TU Clausthal

A Convolutional Neural Network for Image Classification of Cats and Dogs

Final presentation

Aditya Raj, Sören Schleibaum
Institut für Informatik

INTRODUCTION

Structure

1. Introduction

1 → 2 → 3 → 4 → 5 → 6 → 7 → 8

Structure

1. Introduction
2. Neural Networks (NN)
3. Math behind NN
4. Convolutional NN (CNN)
5. Problem
6. Design
7. Evaluation
8. Summary

NEURAL NETWORKS

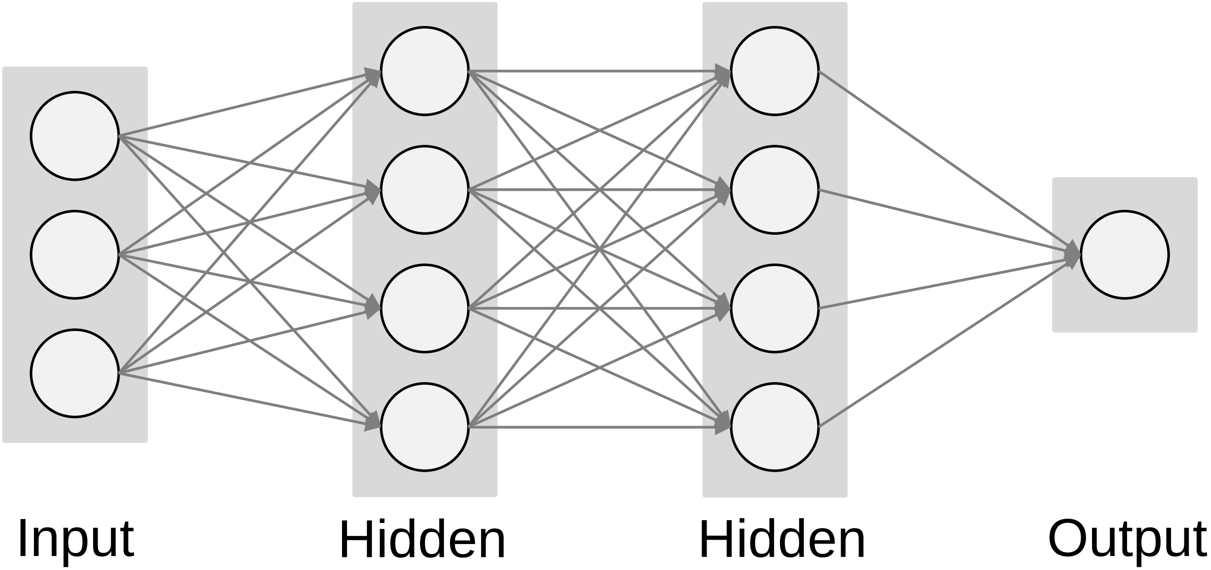
Structure

2. Neural Networks

- Overview
- Concept of Neurons

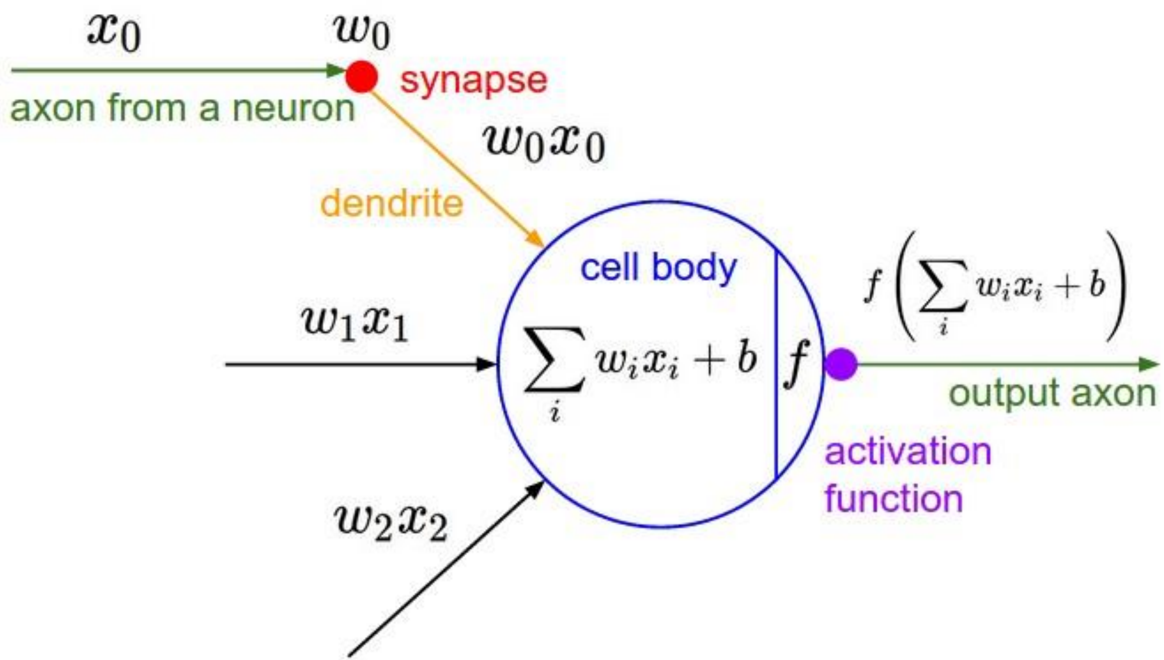
1 → **2** → 3 → 4 → 5 → 6 → 7 → 8

Overview



http://cs231n.github.io/convolutional-networks/

Concept of Neurons



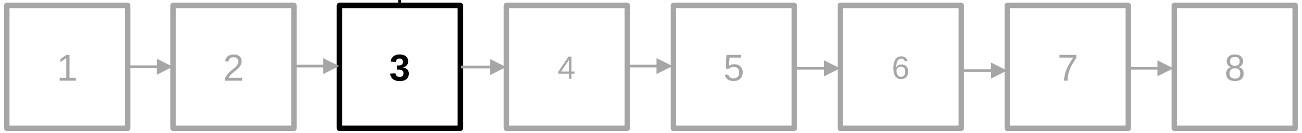
http://cs231n.github.io/convolutional-networks/

MATH BEHIND NEURAL NETS

Structure

3. Math behind NN

- Introduction
- Activation Function
- Backpropagation
- Loss Function
- Summarize



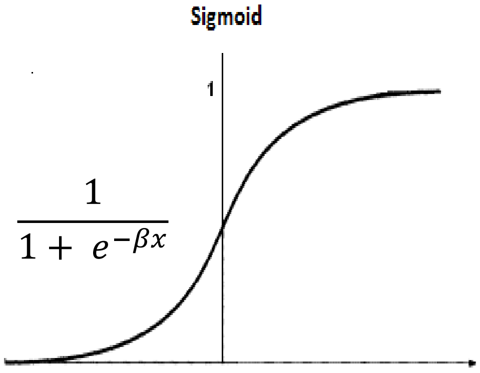
Introduction

- Actual output, weights
- Activation function
- Measure how much we missed (cost function)
- Multiply error by the Sigmoid slope
- Update weights (backpropagation)

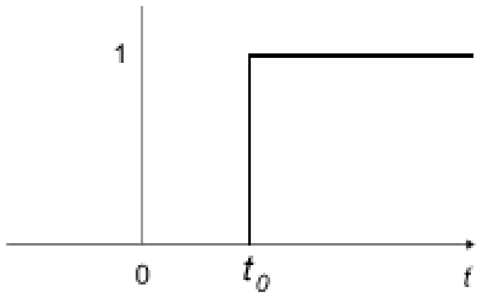
Activation Functions

- Non-Linear
  - Ex: Sigmoid, tanh

$$f(x) = \frac{1}{1 + e^{-\beta x}}$$



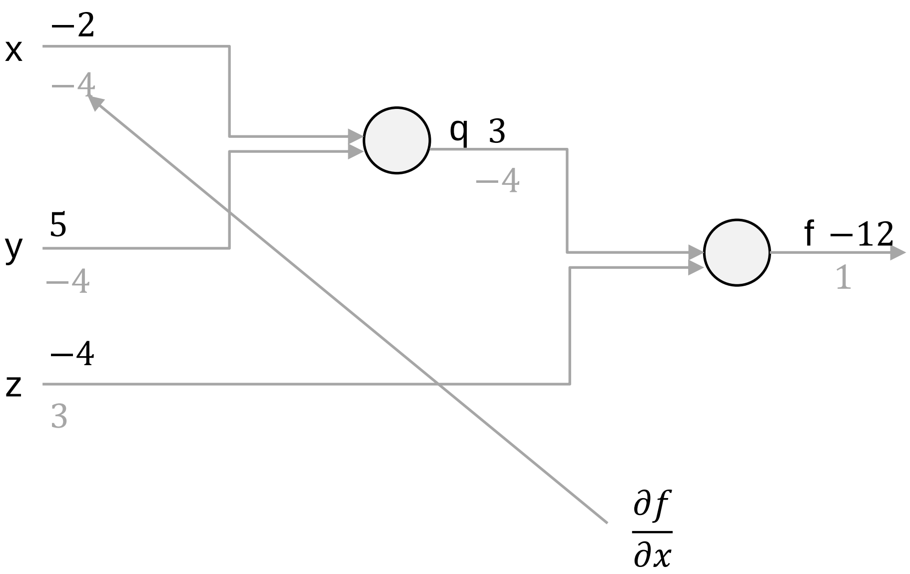
- Continuous but not everywhere differentiable function
  - Cons: descent gradient cannot be obtained
  - Ex: Relu



http://cs231n.github.io/convolutional-networks/

Back Propagation

- $f(x, y, z) = (x + y)z$
- $x = -2,\ y = 5, z = -4$
- $q = x + y,\ \frac{\partial q}{\partial x} = 1,\ \frac{\partial q}{\partial y} = 1$
- $f = qz, \quad \frac{\partial f}{\partial q} = z,\ \frac{\partial q}{\partial z} = q$
- Desired
  - $\frac{\partial f}{\partial x} = \frac{\partial f}{\partial q} \frac{\partial q}{\partial x}$
- Similarly propagate
  - $\frac{\partial f}{\partial y}$ and $\frac{\partial f}{\partial z}$



http://cs231n.github.io/convolutional-networks/

Loss Function

- Squared Error Measure
- SoftMax

Loss Function - Squared Error Measure

- $Error = \frac{1}{2}(Y_{actual} - Y_{predicted})^2$
- Drawbacks
  - No gradient to get from 0.000…1 to 1.
    - To do so it will take quite longer.
  - Deprives NN of probability information.

Summarize

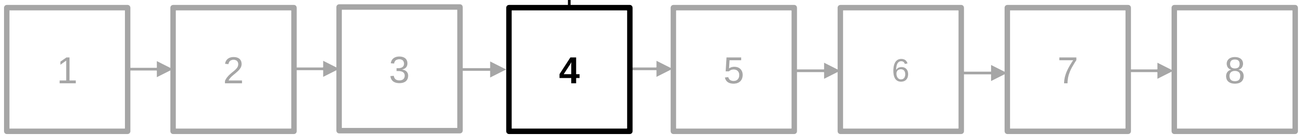
- Actual output, weights $l_0 = T_i, W = rand()$
- Activation function $l_1 = f(X_i \,.\, W)$
- Measure how much we missed (cost function) $Err = (l_0, l_1)$
- Multiply error by the Sigmoid slope $\Delta l_1 = \mathrm{Err} \times \Delta\big(f(Err)\big)$
- Update weights (backpropagation) $\mathrm{W} = \mathrm{W} + \alpha(l_0 . \Delta l_1)$

CONVOLUTIONAL NN

Structure

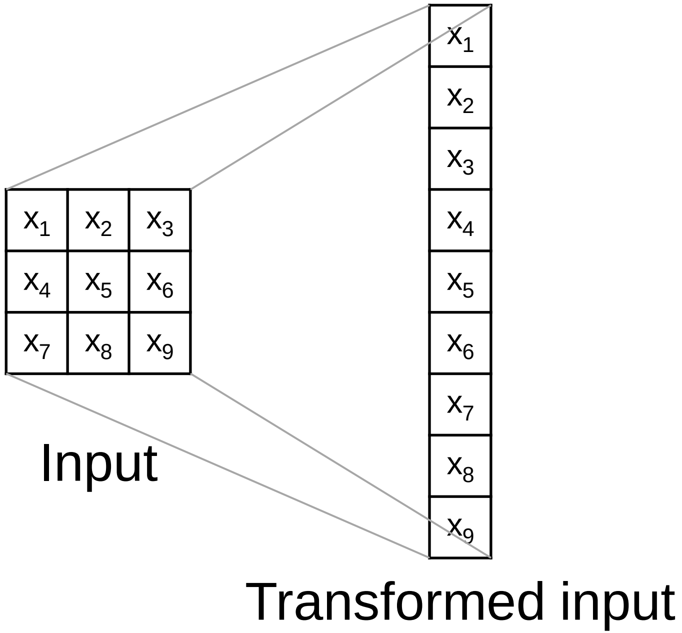
4. Convolutional NN

- Motivation
- Description of Layers
- Hyperparameters



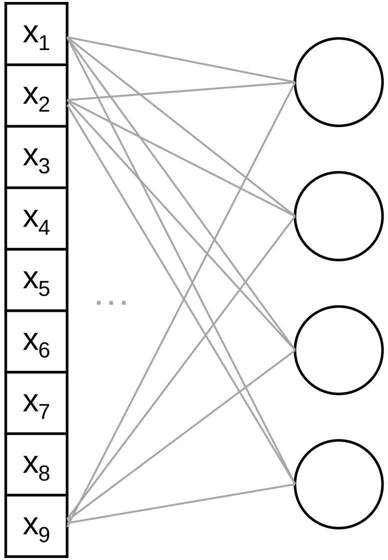
Motivation

- Number of parameters



Motivation

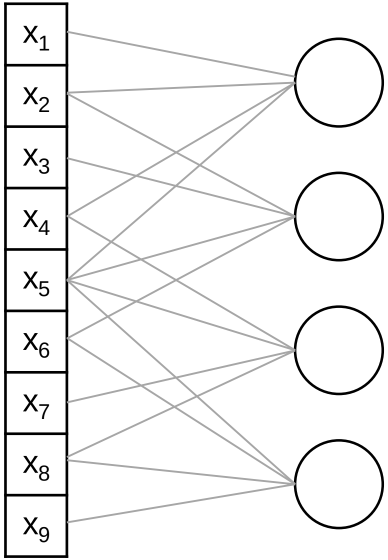
- NN
  - High number of params



Number of weights: 36

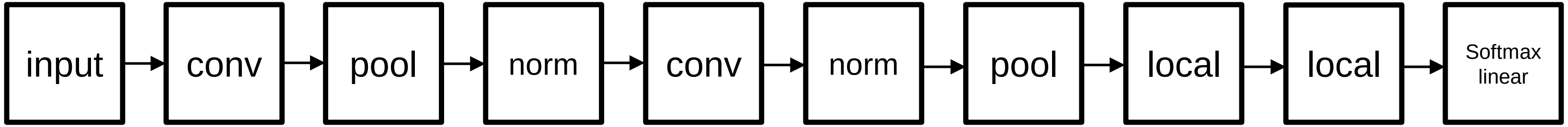
- CNN
  - Lower number of params

x_1	x_2	x_3
x_4	x_5	x_6
x_7	x_8	x_9



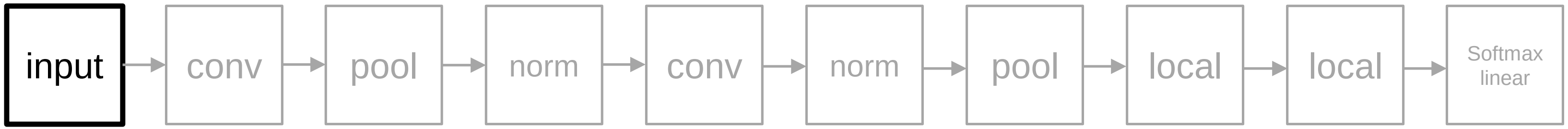
Number of weights: 4

Description of Layers



Input Layer

- Image cropping
- Distortions
  - Randomly flipping
  - Randomly changing brightness
  - Randomly changing contrast



Convolutional Layer - Filter

Input

0	0	0	0	0	0	0
0	2	1	0	0	2	0
0	2	2	2	0	2	0
0	1	2	0	2	0	0
0	2	1	2	**1**	**2**	**0**
0	1	2	1	**0**	**0**	**0**
0	0	0	0	**0**	**0**	**0**

Filter

0	0	0
0	2	1
0	2	2

Bias

1

Output

1

$$1*0 + 2*0 + 0*0 + 0*0 + 0*2 + 0*1 + 0*0 + 0*2 + 0*2 + 1 = \mathbf{1}$$

http://cs231n.github.io/convolutional-networks/

input → **conv** → pool → norm → **conv** → norm → pool → local → local → Softmax linear

Aditya Raj, Sören Schleibaum
Institut für Informatik

Convolutional Layer - Parameters

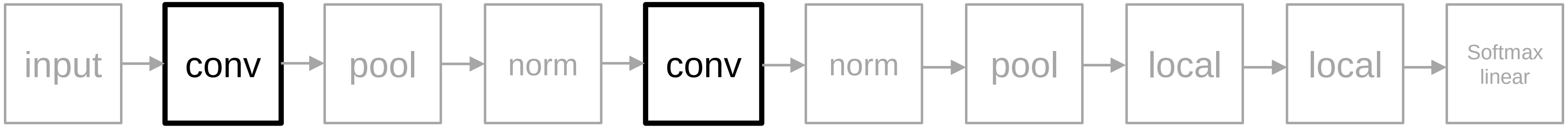
- Input volume size
- Number of filters
- Filter size
- Step size
- Zero padding

Input

0	0	0	0	0	0	0
0	2	1	0	0	2	0
0	2	2	2	0	2	0
0	1	2	0	2	0	0
0	2	1	2	**1**	**2**	**0**
0	1	2	1	**0**	**0**	**0**
0	0	0	0	**0**	**0**	**0**

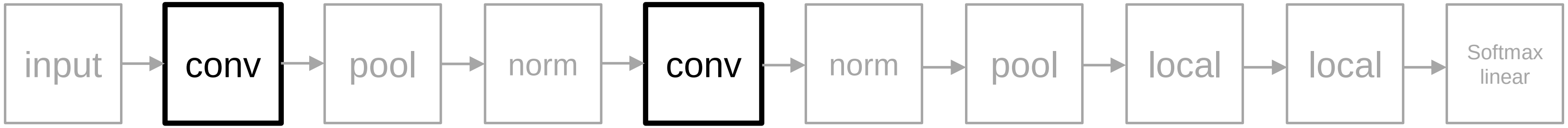
Filter

0	0	0
0	2	1
0	2	2



Convolutional Layer – Activation function

- Rectified linear
  - Element wise $\max(0, x)$
- Leaky ReLu
  - If $\mathrm{x} < 0$ then $\mathrm{f(x)} = 0.01x$
  - Non-zero gradient when the input is negative



Pool Layer – Max Pooling

- Reduce the spatial dimension of an image

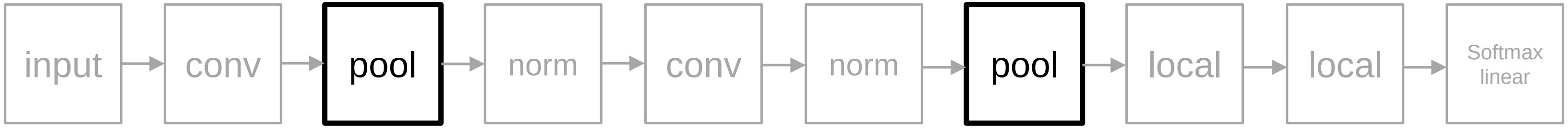
Input

1	1	2	4
5	6	7	8
3	2	1	0
1	2	3	4

Output

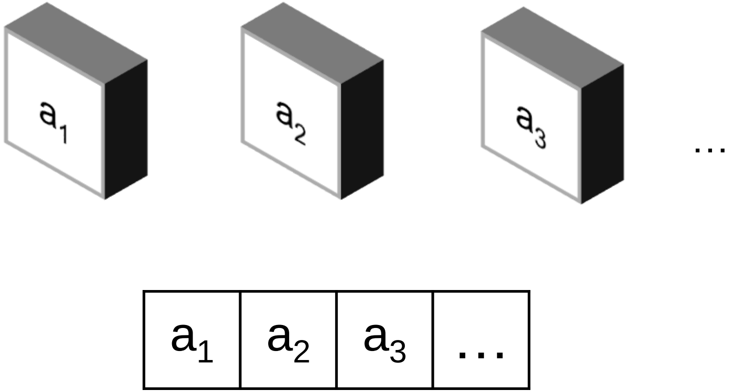
6	8
3	4

http://cs231n.github.io/convolutional-networks/



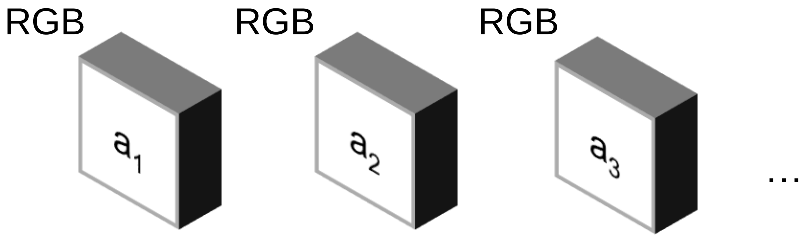
Norm Layer

- 4D-array
- Normalize each element of this array



input → conv → pool → **norm** → conv → **norm** → pool → local → local → Softmax linear

Norm Layer



- Normalize each element of this array
- $a_1 = \left(\left(\frac{R}{\sqrt{R^2+G^2+B^2}}\right), \left(\frac{G}{\sqrt{R^2+G^2+B^2}}\right), \left(\frac{B}{\sqrt{R^2+G^2+B^2}}\right)\right)$

a_1	a_2	a_3	…

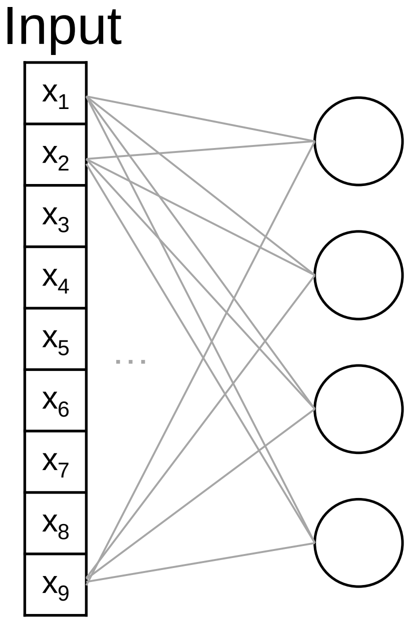
input → conv → pool → **norm** → conv → **norm** → pool → local → local → Softmax linear

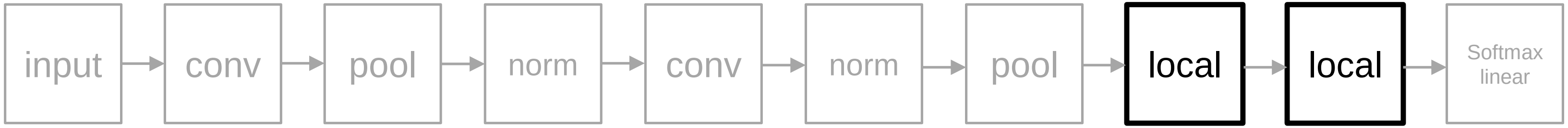
Local Layer

- Also named fully connected layer

Input

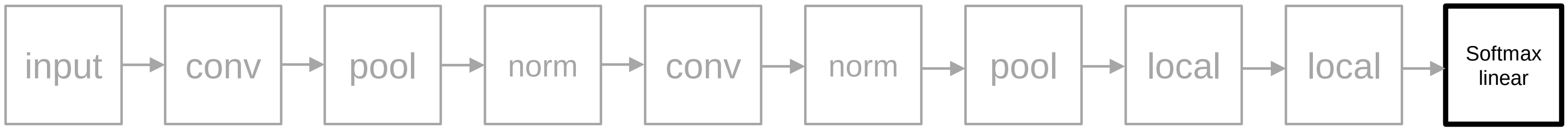
x_1	x_2	x_3
x_4	x_5	x_6
x_7	x_8	x_9





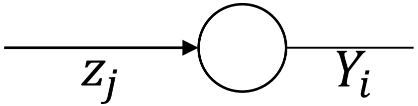
Softmax-Linear Layer

- Softmax output function
- Cost measure for softmax



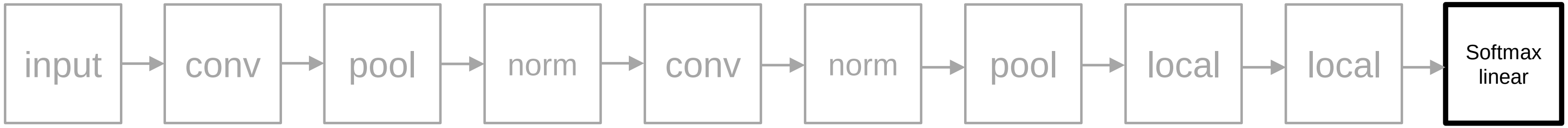
Softmax Output Function

- Soft continuous version of Max Function



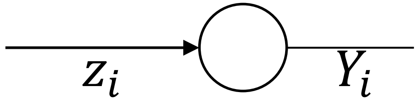
$$Y_i = \frac{e^{z_i}}{\sum_j e^{z_j}}$$

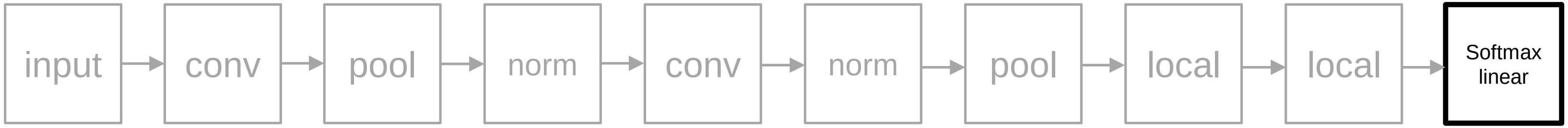
- Forces $\sum(Y_i) = 1$.



Softmax Output Function

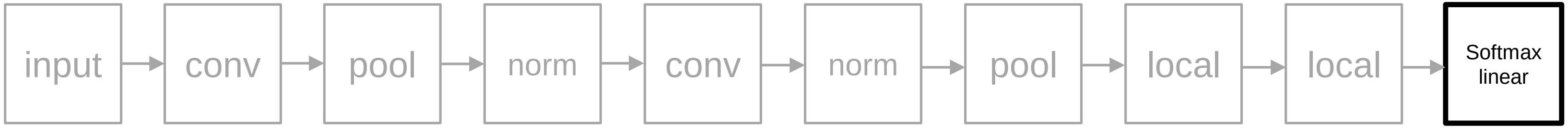
- $\frac{\delta Y_i}{\delta z_i} = Y_i(1 - Y_i)$
- Nice Simple derivative
- Even though Y_i depends of Z_i
  - Derivative
    - For an individual neuron
    - Of an O/P in respect to I/P is just $Y_i(1 - Y_i)$





Cost Measure for Softmax

- Cross entropy cost function
  - $C = -\sum_j T_j \log Y_j$
    - Negative log probability of correct answer
  - Maximise the log probability of getting answer right
  - Very big gradient when O/P is 1 and target is 0
- $\frac{\delta C}{\delta Z_i} = T_i - T_j$
  - Slope is -1 when target values and actual value is opposite



Hyperparameters - Learning Rate

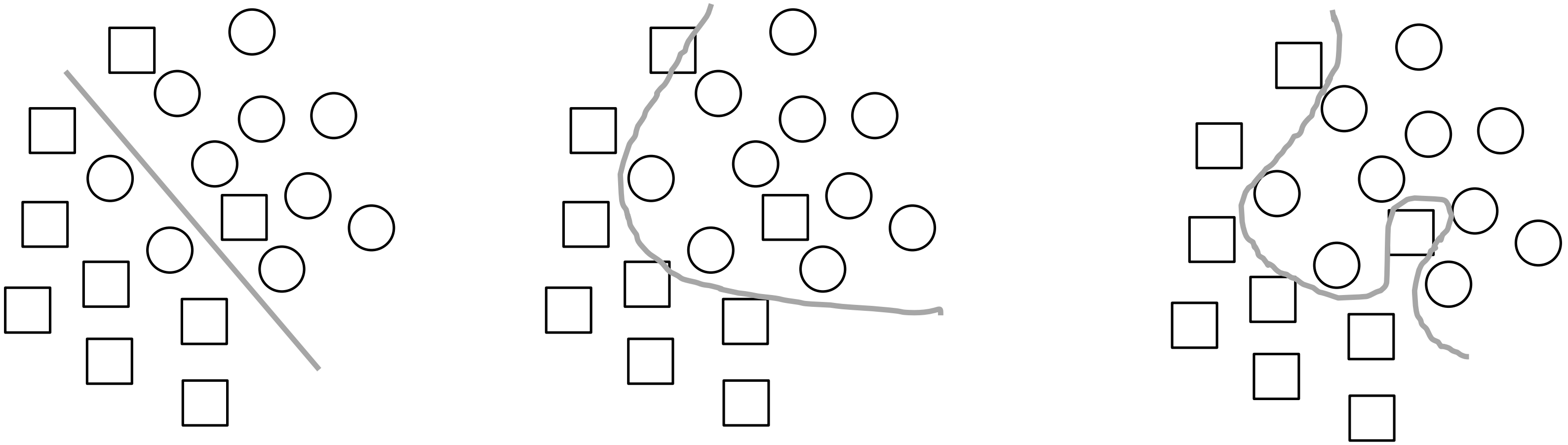
- How fast the network trains
- High learning rate
  - Convergence or global minimum finding is problem
- Low learning rate
  - High training times

TU Clausthal

Hyperparameters - Learning Rate Decay

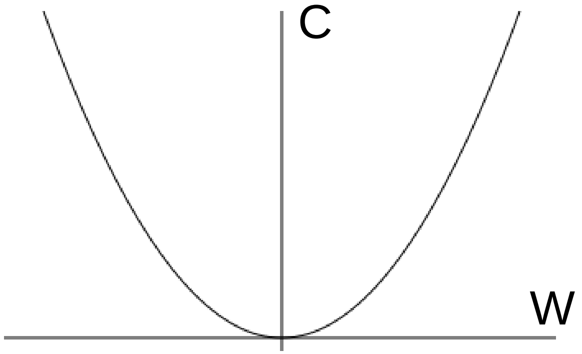
- Learning rate decay means the learning rate decreases over time
  - higher learning rate is well suited to get close to the global minimum
  - small learning rate is better at fine tuning the global minimum
- Several way
  - Exponential decay, reduction by factor of *n*
  - Function to decrease the learning rate by 4%

Hyperparameters - Overfitting or Underfitting



Hyperparameters - Weight Penalty

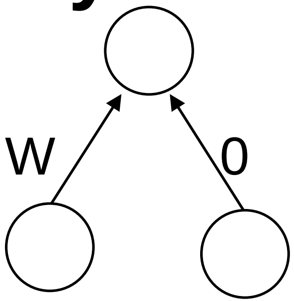
- Adding λ to penalise
  - Keeps weight small
  - Big error derivatives

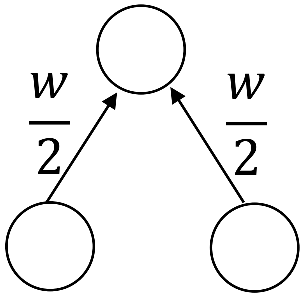


- $C = E + \frac{\lambda}{2}\sum_{i=1} w_i^2$
- $\frac{\partial C}{\partial w_i} = \frac{\partial E}{\partial w_i} + \lambda w_i$
- When $\frac{\partial C}{\partial w_i}$ =0;
  - $w_i = -\frac{1}{\lambda}\frac{\partial E}{\partial w_i}$
  - At minimum of cost function if $\frac{\partial E}{\partial w_i}$ is large, the weights are large

Hyperparameters - Weight Penalty

- Preventing network from the weights it does not need
  - Don't have a lot of weights not doing anything
  - So output changes more slowly as input changes.
- Putting half the weight on each and not on one





PROBLEM

TU Clausthal

Structure

5. Problem

- Introduction
- Data

1 → 2 → 3 → 4 → **5** → 6 → 7 → 8

Introduction

Data

- Images of cats and dogs
- File format is *.jpg
- Color space is RGB

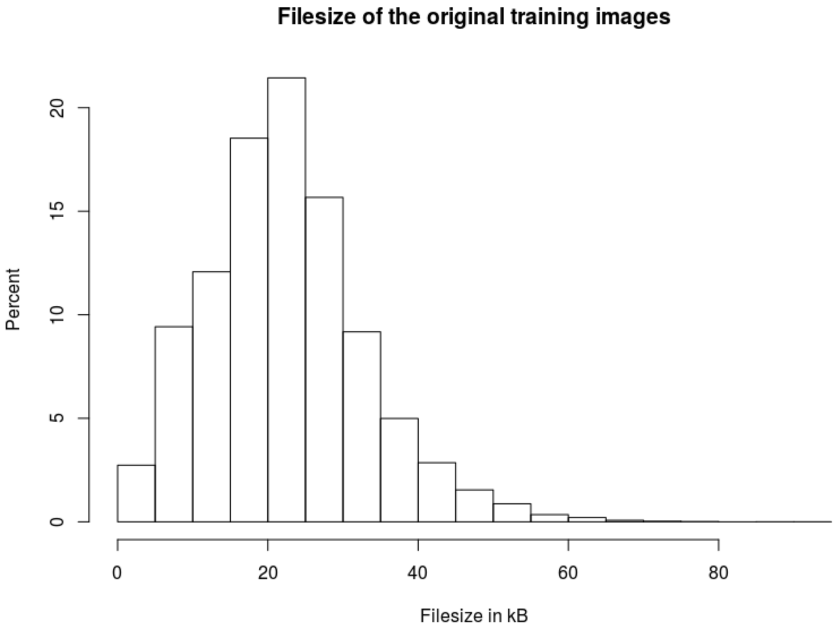
dog1.jpg

cat10.jpg

TU Clausthal

Data

- 25,000 images
  - 12,500 of dogs
  - 12,500 of cats
- Avg. file size
  - 22.34 kB



DESIGN

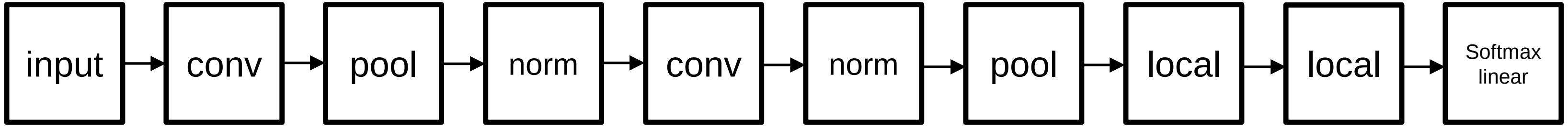
Structure

6. Design

- Implementation of CNN architecture
- System model
- Implemented architectures

1 → 2 → 3 → 4 → 5 → **6** → 7 → 8

Implementation of CNN Architecture



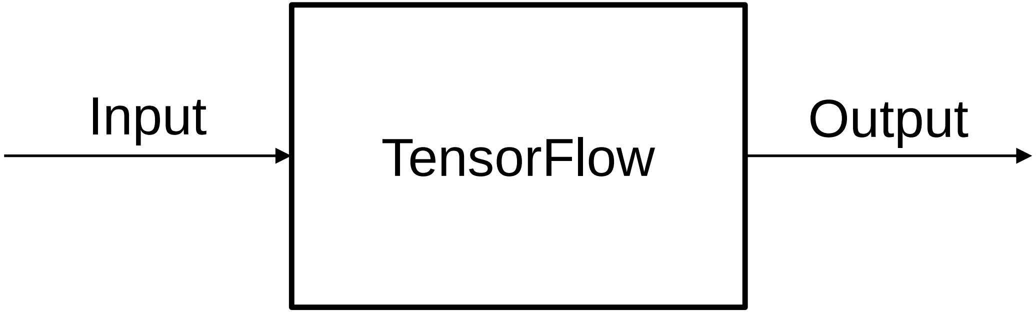
Implementation of CNN Architecture

- TensorFlow was developed by Google Brain team
- Version 1.0
- Use cases
  - Handwritten patterns, image recognition, Word2Vec
- Input data
  - Audio, image, text
- Used techniques
  - Linear classifiers, NN, CNN

Implementation of CNN Architecture

- Input: Raw model
- Graph
  - Architecture of nodes and edges (like NN structure)
  - Session is placed on device
    - Initialise variables randomly
    - Run
  - Let tensors pass through the graph
- Output: Trained model

System Model



System Model -Train vs. Test Data

- Split data
  - Train data
    - 20,000 images (80 percent)
    - Divide into 5 batches containing 4,000 each
  - Test data
    - 5,000 images (20 percent)

Process images

- Resize to 32 * 32 * 3 = 3,072
- Convert to array
  - 25,000 * 3,073

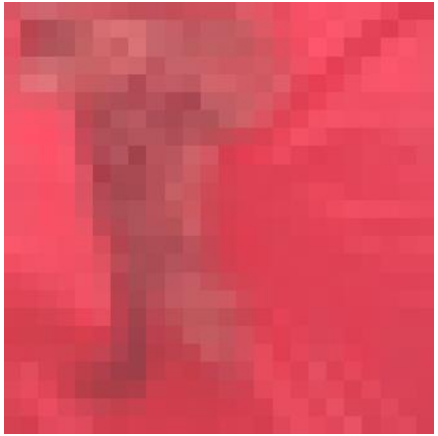
cat10.jpg

dog1.jpg

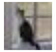
Process images

- Resize to 32 * 32 * 3
- Convert to array
  - 25,000 * 3,073
- Example
  - 1; 22; 11; 123; …
  - 0; 256; 255; 0; …

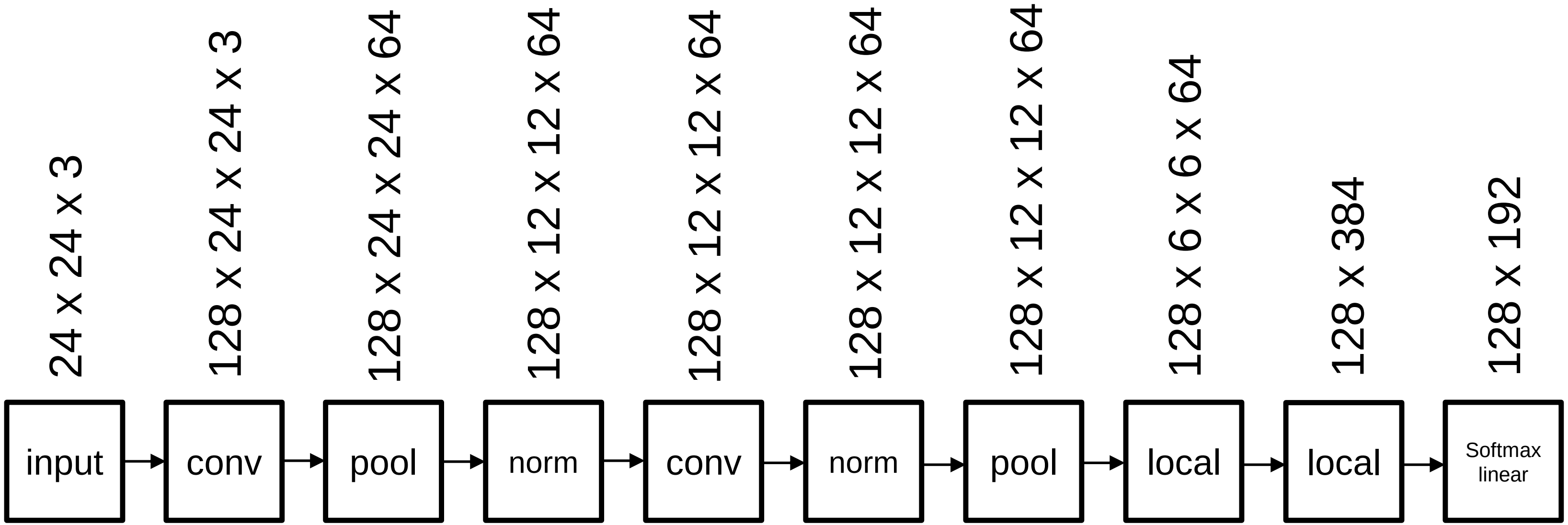
System Model - Random distorsion



System Model - Structure of CNN

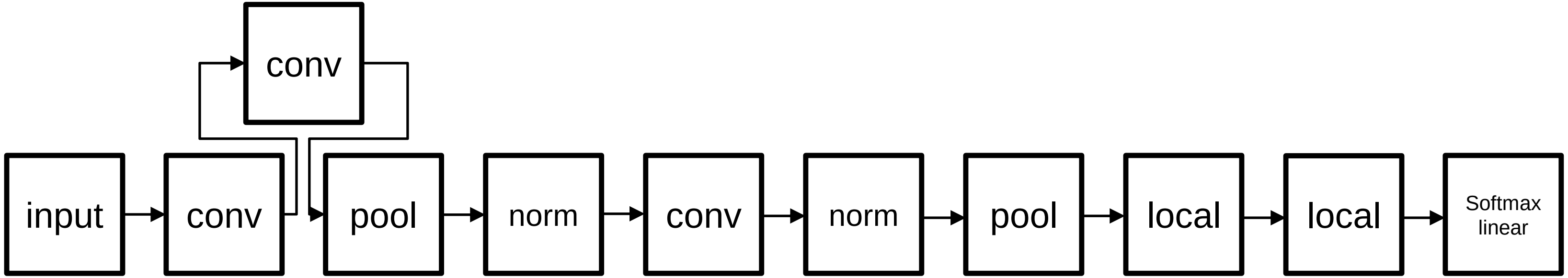


Output: 128 x 2

TU Clausthal

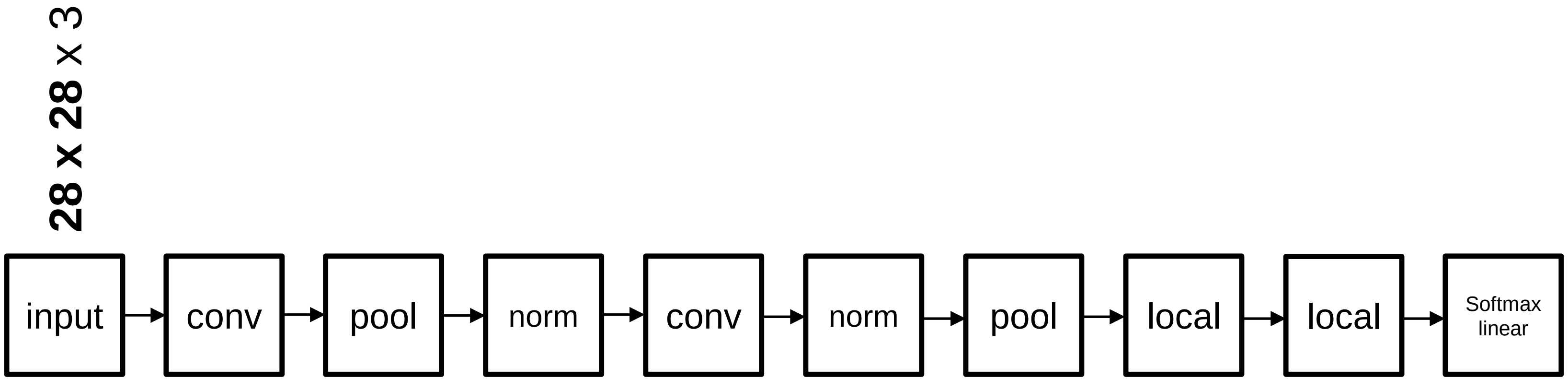
Implemented Architectures – Added Conv Layer

- Input: 128 x 24 x 24 x 3
- Output: 128 x 24 x 24 x 3



Implemented Architectures – Increased size

- Original input: 24 x 24 x 3
- New input: 28 x 28 x 3



EVALUATION

Structure

7. Evaluation

- Automated verification
- Manual verification

1 → 2 → 3 → 4 → 5 → 6 → **7** → 8

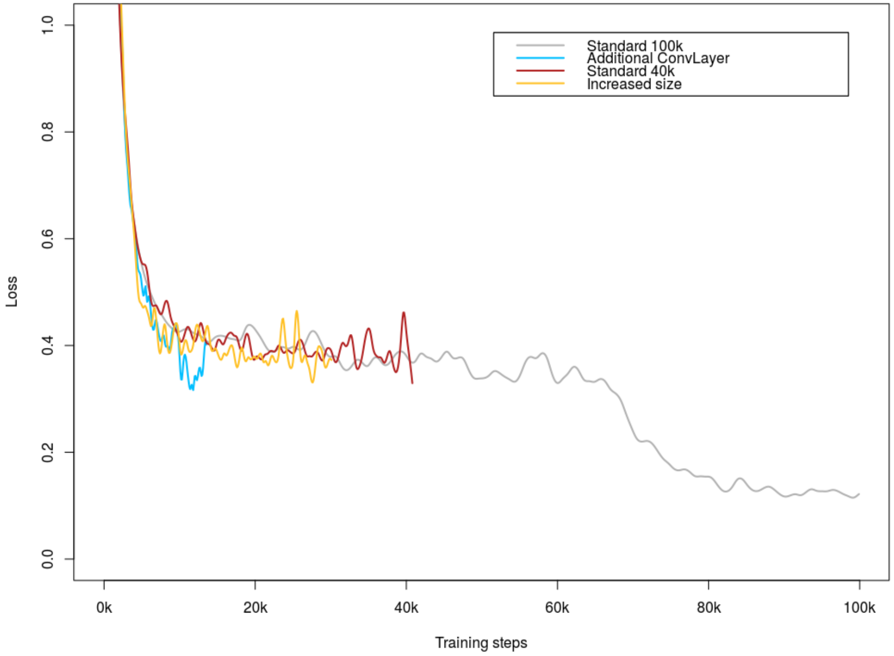
Automated Verification

	Number of steps	Total loss	Time	Machine
Standard 100k	99,900	0.1132	6h 15m 50s	Windows
Standard 40k	40,600	0.3316	8h 26m 58s	Linux
Additional ConvLayer	13,500	0.3128	9h 19m 34s	Linux
Increased size	30,100	0.3446	8h 37m 30s	Linux

TU Clausthal

Automated Verification

- Accuracy of 85 percent



Manual Verification

- For 5000 images accurancy of 97 percent
- Manually verified 100 images
  - Seven were predicted wrong
  - 93 were predicted correctly

Manual Verification – Correctly Predicted

Manual Verification – Wrongly Predicted







Aditya Raj, Sören Schleibaum
Institut für Informatik

Manual Verification – Confusing Images

SUMMARY

Structure

8. Summary

1 → 2 → 3 → 4 → 5 → 6 → 7 → **8**

Summary

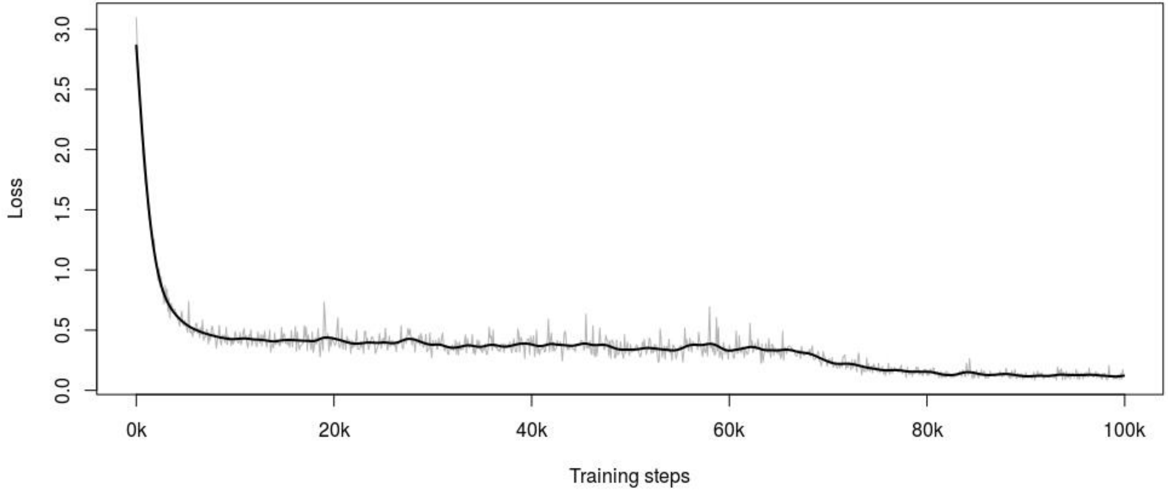
1. Introduction
2. Neural Networks (NN)
3. Math behind NN
4. Convolutional NN (CNN)
5. Problem
6. Design
7. Evaluation
8. Summary

QUESTIONS

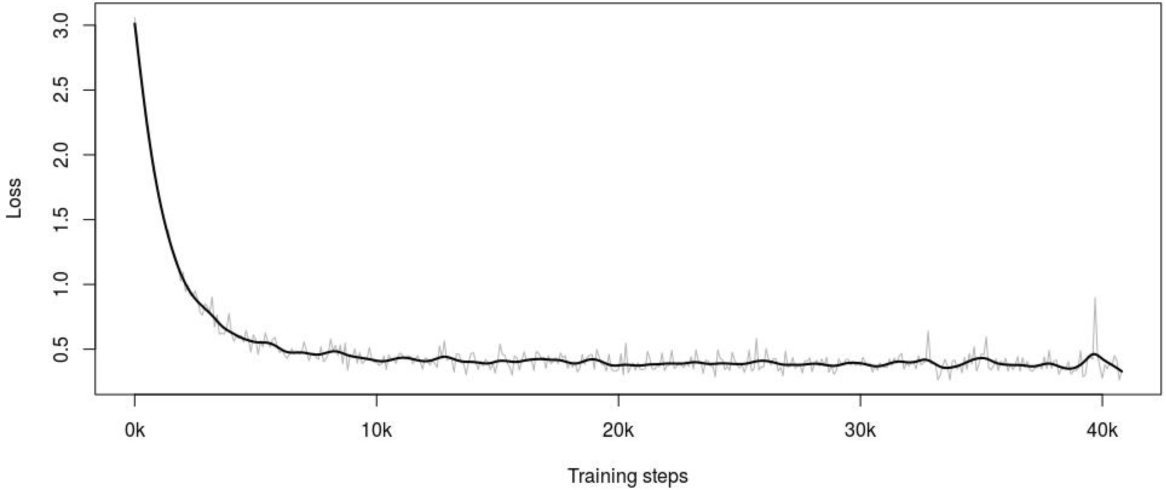
Quellen

- http://cs231n.github.io/convolutional-networks/
- https://www.tensorflow.org/tutorials/deep_cnn/
- Maas, Andrew L., Awni Y. Hannun, and Andrew Y. Ng. "Rectifier nonlinearities improve neural network acoustic models." *Proc. ICML*. Vol. 30. No. 1. 2013.

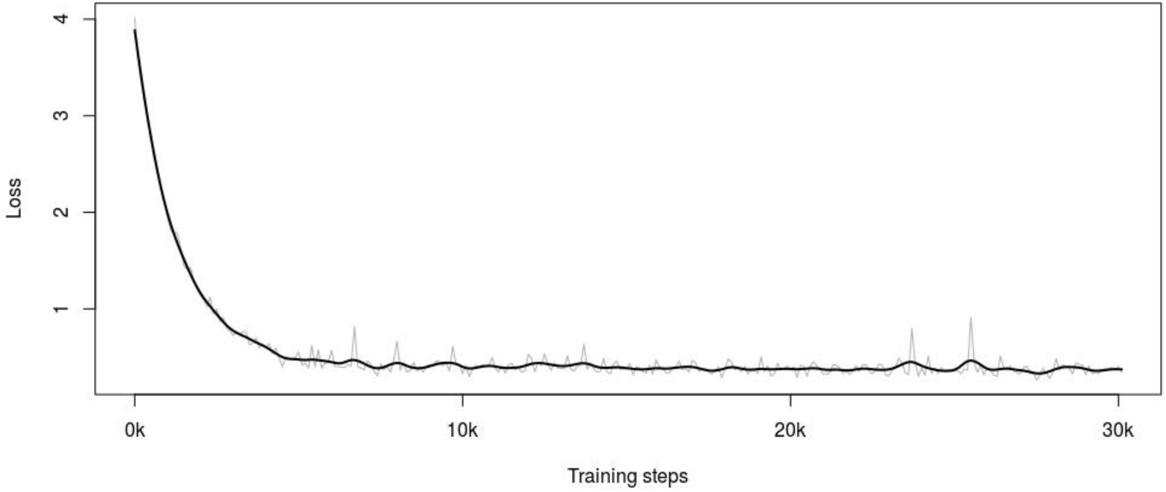
Standard 100k



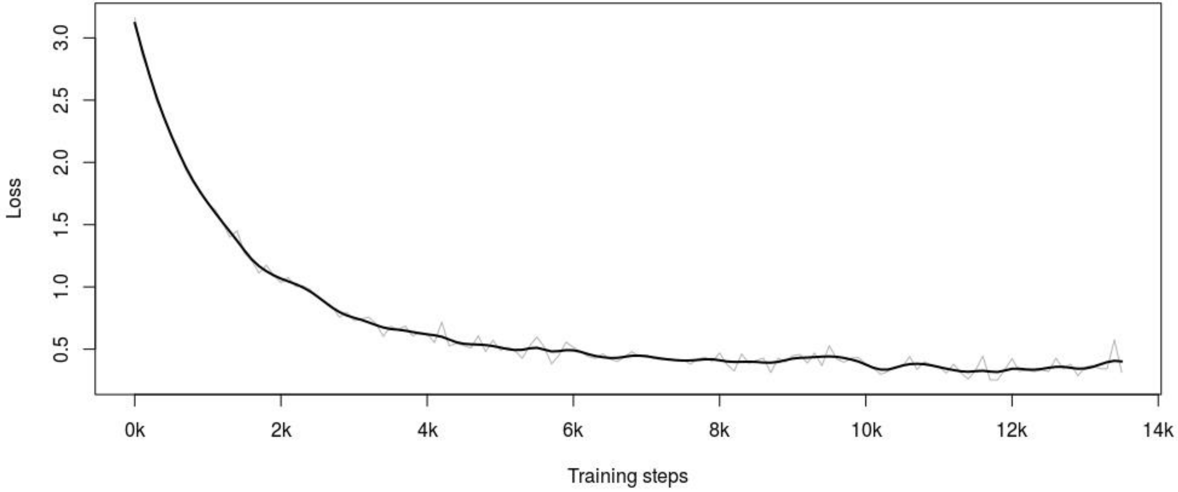
Standard 40k



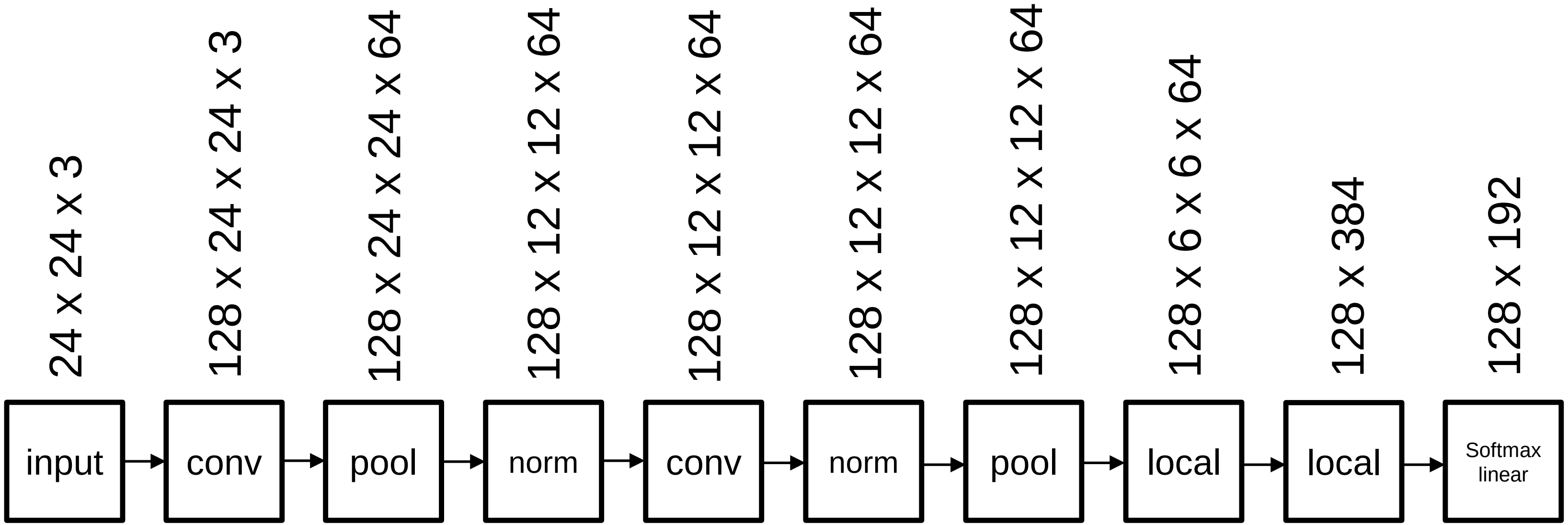
Increased Image Size



Added Convolutional Layer

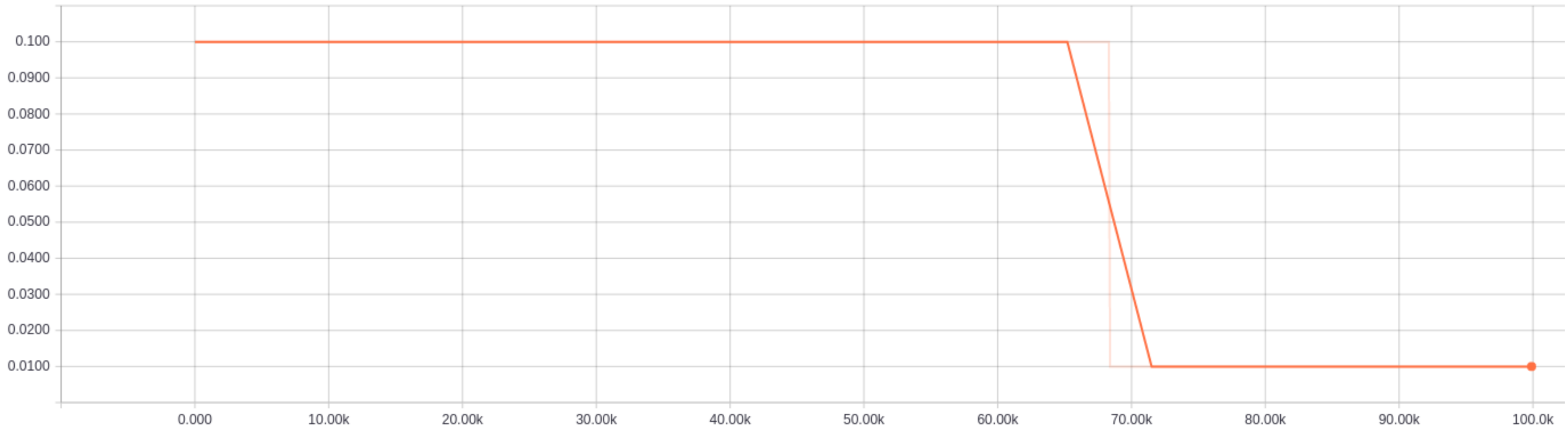


Structure of the CNN we used

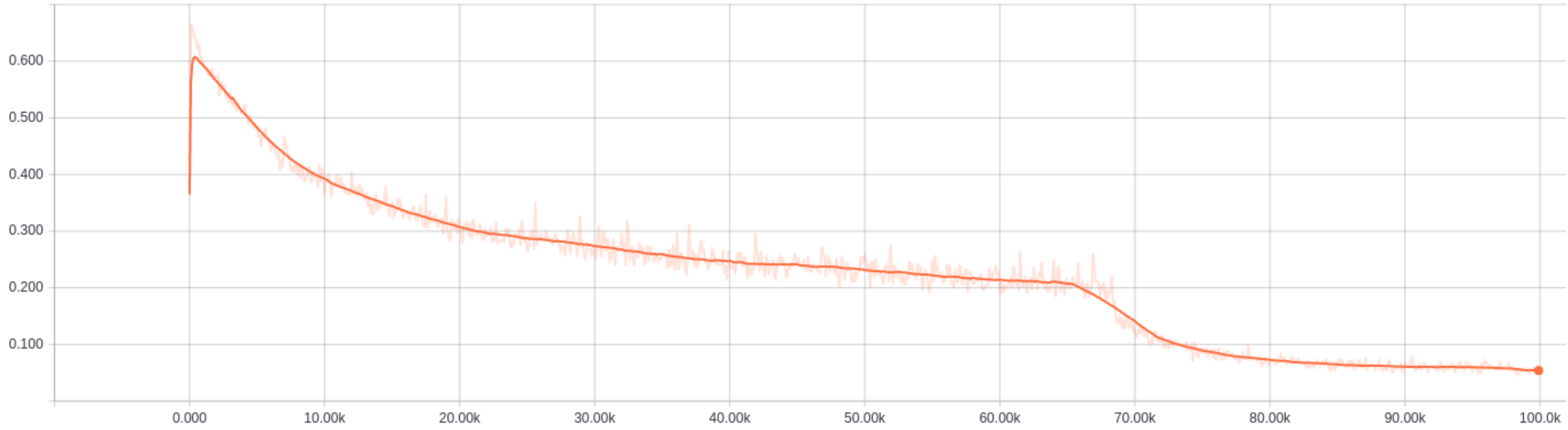


Output: 128 x 2

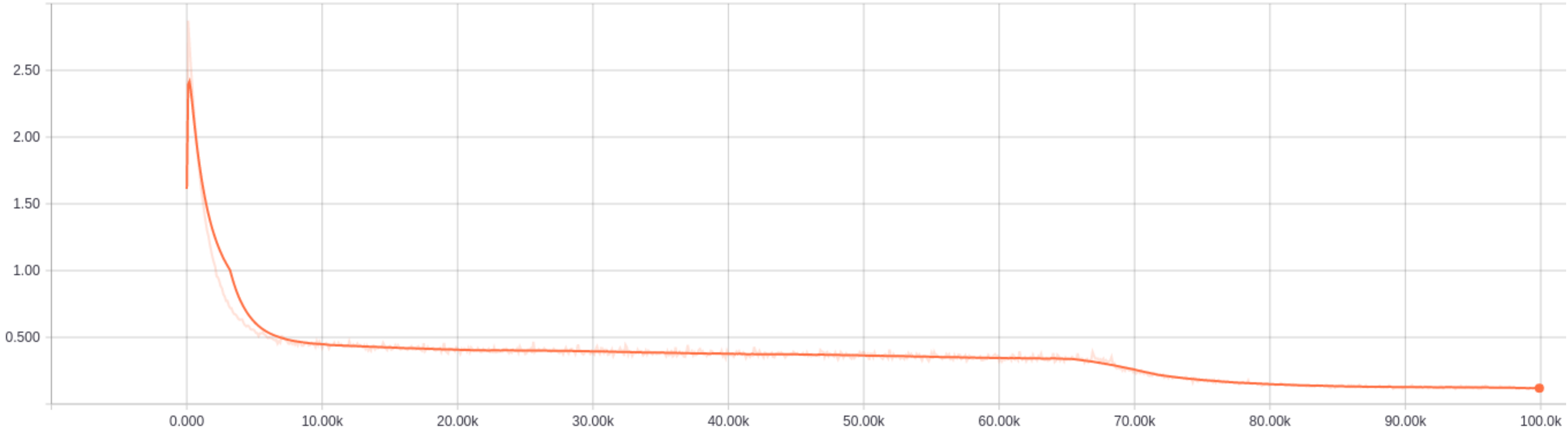
Learning rate



Cross-entropy



Total loss



Total loss after 100k steps roughly above 0.1