Convolutional neural networks: Finding Santa in Images

December 17, 2018

FLATIRON SCHOOL

Who am I?

- Lore Dirick
  - Lore → not “Lohr” or “Lori” but “Lora”
- PhD in Business Economics, KU Leuven (Belgium)
- Worked as a curriculum developer / curriculum lead at DataCamp (Boston/NYC)
- Joined Flatiron School as a Senior Data Science Curriculum Developer in NYC in April 2018

About Flatiron School

- Coding bootcamp for web development, recently added data science course offerings
  - Intro to data science (60 hours)
  - Data Science bootcamp (15 weeks)
- On-line and in-person
- https://flatironschool.com/
- Several scholarship programmes to make gender parity in tech a reality (https://flatironschool.com/scholarships/women-take-tech/)
- We’re hiring!! https://flatironschool.com/careers/

Outline

- Landscape of AI and where deep learning comes in
- How do computers read images? + how to in Keras
- A densely connected network + how to in Keras
- Introduction to CNNs + how to in Keras

- Some questions:
    - Who has some experience with data science?
    - Who has worked with images before?
    - Who has worked with (densely connected) neural networks before?
    - Who has worked with convolutional neural networks before?

Main resources

- Andrew Ng’s Deep learning course in Coursera
- Francois Chollet “Deep Learning in Python”
  - He is also the author of Keras

For this tutorial

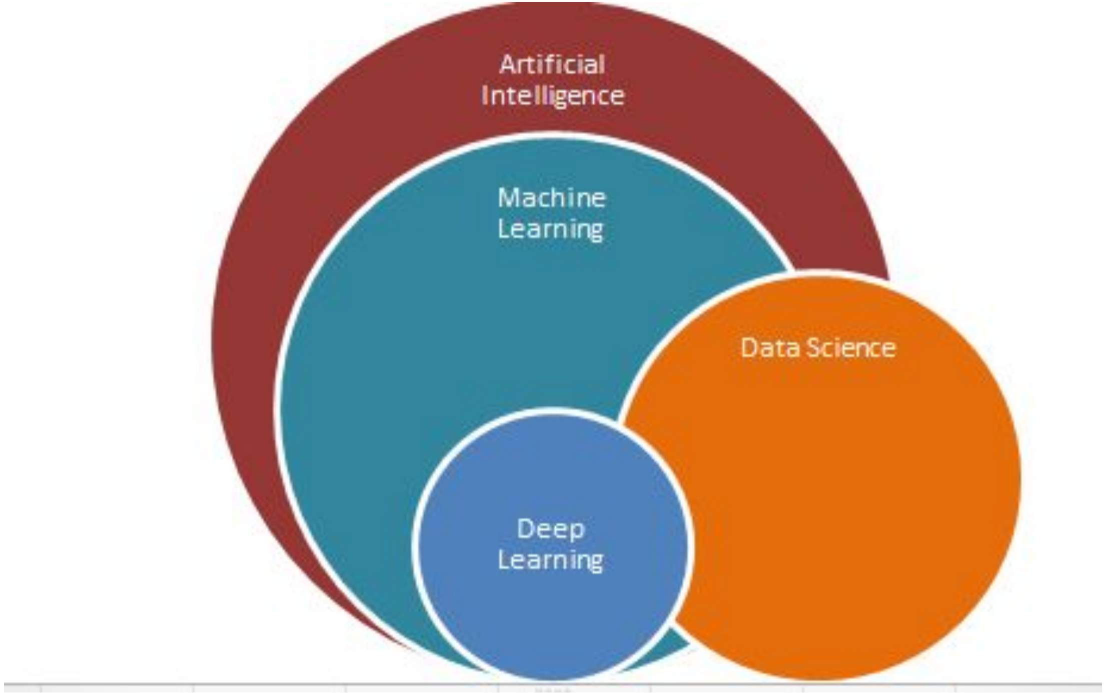
- https://github.com/LoreDirick/Meetup-Santa-Images
- Install Keras (https://keras.io/#installation) -- if you do not already have a preference we recommend choosing the TensorFlow backend.

About Keras

- Keras is a high-level neural networks API, written in Python
- Keras (currently) runs on three backend engines: TensorFlow, Theano, or CNTK.
- Allows the same code to run on CPU or GPU.
- Makes it easy to quickly prototype deep-learning models.
- Installation instructions: https://keras.io/

What is AI?

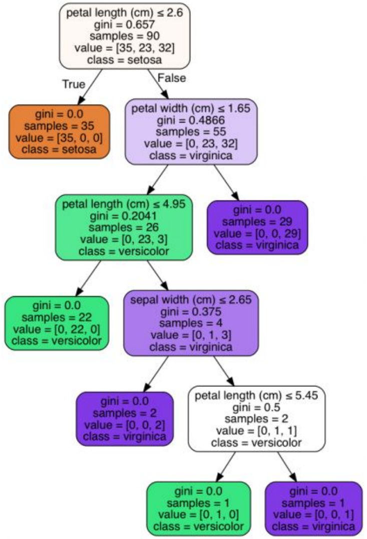
A branch of computer science dealing with the simulation of intelligent behavior in computers



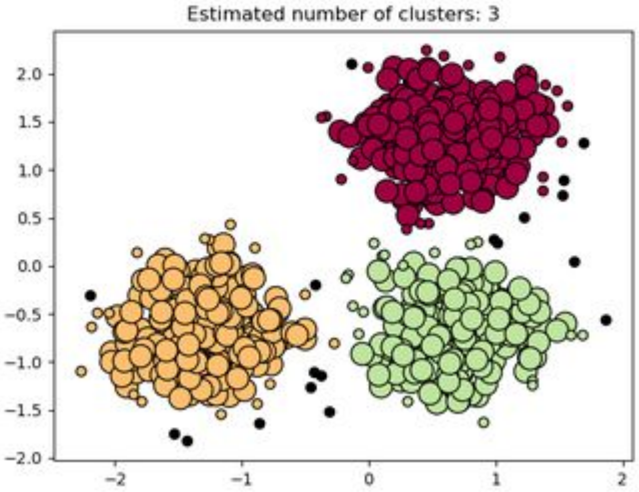
What is Machine Learning?

An algorithm that learns from experience with respect to a given task. Performance of the task improves as the algorithm gains more experience for that task.

Supervised Learning



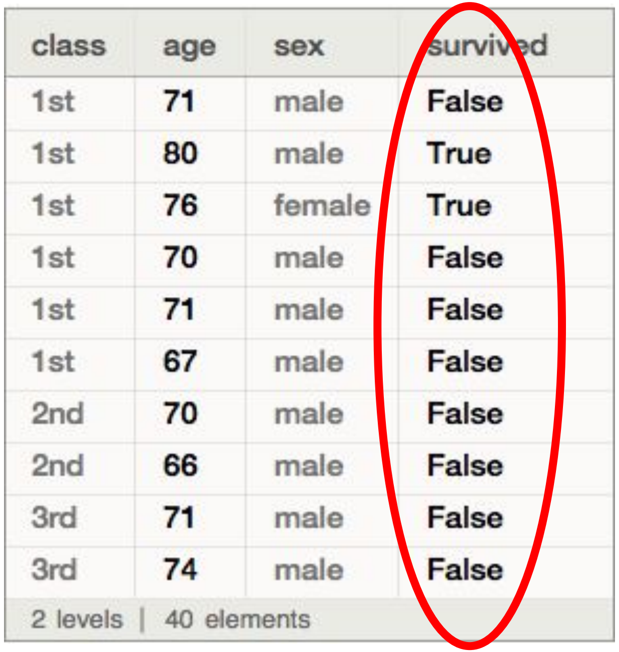
Unsupervised Learning



Supervised Learning

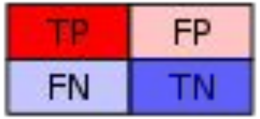
A subclass of Machine Learning algorithms that requires labeled training data.

Titanic Dataset



class	age	sex	survived
1st	71	male	False
1st	80	male	True
1st	76	female	True
1st	70	male	False
1st	71	male	False
1st	67	male	False
2nd	70	male	False
2nd	66	male	False
3rd	71	male	False
3rd	74	male	False

2 levels | 40 elements



TP	FP
FN	TN

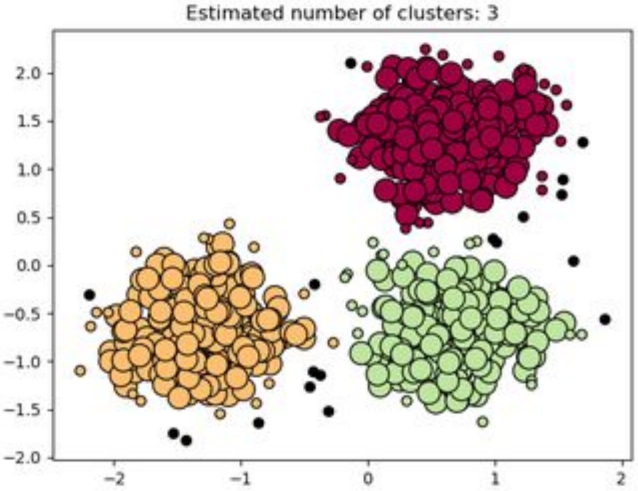
Questions It Can Answer:

- Classification--A or B?
- Regression--How much, or how many?
- Anomaly Detection--Is this weird?

Unsupervised Learning

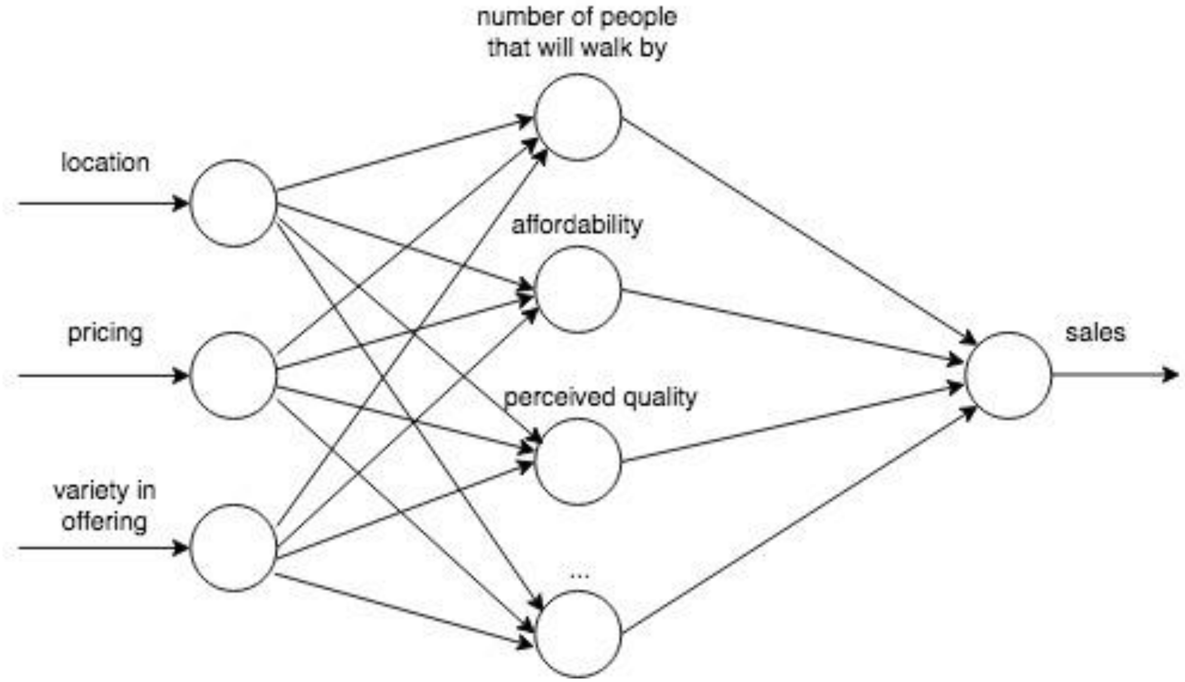
A subclass of machine learning algorithms that does not require labeled training data.

Clustering



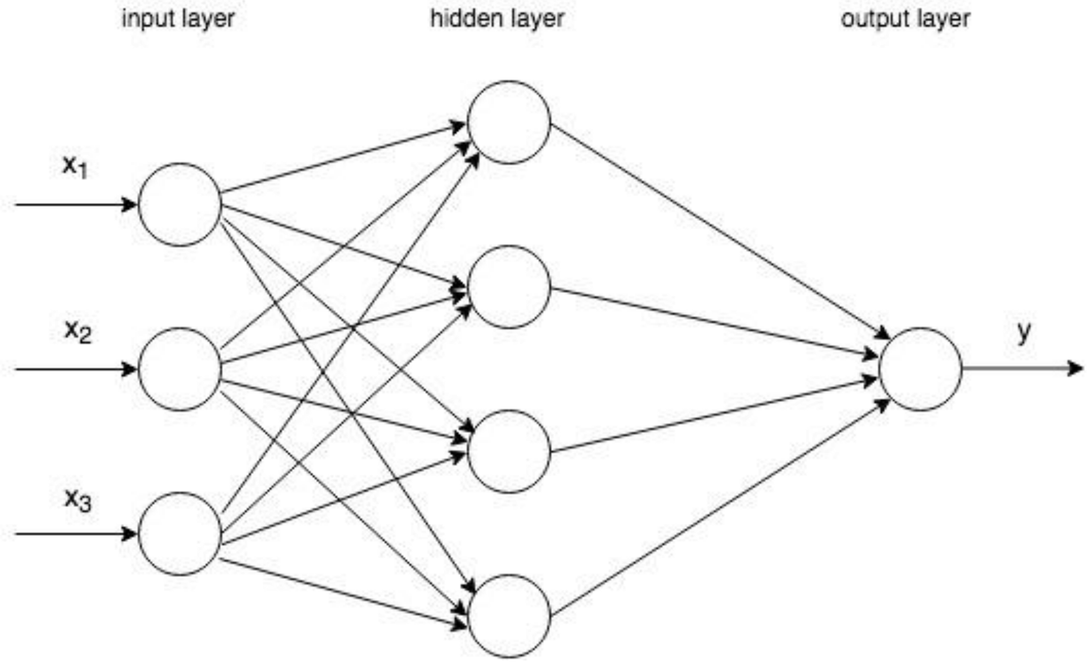
Deep Learning

Machine Learning algorithms that make use of Neural Networks with 1 or more “hidden” layers.



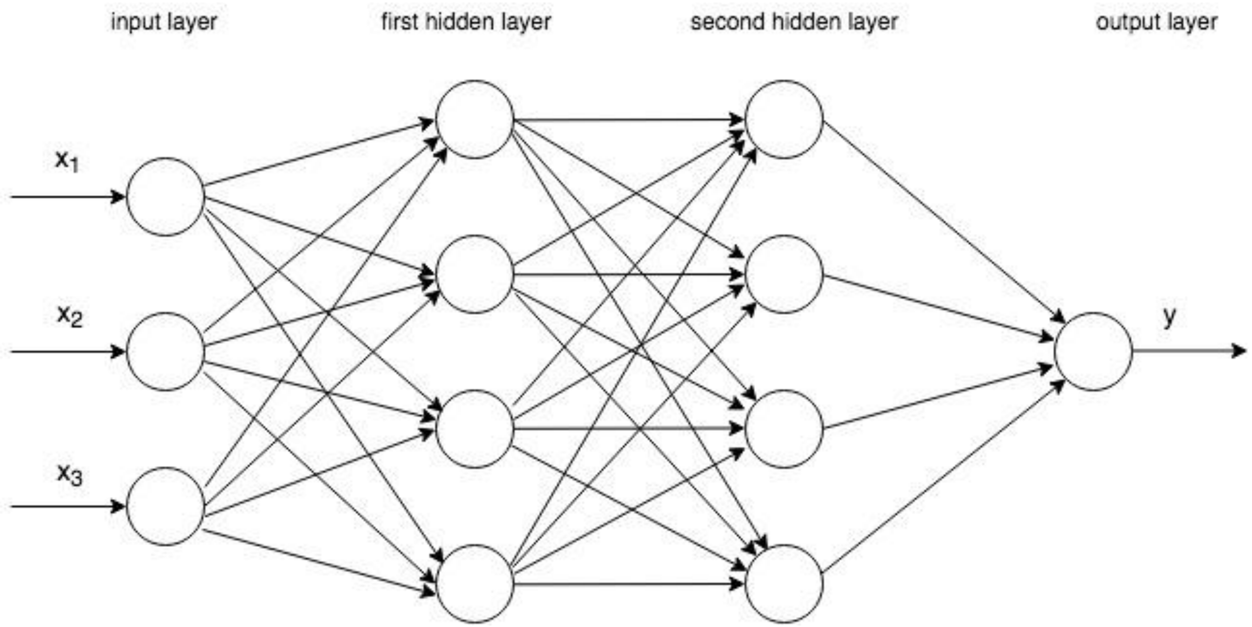
Deep Learning

Machine Learning algorithms that make use of Neural Networks with 1 or more “hidden” layers.



Deep Learning

Machine Learning algorithms that make use of Neural Networks with 1 or more “hidden” layers.



Deep Learning

Deep learning can deal extremely well with **unstructured data**.

Use Cases	**Example**
Computer Vision	Facial Recognition
Sequence Modeling	Google Translate
Natural Language Processing	Siri
Reinforcement Learning	Autonomous Vehicles

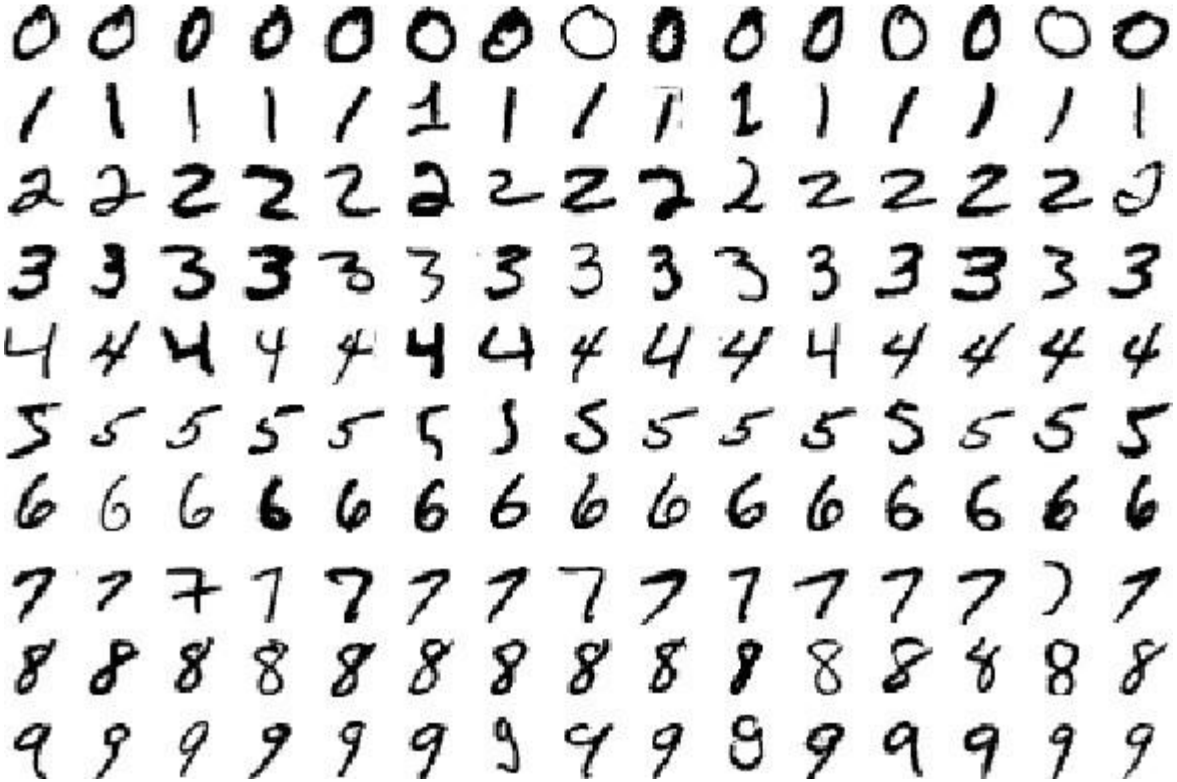
Deep Learning: strengths

- Can deal very well with unstructured data
- Extremely powerful
- Captures nonlinearity well
- Better than human performance on some tasks

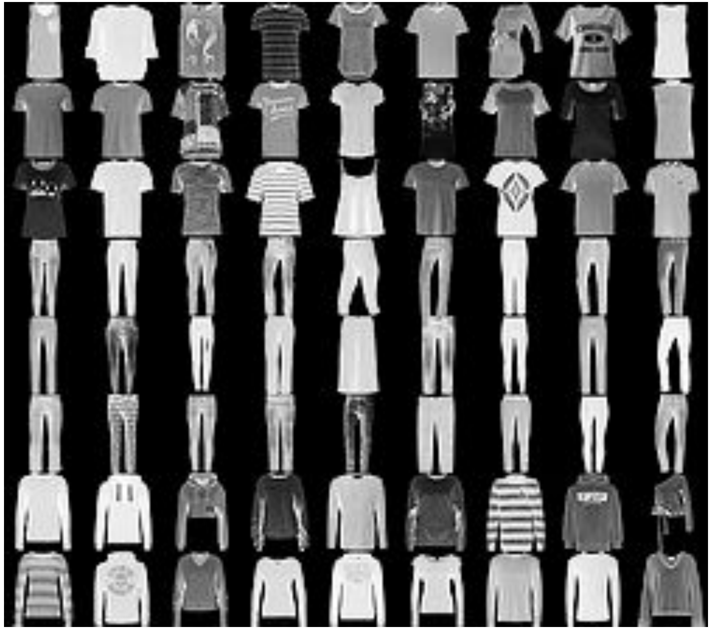
Deep Learning: drawbacks

- Data Hungry
- Computationally Intensive
- “Black Box” Model practically impossible to interpret how it arrived at results

Deep learning for image classification: MNIST



Another example: Fashion MNIST

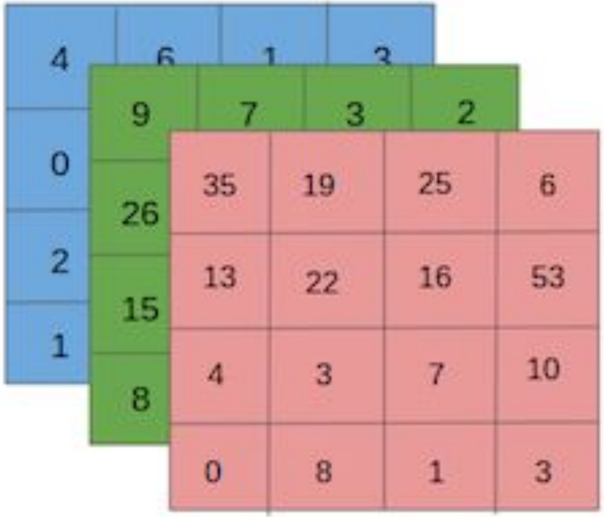


https://github.com/zalandoresearch/fashion-mnist

Deep learning for image classification

Santa or not?

How does a computer read an image?

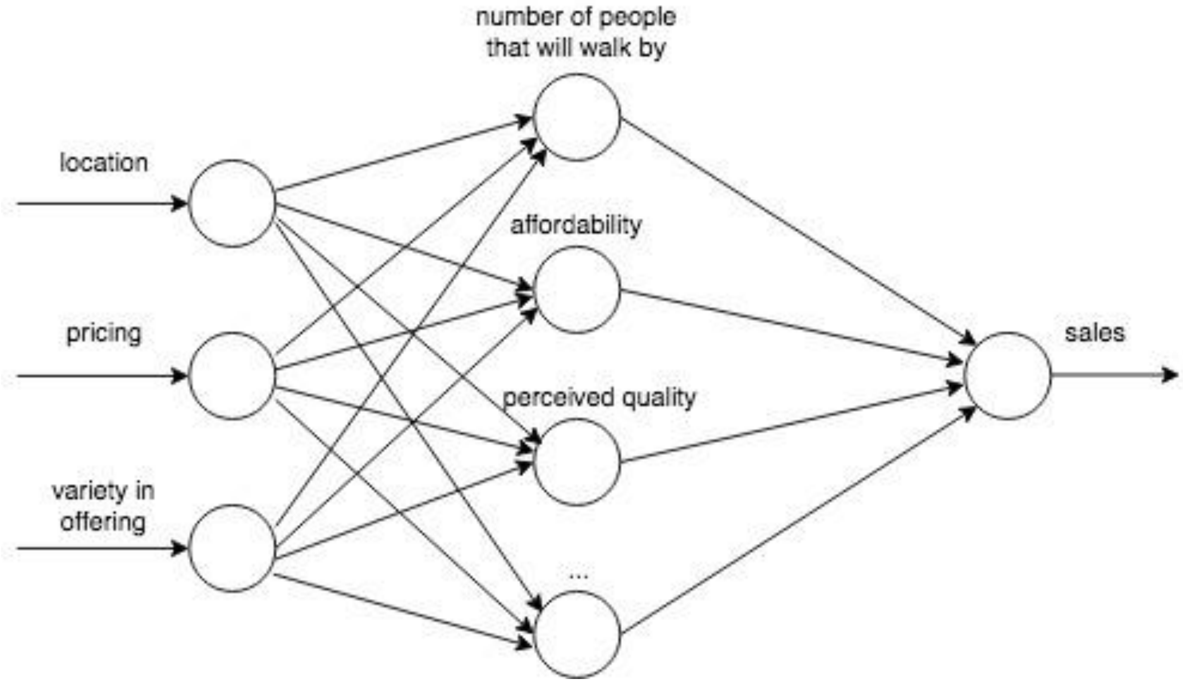


1 image is represented by :

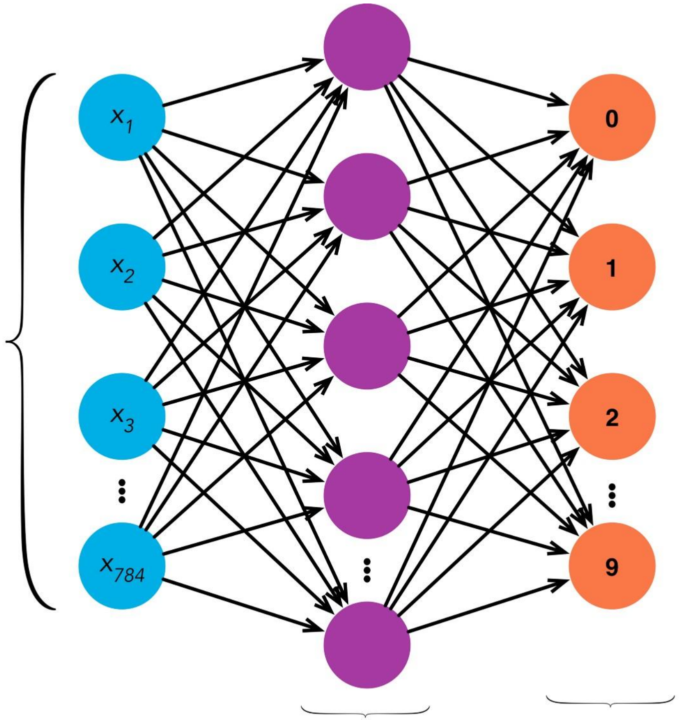
(n_pixels*n_pixels) * 3

Jupyter notebook

How to translate this for an image?



Deep learning for image classification: MNIST



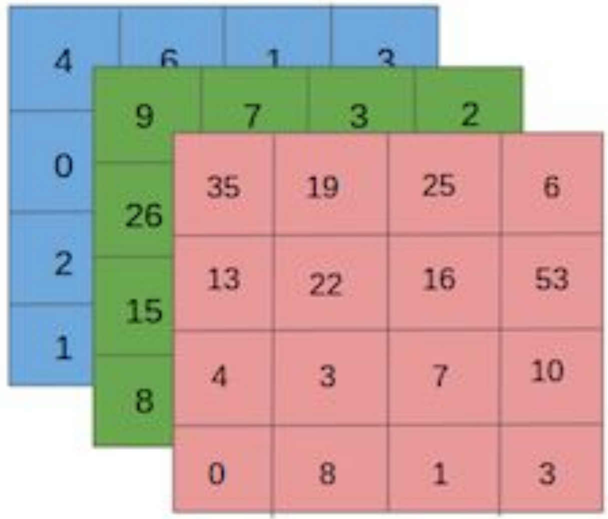
input layer: **784** (28x28) neurons, each with values between 0 and 255

Hidden Layer: ***16*** hidden neurons

Output Layer: ***10*** classifiers for 10 digits

Source: https://towardsdatascience.com/multi-layer-neural-networks-with-sigmoid-function-deep-learning-for-rookies-2-bf464f09eb7f

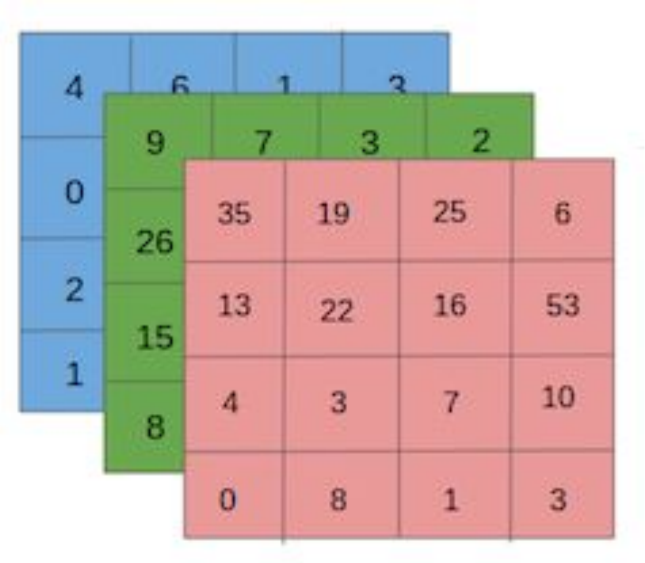
How does a computer read an image?



35	19	25	6
13	22	16	53
4	3	7	10
0	8	1	3

$$x = \begin{bmatrix} 35 \\ 19 \\ \vdots \\ 9 \\ 7 \\ \vdots \\ 4 \\ 6 \\ \vdots \end{bmatrix}$$

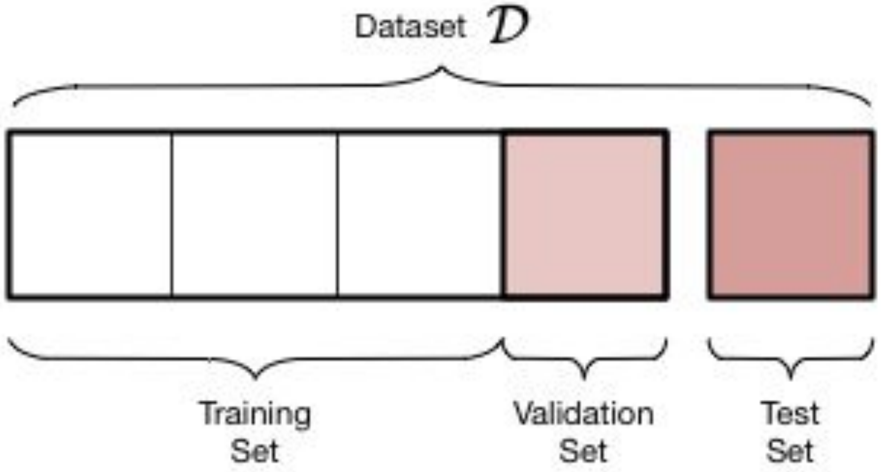
How does a computer read an image?



$$x=\begin{array}{c} \begin{array}{cccc} x^{(1)} & x^{(2)} & & x^{(l)} \end{array} \\ \begin{bmatrix} 35 & 23 & \cdots & 1 \\ 19 & 88 & \cdots & 230 \\ \vdots & \vdots & \ddots & \vdots \\ 9 & 3 & \cdots & 222 \\ 7 & 166 & \cdots & 43 \\ \vdots & \vdots & \ddots & \vdots \\ 4 & 202 & \cdots & 98 \\ 6 & 54 & \cdots & 100 \\ \vdots & \vdots & \ddots & \vdots \end{bmatrix} \end{array}$$

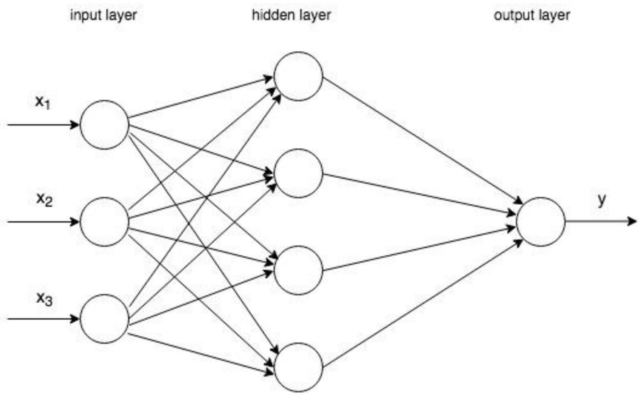
$$y=\begin{bmatrix} 1 & 0 & \cdots & 1 \end{bmatrix}$$

Model building



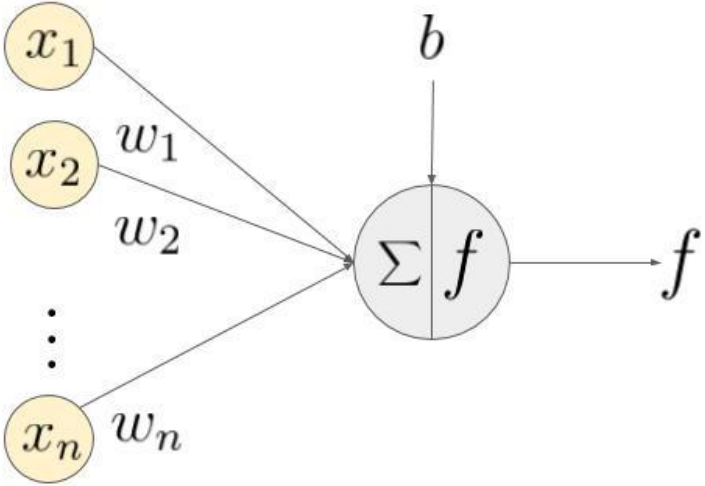
Jupyter notebook

A densely connected network



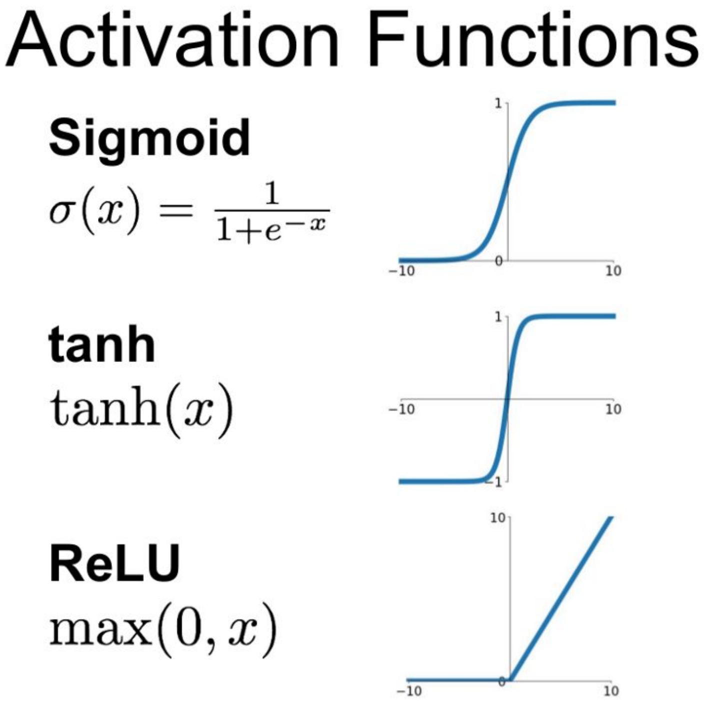
- How many layers/number of nodes in a layer
- Define an activation function (intuition: “activate” a neuron - 0/1)
- Loss function: How the network will be able to measure its performance on the training data, and steer itself in the right direction.
- An optimizer: the mechanism through which the network will update itself based on the data it sees and its loss function.

A densely connected network



$$f\left(b+\sum_{i=1}^{n} x_i w_i\right)$$

A densely connected network



Activation Functions

Sigmoid

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

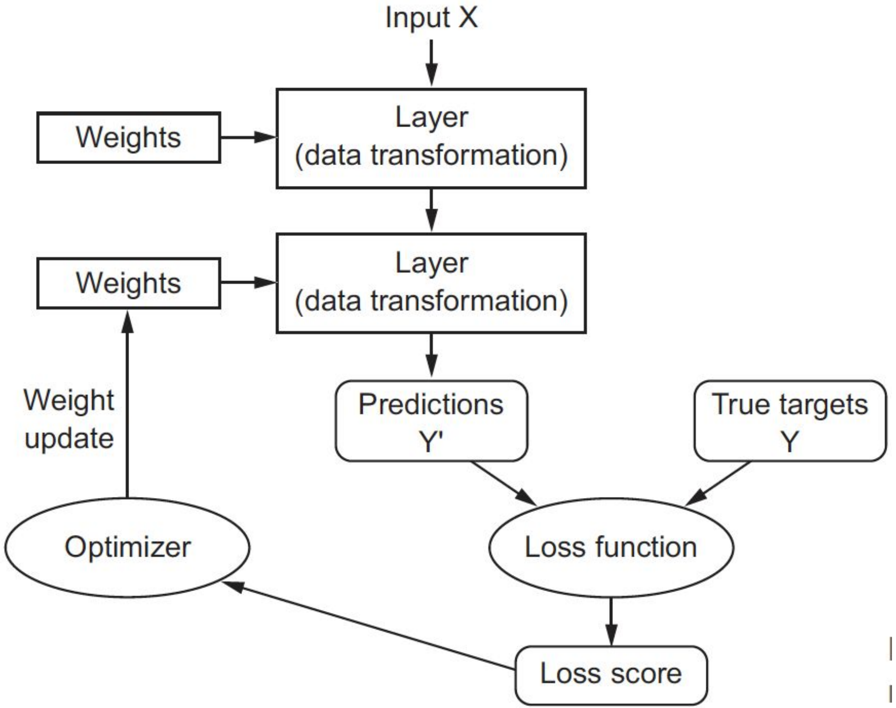
tanh

$\tanh(x)$

ReLU

$\max(0, x)$

A densely connected network



A densely connected network

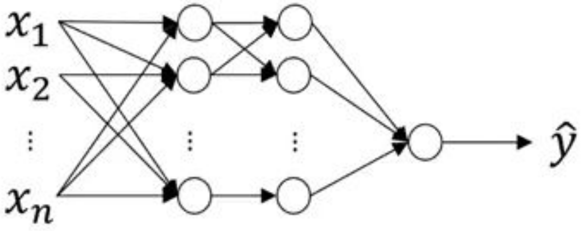
- one **epoch** = one forward and backward pass of *all* training examples
- **batch size** = # training examples in one forward/backward pass.

  Higher batch size = bigger memory requirement
- number of **iterations** = number of passes (backward + forward), each pass using [batch size] number of examples.

Example: if you have 1000 training examples, and your batch size is 500, then it will take 2 iterations to complete 1 epoch.

Jupyter notebook

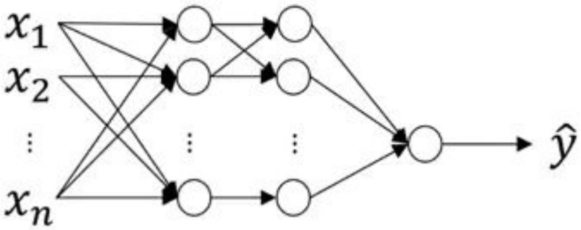
Convolutional Neural Networks: why?



- Santa example: n = 64*64*3= 12288
- 128 nodes in the first hidden layer
- Weight matrix: ~ 1.5 M weights!!

Convolutional Neural Networks: why?

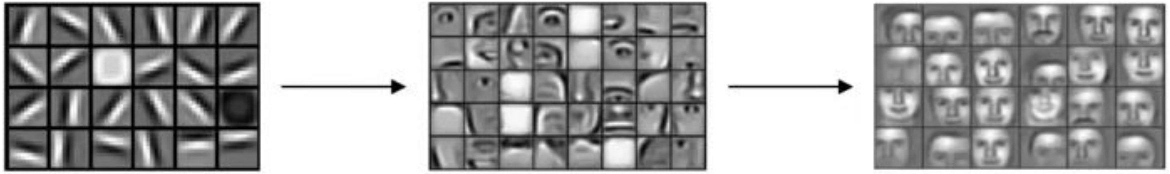
Convolutional Neural Networks: why?



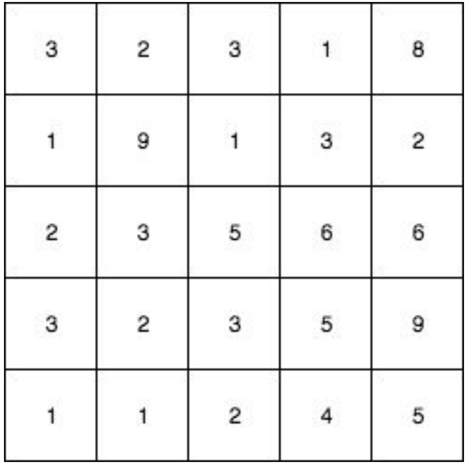
- 1000*1000*3
- 1000 nodes in the first hidden layer
- Weight matrix: 3 Billion parameters
- And this is just 1 layer!

Convolutional Neural Networks: why?

- Dense layers learn global patterns in their input feature space
- Convolution layers learn local patterns, and this leads to the following interesting features:
    - when a convolutional neural network recognizes a patterns in eg. upper-right corner of a picture, it can recognize it anywhere else.
    - Deeper convolutional neural networks can learn spatial hierarchies



The convolution operation



3	2	3	1	8
1	9	1	3	2
2	3	5	6	6
3	2	3	5	9
1	1	2	4	5

The convolution operation

3	2	3	1	8
1	9	1	3	2
2	3	5	6	6
3	2	3	5	9
1	1	2	4	5

*

1	0	-1
1	0	-1
1	0	-1

filters acts as feature detectors
from the original input image

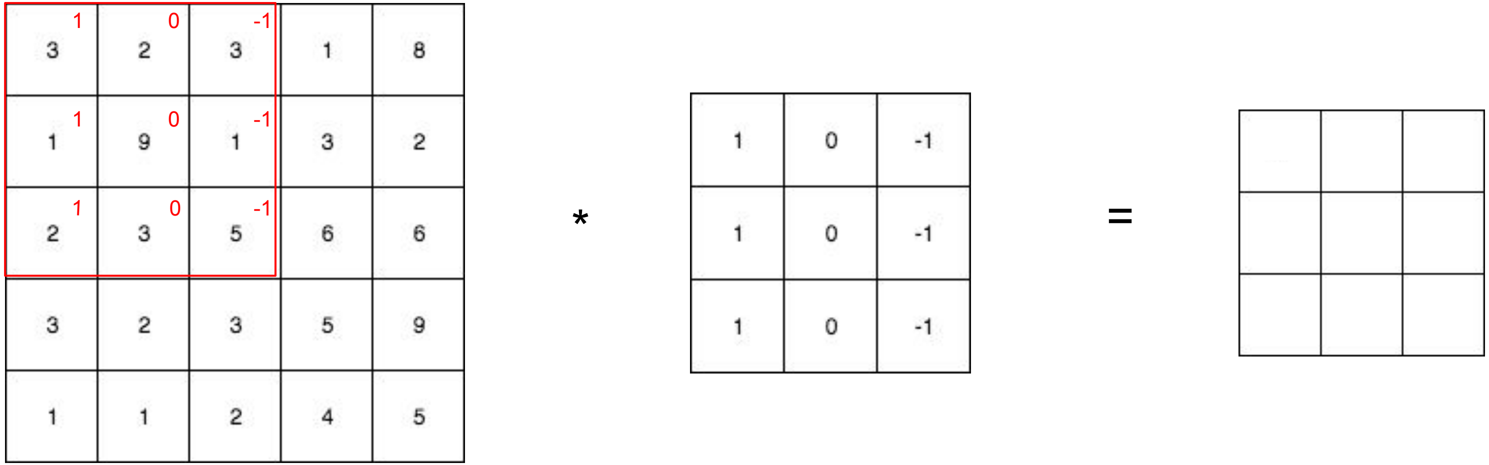
The convolution operation

3^{1}	2^{0}	3^{-1}	1	8
1^{1}	9^{0}	1^{-1}	3	2
2^{1}	3^{0}	5^{-1}	6	6
3	2	3	5	9
1	1	2	4	5

*

1	0	-1
1	0	-1
1	0	-1

The convolution operation



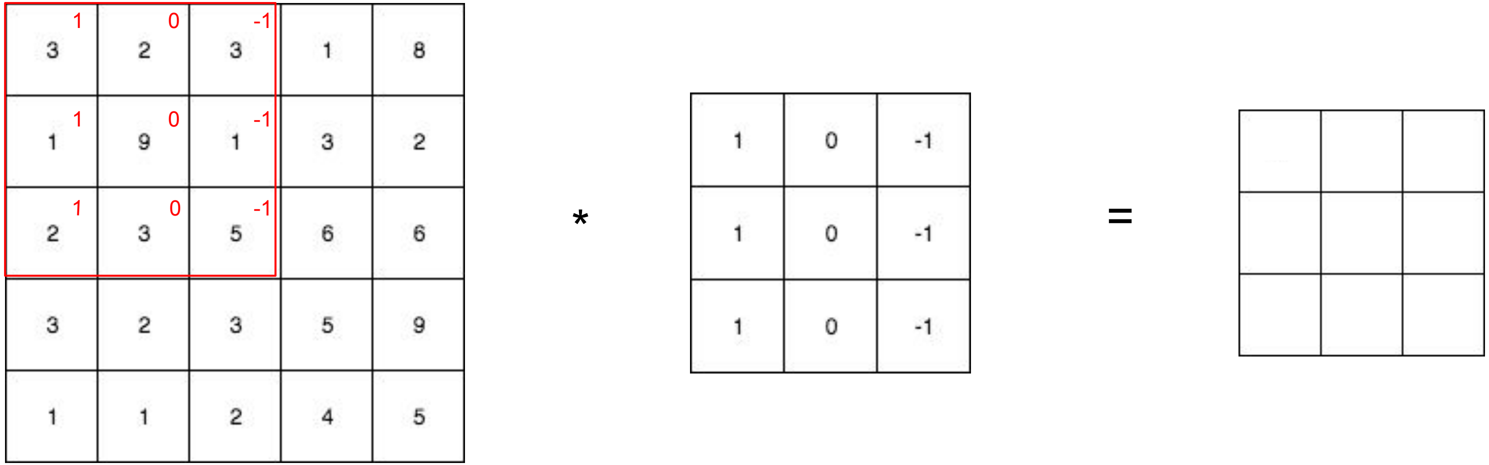
3^{1}	2^{0}	3^{-1}	1	8
1^{1}	9^{0}	1^{-1}	3	2
2^{1}	3^{0}	5^{-1}	6	6
3	2	3	5	9
1	1	2	4	5

*

1	0	-1
1	0	-1
1	0	-1

=

The convolution operation



1 3	0 2	-1 3	1	8
1 1	0 9	-1 1	3	2
1 2	0 3	-1 5	6	6
3	2	3	5	9
1	1	2	4	5

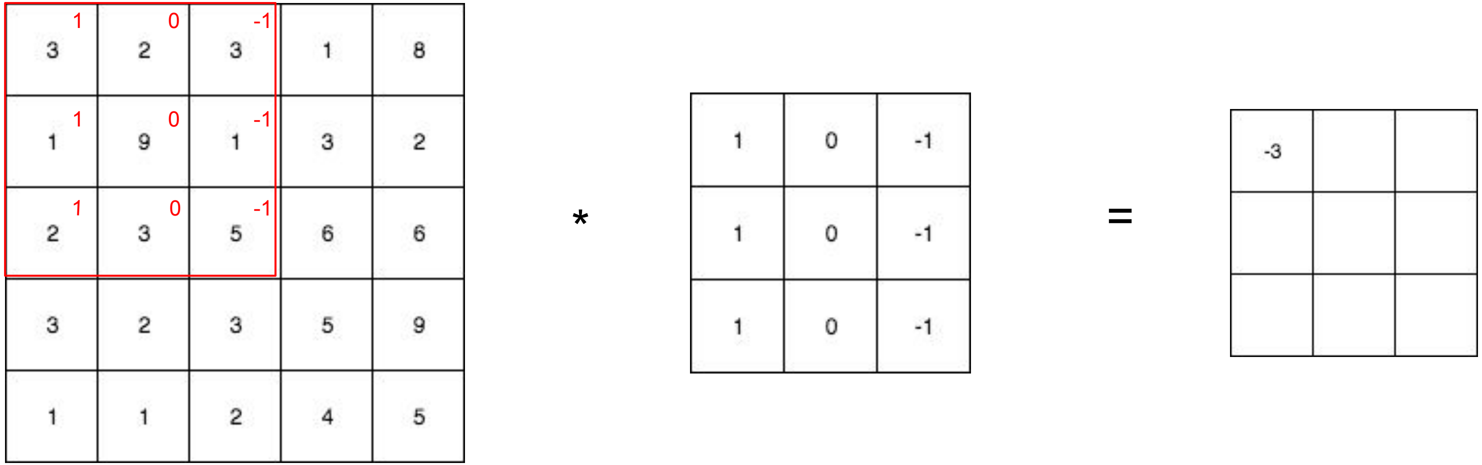
*

1	0	-1
1	0	-1
1	0	-1

=

3*1+1*1+2+1+0*2+0*9+0*3+3*(-1)+1*(-1)+5*(-1) = -3

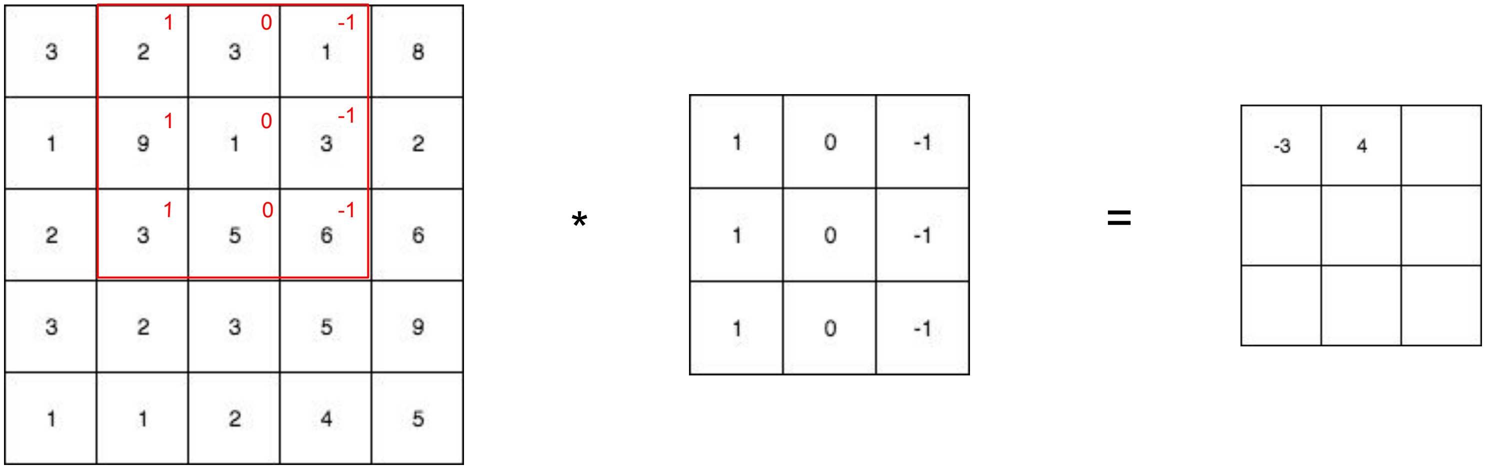
The convolution operation



$$\begin{array}{|c|c|c|c|c|}\hline 3^{1} & 2^{0} & 3^{-1} & 1 & 8 \\ \hline 1^{1} & 9^{0} & 1^{-1} & 3 & 2 \\ \hline 2^{1} & 3^{0} & 5^{-1} & 6 & 6 \\ \hline 3 & 2 & 3 & 5 & 9 \\ \hline 1 & 1 & 2 & 4 & 5 \\ \hline \end{array} * \begin{array}{|c|c|c|}\hline 1 & 0 & -1 \\ \hline 1 & 0 & -1 \\ \hline 1 & 0 & -1 \\ \hline \end{array} = \begin{array}{|c|c|c|}\hline -3 & & \\ \hline & & \\ \hline & & \\ \hline \end{array}$$

3*1+1*1+2*1+0*2+0*9+0*3+3*(-1)+1*(-1)+5*(-1) = -3

The convolution operation



3	2	3	1	8
1	9	1	3	2
2	3	5	6	6
3	2	3	5	9
1	1	2	4	5

1	0	-1
1	0	-1
1	0	-1

-3	4	

2*1+9*1+3*1+0*3+0*1+0*5+1*(-1)+3*(-1)+6*(-1) = 4

The convolution operation

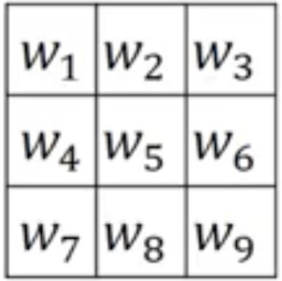
$$\begin{array}{|c|c|c|c|c|}\hline 3 & 2 & 3 & 1 & 8 \\ \hline 1 & 9 & 1 & 3 & 2 \\ \hline 2 & 3 & 5 & 6 & 6 \\ \hline 3 & 2 & 3 & 5 & 9 \\ \hline 1 & 1 & 2 & 4 & 5 \\ \hline \end{array} * \begin{array}{|c|c|c|}\hline 1 & 0 & -1 \\ \hline 1 & 0 & -1 \\ \hline 1 & 0 & -1 \\ \hline \end{array} = \begin{array}{|c|c|c|}\hline -3 & 4 & -7 \\ \hline -3 & 0 & -8 \\ \hline -4 & -9 & -10 \\ \hline \end{array}$$

What if there is an edge?

$$\begin{bmatrix} 8 & 8 & 8 & 0 & 0 & 0 \\ 8 & 8 & 8 & 0 & 0 & 0 \\ 8 & 8 & 8 & 0 & 0 & 0 \\ 8 & 8 & 8 & 0 & 0 & 0 \\ 8 & 8 & 8 & 0 & 0 & 0 \\ 8 & 8 & 8 & 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix} = \begin{bmatrix} 0 & 24 & 24 & 0 \\ 0 & 24 & 24 & 0 \\ 0 & 24 & 24 & 0 \\ 0 & 24 & 24 & 0 \end{bmatrix}$$

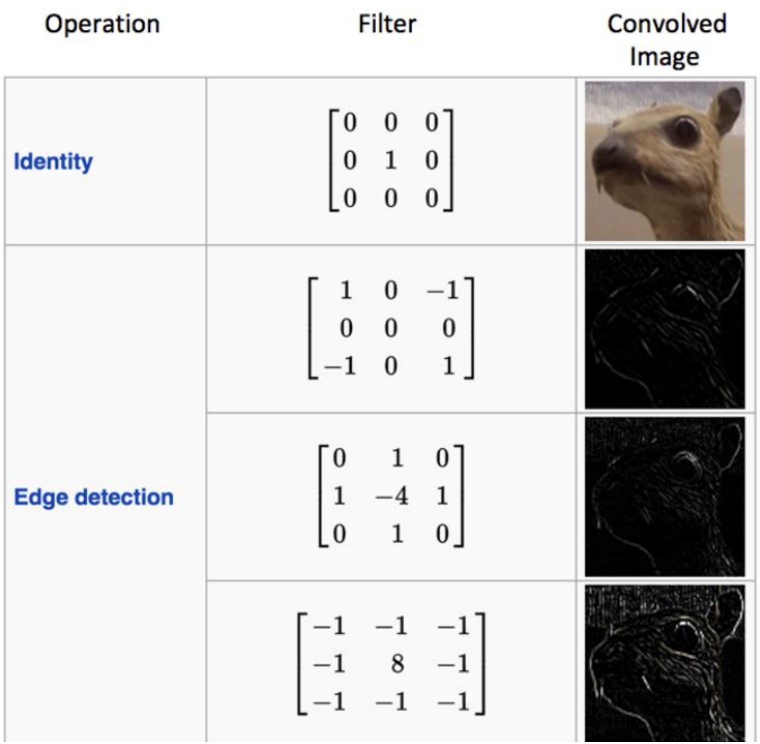
Other filters

- Horizontal edges
- Diagonal edges
- ...
- Make network learn filters!

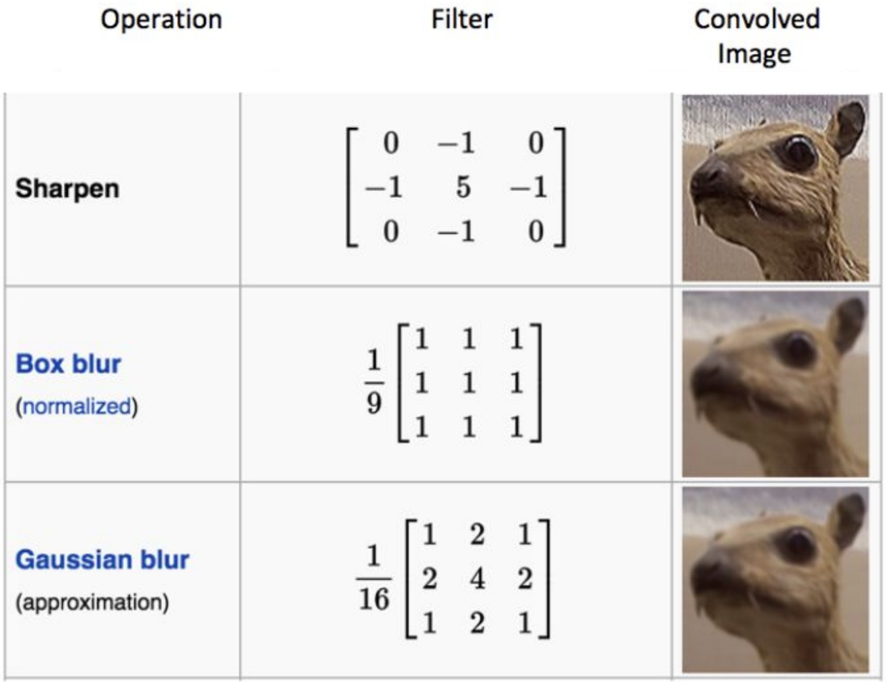


w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

Other filters



Operation	Filter	Convolved Image
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Edge detection	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	



Operation	Filter	Convolved Image
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Box blur (normalized)	$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	
Gaussian blur (approximation)	$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	

Problems with convolutions

- Deep networks: images **shrink considerably**
- More central pixels are used much more in output

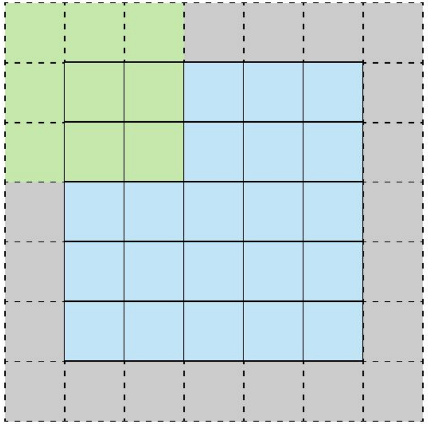
Solution: padding!

Problems with convolutions

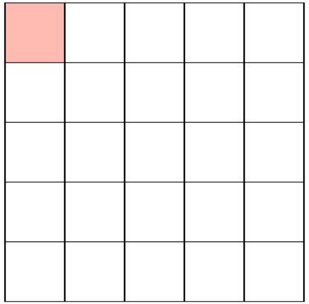
- Deep networks: images **shrink considerably**
- More central pixels are used much more in output

Solution: padding!

- Valid
- Same

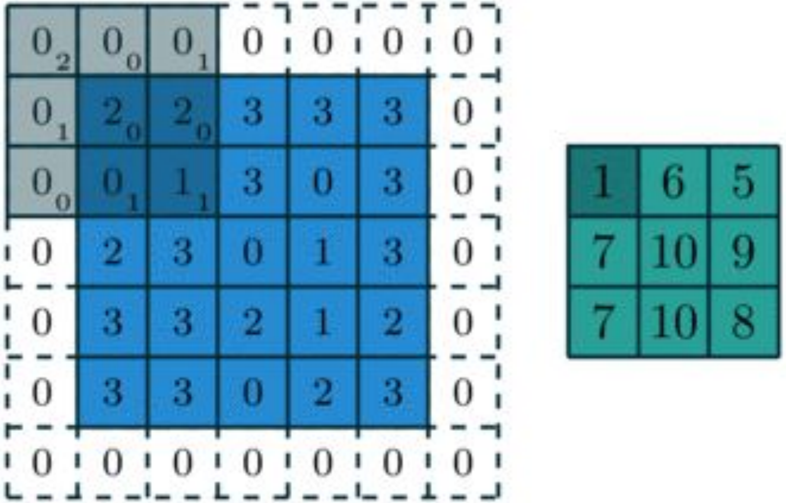


Stride 1 with Padding



Feature Map

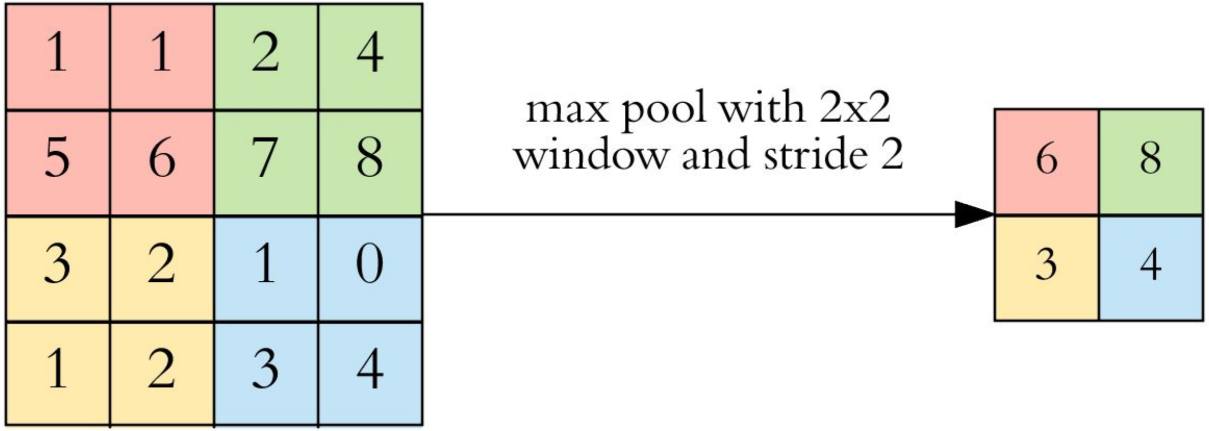
Strided convolutions



0_2	0_0	0_1	0	0	0	0
0_1	2_0	2_0	3	3	3	0
0_0	0_1	1_1	3	0	3	0
0	2	3	0	1	3	0
0	3	3	2	1	2	0
0	3	3	0	2	3	0
0	0	0	0	0	0	0

1	6	5
7	10	9
7	10	8

Pooling layers



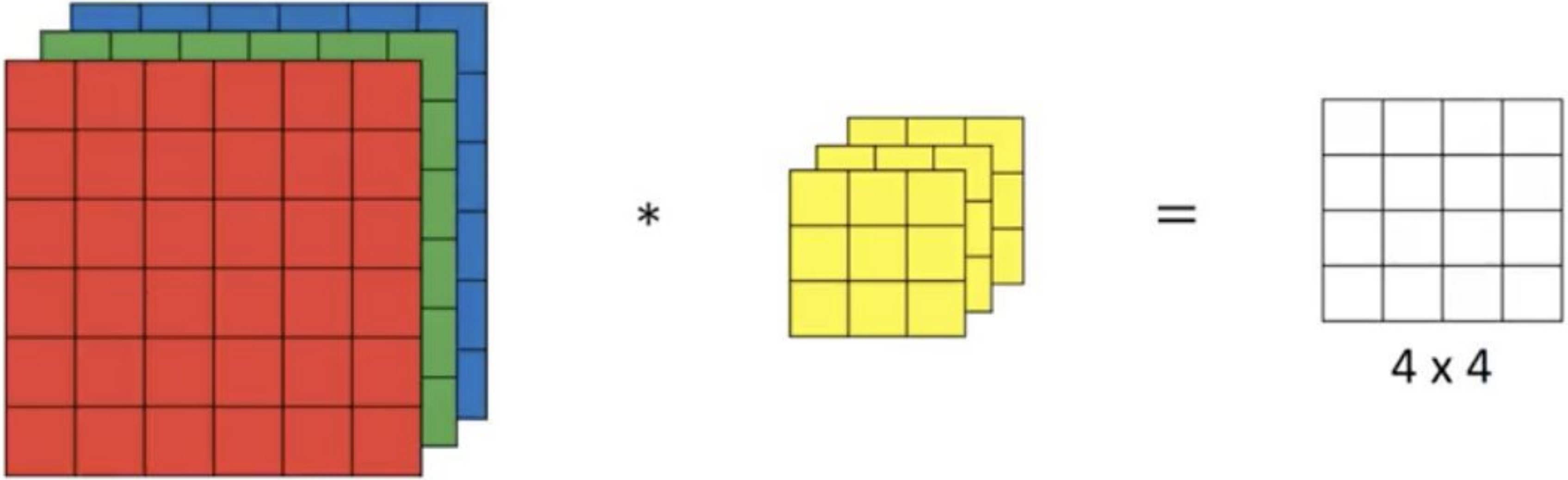
1	1	2	4
5	6	7	8
3	2	1	0
1	2	3	4

max pool with 2x2
window and stride 2

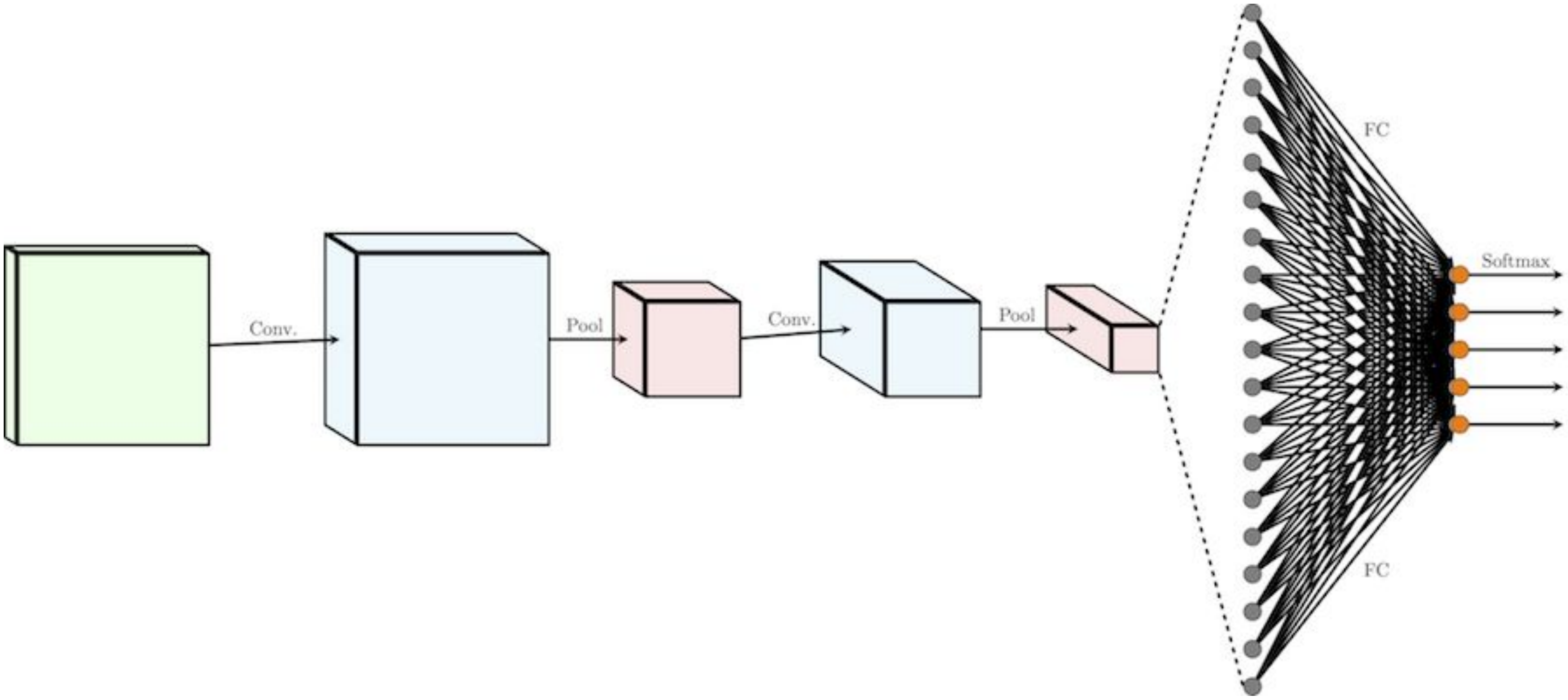
6	8
3	4

- Drastically downsizes feature maps
- Summarizes whether features are detected somewhere
- Is found to work well in a lot of experiments
- Only hyperparameters - no parameters to learn!

Convolutions on RGB images



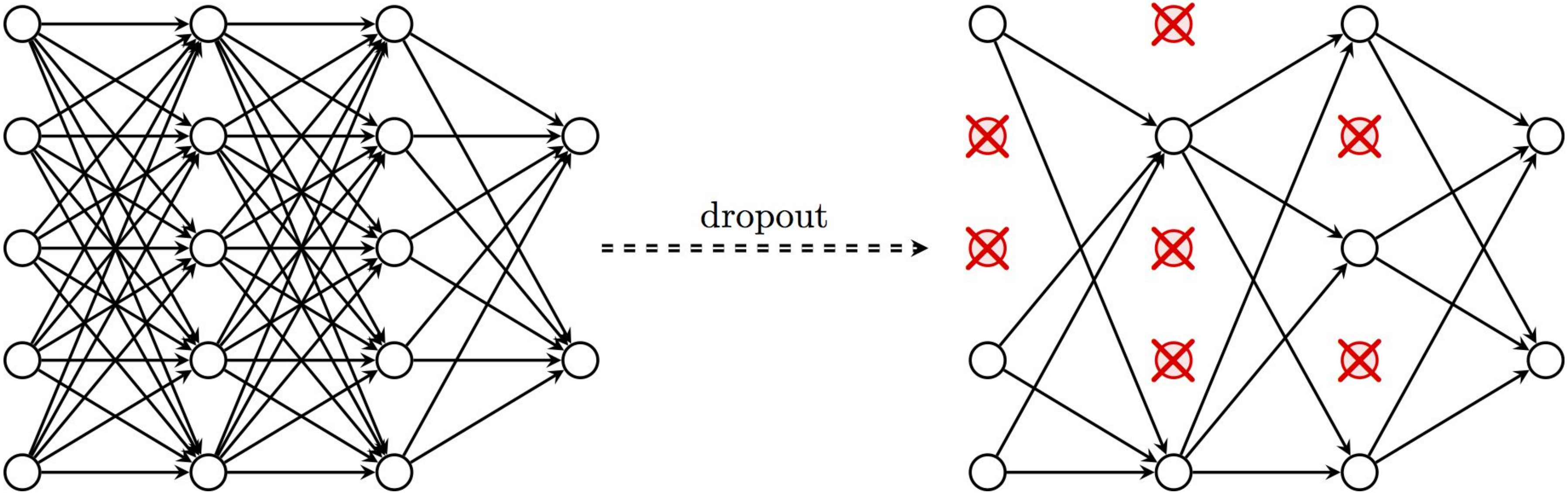
A full convolutional network



Back to our example!

Regularization

Regularization

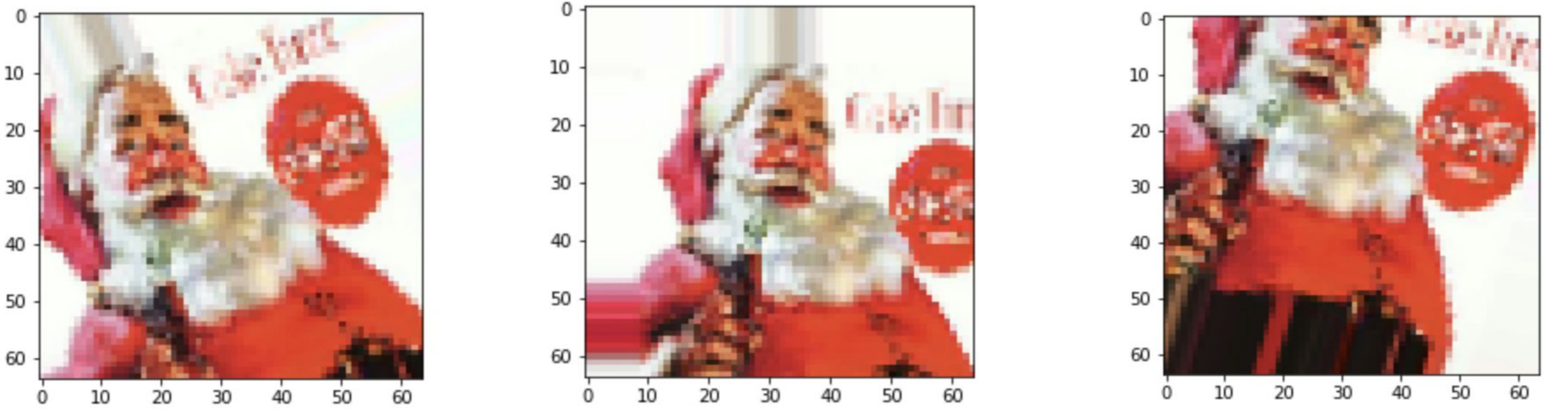


Other ways to tune/improve your model

- Try different architectures: add/remove layers
- L1/L2 regularization
- Try different hyperparameters (such as the number of units per layer, optimizers,...)
- Use data augmentation

Other ways to tune/improve your model

- Try different architectures: add/remove layers
- L1/L2 regularization
- Try different hyperparameters (such as the number of units per layer, optimizers,...)
- Use data augmentation



Questions? Comments? Feedback?

Lore Dirick
Senior Data Science Curriculum Developer
lore.dirick@flatironschool.com

Q & A

THANK YOU !