technische universität
dortmund

MNIST Training for BNN

Jack Diep, Florian Köhler, Yannick Naumann

September 5, 2021

Design Your Own CPU - Design of Embedded Systems

Content

1. **Neural Networks**
   - What is a neural network?
   - Training
2. **BNN Design**
3. **BNN Training Analysis**
   - Layer Analysis
   - Parameter Analysis

What is a neural network?

What is a neural network?

- The heart of deep learning

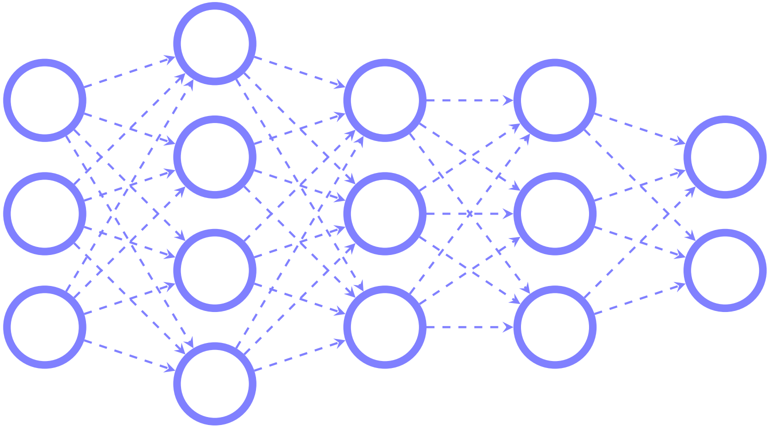
What is a neural network?

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

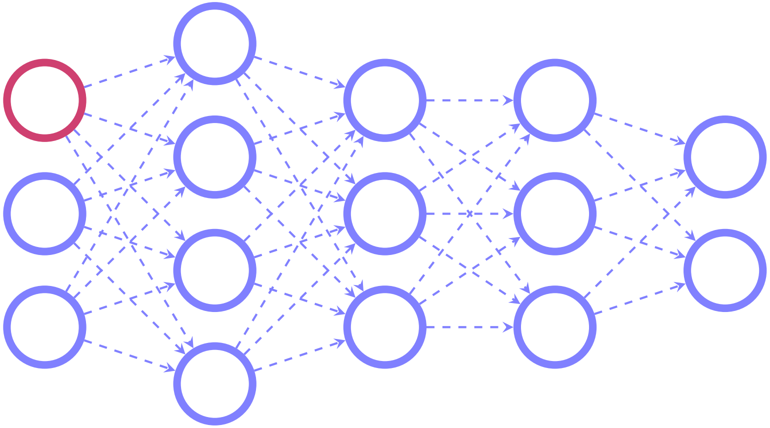
What is a neural network?

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

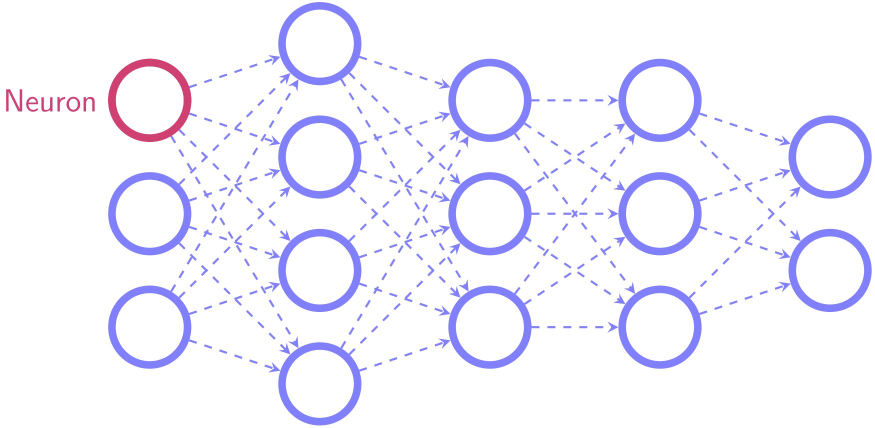
What is a neural network?



What is a neural network?

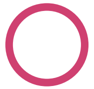


What is a neural network?



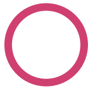
Neuron

- Holds a single value $v \in V_L$



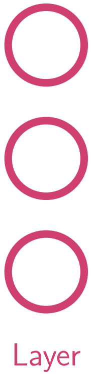
Neuron

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



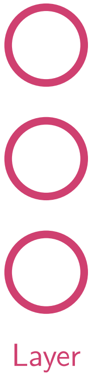
Layer

- Layer of neurons

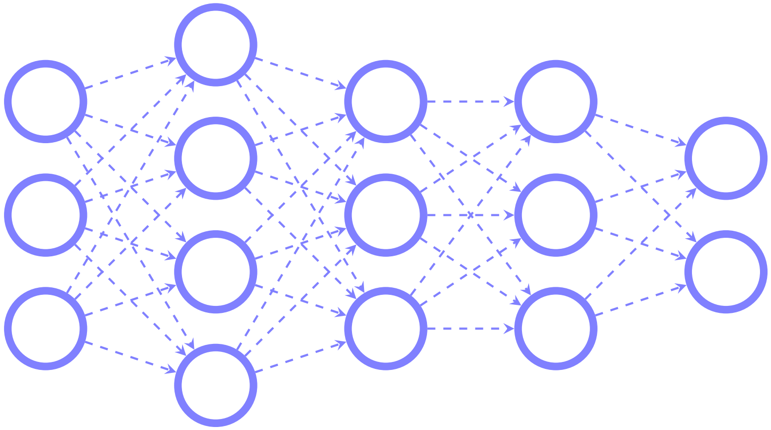


Layer

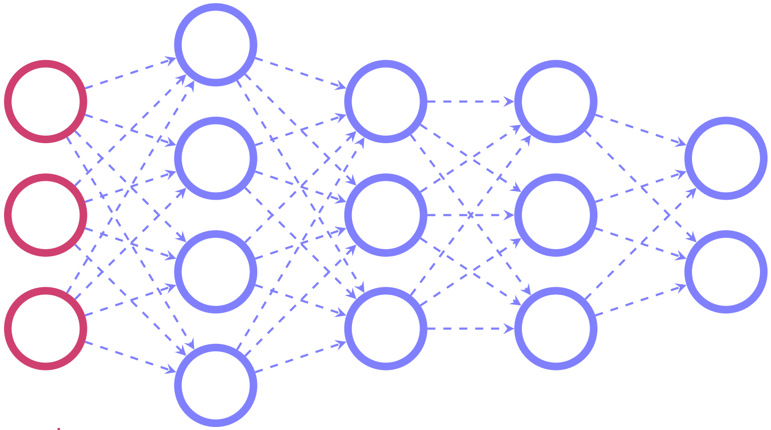
- Layer of neurons
- Three types:
  - Input layer: *Network input neurons*
  - Hidden layer: *Feature neurons*
  - Output layer: *Network output neurons*



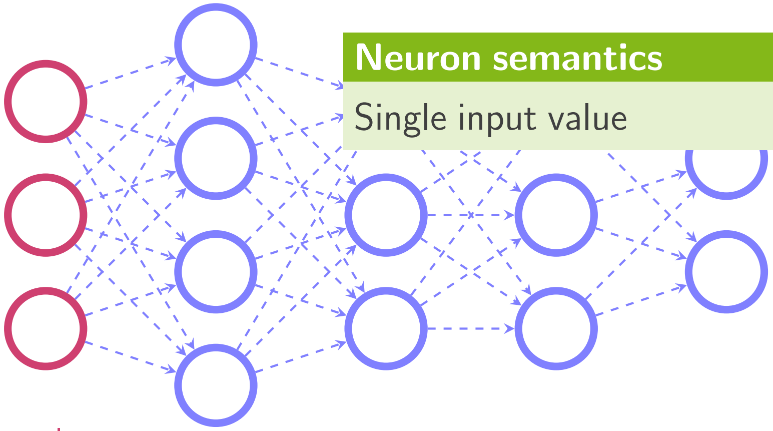
Classes of layers



Classes of layers


Input layer

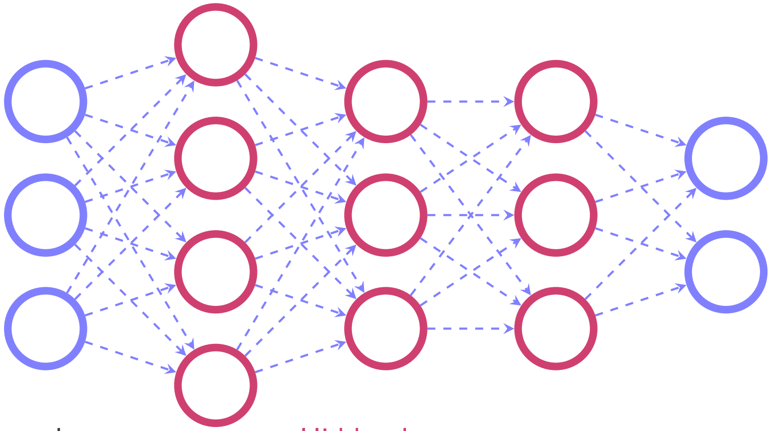
Classes of layers



Neuron semantics

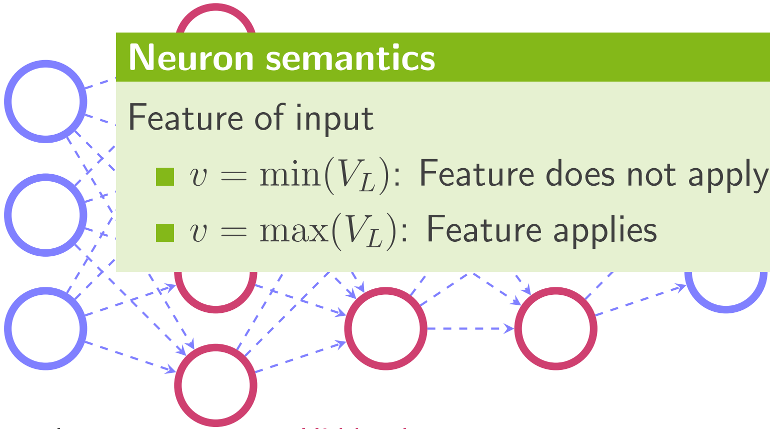
Single input value

Classes of layers



Input layer ← Hidden layers →

Classes of layers

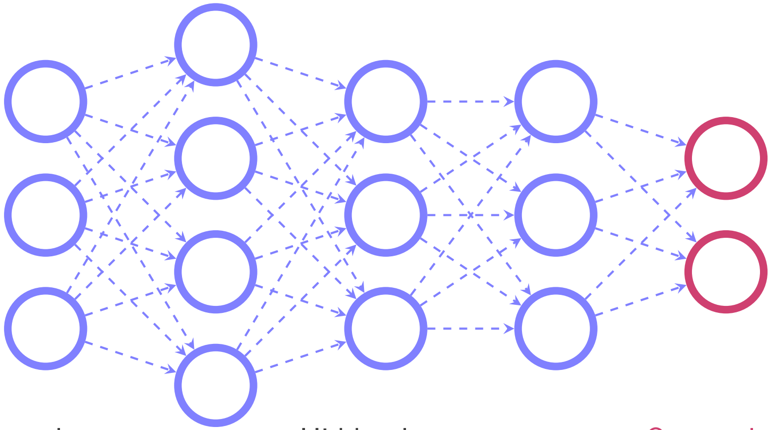


Neuron semantics

Feature of input

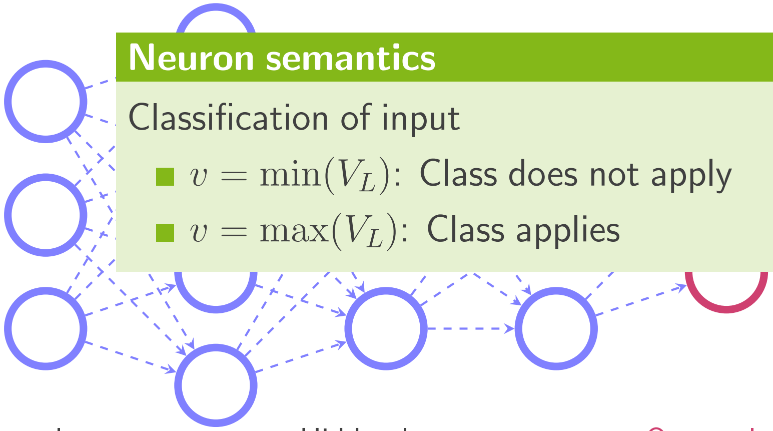
- $v = \min(V_L)$: Feature does not apply
- $v = \max(V_L)$: Feature applies

Classes of layers



Input layer ← Hidden layers → Output layer

Classes of layers

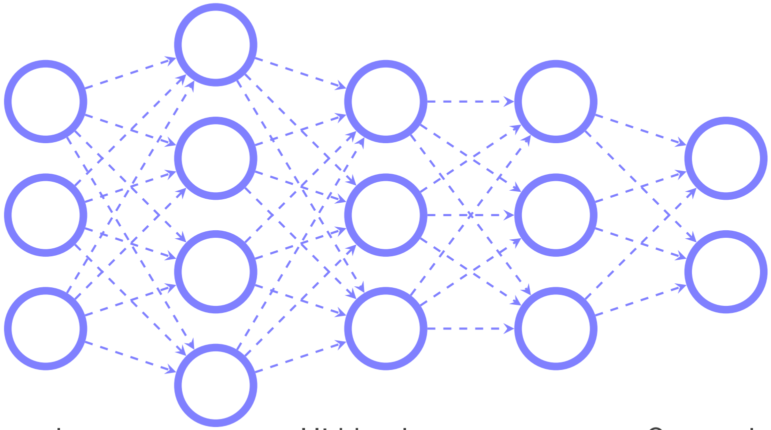


Neuron semantics

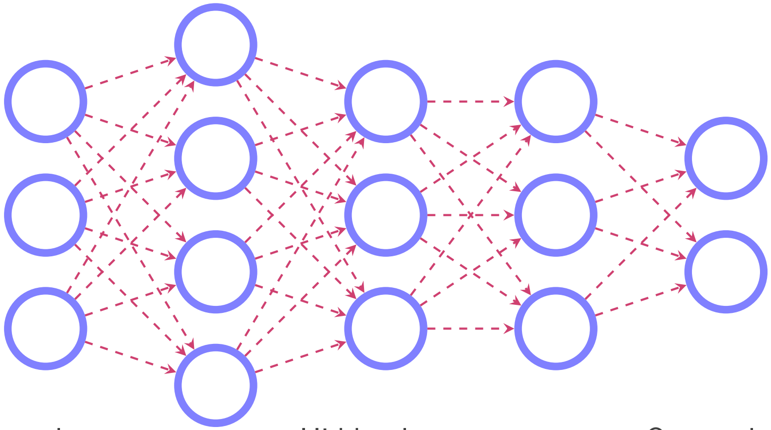
Classification of input

- $v = \min(V_L)$: Class does not apply
- $v = \max(V_L)$: Class applies

Classes of layers

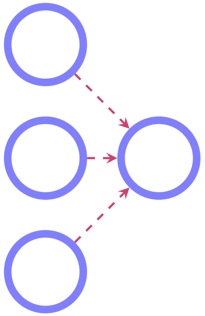


Classes of layers



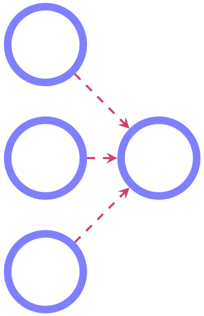
Edges

- Connects all neurons between subsequent layers



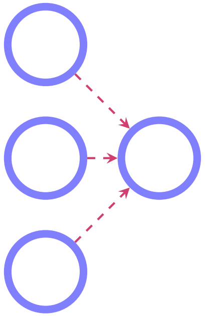
Edges

- Connects all neurons between subsequent layers
- Weighted



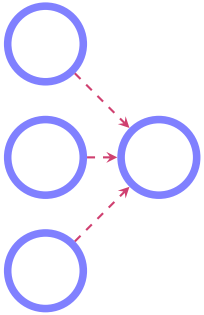
Edges

- Connects all neurons between subsequent layers
- Weighted
- Semantics:
  Higher weight
  $\rightarrow$ higher feature significance



Edges

- Connects all neurons between subsequent layers
- Weighted
- Semantics:
  Higher weight
  $\rightarrow$ 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
   $\rightarrow$ Calculate error ($|v - \text{expected}|$)

Training (Cycle)

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

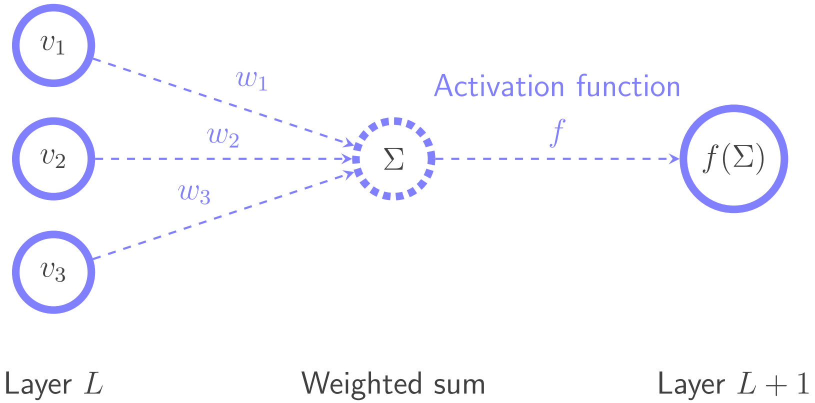
Training (Cycle)

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

Training (Cycle)

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

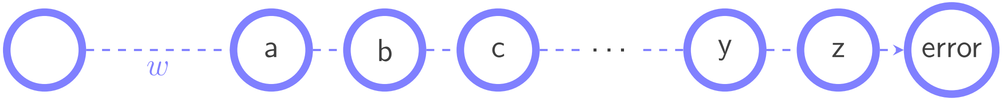
Training (Cycle)

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

Adjusting weights

Backpropagation

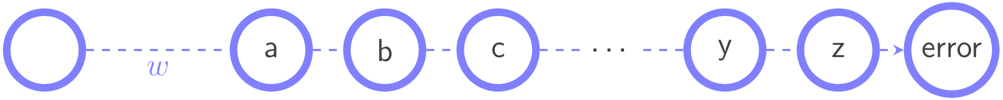
Calculate change of error when adjusting some weight
$\rightarrow$ *Slope*



Adjusting weights

Backpropagation

Calculate change of error when adjusting some weight
$\rightarrow$ *Slope*



Chain rule

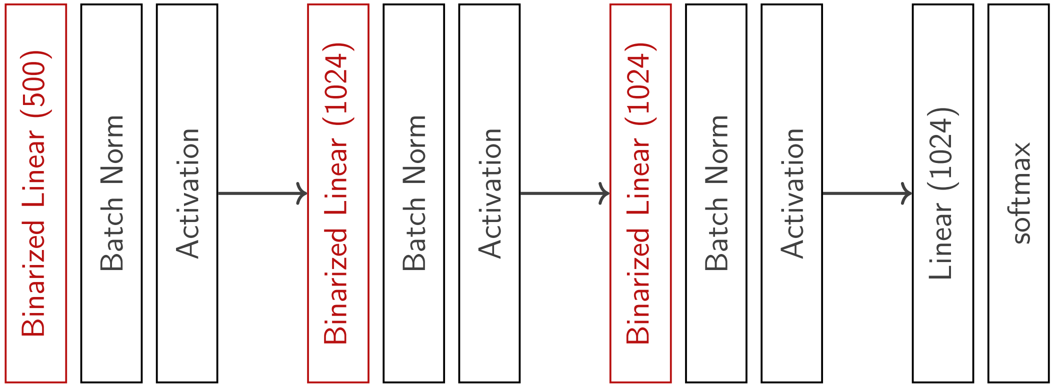
$$\frac{\delta \text{error}}{\delta w} = \frac{\delta a}{\delta w} \cdot \frac{\delta b}{\delta a} \cdot \frac{\delta c}{\delta b} \cdot \ldots \cdot \frac{\delta z}{\delta y} \cdot \frac{\delta \text{error}}{\delta z}$$

1. **Neural Networks**
   - What is a neural network?
   - Training

2. **BNN Design**

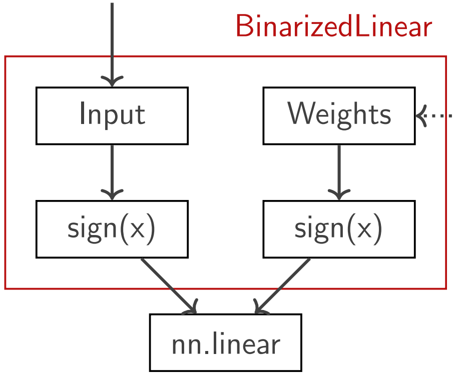
3. **BNN Training Analysis**
   - Layer Analysis
   - Parameter Analysis

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

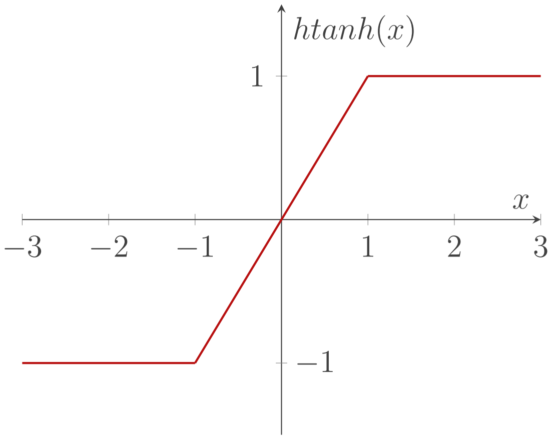
- binarisation of weights
- binarisation of input data for hidden layers
- calculation through *nn.linear*

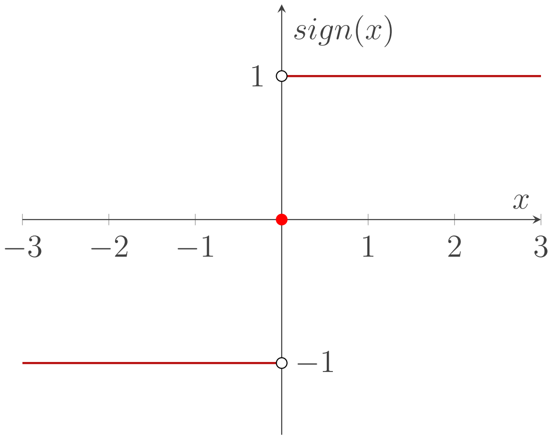


Batch Norm (BN)

- In NN
  - normalize batches
  - mean 0
  - standard derivation 1
- In BNN
  - prevent *expolding gradient*

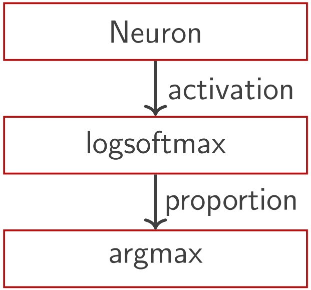
Activation





Evaluation of last layer

- normalisation of activation
- decision of the network



1. **Neural Networks**
- What is a neural network?
- Training

2. **BNN Design**

3. **BNN Training Analysis**
- Layer Analysis
- Parameter Analysis

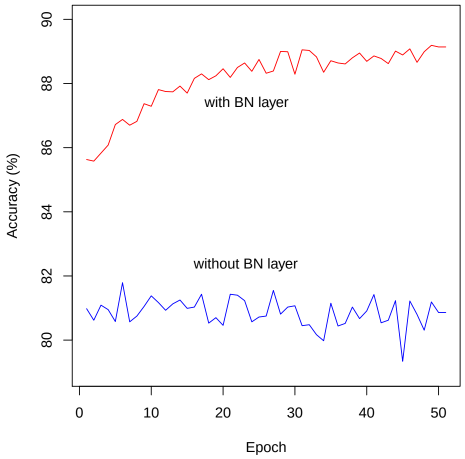
Consequences of linear layer binarisation

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 expoldging gradient

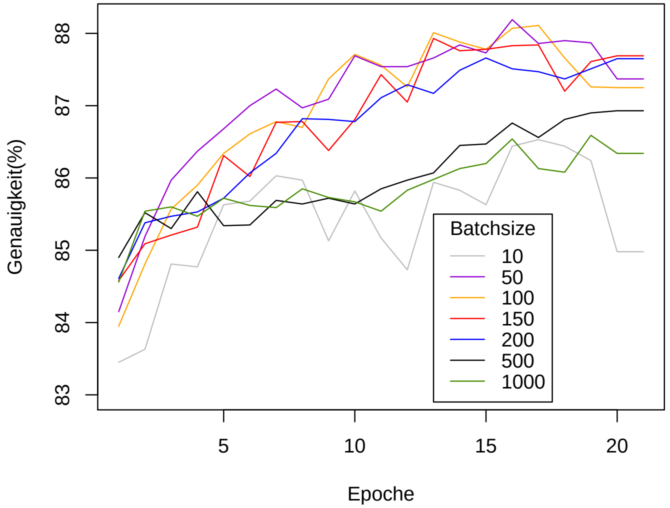


Batch size

- frequency of error calculation
- rate of parallelization

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

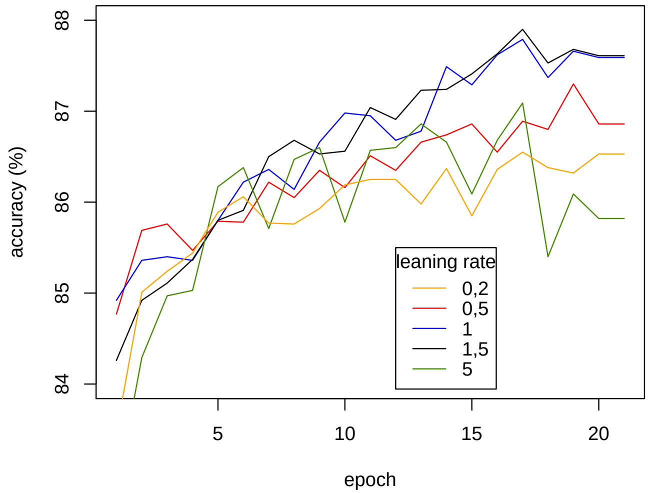
Evaluation of Batch size



Learning rate

- higher value $\rightarrow$ more weights are updated
- balance between over- and underfitting

evaluation learning rate


accuracy (%)
epoch
leaning rate