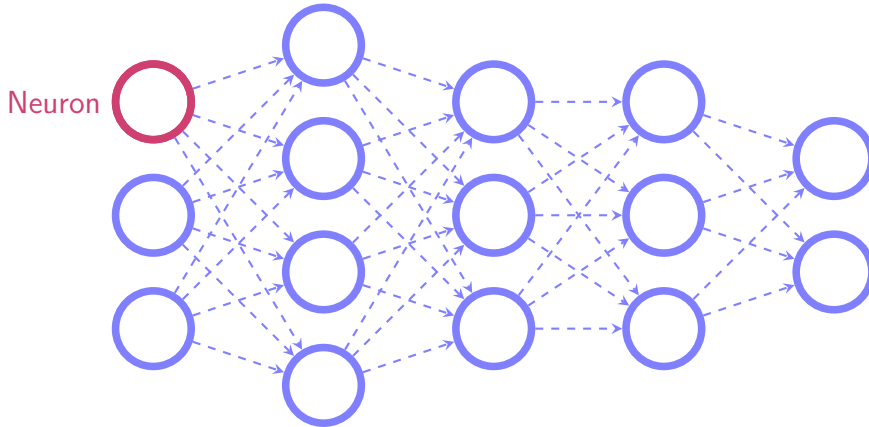
What is a neural network?



What is a neural network?



What is a neural network?



Neuron

- Holds a single value $v \in V_L$



Neuron

Neuron

- Holds a single value $v \in V_L$
- Semantics depend on class of layer



Neuron

Layer

- Layer of neurons



Layer

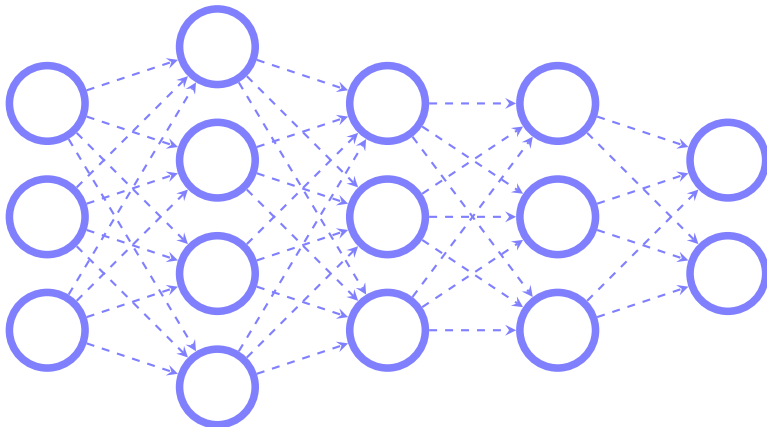
Layer

- Layer of neurons
- Three types:
 - Input layer: *Network input neurons*
 - Hidden layer: *Feature neurons*
 - Output layer: *Network output neurons*

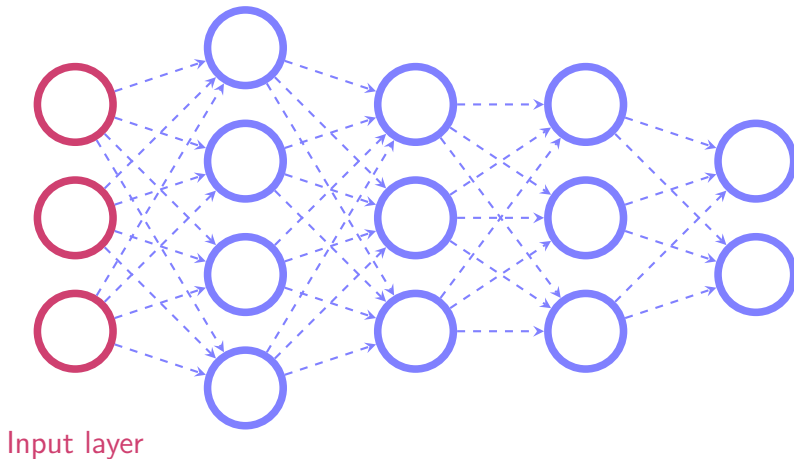


Layer

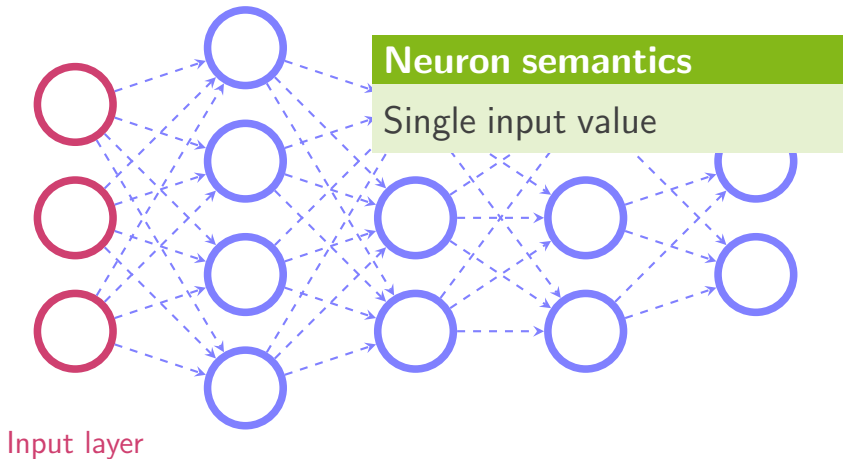
Classes of layers



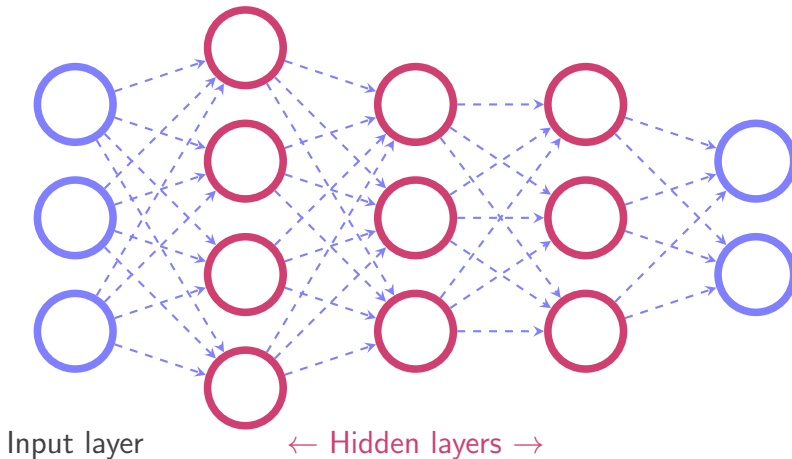
Classes of layers



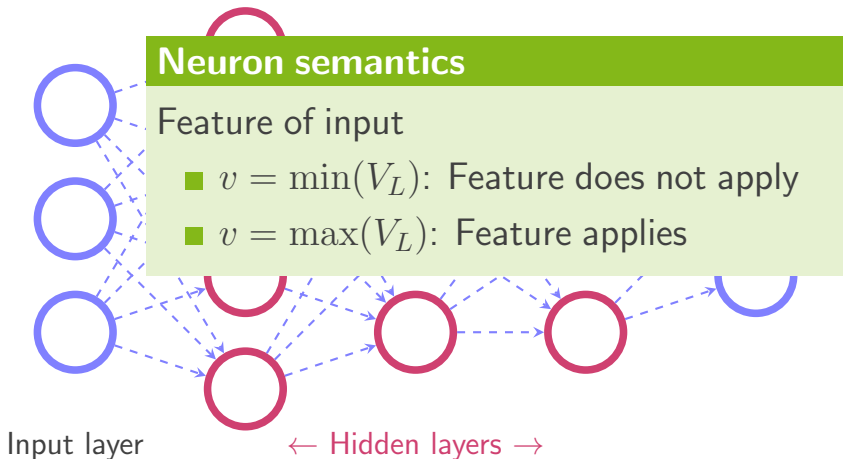
Classes of layers



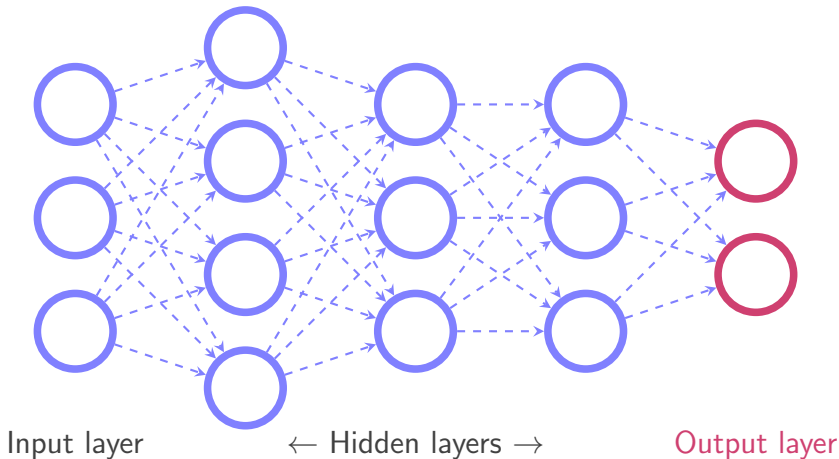
Classes of layers



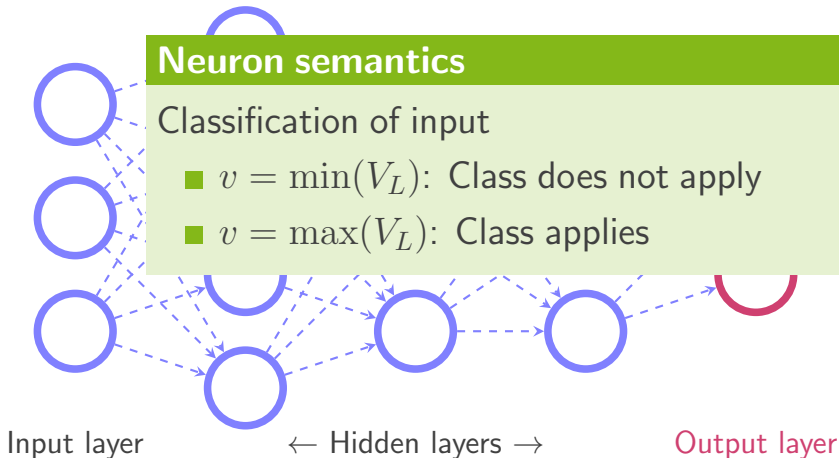
Classes of layers



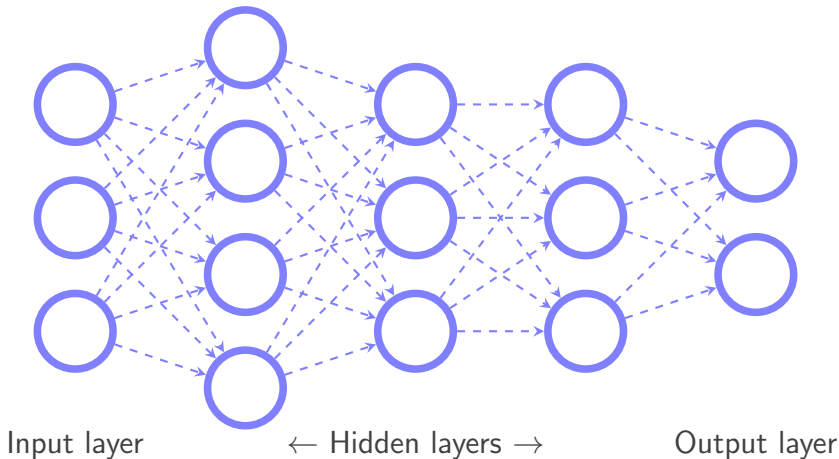
Classes of layers



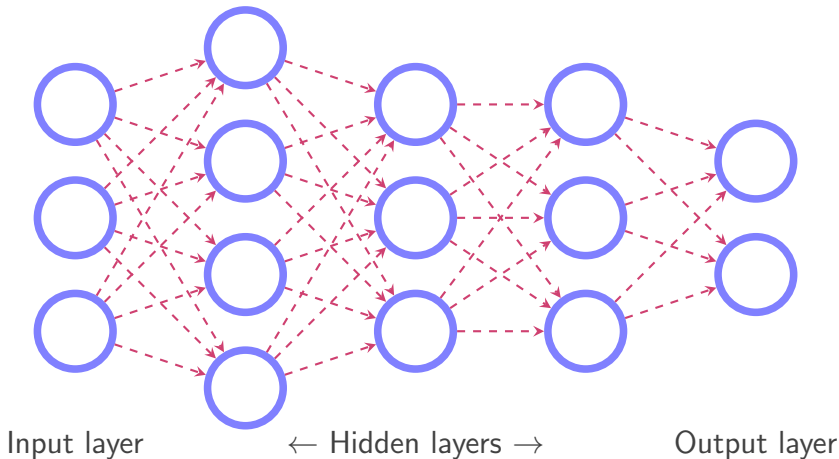
Classes of layers



Classes of layers

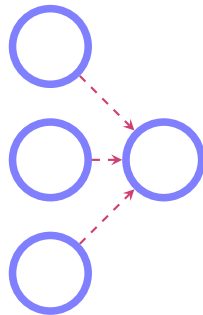


Classes of layers



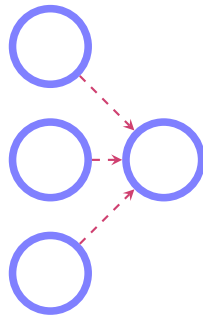
Edges

- Connects all neurons between subsequent layers



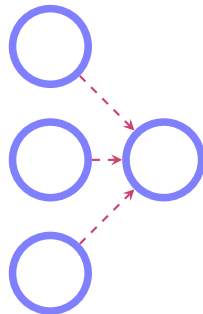
Edges

- Connects all neurons between subsequent layers
- Weighted



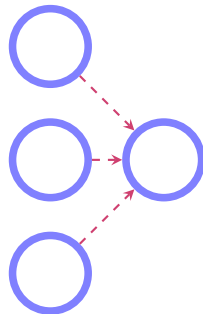
Edges

- Connects all neurons between subsequent layers
- Weighted
- Semantics:
Higher weight
→ higher feature significance



Edges

- Connects all neurons between subsequent layers
- Weighted
- Semantics:
Higher weight
→ higher feature significance
- **Training: Optimize weights!**



Training

Training (Cycle)

1. Input data

Training (Cycle)

1. Input data
2. Run the network

Training (Cycle)

1. Input data
2. Run the network
3. Compare output with expected values
→ Calculate error ($|v - \text{expected}|$)

Training (Cycle)

1. Input data
2. Run the network
3. Compare output with expected values
→ Calculate error ($|v - \text{expected}|$)
4. Run error back through network, adjust weights

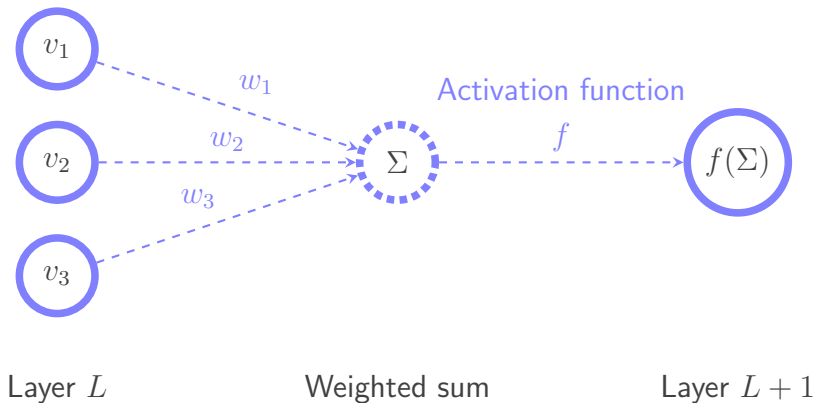
Training (Cycle)

1. Input data ✓
2. Run the network
3. Compare output with expected values
→ Calculate error ($|v - \text{expected}|$)
4. Run error back through network, adjust weights

Training (Cycle)

1. Input data ✓
2. Run the network?
3. Compare output with expected values
→ Calculate error ($|v - \text{expected}|$)
4. Run error back through network, adjust weights

Run the network



Training (Cycle)

1. Input data ✓
2. Run the network ✓
3. Compare output with expected values
→ Calculate error ($|v - \text{expected}|$)
4. Run error back through network, adjust weights

Training (Cycle)

1. Input data ✓
2. Run the network ✓
3. Compare output with expected values
→ Calculate error ($|v - \text{expected}|$) ✓
4. Run error back through network, adjust weights

Training (Cycle)

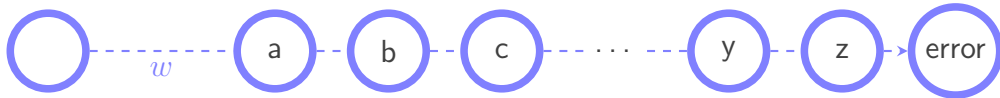
1. Input data ✓
2. Run the network ✓
3. Compare output with expected values
→ Calculate error ($|v - \text{expected}|$) ✓
4. Run error back through network, adjust weights?

Adjusting weights

Backpropagation

Calculate change of error when adjusting some weight

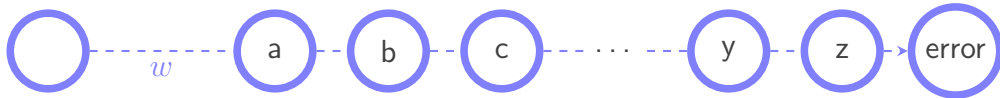
→ *Slope*



Adjusting weights

Backpropagation

Calculate change of error when adjusting some weight
→ *Slope*

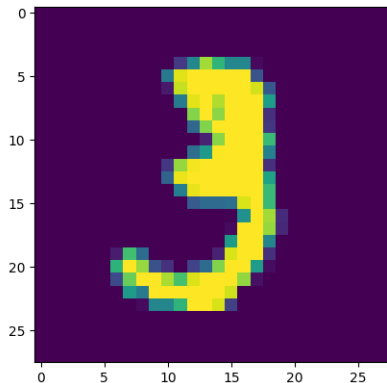


Chain rule

$$\frac{\delta \text{error}}{\delta w} = \frac{\delta a}{\delta w} \cdot \frac{\delta b}{\delta a} \cdot \frac{\delta c}{\delta b} \cdot \dots \cdot \frac{\delta z}{\delta y} \cdot \frac{\delta \text{error}}{\delta z}$$

Our Goal

- Create a BNN in PyTorch
- Image recognition on MNIST-Dataset
- Keep an accuracy of at least 90%
- Export trained BNN



1. Neural Networks

- What is a neural network?
- Training
- Our Goal

2. BNN Design

- The Network
- Layers
- Binarization of Input data

3. BNN Training Analysis

- Layer Analysis
- Parameter Analysis

The Network

Binarized Linear (500)

Batch Norm

Activation

Binarized Linear (1024)

Batch Norm

Activation

Binarized Linear (1024)

Batch Norm

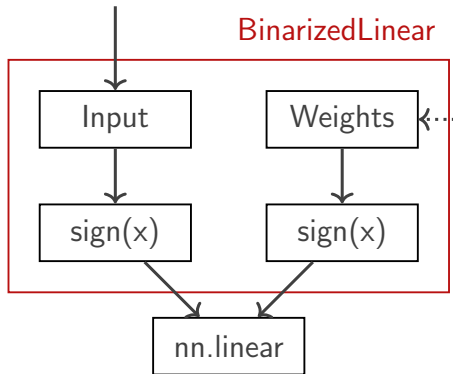
Activation

Linear (1024)

softmax

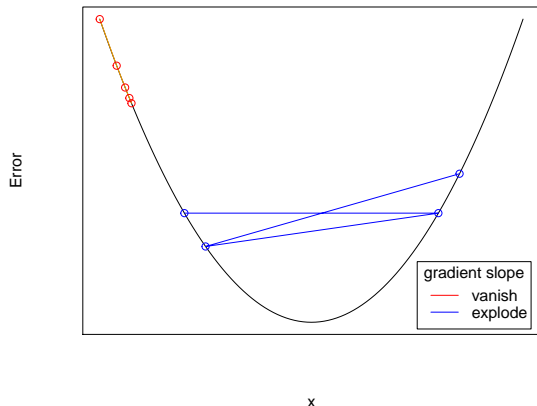
Binarisation of Linear Layer

- binarisation of weights
- binarisation of input data for hidden layers
- calculation through *nn.linear*

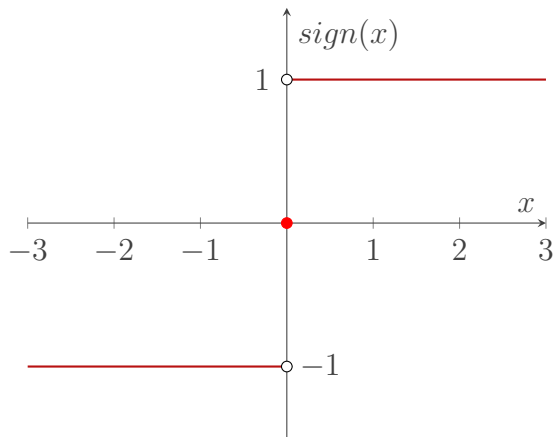
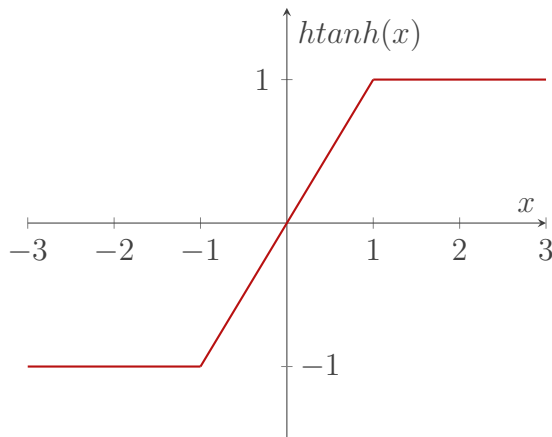


Batch Norm (BN)

- Batches = data calculated in parallel
- In NN
 - normalize batches
 - mean 0
 - standard derivation 1
- In BNN
 - prevent *explosing gradient*



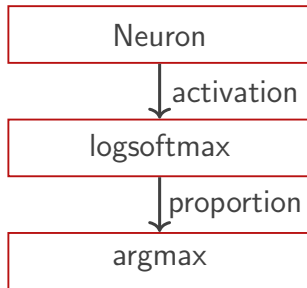
Activation



Evaluation of last layer

- normalisation of activation
- decision of the network

$$\text{LogSoftmax}(x_i) = \log\left(\frac{e^{x_i}}{\sum_j e^{x_j}}\right)$$



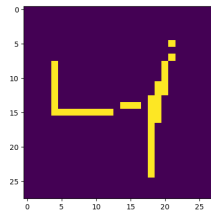
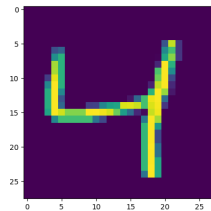
Binarization of Input data

Binarization of Input data

- Mapping 255 values to 0,1
- minimize accuracy losses
- 2 approaches
 - Threshold
 - Probability

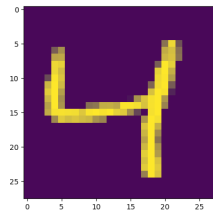
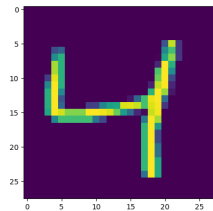
Threshold-Binarization

- define static threshold
- filter pixel-array via:
 $\text{pixel} > \text{threshold}$



Probability-Binarization

- each pixelvalue dictates its prob for being 1
- binarize same trainingset multiple times
 - Run each epoche with all trainingsets



Comparison Threshold, Prob

■ Threshold

- Using integrated tensor-functions
- 150ms per iteration
- Convergence after approx. 100 epochs

■ Probability

- Iterate through tensor manually
- 250ms per iteration
- Convergence after approx. 20*30 iterations

Evaluating Accuracy-Loss

Run	Non-Binarized	threshold	prob
1	91.99%	89.24%	92.52%
2	91.99%	89.24%	91.82%
3	91.99%	89.24%	92.56%
4	91.99%	89.24%	91.02%
avg	91.99%	89.24%	91.98%

- 600 epochs for threshold, default
- 20 epochs, 30 trainingsets for prob

1. Neural Networks

- What is a neural network?
- Training
- Our Goal

2. BNN Design

- The Network
- Layers
- Binarization of Input data

3. BNN Training Analysis

- Layer Analysis
- Parameter Analysis

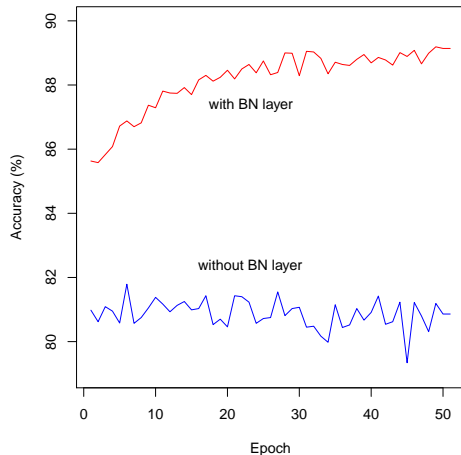
Consequences of linear layer binarisation

Run	binary	normal
1	88.29%	97.43%
2	87.32%	96.98%
3	87.19%	97.2%

- training for 50 epochs
- mean loss of 9,6%
- loss in granularity

Effect of Batch Norm

- 7.4% improved peak performance
- Less jitter with BN
- Reduced exploding gradient



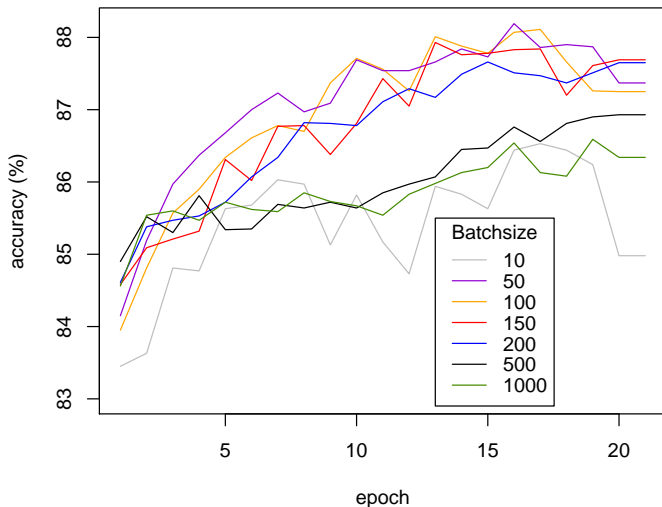
Parameter Analysis

Batch size

- frequency of error calculation
- normalisation though Batch Norm
- rate of parallelization

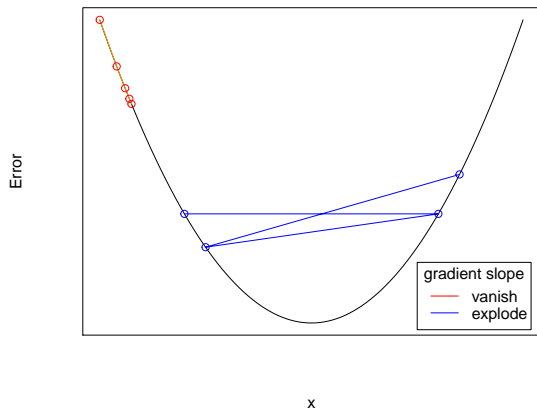
Batchsize	Time (s)
10	30,68
50	11,33
100	8,76
150	7,95
200	7,63
500	6,66
1000	6,39

Evaluation of Batch size



Learning rate

- higher value \rightarrow more weights are updated
- balance between vanishing- and exploding gradient



Evaluation learning rate

