`tf.talk()`

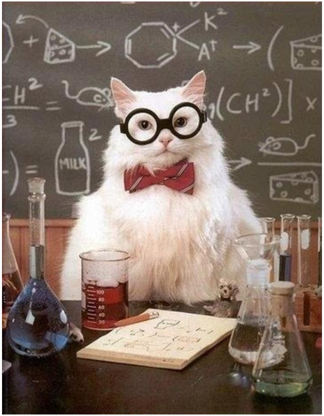
An Introduction to Deep Learning with TensorFlow

Peter Goldsborough

September 16, 2016

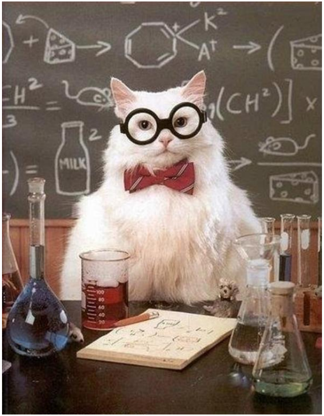
Table of Catents

Table of Catents

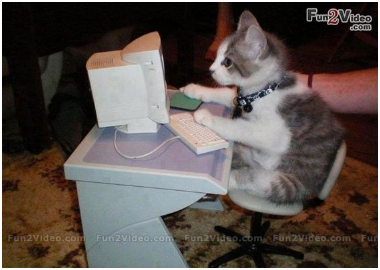


Theory

Table of Catents



Theory



Practice

Background

Background

- CS Student @ TUM

Background

- CS Student @ TUM
- Google & Bloomberg Intern

Background

- CS Student @ TUM
- Google & Bloomberg Intern

Seminar Topic: *Deep Learning With TensorFlow*

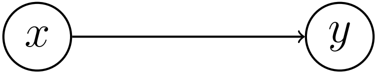
`github.com/peter-can-write/tensorflow-paper`

Neural Networks

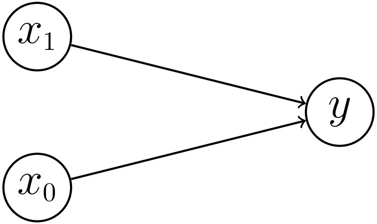
Neural Networks

x

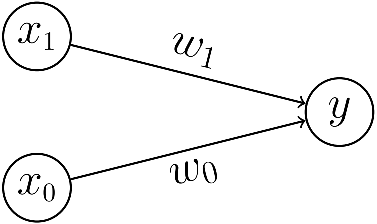
Neural Networks



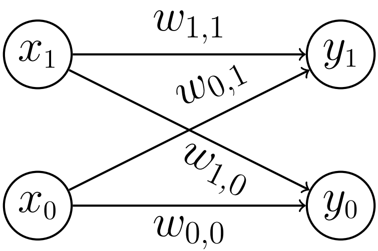
Neural Networks



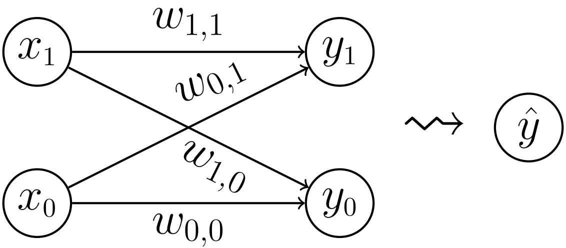
Neural Networks



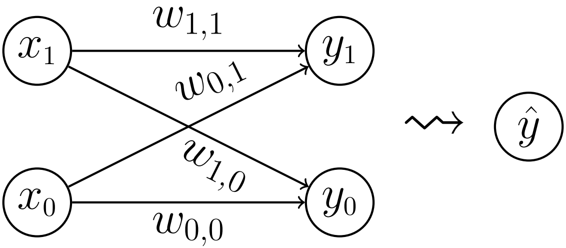
Neural Networks



Neural Networks

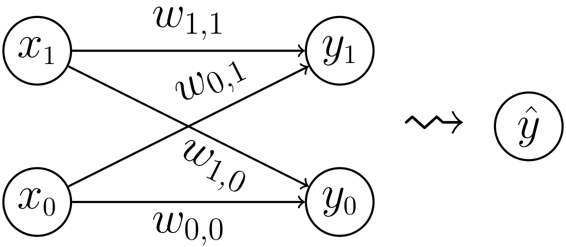


Neural Networks



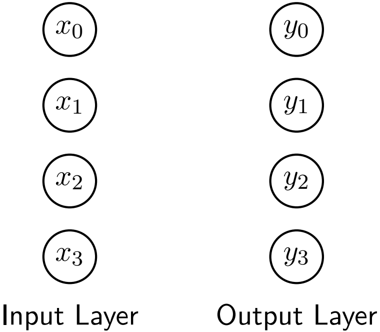
$$\underbrace{\begin{bmatrix} x_0 & x_1 \end{bmatrix}}_{\mathbf{x}} \times \underbrace{\begin{bmatrix} w_{0,0} & w_{0,1} \\ w_{1,0} & w_{1,1} \end{bmatrix}}_{\mathbf{W}} + \underbrace{\begin{bmatrix} b_0 \\ b_1 \end{bmatrix}}_{\mathbf{b}} = \underbrace{\begin{bmatrix} y_0 \\ y_1 \end{bmatrix}}_{\mathbf{y}}$$

Neural Networks

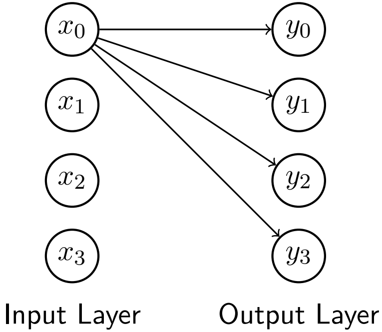


$$\underbrace{\begin{bmatrix} x_0 & x_1 \end{bmatrix}}_{\mathbf{x}} \times \underbrace{\begin{bmatrix} w_{0,0} & w_{0,1} \\ w_{1,0} & w_{1,1} \end{bmatrix}}_{\mathbf{W}} + \underbrace{\begin{bmatrix} b_0 \\ b_1 \end{bmatrix}}_{\mathbf{b}} = \underbrace{\begin{bmatrix} y_0 \\ y_1 \end{bmatrix}}_{\mathbf{y}} \rightsquigarrow \hat{\mathbf{y}}$$

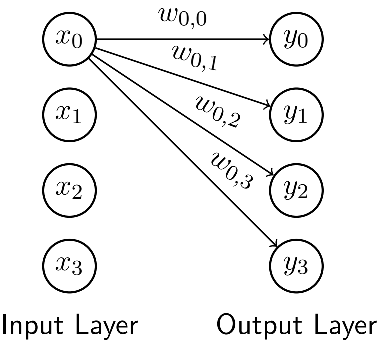
Neural Networks



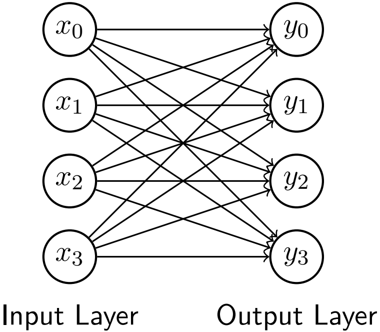
Neural Networks



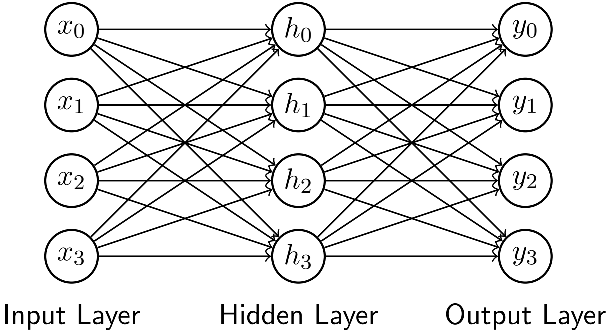
Neural Networks



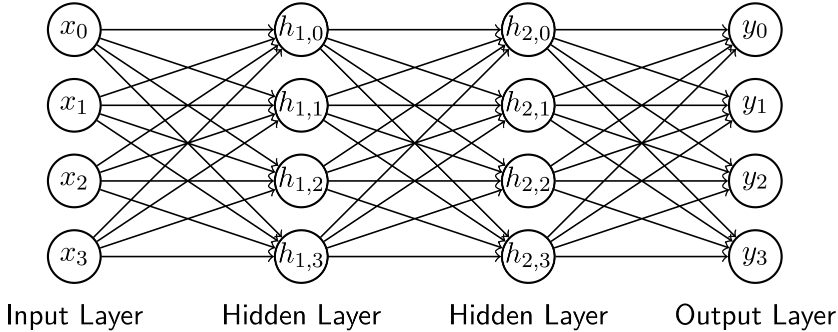
Neural Networks



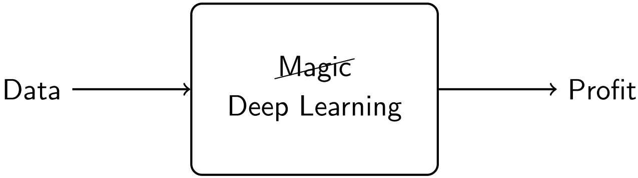
Deep Neural Networks



Deep Neural Networks



Deep Neural Networks



Deep Neural Networks

Deep Learning assumes that data is structured

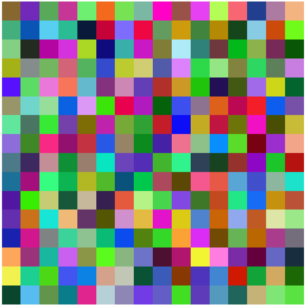
Deep Learning assumes that data is structured

hierarchically

Deep Neural Networks

Deep Neural Networks

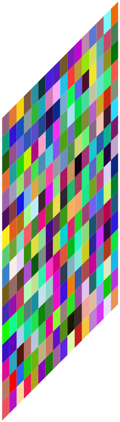
Deep Neural Networks



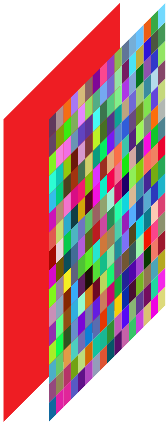
Deep Neural Networks

47	6	89	83	19	73	78	39	62	96	53	47	96	45	91	92
21	88	79	48	24	74	26	81	47	22	33	57	52	30	77	47
26	10	70	63	61	68	75	82	73	22	66	14	21	42	87	8
28	28	40	32	34	8	4	33	62	59	12	62	56	10	72	49
35	60	75	72	36	37	60	73	63	96	87	26	45	48	36	66
37	50	77	54	37	65	69	86	43	96	45	76	44	40	56	28
2	15	29	82	29	29	53	76	57	56	73	16	36	87	71	10
3	63	75	73	23	45	24	32	82	72	32	11	65	93	59	46
29	43	77	89	97	21	45	95	96	84	54	61	79	34	20	87
75	93	97	18	73	80	19	35	97	16	21	87	61	39	5	24
38	83	83	97	53	13	32	75	3	91	7	60	70	83	26	79
40	50	74	42	18	6	49	1	3	20	51	26	63	41	17	17
44	85	5	20	91	68	14	68	39	45	43	89	14	77	46	80
79	71	74	38	70	91	47	29	40	5	25	29	24	62	59	76
64	14	24	95	8	97	42	96	81	93	33	50	56	23	35	3
11	70	85	59	57	91	92	1	57	76	16	12	64	70	27	22

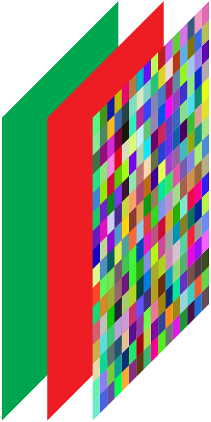
Deep Neural Networks



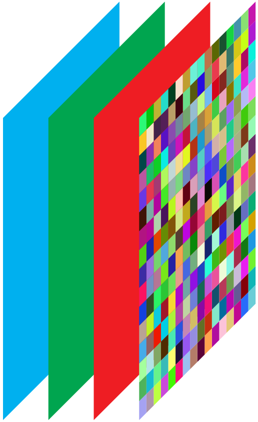
Deep Neural Networks



Deep Neural Networks



Deep Neural Networks



Deep Neural Networks

- We want to classify images into one of k classes

Deep Neural Networks

- We want to classify images into one of k classes
- Extract hierarchical features

Deep Neural Networks

- We want to classify images into one of k classes
- Extract hierarchical features

Edges

Deep Neural Networks

- We want to classify images into one of k classes
- Extract hierarchical features

$\boxed{\text{Edges}} \longrightarrow \boxed{\text{Contours}}$

Deep Neural Networks

- We want to classify images into one of k classes
- Extract hierarchical features

Edges $\longrightarrow$ Contours $\longrightarrow$ Parts

Deep Neural Networks

- We want to classify images into one of k classes
- Extract hierarchical features

Edges $\longrightarrow$ Contours $\longrightarrow$ Parts $\longrightarrow$ Objects

Why reinvent the wheel?

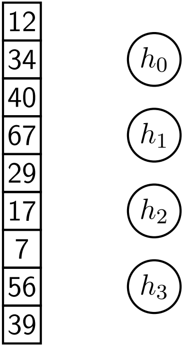
Why reinvent the wheel?

78	87	55
60	53	46
88	63	91

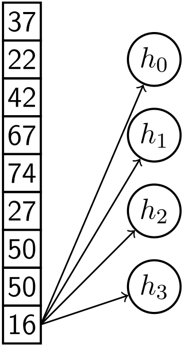
Why reinvent the wheel?

42
3
32
23
39
39
73
16
5

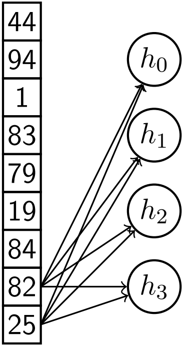
Why reinvent the wheel?



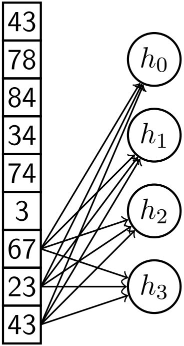
Why reinvent the wheel?



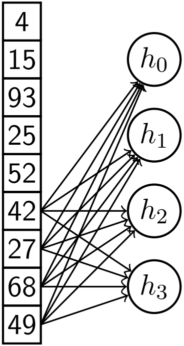
Why reinvent the wheel?



Why reinvent the wheel?

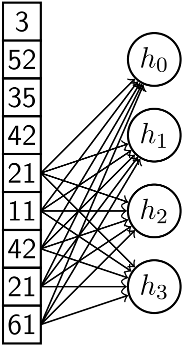


Why reinvent the wheel?

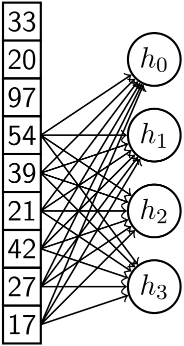


Deep Neural Networks

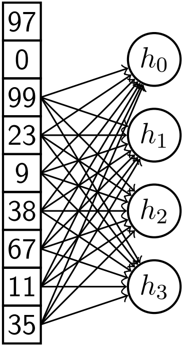
Why reinvent the wheel?



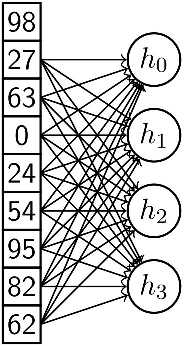
Why reinvent the wheel?



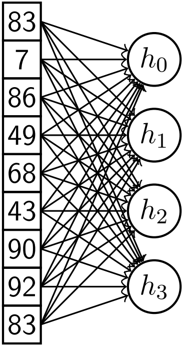
Why reinvent the wheel?



Why reinvent the wheel?

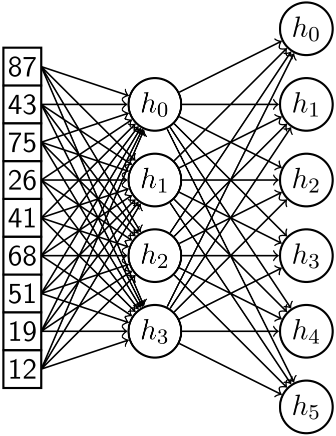


Why reinvent the wheel?

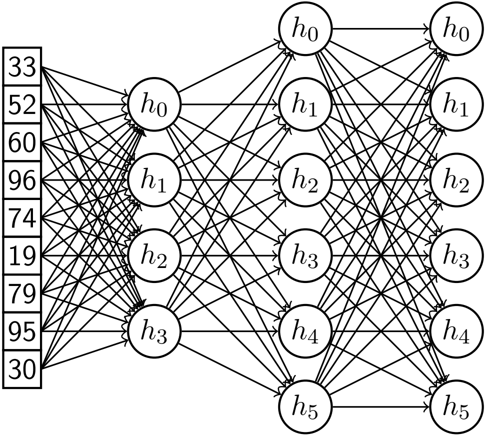


Deep Neural Networks

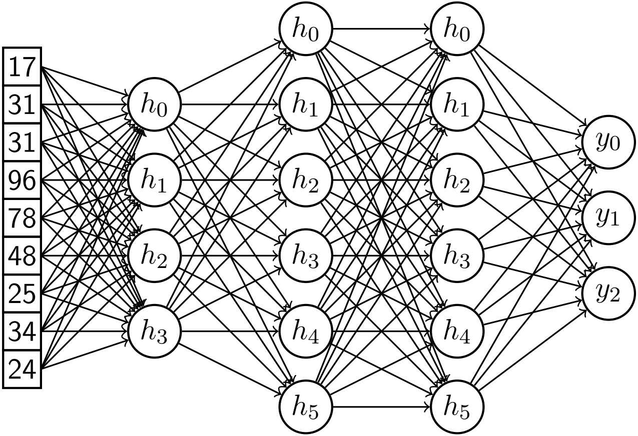
Why reinvent the wheel?



Why reinvent the wheel?

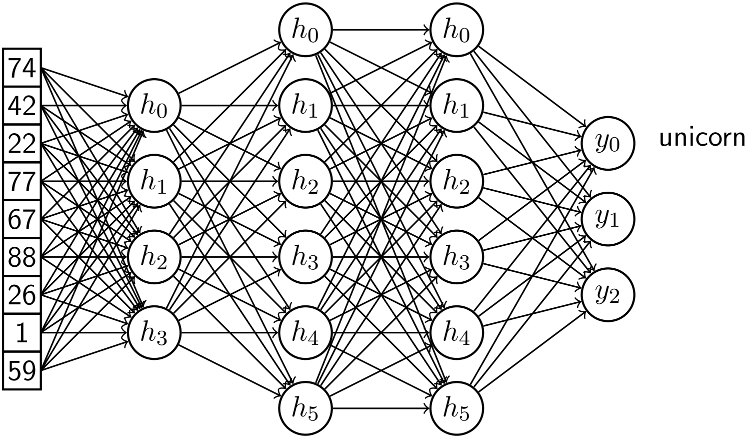


Why reinvent the wheel?



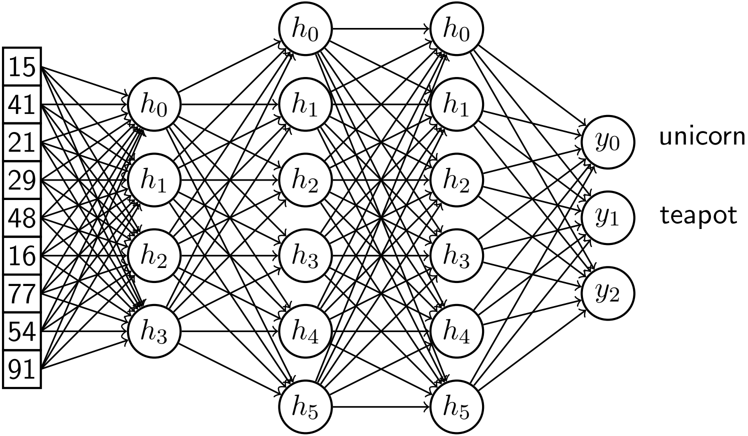
Deep Neural Networks

Why reinvent the wheel?



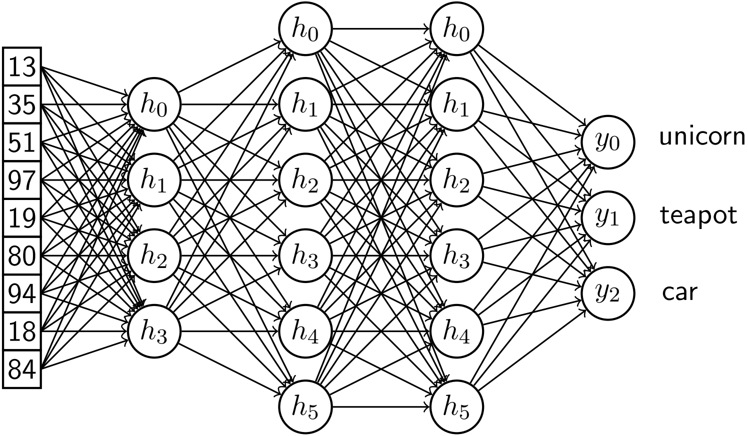
Deep Neural Networks

Why reinvent the wheel?



Deep Neural Networks

Why reinvent the wheel?



Time for ze kitteh

This is a cat ♥

Time for ze kitteh

Still a cat ♥♥

Time for ze kitteh

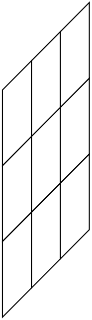
Half cat / half salad ♥♥♥

Time for ze kitteh

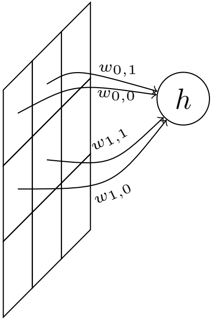
Many cats ♥♥♥♥

Weight Sharing

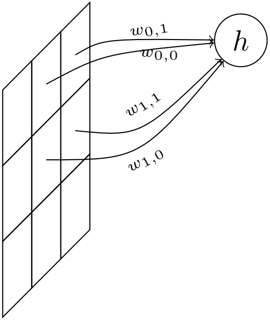
Weight Sharing



Weight Sharing

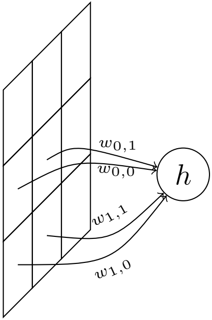


Weight Sharing

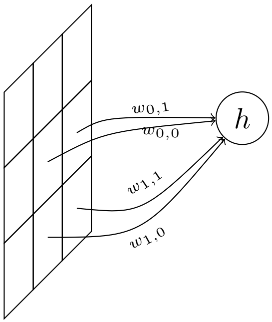


Convolutional Neural Networks: Mechanics

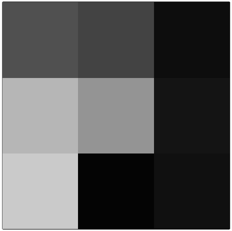
Weight Sharing



Weight Sharing



Convolutional Neural Networks: Mechanics



Image

Convolutional Neural Networks: Mechanics

0.4	0.9	0.1
0.7	0.2	0.6
0.8	0.3	0.5

Image

Convolutional Neural Networks: Mechanics

0.4	0.9	0.1
0.7	0.2	0.6
0.8	0.3	0.5

Image

5.7	2.4
3.1	0.9

Kernel

Convolutional Neural Networks: Mechanics

$5.7 \cdot 0.4$	$2.4 \cdot 0.9$	0.1
$3.1 \cdot 0.7$	$0.9 \cdot 0.2$	0.6
0.8	0.3	0.5

Image

6.79

Output

Convolutional Neural Networks: Mechanics

0.4	$5.7 \cdot 0.9$	$2.4 \cdot 0.1$
0.7	$3.1 \cdot 0.2$	$0.9 \cdot 0.6$
0.8	0.3	0.5

Image

6.79	6.53

Output

Convolutional Neural Networks: Mechanics

0.4	0.9	0.1
$5.7 \cdot 0.7$	$2.4 \cdot 0.2$	0.6
$3.1 \cdot 0.8$	$0.9 \cdot 0.3$	0.5

Image

6.79	6.53
7.67	

Output

Convolutional Neural Networks: Mechanics

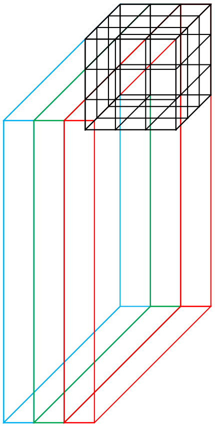
0.4	0.9	0.1
0.7	$5.7 \cdot 0.2$	$2.4 \cdot 0.6$
0.8	$3.1 \cdot 0.3$	$0.9 \cdot 0.5$

Image

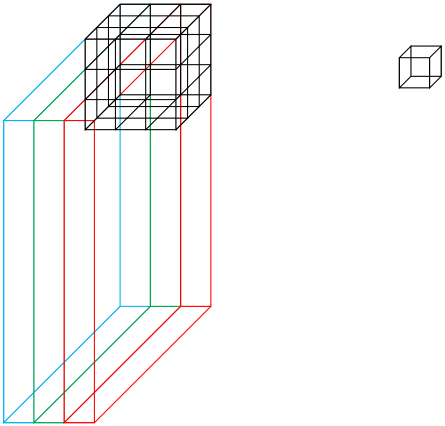
6.79	6.53
7.67	3.96

Output

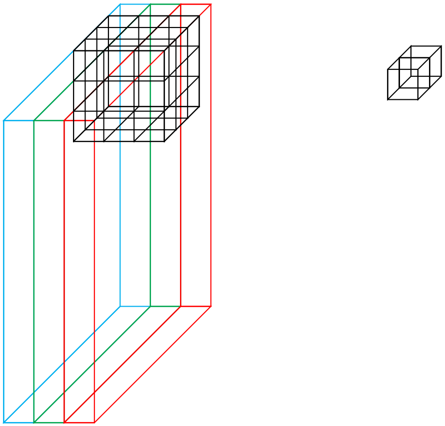
Convolutional Neural Networks: Mechanics



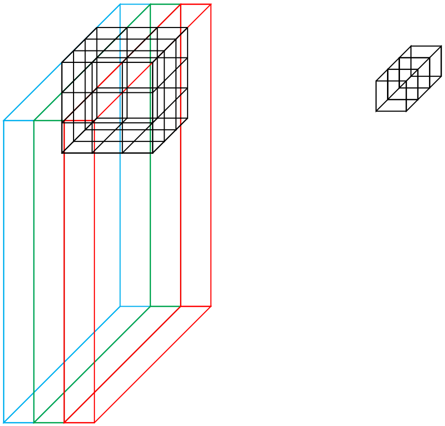
Convolutional Neural Networks: Mechanics



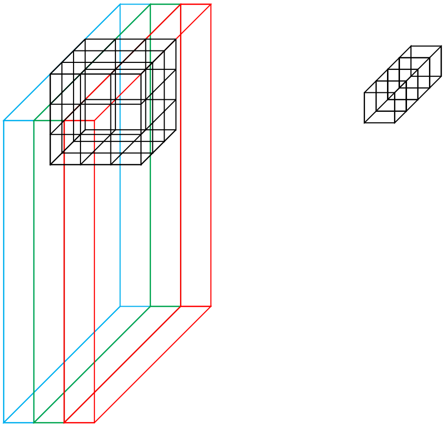
Convolutional Neural Networks: Mechanics



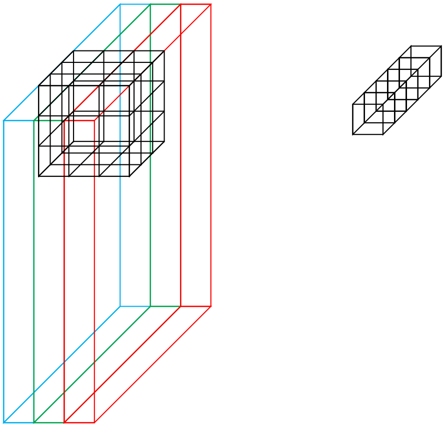
Convolutional Neural Networks: Mechanics



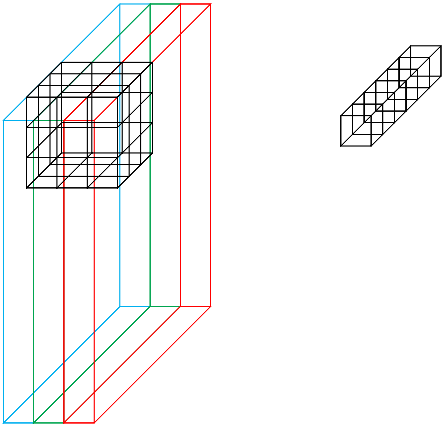
Convolutional Neural Networks: Mechanics



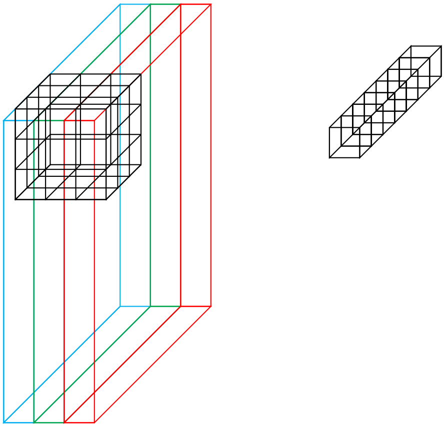
Convolutional Neural Networks: Mechanics



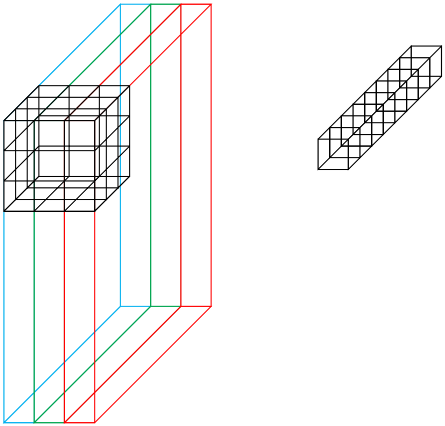
Convolutional Neural Networks: Mechanics



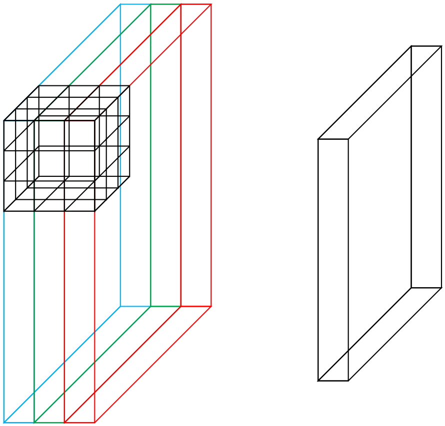
Convolutional Neural Networks: Mechanics



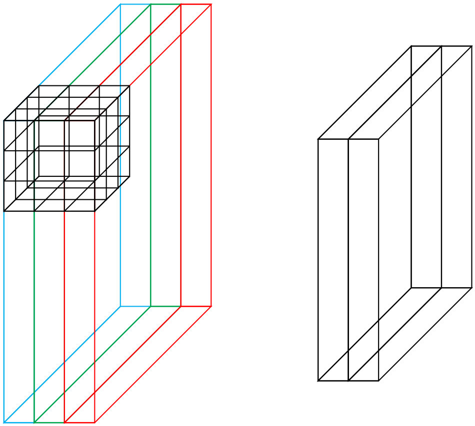
Convolutional Neural Networks: Mechanics



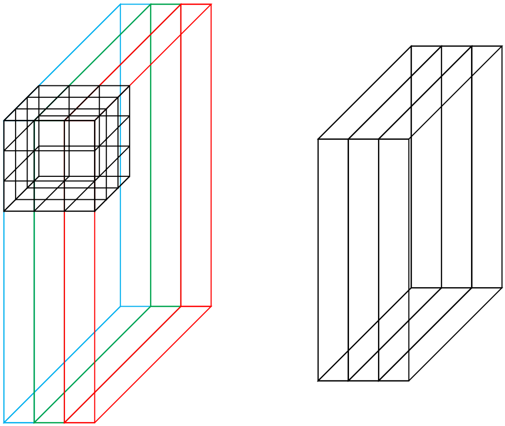
Convolutional Neural Networks: Mechanics



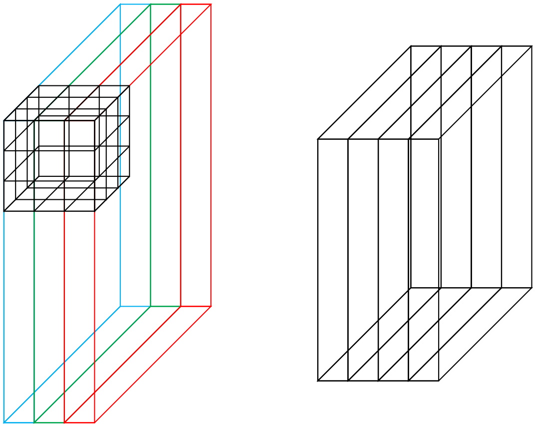
Convolutional Neural Networks: Mechanics



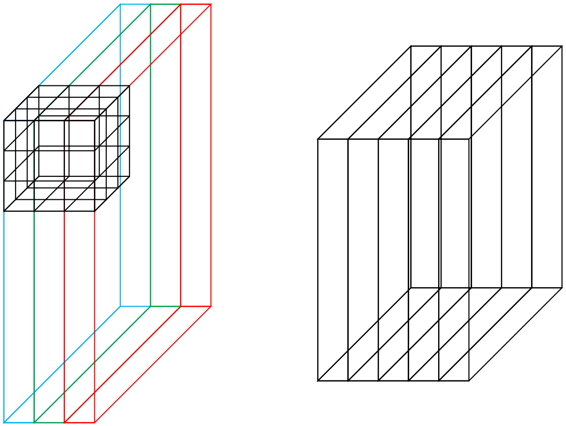
Convolutional Neural Networks: Mechanics



Convolutional Neural Networks: Mechanics



Convolutional Neural Networks: Mechanics



Recipe for a Convolutional Neural Network Layer

Recipe for a Convolutional Neural Network Layer

- Ingredients

Recipe for a Convolutional Neural Network Layer

- Ingredients
  1. Image I with dimension $w \times h \times d$

Recipe for a Convolutional Neural Network Layer

- Ingredients
  1. Image I with dimension $w \times h \times d$
  2. A kernel (filter) K of size $k \times l \times m$

Recipe for a Convolutional Neural Network Layer

- Ingredients
  1. Image I with dimension $w \times h \times d$
  2. A kernel (filter) K of size $k \times l \times m$
- Cooking

Recipe for a Convolutional Neural Network Layer

- Ingredients
  1. Image I with dimension $w \times h \times d$
  2. A kernel (filter) K of size $k \times l \times m$
- Cooking
  - Put the image into the oven at 150°C

Recipe for a Convolutional Neural Network Layer

- Ingredients
  1. Image I with dimension $w \times h \times d$
  2. A kernel (filter) K of size $k \times l \times m$
- Cooking
  - Don't put the image into the oven at 150°C

Recipe for a Convolutional Neural Network Layer

- Ingredients
  1. Image I with dimension $w \times h \times d$
  2. A kernel (filter) K of size $k \times l \times m$
- Cooking
  - Don't put the image into the oven at 150°C
  - Slide the kernel across the image

Recipe for a Convolutional Neural Network Layer

- Ingredients
  1. Image I with dimension $w \times h \times d$
  2. A kernel (filter) K of size $k \times l \times m$
- Cooking
  - Don't put the image into the oven at 150°C
  - Slide the kernel across the image
  - Compute the "dot product" for each configuration

Convolutional Neural Networks: Pooling

66	2
6	32

Convolutional Neural Networks: Pooling

66	2
6	32

Convolutional Neural Networks: Pooling

6	2
66	32

Convolutional Neural Networks: Pooling

2	66
6	32

Convolutional Neural Networks: Pooling

32	6
2	66

Convolutional Neural Networks: Pooling

5	19	69
66	2	79
6	32	128

Convolutional Neural Networks: Pooling

5	19	69
66	2	79
6	32	128

Convolutional Neural Networks: Pooling

5	19	69
66	2	79
6	32	128

66

Convolutional Neural Networks: Pooling

5	19	69
66	2	79
6	32	128

66	79

Convolutional Neural Networks: Pooling

5	19	69
66	2	79
6	32	128

66	79
66	

Convolutional Neural Networks: Pooling

5	19	69
66	2	79
6	32	128

66	79
66	128

Convolutional Neural Networks: Pooling

5	19	69
66	2	79
6	32	128

66	79
66	128

- *Pooling* achieves translational invariance

Convolutional Neural Networks: Pooling

5	19	69
66	2	79
6	32	128

66	79
66	128

- *Pooling* achieves translational invariance
- A form of downsampling

Convolutional Neural Networks: Pooling

5	19	69
66	2	79
6	32	128

66	79
66	128

- *Pooling* achieves translational invariance
- A form of downsampling
- Other pooling functions possible

Convolutional Neural Networks: Architecture

```
INPUT -> [CONV+ -> POOL?]+ -> FC+ -> OUTPUT
```

Convolutional Neural Networks: Intuition

What's in a kernel?

Convolutional Neural Networks: Intuition

0	1	0
0	1	0
0	1	0

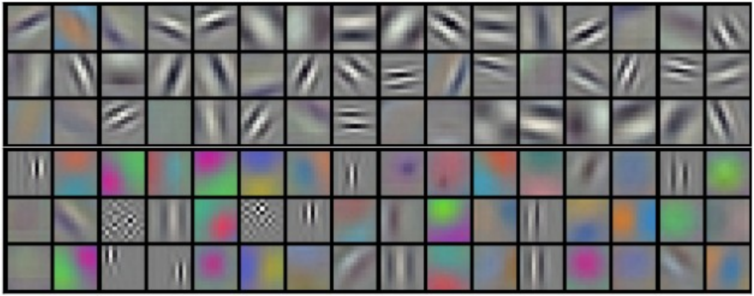
Patterns

Convolutional Neural Networks: Intuition

0	0	0
-1	1	0
0	0	0

Features

Convolutional Neural Networks: Intuition



TensorFlow

TensorFlow

- An open source deep learning library

TensorFlow

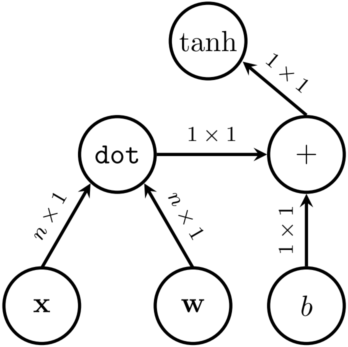
- An open source deep learning library
- Released by Google in November 2015

TensorFlow

- An open source deep learning library
- Released by Google in November 2015
- Especially suited to:
  - "Large-scale machine learning on
  - heterogeneous distributed systems"

Computational Paradigms

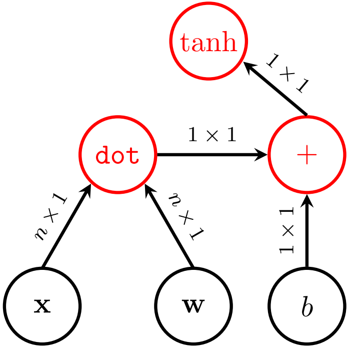
Computational Paradigms



$$\hat{y} = \tanh(\mathbf{x}^\top \mathbf{w} + b)$$

Computational Graphs

Computational Paradigms

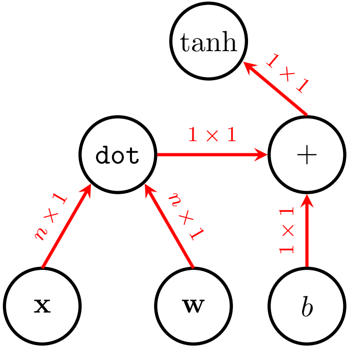


$$\hat{y} = \tanh(\mathbf{x}^\top \mathbf{w} + b)$$

Computational Graphs

1. Operations

Computational Paradigms

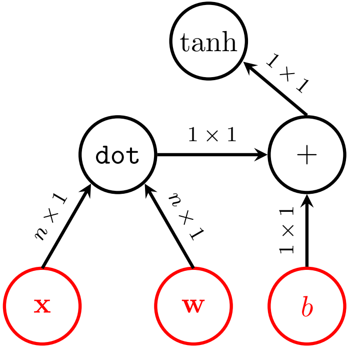


$$\hat{y} = \tanh(\mathbf{x}^\top \mathbf{w} + b)$$

Computational Graphs

1. Operations
2. Tensors

Computational Paradigms

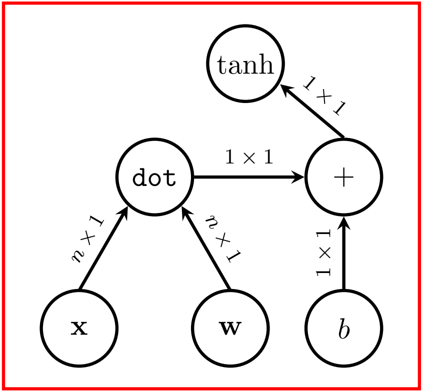


$$\hat{y} = \tanh(\mathbf{x}^\top \mathbf{w} + b)$$

Computational Graphs

1. Operations
2. Tensors
3. Variables

Computational Paradigms



$$\hat{y} = \texttt{session.run}(\tanh(\mathbf{x}^\top \mathbf{w} + b))$$

Computational Graphs

1. Operations
2. Tensors
3. Variables
4. Sessions

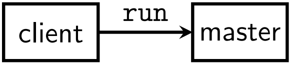
Execution Model

Execution Model

client

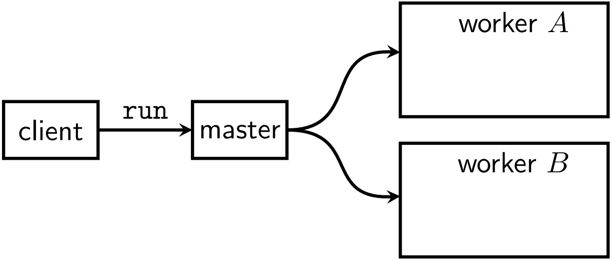
1. Client

Execution Model



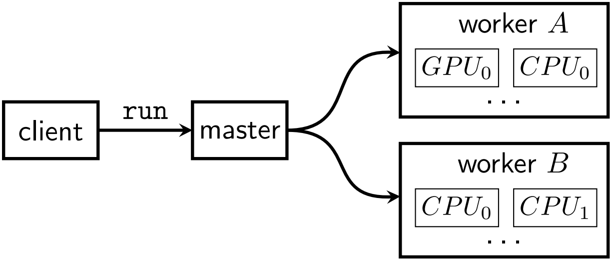
1. Client
2. Master

Execution Model



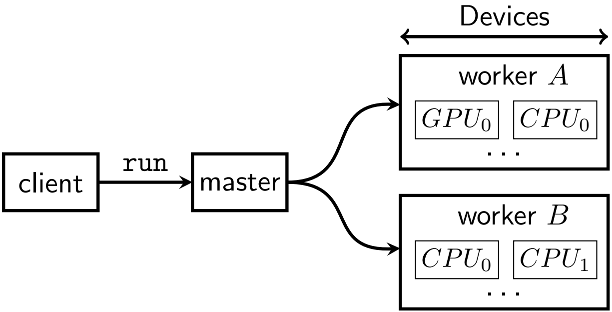
1. Client
2. Master
3. Workers

Execution Model



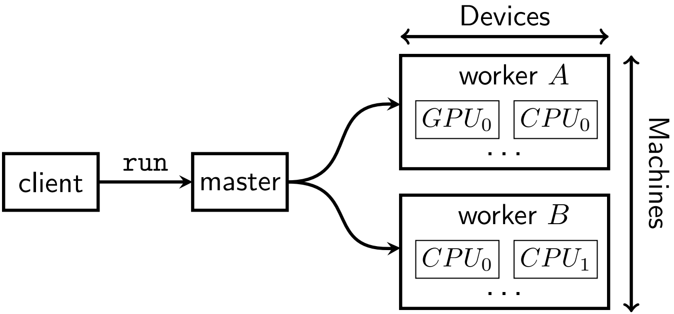
1. Client
2. Master
3. Workers
4. Devices

Execution Model



1. Client
2. Master
3. Workers
4. Devices

Execution Model



1. Client
2. Master
3. Workers
4. Devices

Visualization Tools

Visualization Tools

- Deep Neural Networks have the tendency of being . . . deep

Visualization Tools

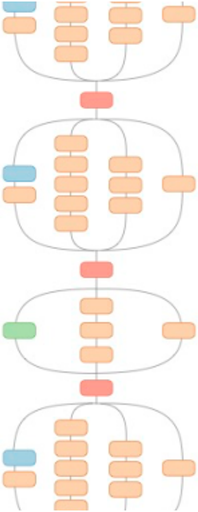
- Deep Neural Networks have the tendency of being ... deep
- Easy to drown in the complexity of an architecture

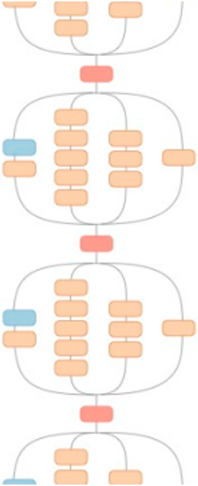
Visualization Tools

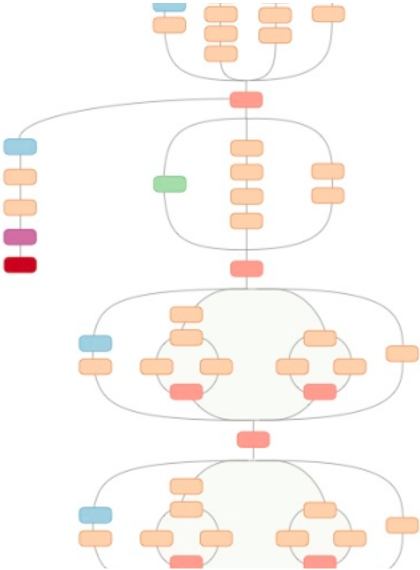
- Deep Neural Networks have the tendency of being . . . deep
- Easy to drown in the complexity of an architecture
- $>$ 36,000 nodes for Google's *Inception* model

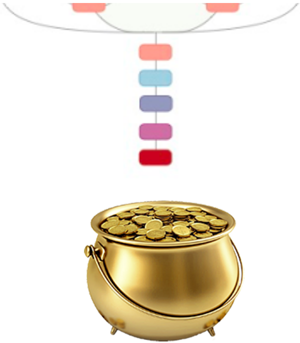
Visualization Tools

- Deep Neural Networks have the tendency of being . . . deep
- Easy to drown in the complexity of an architecture
- $>$ 36,000 nodes for Google's *Inception* model









TensorBoard to the Rescue

Walkthrough

How do I continue?

Resources

Resources

- MOOCs
  - Machine Learning by Andrew Ng @ Coursera
  - Deep Learning by Google @ Udacity
  - Machine Learning Nanodegree @ Udacity

Resources

- MOOCs
  - Machine Learning by Andrew Ng @ Coursera
  - Deep Learning by Google @ Udacity
  - Machine Learning Nanodegree @ Udacity
- Websites
  - http://colah.github.io
  - http://cs231n.github.io
  - http://karpathy.github.io
  - http://www.deeplearningbook.org
  - https://www.kaggle.com
  - https://www.tensorflow.org

PyCon Germany

- 29-30 October in Munich
- Talks on machine learning, deep learning and data science
- 15% off: VISITMUC16

Stay in Touch!

- peter@goldsborough.me
- linkedin.com/in/petergoldsborough
- github.com/goldsborough

Stay in Touch!

- peter@goldsborough.me
- linkedin.com/in/petergoldsborough
- github.com/goldsborough

github.com/peter-can-talk/pycon-uk-2016

Q & A