

# Deep learning: an overview of datasets and algorithms

## DL taskforce Caterpillar

Rob Romijnders

Eindhoven, University of Technology

*RomijndersRob@gmail.com*

May 25, 2017

# Overview

- 1 Why
- 2 How
- 3 Datasets
- 4 What
- 5 Wrap up
- 6 Questions

# Object detection

## Faster R-CNN

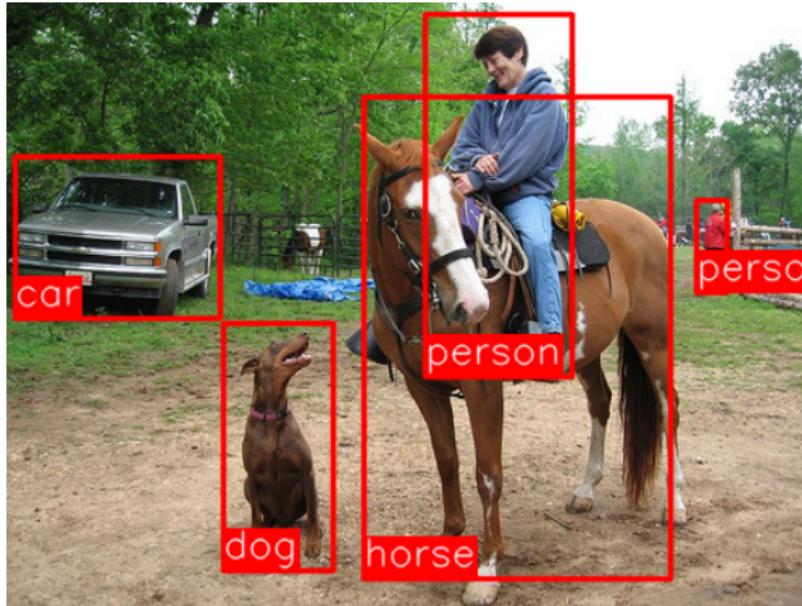


Figure: faster rcnn (image from Github *mitmul*)

# Apple siri



Figure: Photo: cultofmac.com

# GMail reply

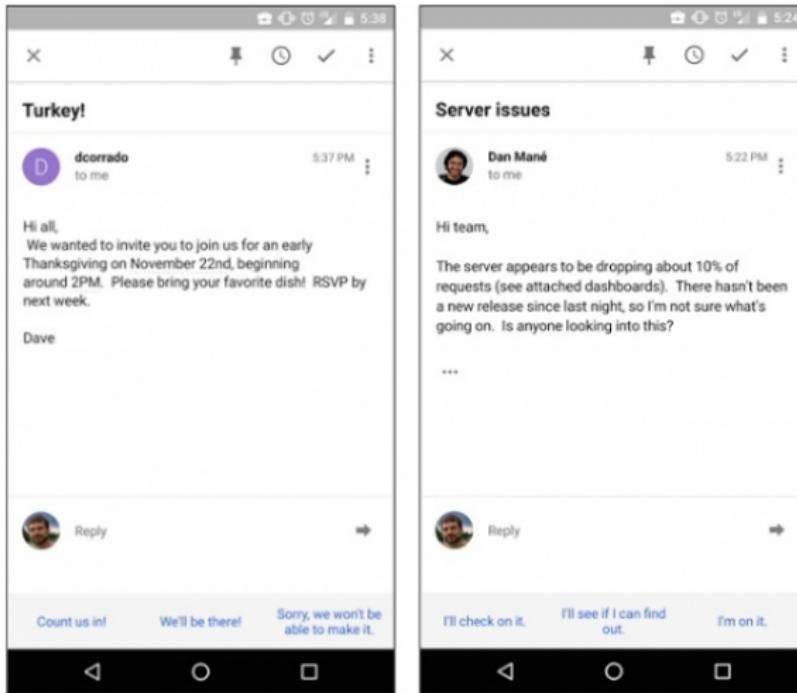
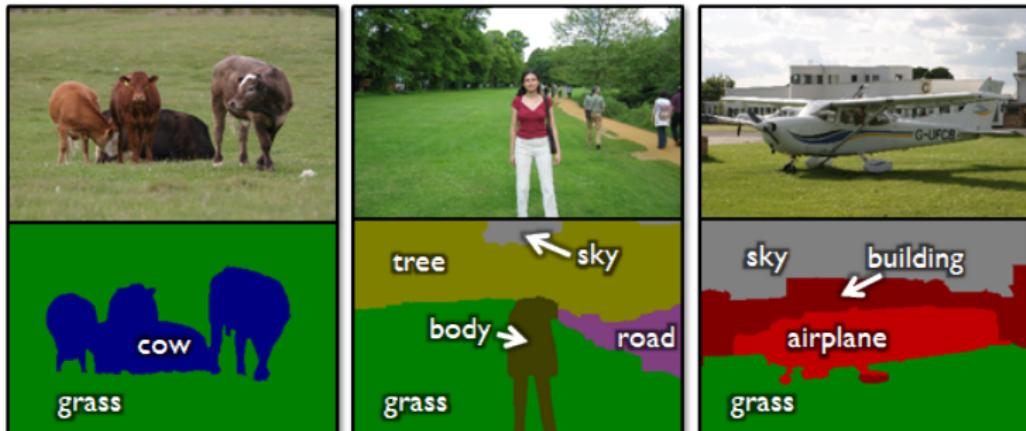


Figure: Photo: Greg Corrado, Google Research Blog

# Segmentation



object classes	building	grass	tree	cow	sheep	sky	airplane	water	face	car
bicycle	flower	sign	bird	book	chair	road	cat	dog	body	boat

Figure: Semantic Segmentation with CNN (image from jamie.shotton.org)

# Generation

THE MULTIVERSE —

## Movie written by algorithm turns out to be hilarious and intense

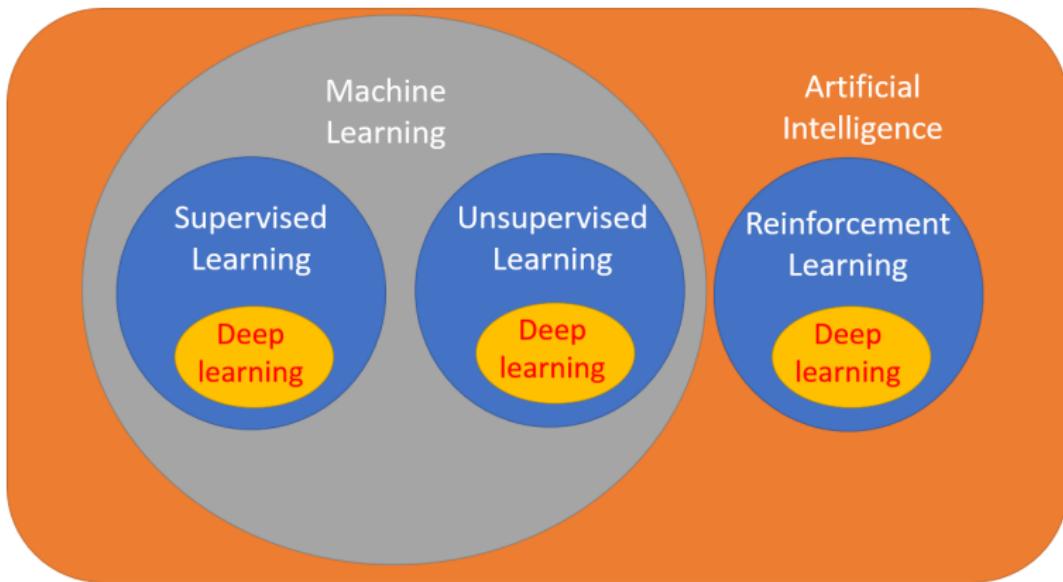
For *Sunspring*'s exclusive debut on Ars, we talked to the filmmakers about collaborating with an AI.

ANNALEE NEWITZ - 6/9/2016, 12:30 PM

<http://arstechnica.com/the-multiverse/2016/06/an-ai-wrote-this-movie-and-its-strangely-moving/>

# Why not

# Overview of the field



# Which data not

- ① YES: media type data
  - ① Text, language, speech
  - ② Images, video, maps
  - ③ time-series, stocks, valuta
- ② NO: categorical data
  - ① Properties of instances
  - ② Features of instances
  - ③ Categories of products

# Neural Networks

## Neural nets

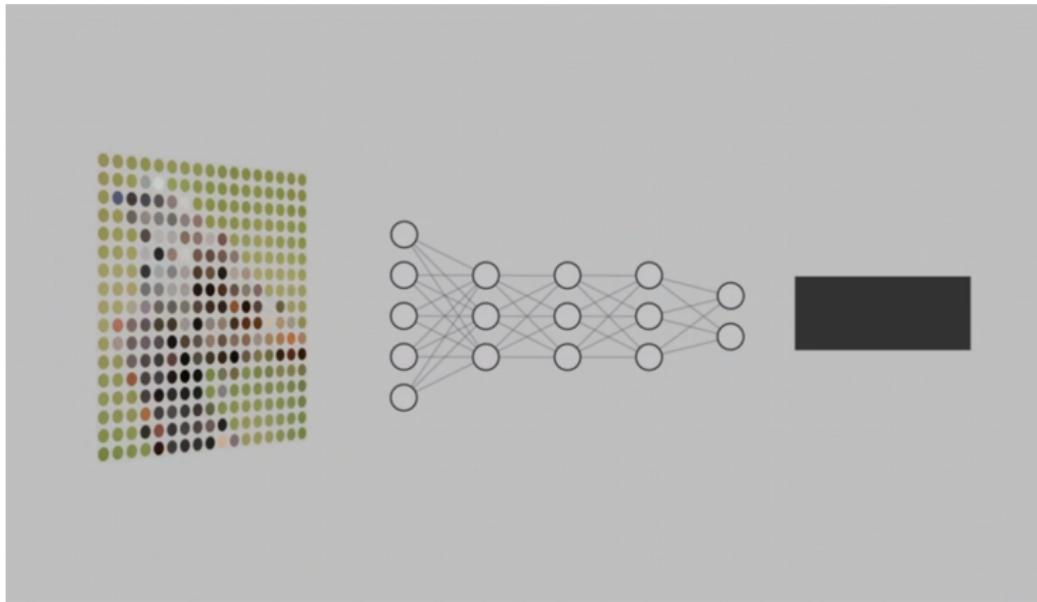


Figure: Neural network explained (credits: Blaise Aguera y Arcas)

## Basic equation

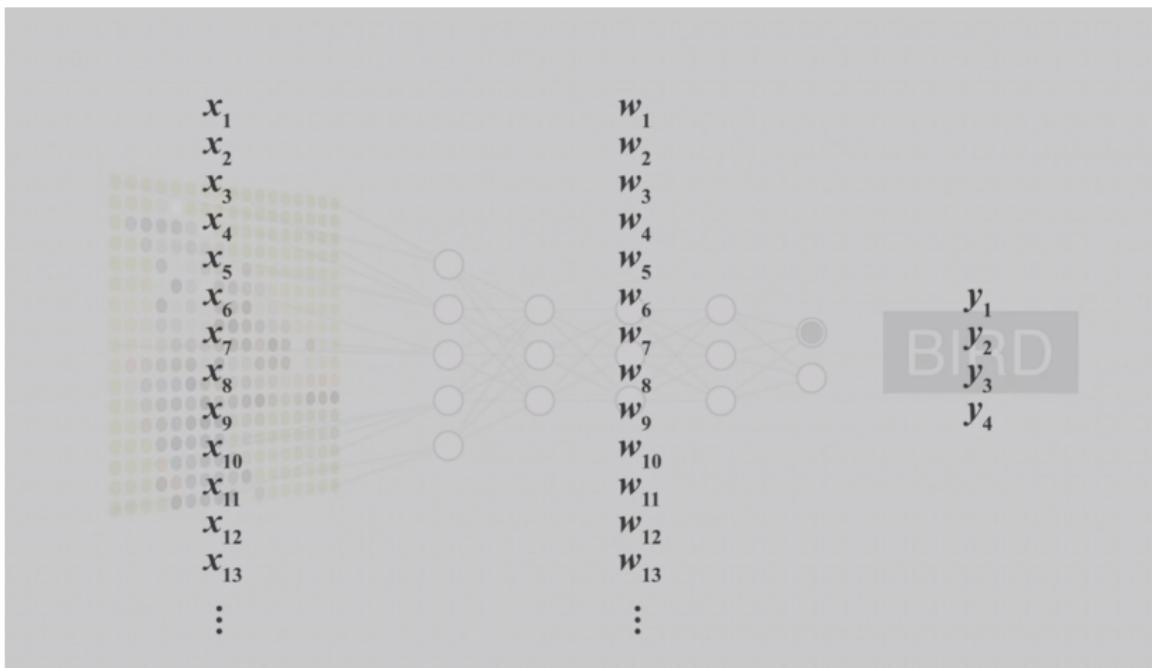


Figure: Neural net in algebraic form (credits: Blaise Aguera y Arcas)

## Template equation



Figure: Template equation neural net (credits: Blaise Aguera y Arcas)

# How to use

# How to use them

$$w \cdot x = y$$

$$2 \cdot 3 = y$$

# Forward inference

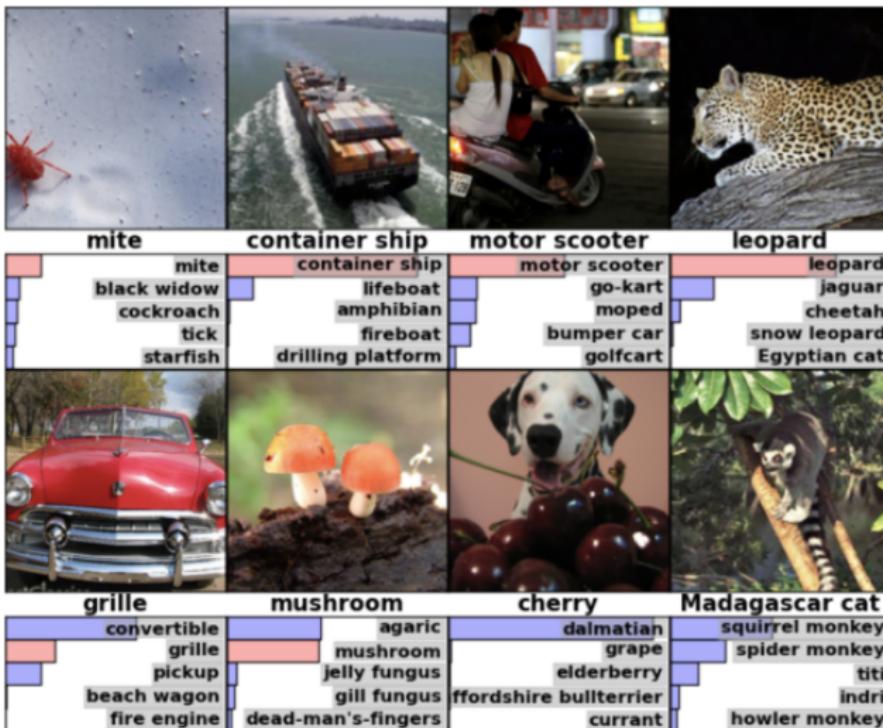


Figure: Forward inference CNN (Krizhevsky et al. 2012)

# How to train

# How to train them

$$w \cdot x = y$$

$$w \cdot 3 = 6$$

$$y \div x = w$$

$$\begin{aligned}0 &= w \cdot x - y \\0 &= w \cdot 3 - 6\end{aligned}$$

$$\text{error} = |w \cdot x - y| \Rightarrow 0$$

# Loss function

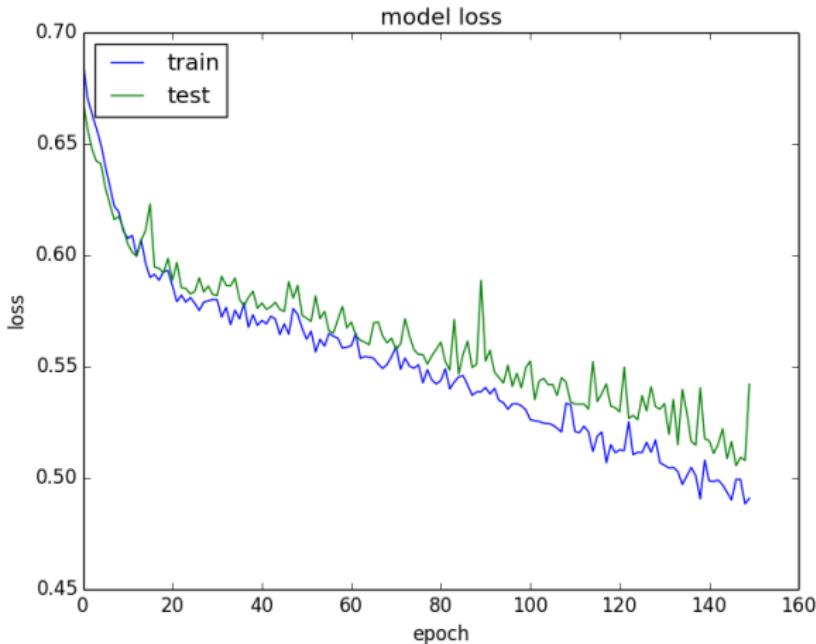


Figure: (<http://machinelearningmastery.com/>)

# Architecture

# Architecture

# Architecture

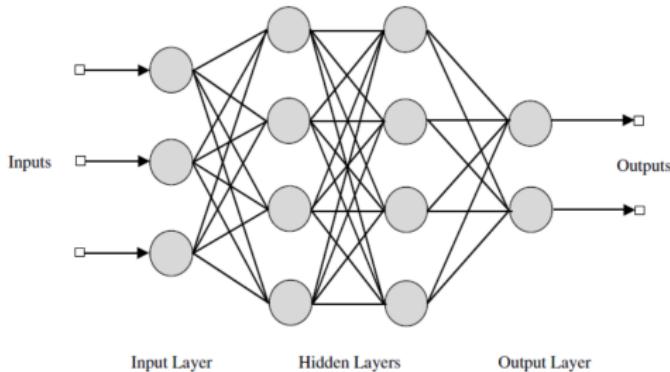


Figure: Feedforward neural network

$$y = w \cdot x$$



# Two architectures

## ① Convolution

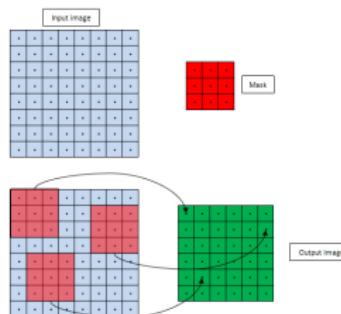


Figure: developer.amd.com

## ② Recursion

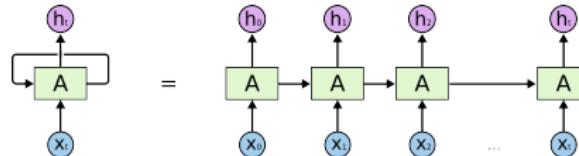


Figure: credits: colah.github.io

# Convolution

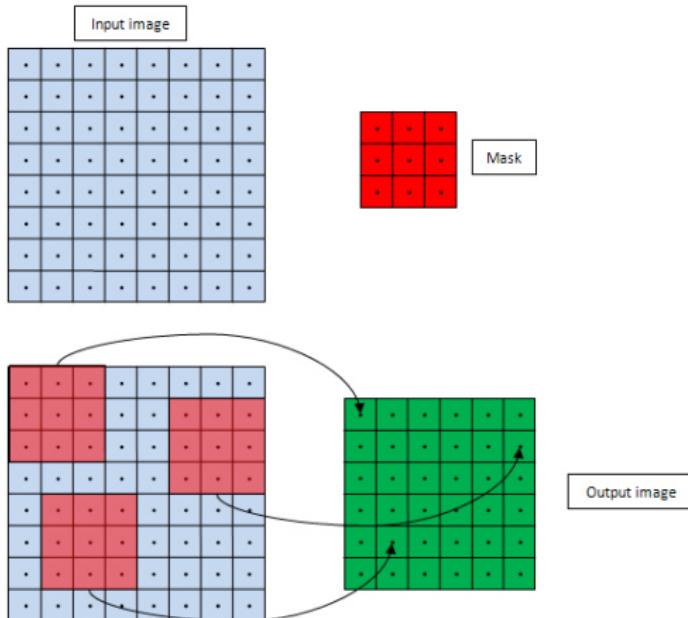


Figure: developer.amd.com

# CNN

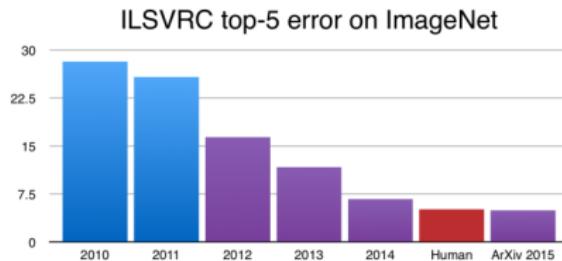


Figure: ImageNet results (credits jackkelly.github.io/)



Figure: Forward inference CNN (Krizhevsky et al. 2012)

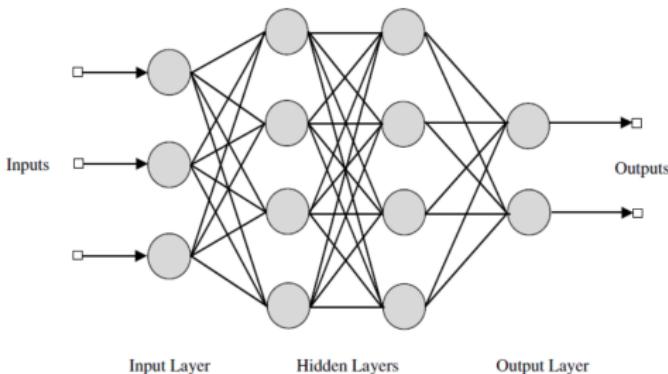


Figure: Feedforward neural network

$$32 \cdot 32 \cdot 100 = 102400$$

$$5 \cdot 5 = 25$$

# Recursion

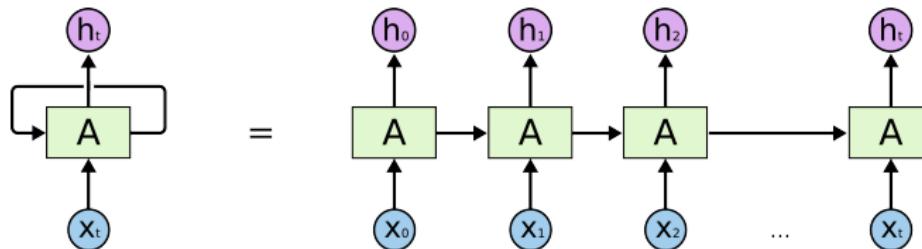


Figure: credits: [colah.github.io](https://colah.github.io)

# Template formula

$$y = w \cdot x$$

$$y_t = w \cdot (x_t, h_{t-1})$$

# Formula

$$h = \phi(W_{xh}x)$$

$$y = \phi(W_{hy}h)$$

$$h_t = \phi(W_{xh}x_t + W_{hh}h_{t-1}) \quad y_t = \phi(W_{hy}h_t)$$

# Python code

```
rnn = RNN()  
y = rnn.step(x)
```

```
class RNN:  
    def step(self, x):  
        self.h = np.tanh(np.dot(self.Whh, self.h)  
                        + np.dot(self.Wxh, x))  
        y = np.dot(self.Why, self.h)  
    return y
```

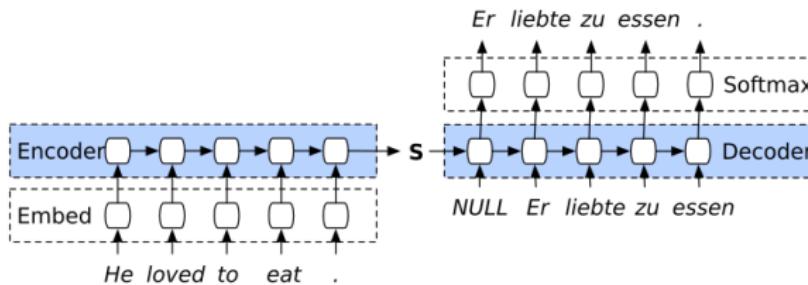


Figure: RNN for Machine Translation (credits: smerity.com)

# Example

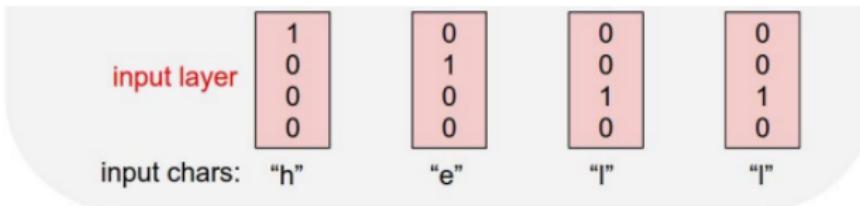


Figure: cs231n, Andrej Karpathy

# Example

$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$

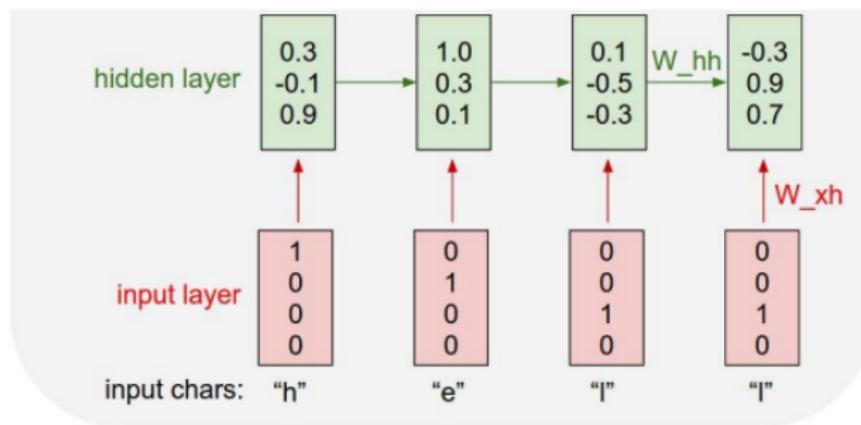


Figure: cs231n, Andrej Karpathy

# Example

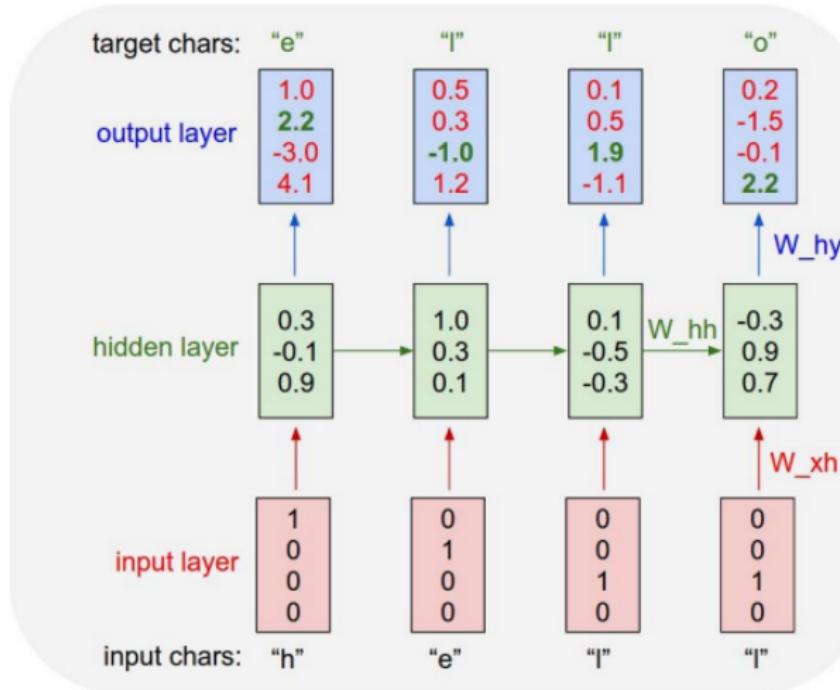


Figure: cs231n, Andrej Karpathy



# Long Short-term memory

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + b_i)$$

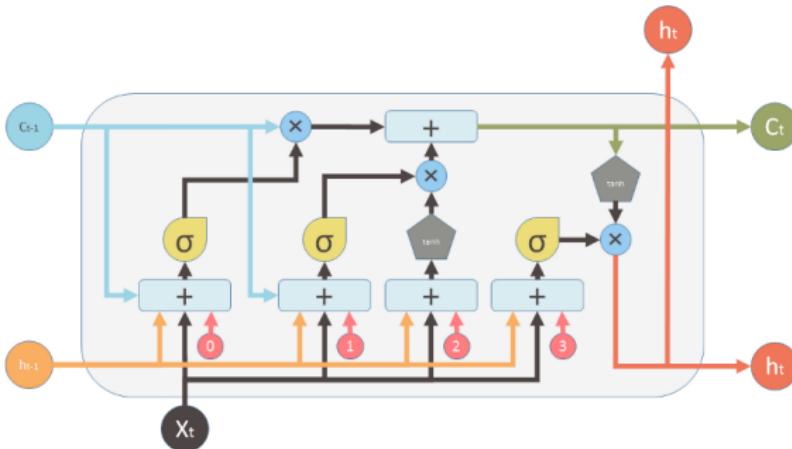
$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + b_f)$$

$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + b_o)$$

$$c_t = f_t c_{t-1} + i_t \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c)$$

$$h_t = o_t \tanh(c_t)$$

# LSTM



Inputs:



Input vector



Memory from previous block



Output of previous block

outputs:



Memory from current block



Output of current block

Nonlinearities:



Sigmoid



Hyperbolic tangent

Vector operations:



Element-wise multiplication



Element-wise Summation / Concatenation

Bias:



Figure: Diagram depicting LSTM block

Author: Shi Yan. Source: [medium.com/@oshiyan/](https://medium.com/@oshiyan/)

Overview			
	FF	RNN	CNN
- Basic computation	Matrix multiplies	Recurrent multiplications	Convolutions
- Typical data	Embedding vectors	Sequences	Images
- Weights ties	No	Sequential	Spatial
- Effective span	Across input	Long term	Local receptive field
- Stationary		Seldom events	Frequent patterns
- How to train	Backprop	Backprop	Backprop
- Performance	No guarantee	No guarantee	No guarantee
- Typical examples	None	Machine translation, speech tagging, text generation	segmentation, image processing, object detection

# Span

## ① CNN

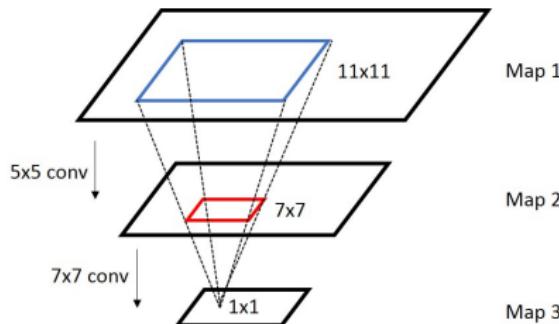


Figure: credits: [cvmarcher.com/](http://cvmarcher.com/)

## ② RNN

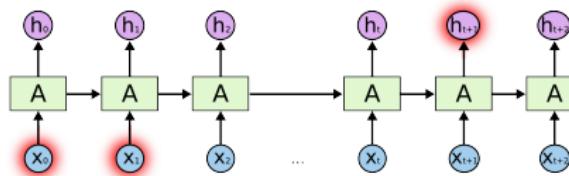


Figure: credits [colah.github.io](https://colah.github.io)

# Datasets and algorithms

# Tagging of clinical events

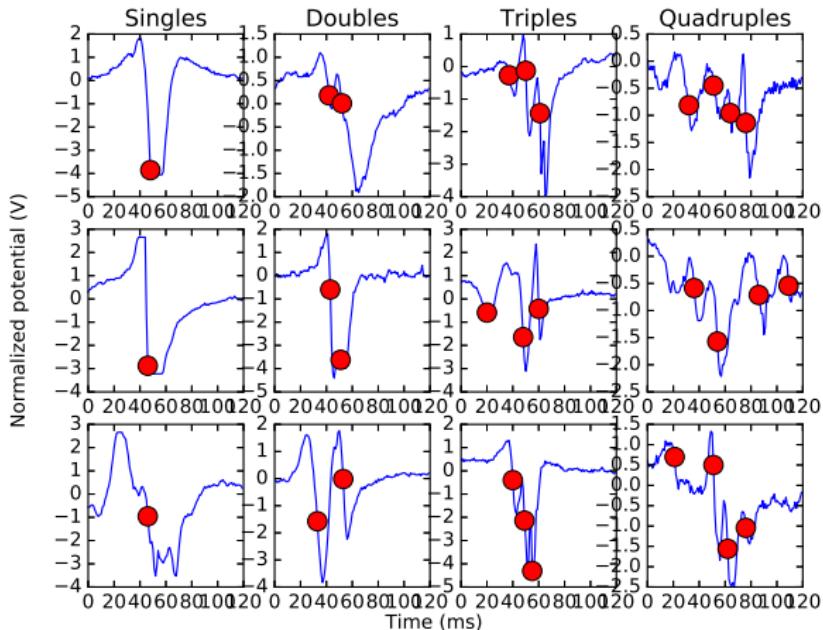


Figure: Classification of fractionated electrograms in epicardial mappings using a recurrent neural network, R Romijnders et al.

Use LSTM's, because the deflections occur as events

# Classifying music

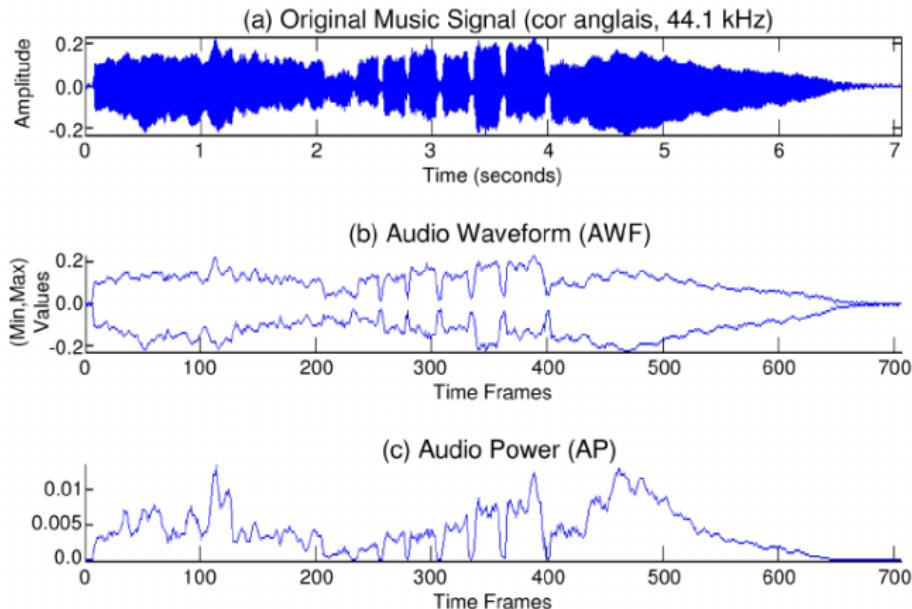


Figure: ML for music



Use CNN's, because music genre follows a certain pattern

- ① *The film did n't move me one way or the other , but it was an honest effort and if you want to see a flick about telemarketers this one will due .*
- ② *For those of us who respond more strongly to storytelling than computer-generated effects , the new Star Wars installment has n't escaped the rut dug by the last one .*

IMDB sentiment challenge, Kaggle

Use LSTM, because a single word could change the sentiment. Compare *I do like this* and *I do not like this*

# Architectures for RNN

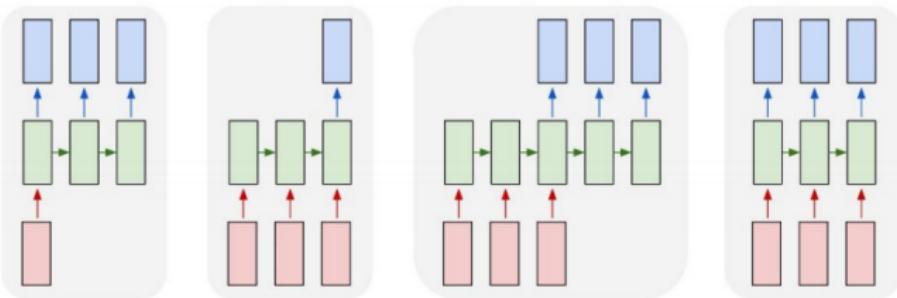


Figure: credits: [Andrej Karpathy](#)

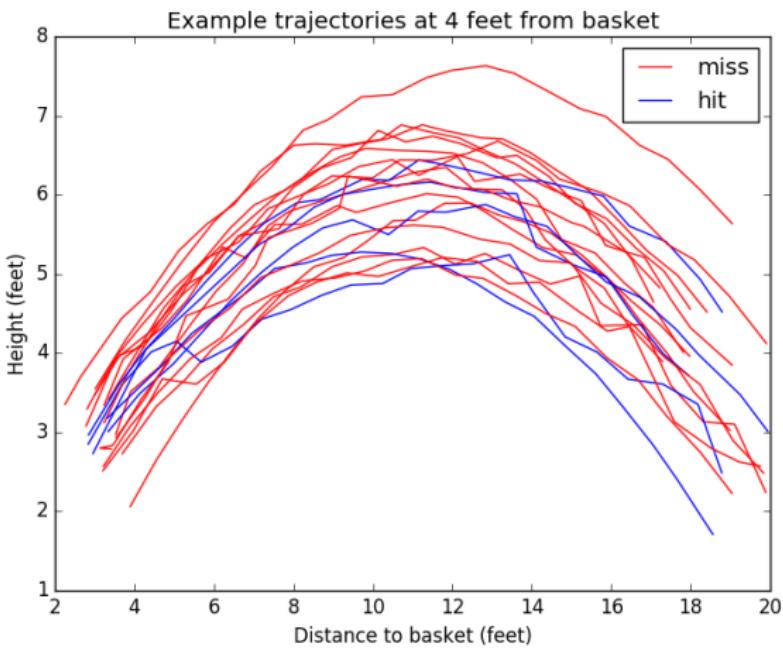


Figure: DL for basketball, Rajiv and Rob



# Works with both CNN and LSTM

PANDARUS:

Alas, I think he shall be come approached and the day  
When little strain would be attain'd into being never fed,  
And who is but a chain and subjects of his death,  
I should not sleep.

Second Senator:

They are away this miseries, produced upon my soul,  
Breaking and strongly should be buried, when I perish  
The earth and thoughts of many states.

DUKE VINCENTIO:

Well, your wit is in the care of side and that.

Second Lord:

They would be ruled after this chamber, and  
my fair nues begun out of the fact, to be conveyed,  
Whose noble souls I'll have the heart of the wars.

Clown:

Come, sir, I will make did behold your worship.

VIOLA:

I'll drink it.

Figure: credits: [Andrej Karpathy](#)

Use LSTM's, because it  
predicts a vector (word) for  
each time step

# Tabular

## Data Dictionary

Variable	Definition	Key
survival	Survival	0 = No, 1 = Yes
pclass	Ticket class	1 = 1st, 2 = 2nd, 3 = 3rd
sex	Sex	
Age	Age in years	
sibsp	# of siblings / spouses aboard the Titanic	
parch	# of parents / children aboard the Titanic	
ticket	Ticket number	
fare	Passenger fare	
cabin	Cabin number	
embarked	Port of Embarkation	C = Cherbourg, Q = Queenstown, S = Southampton

Figure: [Titanic KAGGLE](#)



Do NOT use deep learning. Go  
for Random Forests or SVM

# Captioning



"man in black shirt is playing guitar."



"construction worker in orange safety vest is working on road."



"two young girls are playing with legos toy."



"boy is doing backflip on wakeboard."

Figure: [Link to page](#)

Use CNN for pattern extraction  
from image. Use RNN to  
generate caption.

# DIY

# DIY

- ➊ Gather data
- ➋ Choose architecture
- ➌ Code up neural net
- ➍ Train
- ➎ Add bells and whistles
- ➏ Post processing

# Gather data

## ① SMALL:

- ① MNIST: 60.000 samples
- ② TIMIT: 630 speakers, 10 sentences

## ② NORMAL:

- ① Imagenet: 3.2 million images
- ② FLICKR: 1 million images
- ③ Word vectors on all of Wikipedia

# Amazon Mechanical Turk



**amazon** mechanical turk

beta

## ① Convolution

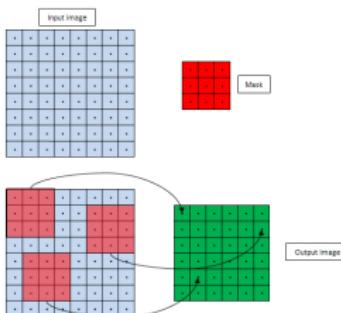


Figure: developer.amd.com

## ② Recursion

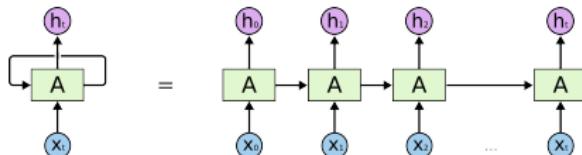


Figure: credits: colah.github.io

# Code

## ① Low level

- ① Tensorflow
- ② Torch, pyTorch

## ② High level

- ① Keras
- ② Scikit flow, pretty tensor, tfslim, ...

# Train

- ➊ No bells and whistles
- ➋ Small sub dataset
- ➌ See if you can overfit



# Bells and whistles

- ➊ Batch normalization, layer normalization
- ➋ Dropout
- ➌ Add more data: semi supervised learning
- ➍ Residual connections
- ➎ etcetera, etcetera



# Activation functions

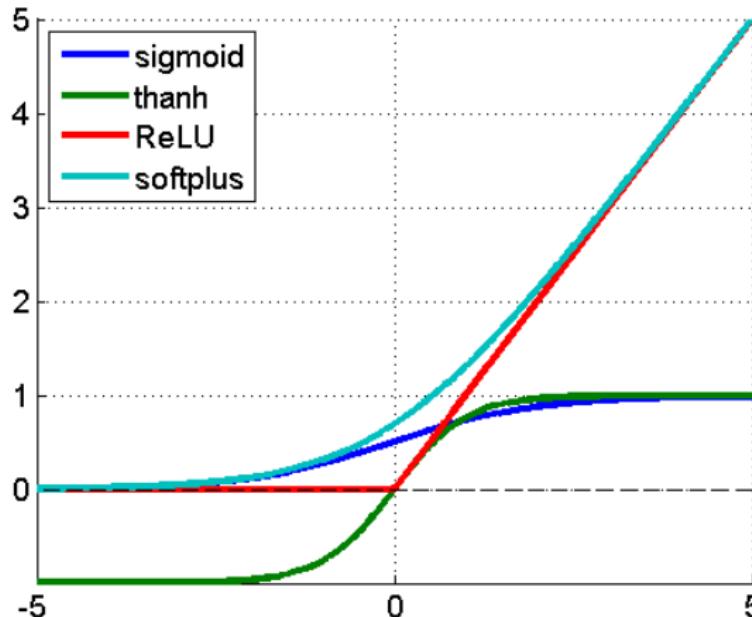


Figure: Common activation functions (credits Vanessa Imiloa)



# MNIST

Random Sampling of MNIST

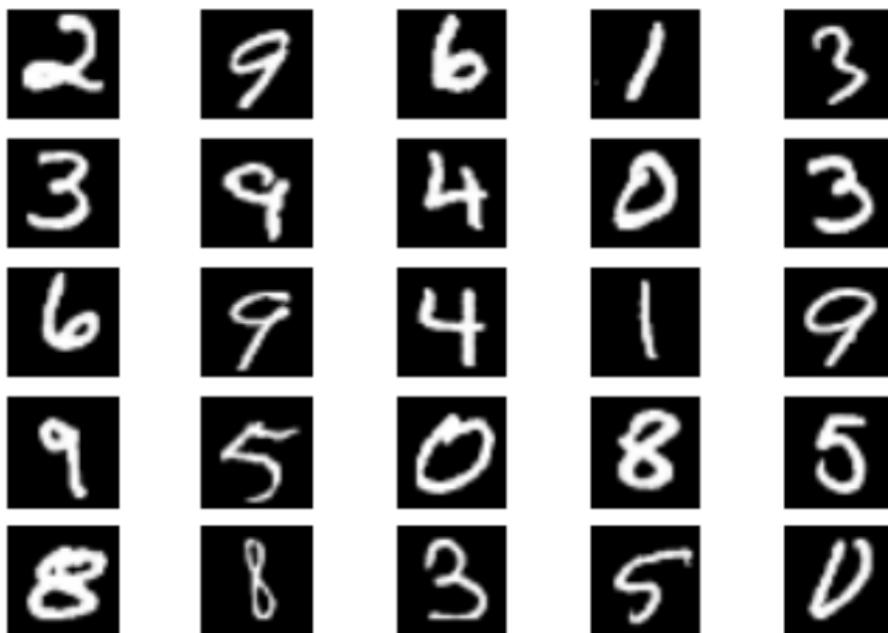


Figure: Caption



# Post processing

- ① Plot low confidence / high uncertainty
- ② Plot confusion matrix
- ③ Inspect neurons

# Regularization



Figure: Example for regularization



Figure: Detecting tanks

# Two architectures

## ① Convolution

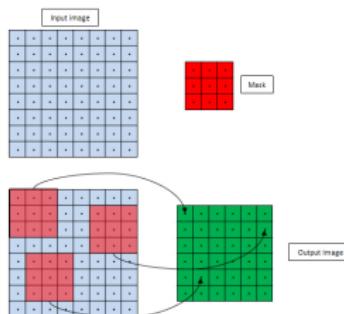


Figure: developer.amd.com

## ② Recursion

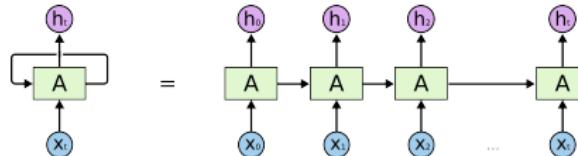


Figure: credits: colah.github.io

Why

How



Datasets



What



Wrap up

Questions

# Questions

# Further reading

## ① RNN

- ① The Unreasonable Effectiveness of Recurrent Neural Networks, Andrej Karpathy
- ② Supervised sequence labelling with RNN, Alex Graves
- ③ Chris Olah's blog ([colah.github.io](http://colah.github.io))

## ② CNN

- ① Course on CNN: cs231n ([cs231n.stanford.edu/syllabus.html](http://cs231n.stanford.edu/syllabus.html))
- ② Udacity on Deep learning ([udacity.com/course/deep-learning-ud730](http://udacity.com/course/deep-learning-ud730))
- ③ My page ([robromijnders.github.io](http://robromijnders.github.io))