Introduction to Machine Learning feat. TensorFlow

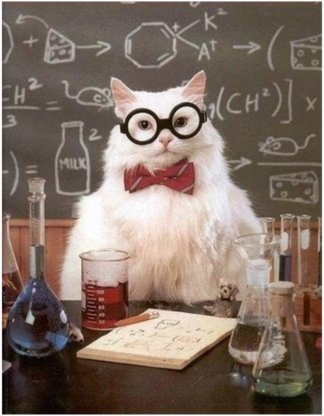
Peter Goldsborough

July 10, 2016

July 10, 2016

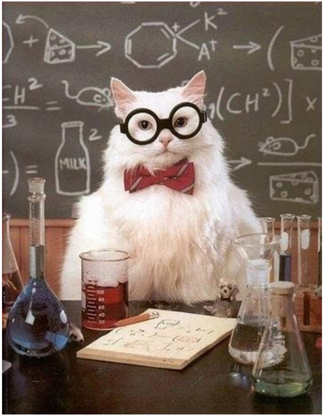
Table of Catents

Table of Catents

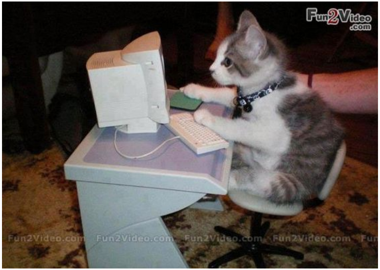


Theory

Table of Catents



Theory



Practice

Background

Background

- CS Student @ TUM

Background

- CS Student @ TUM
- Google & Bloomberg Intern

Background

- CS Student @ TUM
- Google & Bloomberg Intern
- I like cats

Background

- CS Student @ TUM
- Google & Bloomberg Intern
- I like cats

Seminar Topic: *Deep Learning With TensorFlow*

`github.com/peter-can-write/tensorflow-paper` github.com/peter-can-talk/python-meetup-munich-2016

Time to go Deeper

What if the meaning of life is to spend your time thinking about the meaning of life?

Deep Learning

Deep Learning

The why

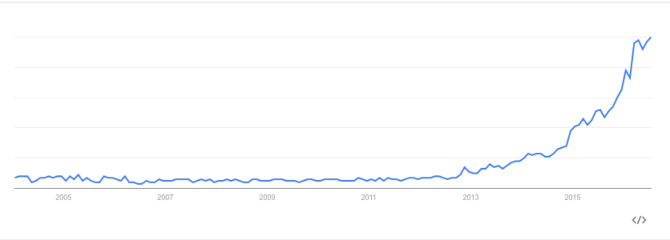
Deep Learning

The why, the what

Deep Learning

The why, the what and the ugly

Deep Learning



Basic Definition

Basic Definition

- Let $\mathcal{L}$ denote the number of hidden layers in a neural network

Basic Definition

- Let $\mathcal{L}$ denote the number of hidden layers in a neural network
- Then we call neural network *deep*, if

Basic Definition

- Let $\mathcal{L}$ denote the number of hidden layers in a neural network
- Then we call neural network *deep*, if
  1. It is trained at more than 10,000m below sea level, or

Basic Definition

- Let $\mathcal{L}$ denote the number of hidden layers in a neural network
- Then we call neural network *deep*, if
  1. It is trained at more than 10,000m below sea level, or
  2. $\mathcal{L} > 1$

Basic Definition

- Let $\mathcal{L}$ denote the number of hidden layers in a neural network
- Then we call neural network *deep*, if
  1. It is trained at more than 10,000m below sea level, or
  2. $\mathcal{L} > 1$
- To really understand why many layers are a good idea, we must understand why few layers might be a bad idea

Universal Approximation Theorem

Universal Approximation Theorem

- Theoretically, a single layer is just enough

Universal Approximation Theorem

- Theoretically, a single layer is just enough
- However, we would need exponentially many units for discrete functions

Universal Approximation Theorem

- Theoretically, a single layer is just enough
- However, we would need exponentially many units for discrete functions
- And infinitely many for continuous functions

Universal Approximation Theorem

- Theoretically, a single layer is just enough
- However, we would need exponentially many units for discrete functions
- And infinitely many for continuous functions
- Just use an infinitely sized hash table

The Curse of Dimensionality

The Curse of Dimensionality

- The number of possible outputs scales exponentially with the dimensionality of our dataset

The Curse of Dimensionality

- The number of possible outputs scales exponentially with the dimensionality of our dataset
- Assume our features have 10 possible values

The Curse of Dimensionality

- The number of possible outputs scales exponentially with the dimensionality of our dataset
- Assume our features have 10 possible values

1 feature $= 10$ outputs

The Curse of Dimensionality

- The number of possible outputs scales exponentially with the dimensionality of our dataset
- Assume our features have 10 possible values

1 feature $= 10$ outputs

The Curse of Dimensionality

- The number of possible outputs scales exponentially with the dimensionality of our dataset
- Assume our features have 10 possible values

1 feature $= 10$ outputs

The Curse of Dimensionality

- The number of possible outputs scales exponentially with the dimensionality of our dataset
- Assume our features have 10 possible values

1 feature $= 10$ outputs

The Curse of Dimensionality

- The number of possible outputs scales exponentially with the dimensionality of our dataset
- Assume our features have 10 possible values

1 feature $= 10$ outputs

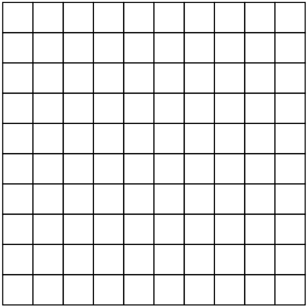
The Curse of Dimensionality

- The number of possible outputs scales exponentially with the dimensionality of our dataset
- Assume our features have 10 possible values

1 feature $= 10$ outputs

The Curse of Dimensionality

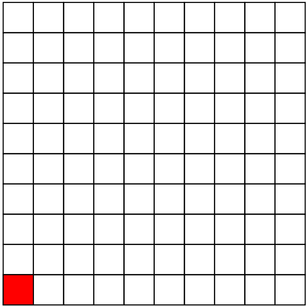
- The number of possible outputs scales exponentially with the dimensionality of our dataset
- Assume our features have 10 possible values



2 features $= 100$ outputs

The Curse of Dimensionality

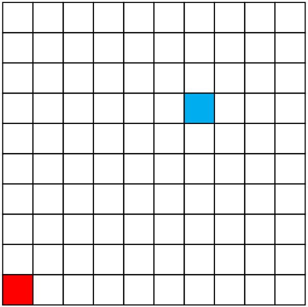
- The number of possible outputs scales exponentially with the dimensionality of our dataset
- Assume our features have 10 possible values



2 features $= 100$ outputs

The Curse of Dimensionality

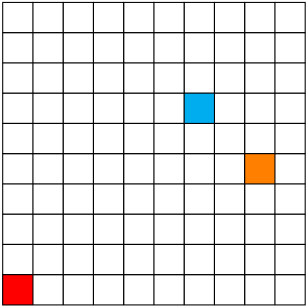
- The number of possible outputs scales exponentially with the dimensionality of our dataset
- Assume our features have 10 possible values



2 features $= 100$ outputs

The Curse of Dimensionality

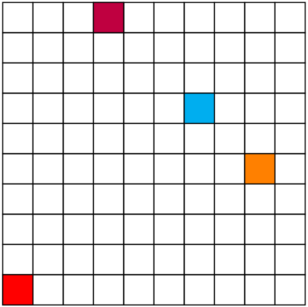
- The number of possible outputs scales exponentially with the dimensionality of our dataset
- Assume our features have 10 possible values



2 features $= 100$ outputs

The Curse of Dimensionality

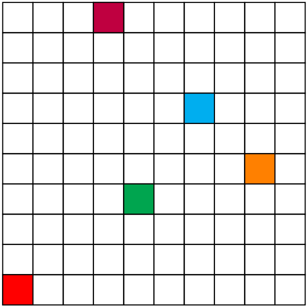
- The number of possible outputs scales exponentially with the dimensionality of our dataset
- Assume our features have 10 possible values



2 features $= 100$ outputs

The Curse of Dimensionality

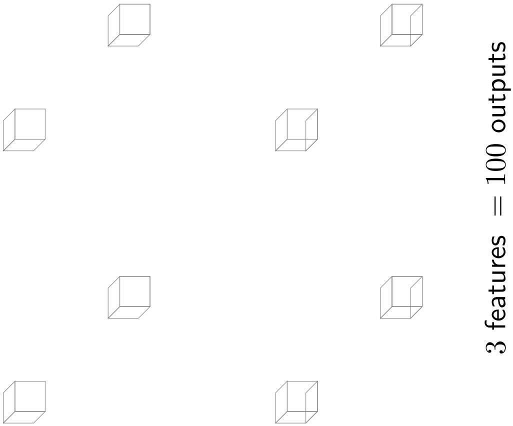
- The number of possible outputs scales exponentially with the dimensionality of our dataset
- Assume our features have 10 possible values



2 features $= 100$ outputs

The Curse of Dimensionality

- The number of possible outputs scales exponentially with the dimensionality of our dataset
- Assume our features have 10 possible values



The Curse of Dimensionality

- Why is this a problem?

The Curse of Dimensionality

- Why is this a problem?
- To predict an output, we must have seen at least one example

The Curse of Dimensionality

- Why is this a problem?
- To predict an output, we must have seen at least one example
- In high dimensions, an algorithm cannot possibly be trained on all possible output

The Curse of Dimensionality

- Why is this a problem?
- To predict an output, we must have seen at least one example
- In high dimensions, an algorithm cannot possibly be trained on all possible output, unless

The Curse of Dimensionality

- Why is this a problem?
- To predict an output, we must have seen at least one example
- In high dimensions, an algorithm cannot possibly be trained on all possible output, unless

We make assumptions about our data

The Curse of Dimensionality

- Assumptions either

The Curse of Dimensionality

- Assumptions either
  - Explicitly reduce the output space, or

The Curse of Dimensionality

- Assumptions either
  - Explicitly reduce the output space, or
  - Create dependencies between outputs

The Curse of Dimensionality

- Assumptions either
  - Explicitly reduce the output space, or
  - Create dependencies between outputs
- *Local constancy* assumption (prior)

$$f^\star(\mathbf{x}) \approx f^\star(\mathbf{x} + \varepsilon)$$

The Curse of Dimensionality

- Assumptions either
  - Explicitly reduce the output space, or
  - Create dependencies between outputs
- *Local constancy* assumption (prior)

$$f^{\star}(\mathbf{x}) \approx f^{\star}(\mathbf{x} + \varepsilon)$$

The Curse of Dimensionality

- Assumptions either
  - Explicitly reduce the output space, or
  - Create dependencies between outputs
- *Local constancy* assumption (prior)

$$f^\star(\mathbf{x}) \approx f^\star(\mathbf{x} + \varepsilon)$$

The Curse of Dimensionality

- Assumptions either
  - Explicitly reduce the output space, or
  - Create dependencies between outputs
- *Local constancy* assumption (prior)

$$f^{\star}(\mathbf{x}) \approx f^{\star}(\mathbf{x} + \varepsilon)$$

- Don't need as much data any more!

The Curse of Dimensionality

Deep Learning assumes that data is structured

The Curse of Dimensionality

Deep Learning assumes that data is structured

hierarchically

Deep Learning assumes that data is structured

hierarchically

Note:
According to the *No Free Lunch Theorem* it is just as bad as flipping a coin.

Convolutional Neural Networks

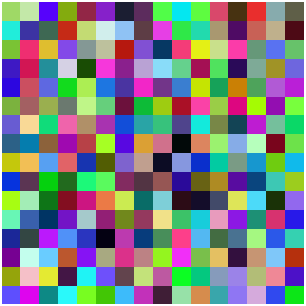
- Convolutional Neural Networks (CNNs; ConvNets) are used in image recognition

Convolutional Neural Networks

- Convolutional Neural Networks (CNNs; ConvNets) are used in image recognition
- They assume their input are images

Convolutional Neural Networks

- Convolutional Neural Networks (CNNs; ConvNets) are used in image recognition
- They assume their input are images



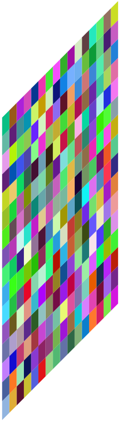
Convolutional Neural Networks

- Convolutional Neural Networks (CNNs; ConvNets) are used in image recognition
- They assume their input are images

71	76	55	37	72	15	90	69	62	79	50	99	15	14	18	97
24	4	35	87	87	58	34	92	97	63	55	95	79	95	24	6
92	93	68	61	57	66	2	93	79	57	33	14	46	49	62	34
5	60	69	40	79	16	59	79	87	51	83	66	81	54	56	12
75	14	65	56	63	99	23	21	33	87	71	48	70	85	64	85
81	92	77	85	91	39	57	47	92	2	69	84	80	95	21	84
49	0	82	22	89	8	37	52	62	54	23	82	52	53	32	91
76	33	64	43	98	83	69	50	9	67	93	30	21	32	13	71
29	85	34	67	42	39	61	4	52	29	14	8	0	10	34	79
63	22	95	87	13	41	98	40	8	50	95	90	49	57	44	19
74	23	79	80	58	2	64	8	30	21	62	47	7	95	21	46
93	41	73	46	76	34	44	74	6	79	15	12	25	64	21	10
41	51	32	98	6	37	77	7	92	54	1	8	3	58	99	49
77	92	49	88	36	56	60	63	62	59	45	41	38	8	87	1
74	99	60	61	73	55	3	23	37	84	94	95	40	80	78	84
60	57	14	60	65	32	2	13	23	23	46	31	30	7	52	42

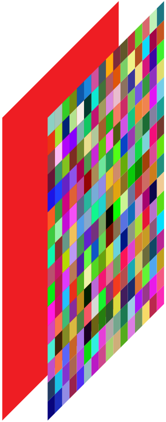
Convolutional Neural Networks

- Convolutional Neural Networks (CNNs; ConvNets) are used in image recognition
- They assume their input are images



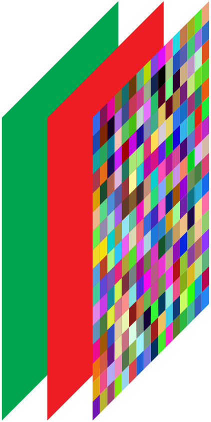
Convolutional Neural Networks

- Convolutional Neural Networks (CNNs; ConvNets) are used in image recognition
- They assume their input are images



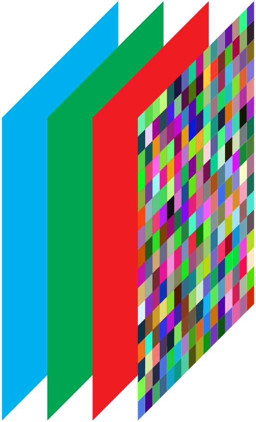
Convolutional Neural Networks

- Convolutional Neural Networks (CNNs; ConvNets) are used in image recognition
- They assume their input are images



Convolutional Neural Networks

- Convolutional Neural Networks (CNNs; ConvNets) are used in image recognition
- They assume their input are images



Convolutional Neural Network

- We want to classify images into one of k classes

Convolutional Neural Network

- We want to classify images into one of k classes
- The model should learn to extract features

Convolutional Neural Network

- We want to classify images into one of k classes
- The model should learn to extract features
- We expect it to exploit the hierarchical nature of the data

Convolutional Neural Network

- We want to classify images into one of k classes
- The model should learn to extract features
- We expect it to exploit the hierarchical nature of the data
    1. Lines and edges
    2. Corners and contours
    3. Abstract components (e.g. noses, ears, feet)
    4. Entire objects (e.g. faces, spaceship)

Convolutional Neural Network

- We want to classify images into one of k classes
- The model should learn to extract features
- We expect it to exploit the hierarchical nature of the data
  1. Lines and edges
  2. Corners and contours
  3. Abstract components (e.g. noses, ears, feet)
  4. Entire objects (e.g. faces, spaceship)
- First idea: just feed the pixels into a neural network

$$\underset{R}{\begin{bmatrix} 116 & 80 \\ 170 & 194 \end{bmatrix}} \quad \underset{G}{\begin{bmatrix} 82 & 78 \\ 5 & 236 \end{bmatrix}} \quad \underset{B}{\begin{bmatrix} 76 & 139 \\ 245 & 236 \end{bmatrix}}$$

Convolutional Neural Network

$$\underset{R}{\begin{bmatrix} 116 & 80 \\ 170 & 194 \end{bmatrix}} \quad \underset{G}{\begin{bmatrix} 82 & 78 \\ 5 & 236 \end{bmatrix}} \quad \underset{B}{\begin{bmatrix} 76 & 139 \\ 245 & 236 \end{bmatrix}}$$

$$\texttt{concatenate(}R\texttt{.flatten, }G\texttt{.flatten, }B\texttt{.flatten)}$$

Convolutional Neural Network

$$\underset{R}{\begin{bmatrix} 116 & 80 \\ 170 & 194 \end{bmatrix}} \quad \underset{G}{\begin{bmatrix} 82 & 78 \\ 5 & 236 \end{bmatrix}} \quad \underset{B}{\begin{bmatrix} 76 & 139 \\ 245 & 236 \end{bmatrix}}$$

$$R.\texttt{flatten} = \begin{bmatrix} 116 & 80 & 170 & 194 \end{bmatrix}$$

Convolutional Neural Network

$$\underset{R}{\begin{bmatrix} 116 & 80 \\ 170 & 194 \end{bmatrix}} \quad \underset{G}{\begin{bmatrix} 82 & 78 \\ 5 & 236 \end{bmatrix}} \quad \underset{B}{\begin{bmatrix} 76 & 139 \\ 245 & 236 \end{bmatrix}}$$

$$R.\texttt{flatten} = \begin{bmatrix} 116 & 80 & 170 & 194 \end{bmatrix}$$
$$G.\texttt{flatten} = \begin{bmatrix} 82 & 78 & 5 & 236 \end{bmatrix}$$

Convolutional Neural Network

$$\underset{R}{\begin{bmatrix} 116 & 80 \\ 170 & 194 \end{bmatrix}} \qquad \underset{G}{\begin{bmatrix} 82 & 78 \\ 5 & 236 \end{bmatrix}} \qquad \underset{B}{\begin{bmatrix} 76 & 139 \\ 245 & 236 \end{bmatrix}}$$

$$R.\texttt{flatten} = \begin{bmatrix} 116 & 80 & 170 & 194 \end{bmatrix}$$
$$G.\texttt{flatten} = \begin{bmatrix} 82 & 78 & 5 & 236 \end{bmatrix}$$
$$B.\texttt{flatten} = \begin{bmatrix} 76 & 139 & 245 & 236 \end{bmatrix}$$

Convolutional Neural Network

$$\underset{R}{\begin{bmatrix}116 & 80\\170 & 194\end{bmatrix}} \qquad \underset{G}{\begin{bmatrix}82 & 78\\5 & 236\end{bmatrix}} \qquad \underset{B}{\begin{bmatrix}76 & 139\\245 & 236\end{bmatrix}}$$

$$R.\texttt{flatten} = \begin{bmatrix}116 & 80 & 170 & 194\end{bmatrix}$$
$$G.\texttt{flatten} = \begin{bmatrix}82 & 78 & 5 & 236\end{bmatrix}$$
$$B.\texttt{flatten} = \begin{bmatrix}76 & 139 & 245 & 236\end{bmatrix}$$

$$\mathbf{x} = \left[116, 80, 170, 194, 82, 78, 5, 236, 76, 139, 245, 236\right]$$

Convolutional Neural Networks

- Why is this a bad idea?

Convolutional Neural Networks

- Why is this a bad idea?
  1. Fully connected NNs scale badly for images

Convolutional Neural Networks

- Why is this a bad idea?
  1. Fully connected NNs scale badly for images
     - $200 \times 200 \times 3 = 120,000$ features

Convolutional Neural Networks

- Why is this a bad idea?
  1. Fully connected NNs scale badly for images
     - $200 \times 200 \times 3 = 120,000$ features
     - With 100 hidden units: $100 \times 120,000 = 12,000,000$

Convolutional Neural Networks

- Why is this a bad idea?
  1. Fully connected NNs scale badly for images
     - $200 \times 200 \times 3 = 120,000$ features
     - With 100 hidden units: $100 \times 120,000 = 12,000,000$
     - With 5 layers: $12,000,000 \times 5 = 60,000,000$ weights

Convolutional Neural Networks

- Why is this a bad idea?
  1. Fully connected NNs scale badly for images
     - $200 \times 200 \times 3 = 120,000$ features
     - With 100 hidden units: $100 \times 120,000 = 12,000,000$
     - With 5 layers: $12,000,000 \times 5 = 60,000,000$ weights
     - $1000 \times 1000 \rightarrow 1,500,000,000$ weights

Convolutional Neural Networks

- Why is this a bad idea?
  1. Fully connected NNs scale badly for images
     - $200 \times 200 \times 3 = 120,000$ features
     - With 100 hidden units: $100 \times 120,000 = 12,000,000$
     - With 5 layers: $12,000,000 \times 5 = 60,000,000$ weights
     - $1000 \times 1000 \rightarrow 1,500,000,000$ weights
  2. It assumes every pixel has entirely new information

Convolutional Neural Networks

This is a cat ♥

Convolutional Neural Networks

Still a cat ♥♥

Convolutional Neural Networks

Half cat / half salad ♥♥♥

Convolutional Neural Networks

Minecraft cat ♥♥♥♥

Convolutional Neural Networks

- Our network learns to detect a feature in one part of the image

Convolutional Neural Networks

- Our network learns to detect a feature in one part of the image
- Wouldn't it make sense to reuse that information?

Convolutional Neural Networks

- Our network learns to detect a feature in one part of the image
- Wouldn't it make sense to reuse that information?
- Yes!

Weight Sharing

Convolutional Neural Network: Mechanics

Recipe for a Convolutional Neural Network

Recipe for a Convolutional Neural Network

- Ingredients

Recipe for a Convolutional Neural Network

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

Recipe for a Convolutional Neural Network

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

Recipe for a Convolutional Neural Network

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

Recipe for a Convolutional Neural Network

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

Recipe for a Convolutional Neural Network

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

Recipe for a Convolutional Neural Network

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

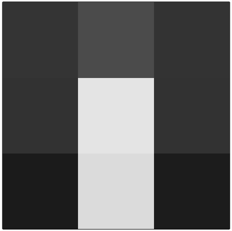
Recipe for a Convolutional Neural Network

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

Recipe for a Convolutional Neural Network

- Ingredients
  1. Image I with dimension $w \times h \times d$
  2. A kernel (filter) K of size $k \times k \times d$
- 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
  - (This is a convolution $I * K$)

Convolutional Neural Network: Mechanics



Image

Convolutional Neural Network: Mechanics

0.4	0.9	0.1
0.7	0.2	0.6
0.8	0.3	0.5

Image

Convolutional Neural Network: 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 Network: 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

Convolutional Neural Network: Mechanics

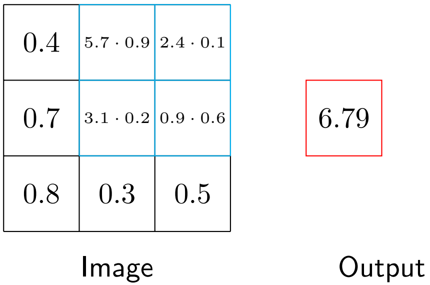
$5.7 \cdot 0.4$	$2.4 \cdot 0.9$	0.1
$3.1 \cdot 0.7$	$0.9 \cdot 0.2$	0.6
0.8	0.3	0.5

Image

6.79

Output

Convolutional Neural Network: Mechanics



Convolutional Neural Network: 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 Network: 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

Output

Convolutional Neural Network: 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 Network: 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	

Output

Convolutional Neural Network: Mechanics

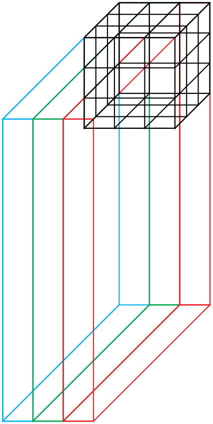
0.4	0.9	0.1
0.7	$5.7 \cdot 0.2$	$2.4 \cdot 0.6$
0.8	$3.1 \cdot 0.3$	$0.9 \cdot 0.5$

Image

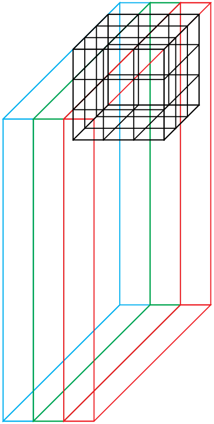
6.79	6.53
7.67	3.96

Output

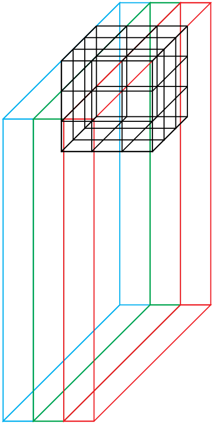
Convolutional Neural Networks: Mechanics



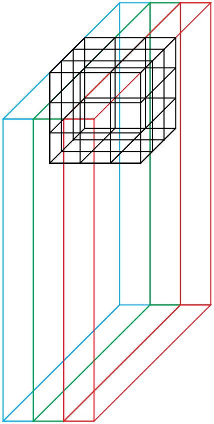
Convolutional Neural Networks: Mechanics



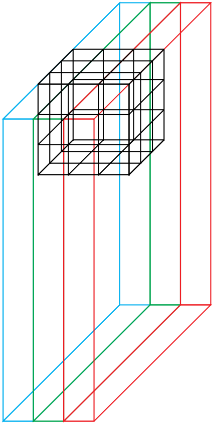
Convolutional Neural Networks: Mechanics



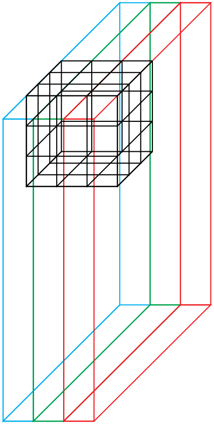
Convolutional Neural Networks: Mechanics



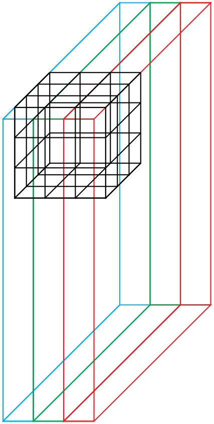
Convolutional Neural Networks: Mechanics



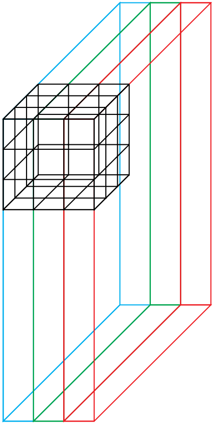
Convolutional Neural Networks: Mechanics



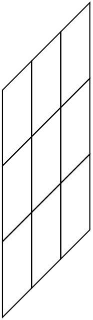
Convolutional Neural Networks: Mechanics



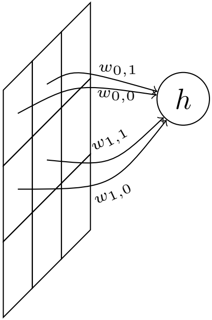
Convolutional Neural Networks: Mechanics



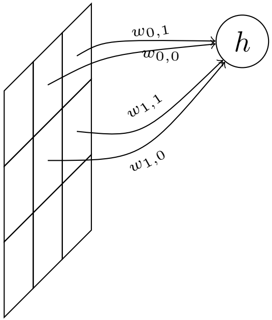
Convolutional Neural Network: Mechanics



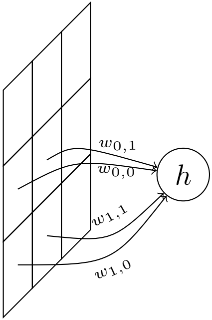
Convolutional Neural Network: Mechanics



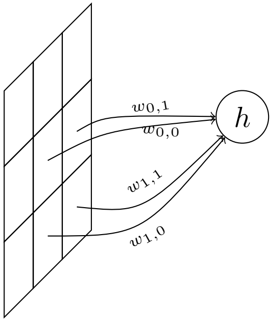
Convolutional Neural Network: Mechanics



Convolutional Neural Network: Mechanics



Convolutional Neural Network: Mechanics



Convolutional Neural Network: Hyperparameters

- Convolutional Neural Networks have three hyperparameters

Convolutional Neural Network: Hyperparameters

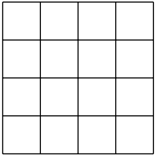
- Convolutional Neural Networks have three hyperparameters
  1. Kernel size

Convolutional Neural Network: Hyperparameters

- Convolutional Neural Networks have three hyperparameters
  1. Kernel size
  2. Kernel stride

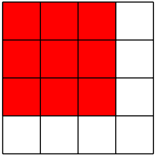
Convolutional Neural Network: Hyperparameters

- Convolutional Neural Networks have three hyperparameters
  1. Kernel size
  2. Kernel stride
  3. Padding (valid or same)



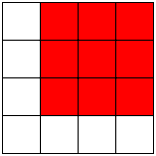
Convolutional Neural Network: Hyperparameters

- Convolutional Neural Networks have three hyperparameters
  1. Kernel size
  2. Kernel stride
  3. Padding (valid or same)



Convolutional Neural Network: Hyperparameters

- Convolutional Neural Networks have three hyperparameters
  1. Kernel size
  2. Kernel stride
  3. Padding (valid or same)



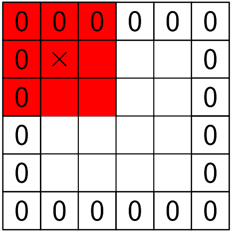
Convolutional Neural Network: Hyperparameters

- Convolutional Neural Networks have three hyperparameters
  1. Kernel size
  2. Kernel stride
  3. Padding (valid or same)

0	0	0	0	0	0
0					0
0					0
0					0
0					0
0	0	0	0	0	0

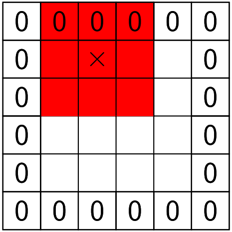
Convolutional Neural Network: Hyperparameters

- Convolutional Neural Networks have three hyperparameters
  1. Kernel size
  2. Kernel stride
  3. Padding (valid or same)



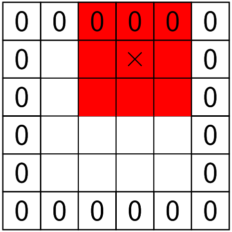
Convolutional Neural Network: Hyperparameters

- Convolutional Neural Networks have three hyperparameters
  1. Kernel size
  2. Kernel stride
  3. Padding (valid or same)



Convolutional Neural Network: Hyperparameters

- Convolutional Neural Networks have three hyperparameters
  1. Kernel size
  2. Kernel stride
  3. Padding (valid or same)



Convolutional Neural Network: Hyperparameters

- Convolutional Neural Networks have three hyperparameters
  1. Kernel size
  2. Kernel stride
  3. Padding (valid or same)

0	0	0	0	0	0
0				×	0
0					0
0					0
0					0
0	0	0	0	0	0

Convolutional Neural Networks: Pooling

- *Pooling* achieves translational invariance

Convolutional Neural Networks: Pooling

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

Convolutional Neural Networks: Pooling

- *Pooling* achieves translational invariance
- A form of downsampling
- The maximum stays the maximum

66	2
6	32

Convolutional Neural Networks: Pooling

- *Pooling* achieves translational invariance
- A form of downsampling
- The maximum stays the maximum

6	2
66	32

Convolutional Neural Networks: Pooling

- *Pooling* achieves translational invariance
- A form of downsampling
- The maximum stays the maximum

2	66
6	32

Convolutional Neural Networks: Pooling

- *Pooling* achieves translational invariance
- A form of downsampling
- The maximum stays the maximum

32	6
2	66

Convolutional Neural Network: Architecture

```
INPUT ->
[[CONV -> RELU]*N -> POOL?]*M ->
[FC -> RELU]*K -> FC
```

[?]

Convolutional Neural Networks: Intuition

Convolutional Neural Networks: Intuition

- Each layer of a ConvNet learns to detect features

Convolutional Neural Networks: Intuition

- Each layer of a ConvNet learns to detect features
- Later layers combine features of earlier layers

Convolutional Neural Networks: Intuition

- Each layer of a ConvNet learns to detect features
- Later layers combine features of earlier layers
- For early layers, we can still gain an intuition of what they do

What's in a kernel?

0	1	0
0	1	0
0	1	0

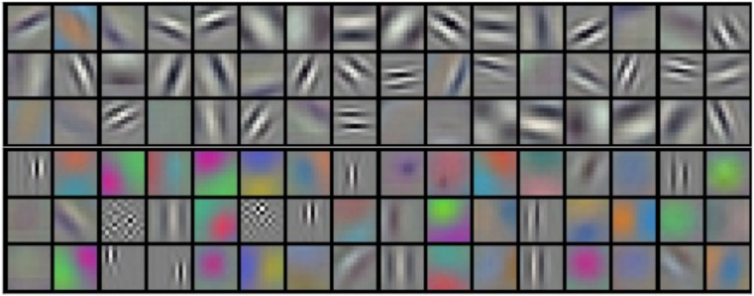
Patterns

What's in a kernel?

0	0	0
-1	1	0
0	0	0

Features

Convolutional Neural Networks: Intuition



Recurrent Neural Networks

LSTMs

Deep Learning

- Why the recent success of deep learning?

Deep Learning

- Why the recent success of deep learning?
- Three reasons

Deep Learning

- Why the recent success of deep learning?
- Three reasons
    1. Better hardware

Deep Learning

- Why the recent success of deep learning?
- Three reasons
  1. Better hardware
  2. More data

Deep Learning

- Why the recent success of deep learning?
- Three reasons
  1. Better hardware
  2. More data
  3. Better methods

The Ugly

References