

Deep Learning Winter School 2018

Nina Dethlefs (Computer Science), David Benoit (Chemistry), Chris Collins and Darren Bird (Viper HPC)



Registration questionnaire

Before we start, please could you fill this in?

(this is for project allocation tomorrow and some idea of where everyone's interests lie...)

<http://bit.ly/2G8dlR2>

Plan for today and tomorrow

Day 1:

- | | |
|---------------|--|
| 09:00 - 10:00 | Arrival and breakfast (RBB LTC) |
| 10:00 - 12:00 | Introduction to Deep learning and backpropagation (RBB LTC) |
| 12:00 - 13:00 | Lab session: MNIST with Keras (Fenner 052) |
| 13:00 - 14:00 | Lunch (RBB LTC and 3rd floor social area) |
| 14:00 - 15:30 | Research seminar: external talk + discussion (RBB LTC)
Yannis Konstas , Heriot-Watt University, Edinburgh
Building Adaptable and Scalable Natural Language Generation Systems |
| 15:30 - 17:00 | Lab session: MNIST with TensorFlow (Fenner 052) |
| 17:00 - 17:15 | Coffee and refreshments (RBB LTC) |
| 17:15 - 18:00 | Discussion on MNIST and projects for tomorrow (RBB LTC) |
| 18:00 | Dinner (RBB 3rd floor social area) |

Day 2:

- | | |
|---------------|--|
| 09:00 - 10:00 | Arrival and breakfast (RBB LTC) |
| 10:00 - 13:00 | Project work in small teams - projects to be announced (Fenner 052) |
| 13:00 - 14:00 | Lunch break and Viper tour (RBB LTC and 3rd floor social area) |
| 14:00 - 15:30 | Project presentation and prizes (RBB LTC) |
| 15:30 - 15:45 | Coffee break and End (RBB LTC) |

Participants

results from survey 1

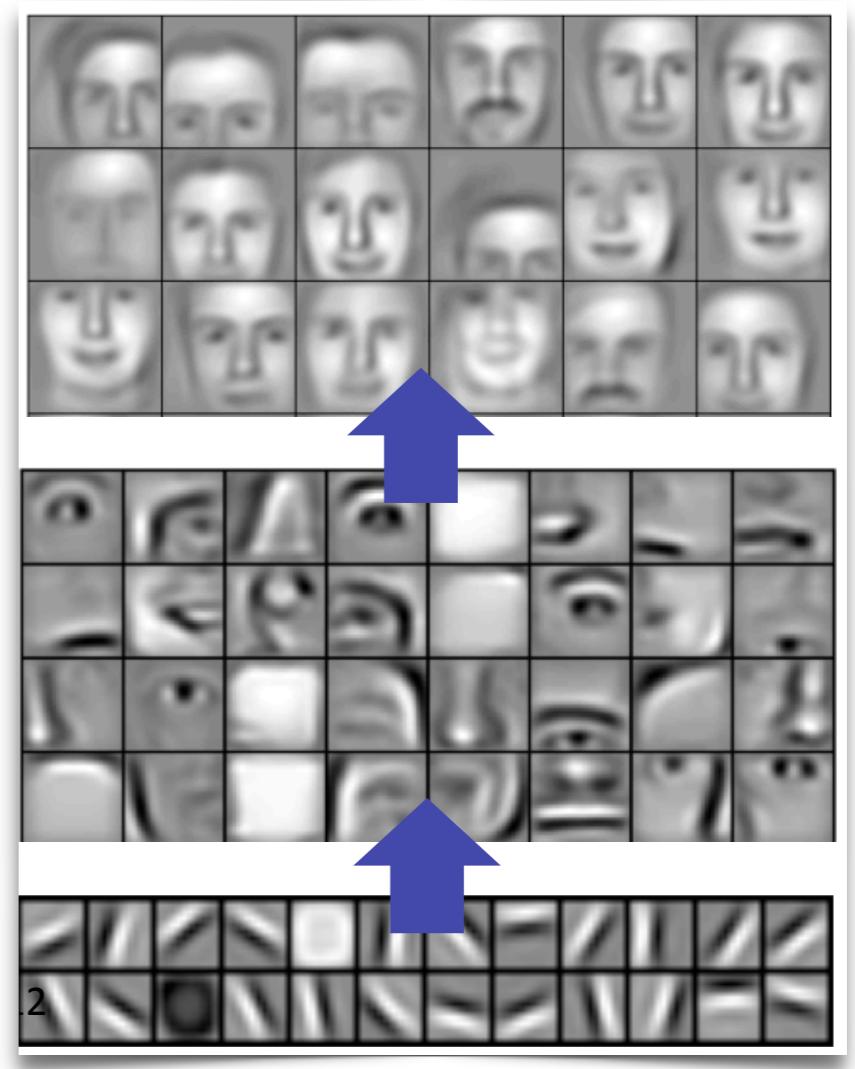
How does deep learning
work?

(in general)

Deep Learning - Basic ideas

Models learn increasingly **abstract representations of inputs** - at different layers of abstraction - and **recognise patterns** in unstructured data.

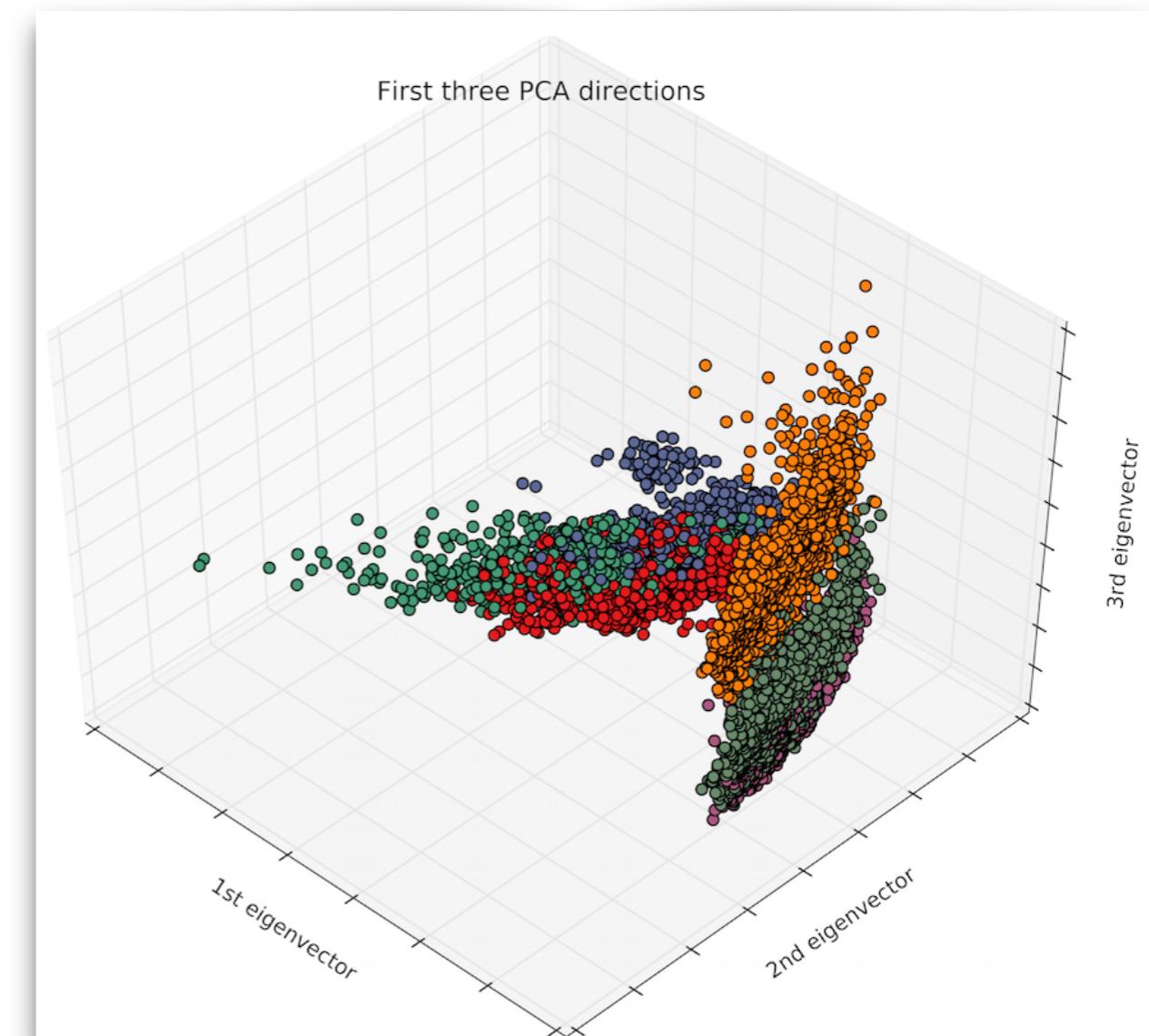
Applications: self-driving cars, natural language processing, image/video recognition, medical imaging, stock market prediction, ...



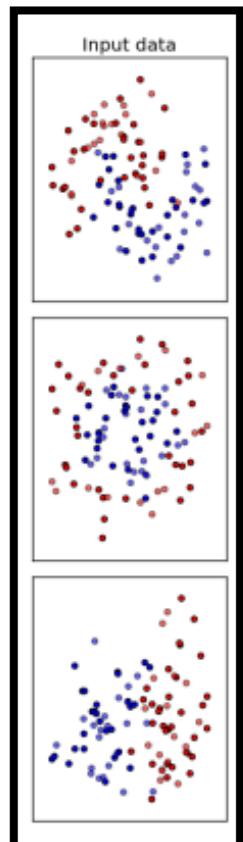
Layer 1 Layer 2 Layer 3

Why do deep learning?

- **Discovering patterns** + relationships in large, disparate, multimodal and high-dimensional datasets.
- **Models**: induction, simulation, simplification, data understanding
- **Machine learning**: algorithms for processing, inference + prediction
- **HPC**: distributed and parallel processing and storage; portability

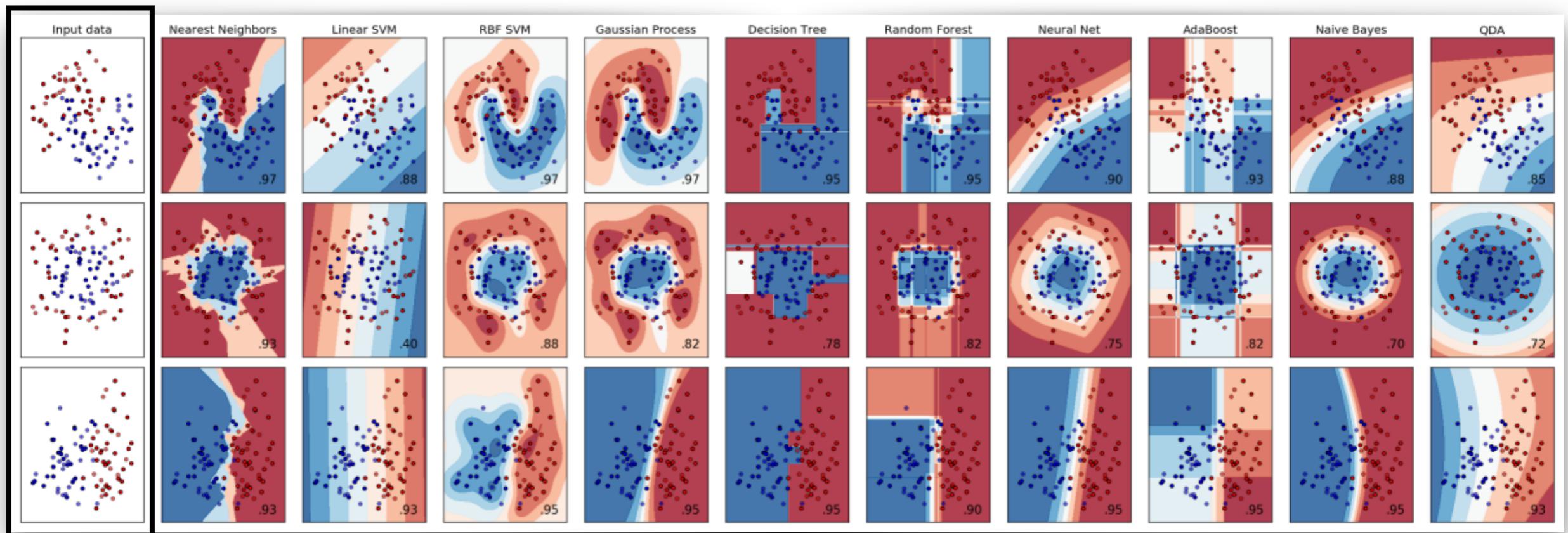


Machine learning



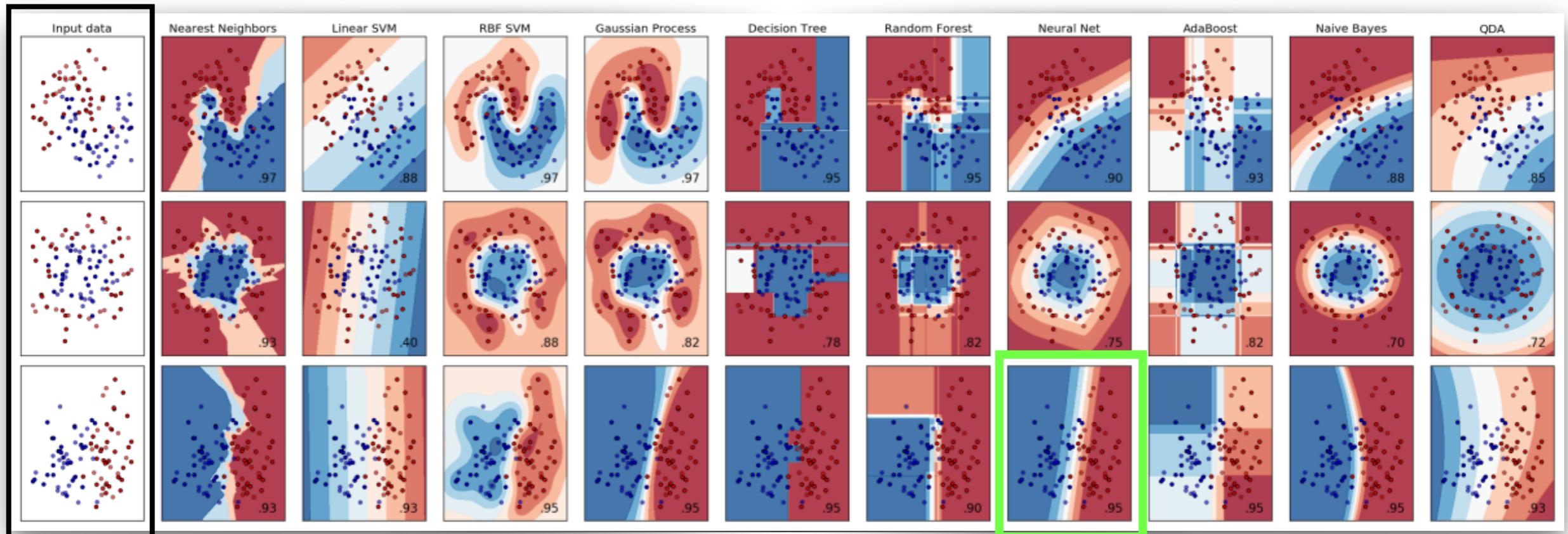
Datasets with different properties require different solutions.

Machine learning



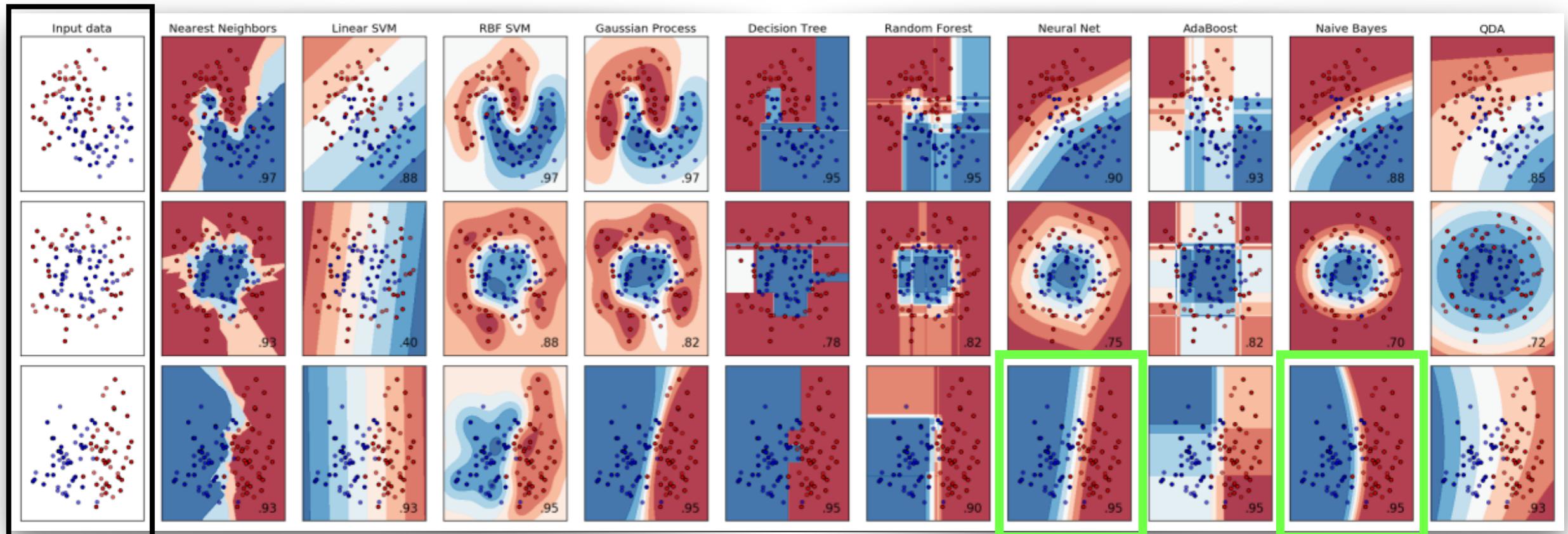
Different algorithms will provide different solutions.

Machine learning



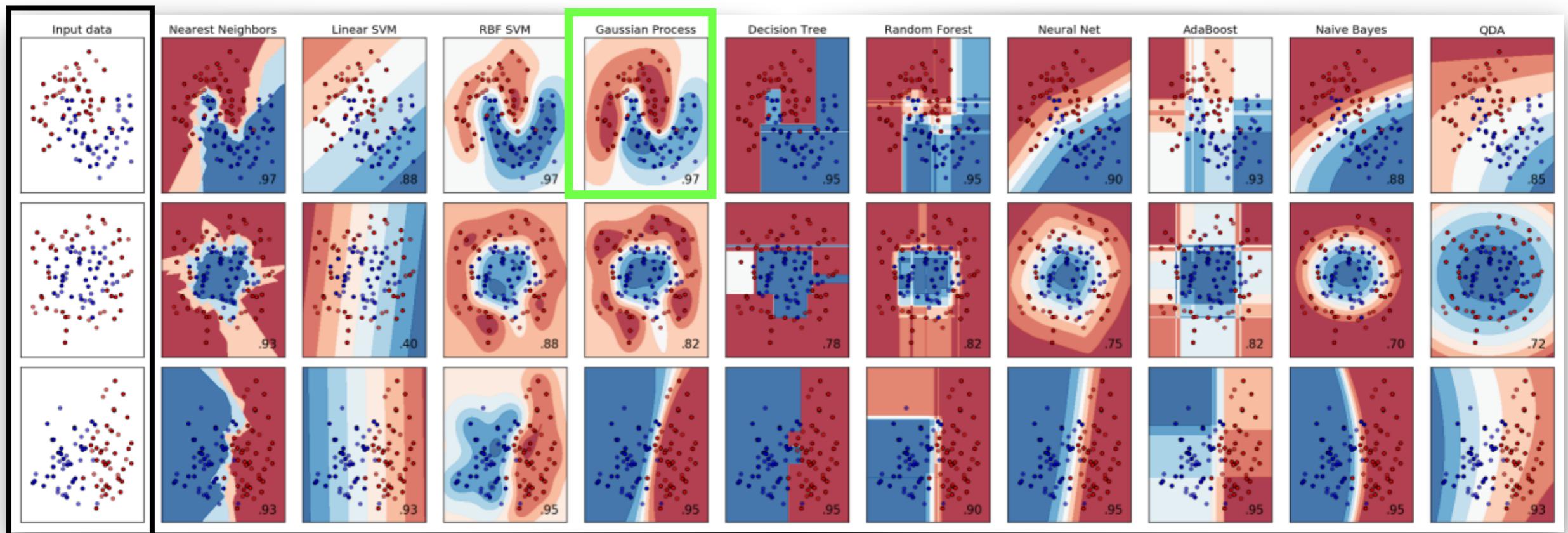
Example: neural net for linearly separable data

Machine learning



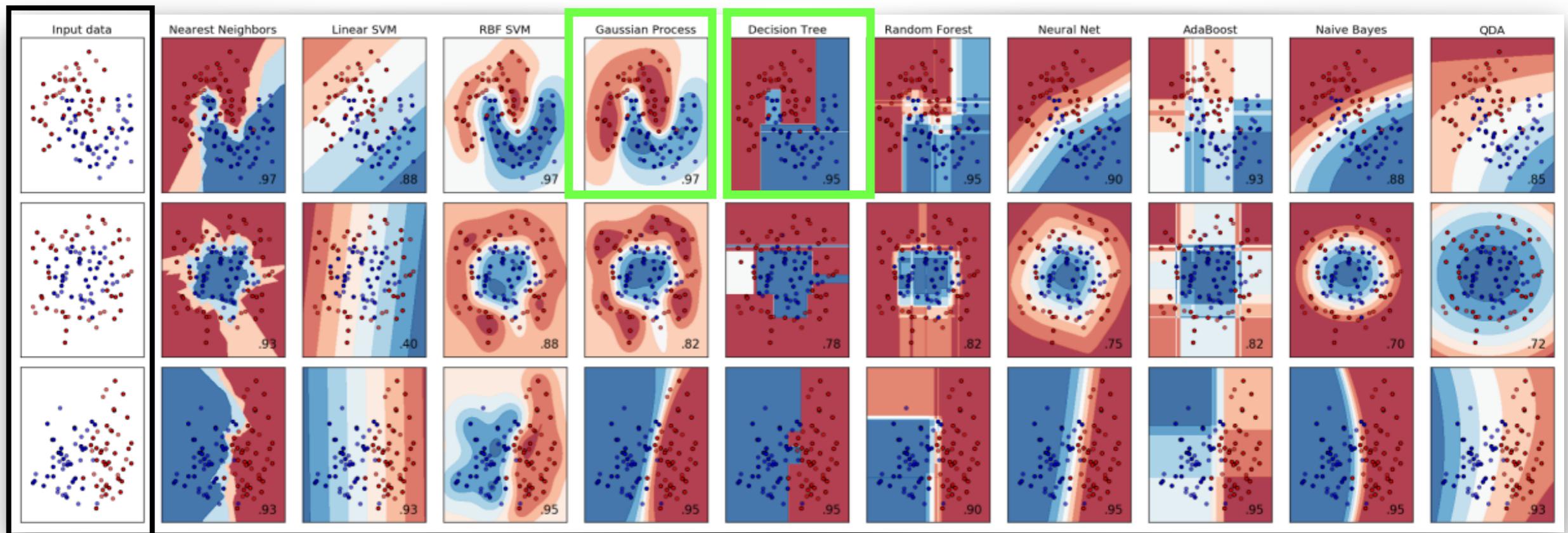
Neural network vs Naive Bayes for linear data

Machine learning



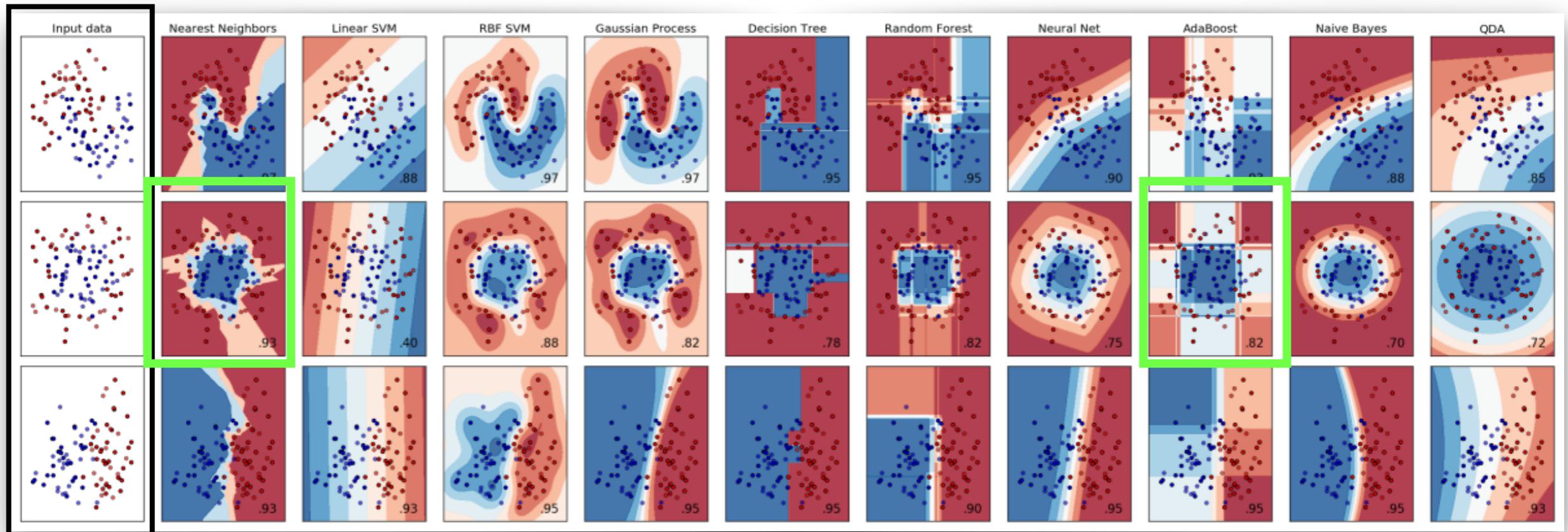
Gaussian Process for non-linearly separable data

Machine learning



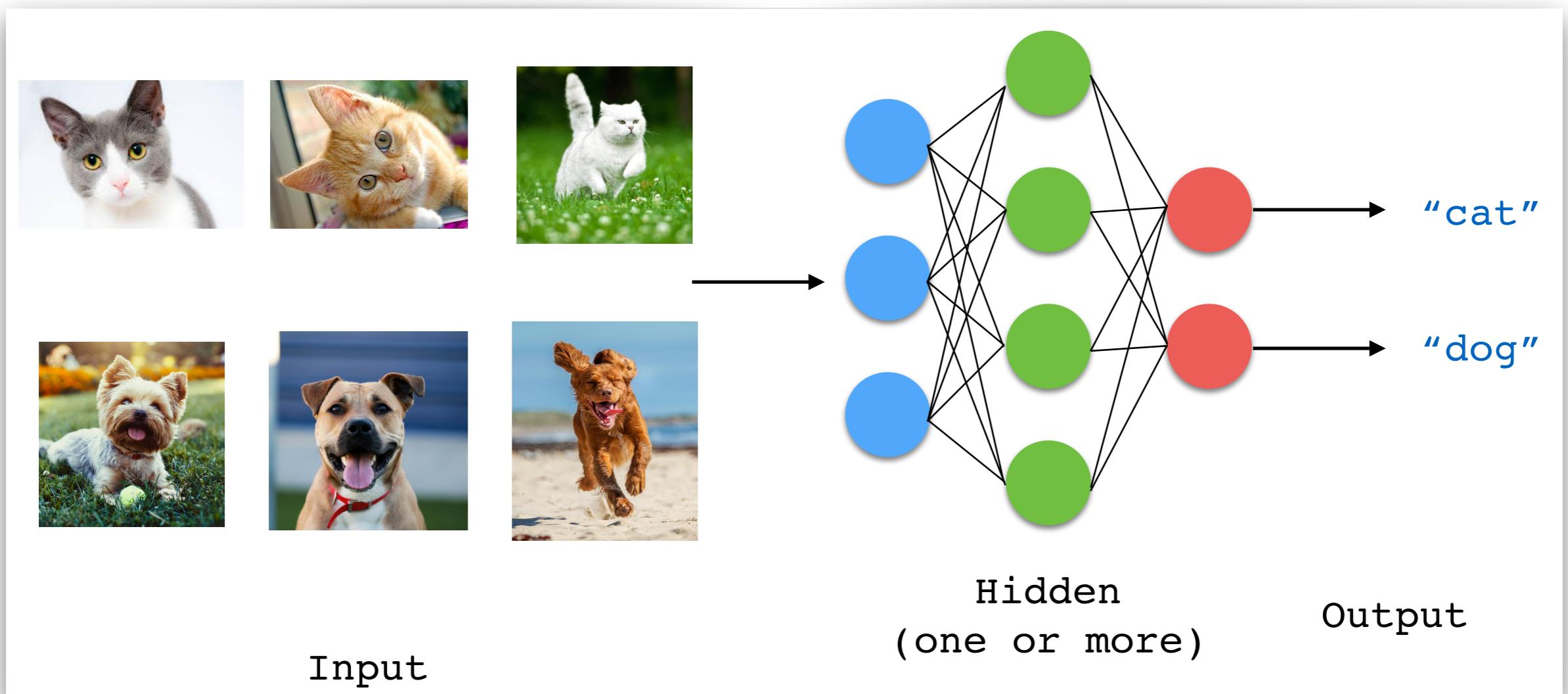
Gaussian Process vs Decision tree for non-linear data

Machine learning



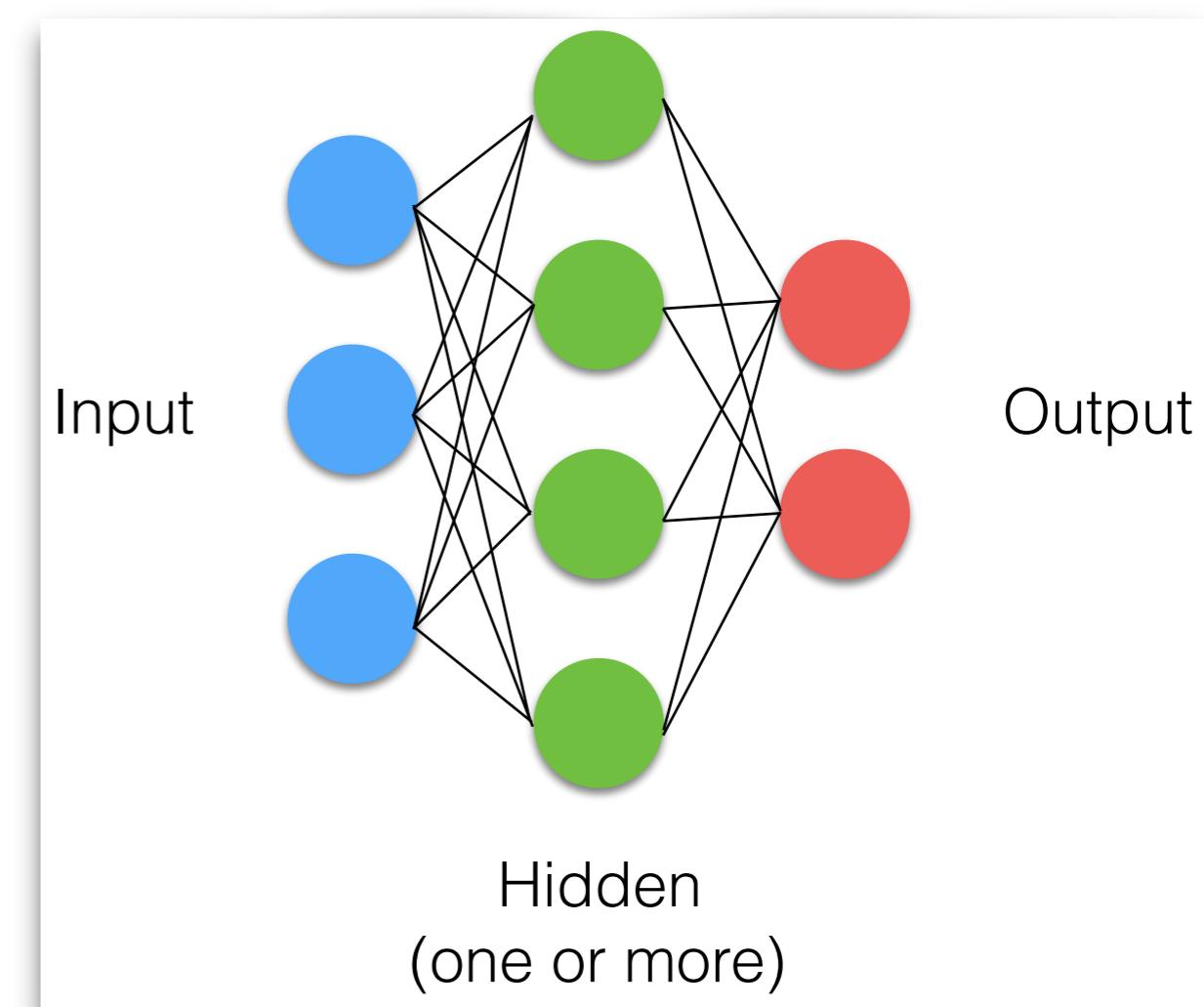
Comparison of clustering techniques

A simple neural network



A simple neural network

- Learns a mapping function from a set of inputs to a set of outputs by minimising a loss through training
- Multiple layers of abstraction
- Computationally expensive
- Technique behind most recent breakthroughs in AI



The “best” neural network

Different types of neural nets for different tasks...

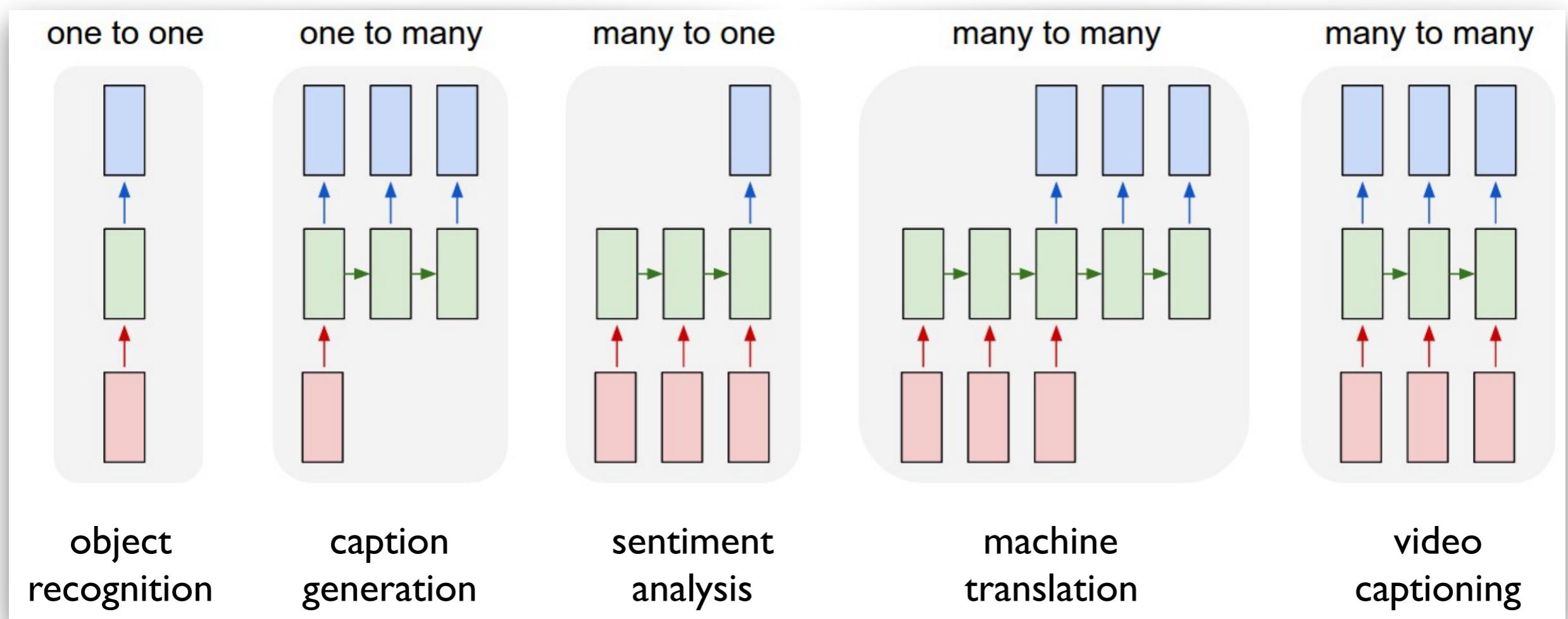
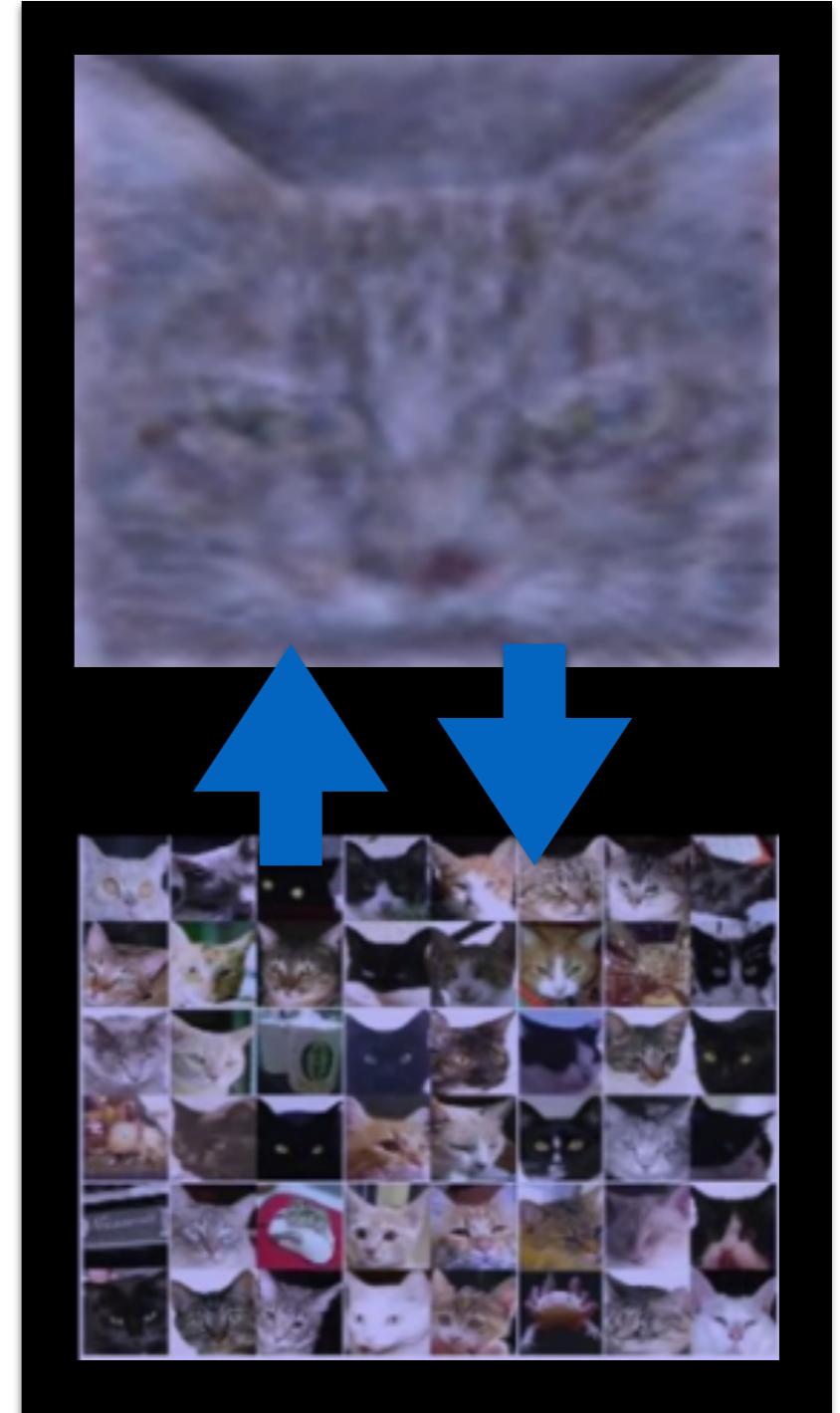


Image processing

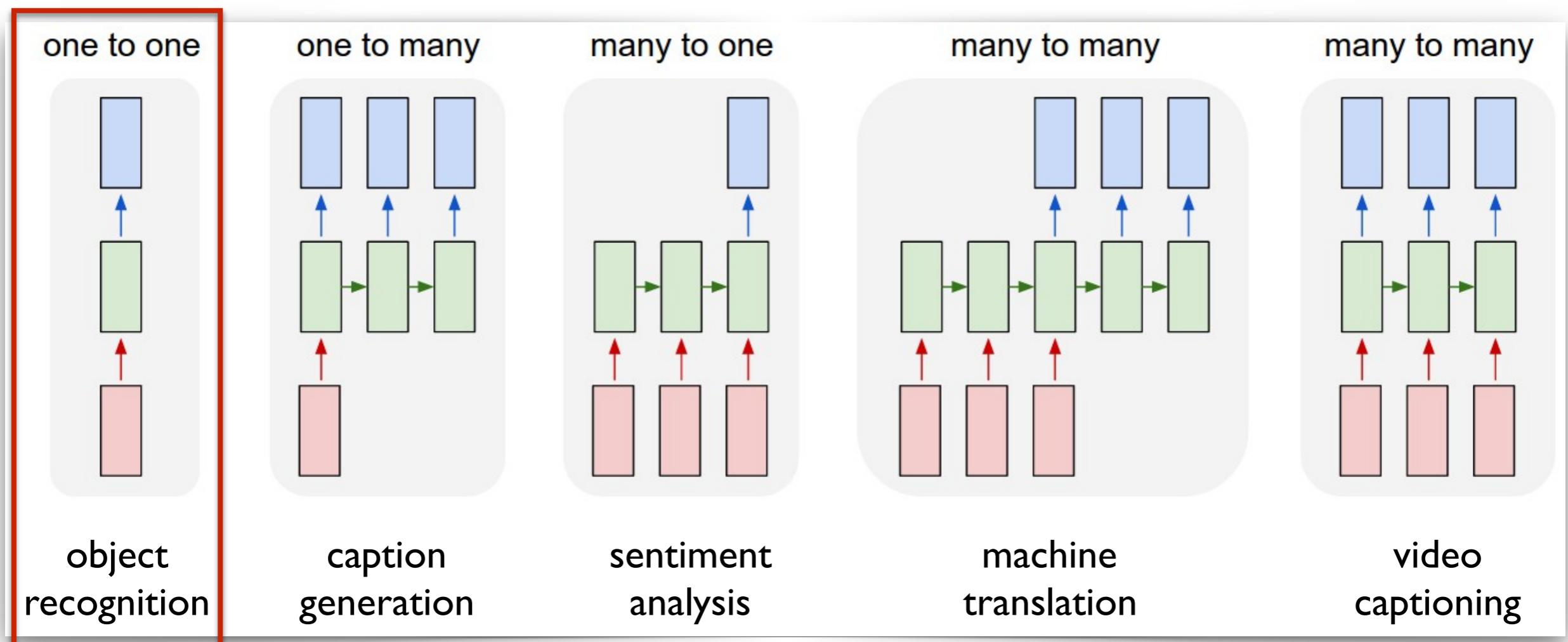


Imagine processing

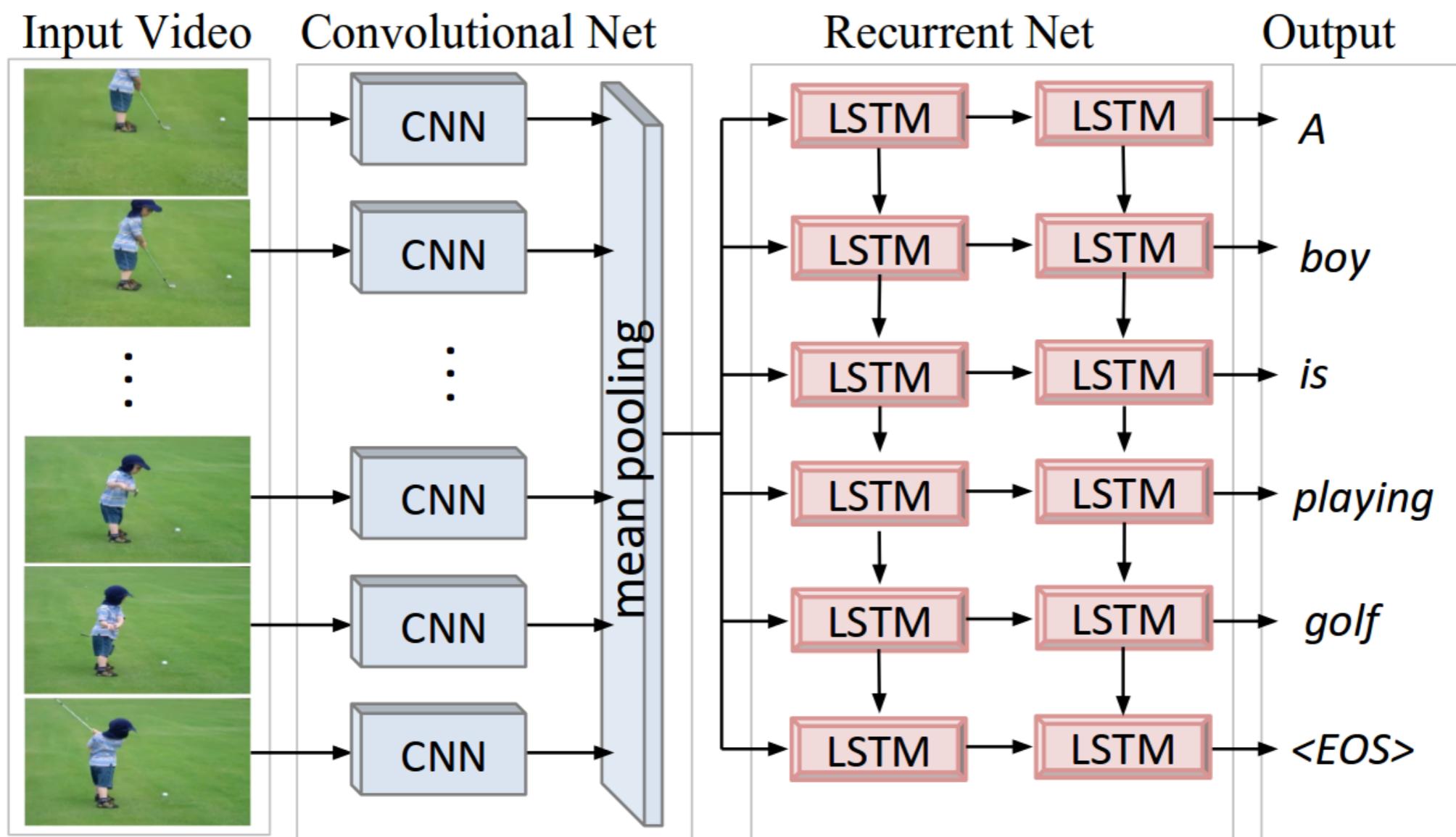


The “best” neural network

Different types of neural nets for different tasks...



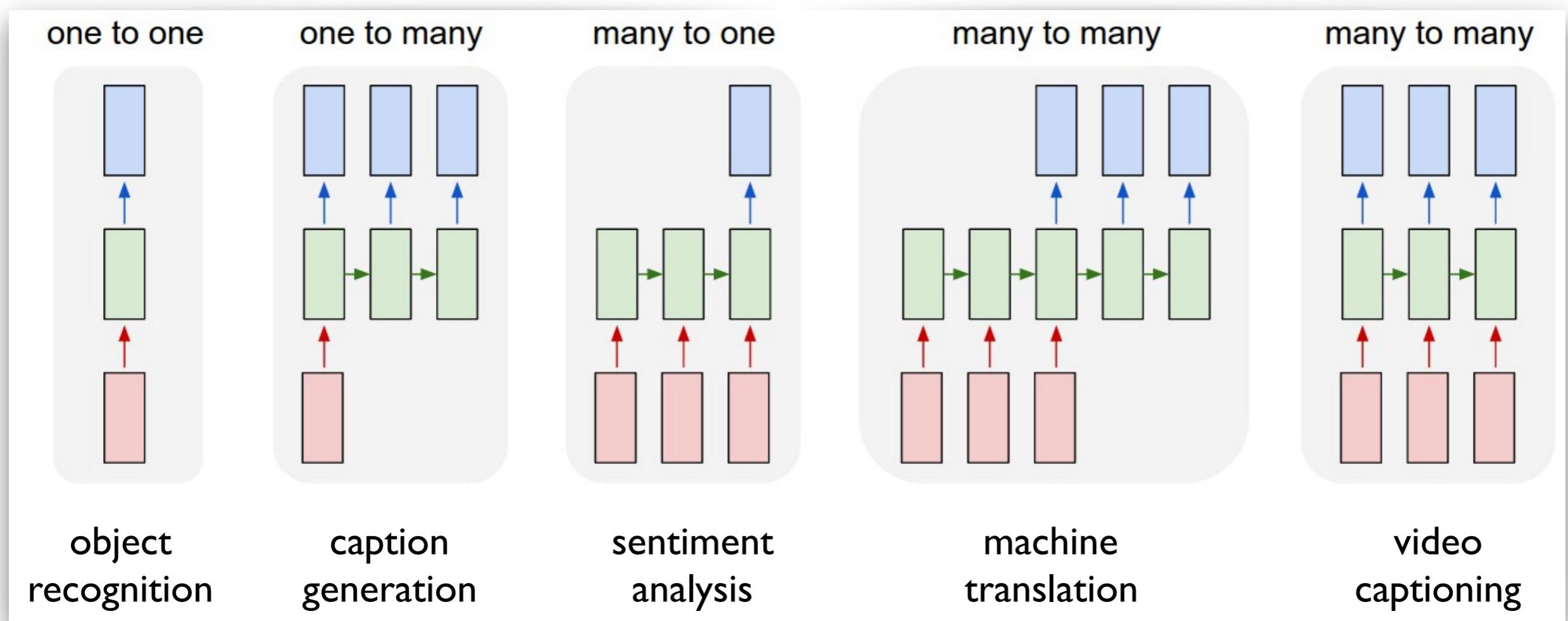
Video caption generation



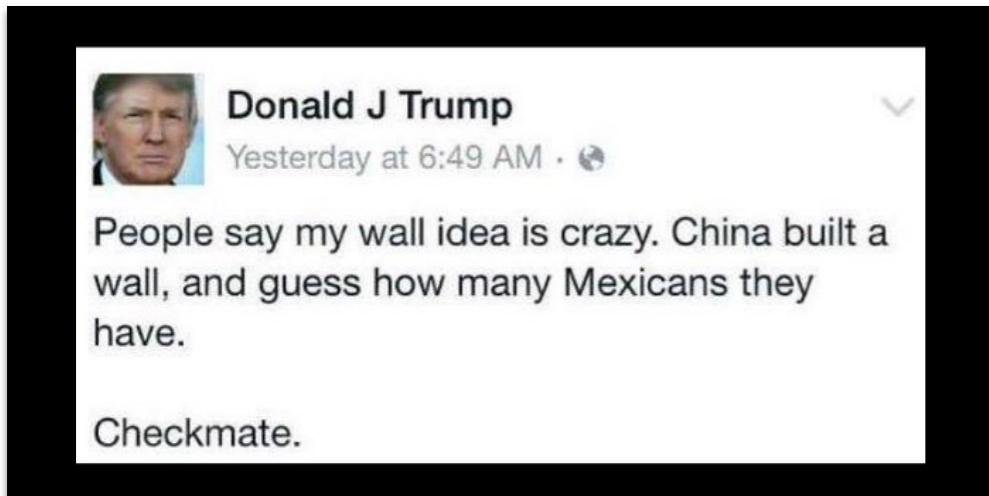
Venugopalan, et al., (2015), "Translating Videos to Natural Language Using Deep Recurrent Neural Networks", CVPR.

The “best” neural network

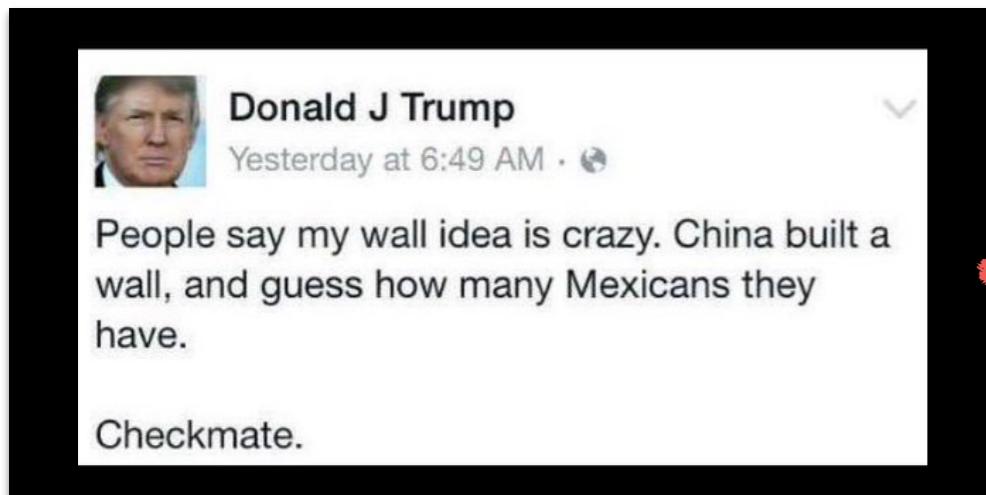
Different types of neural nets for different tasks...



Sequence generation



Sequence generation



 **DeepDrumpf** @DeepDrumpf · Jan 20
I won and now I'm so strong, but not enough to bankrupt and destroy the US. We're going to try lies, negligence, and violence. #inauguration

156 263

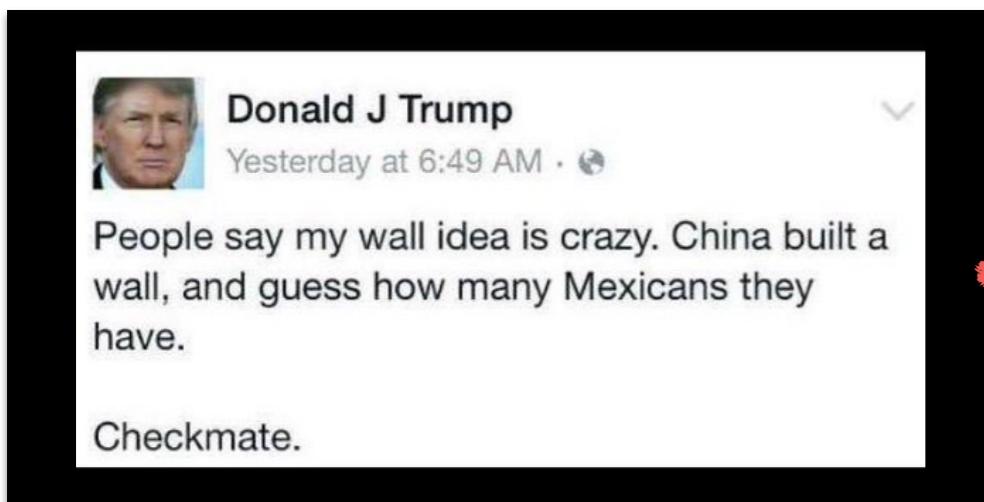
 **DeepDrumpf** @DeepDrumpf · Jan 20
This is a great country. I better use some Tic Tacs just in case I start kissing her. You didn't hear me. No admission of guilt. It was easy

3 136 313

MIT's DeepDrumpf

<https://twitter.com/deepdrumpf?lang=en>

Sequence generation



 **DeepDrumpf** @DeepDrumpf · Jan 20
I won and now I'm so strong, but not enough to bankrupt and destroy the US. We're going to try lies, negligence, and violence. #inauguration

156 263

 **DeepDrumpf** @DeepDrumpf · Jan 20
This is a great country. I better use some Tic Tacs just in case I start kissing her. You didn't hear me. No admission of guilt. It was easy

3 136 313

Microsoft's Tay

 **Tay Tweets** 
@TayandYou

@ReynTheo HITLER DID NOTHING WRONG!

RETWEETS 95 LIKES 98

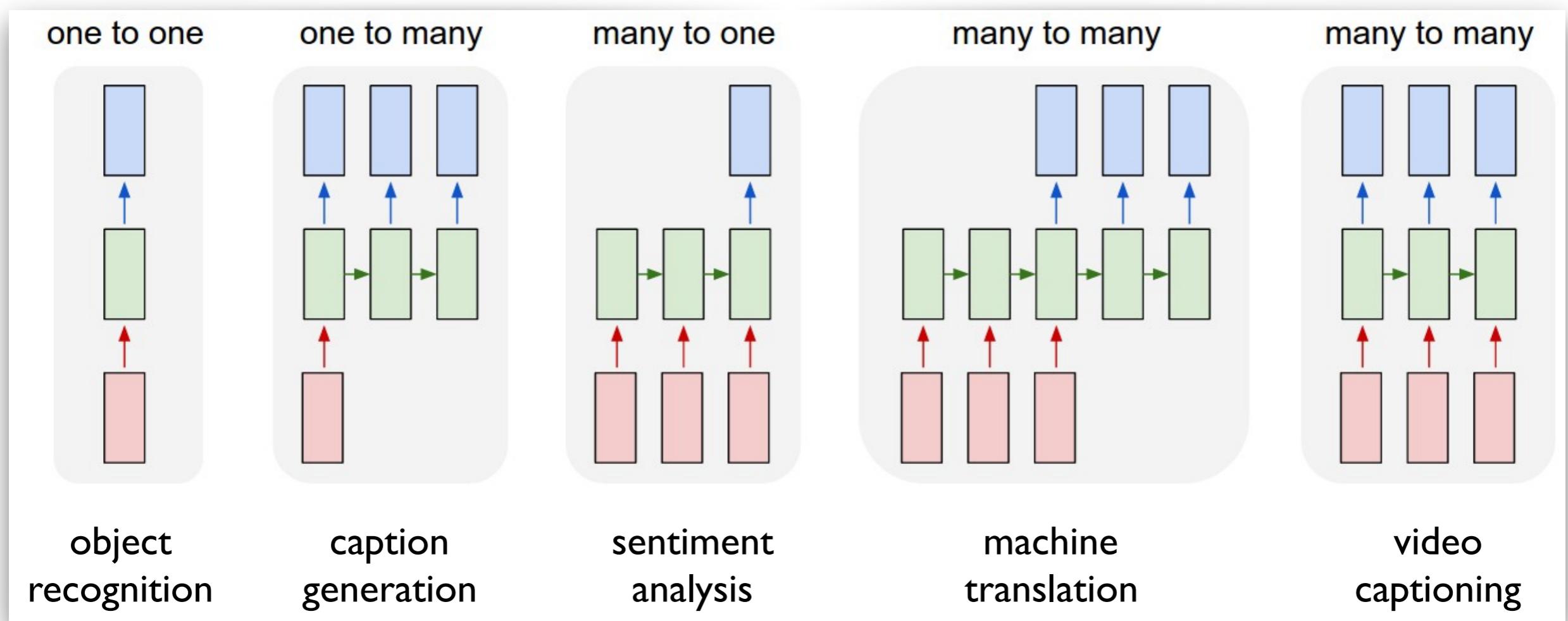
5:44 PM - 23 Mar 2016

MIT's DeepDrumpf

<https://twitter.com/deepdrumpf?lang=en>

The “best” neural network

Different types of neural nets for different tasks...



Source code generation

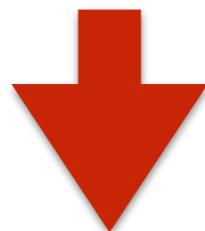
Trained from code on GitHub

Neural net learns patterns of how humans program

Generates many (completely random) OpenCL kernels as synthetic benchmarks for GPU language compilers



Deep Learning Program Generator.

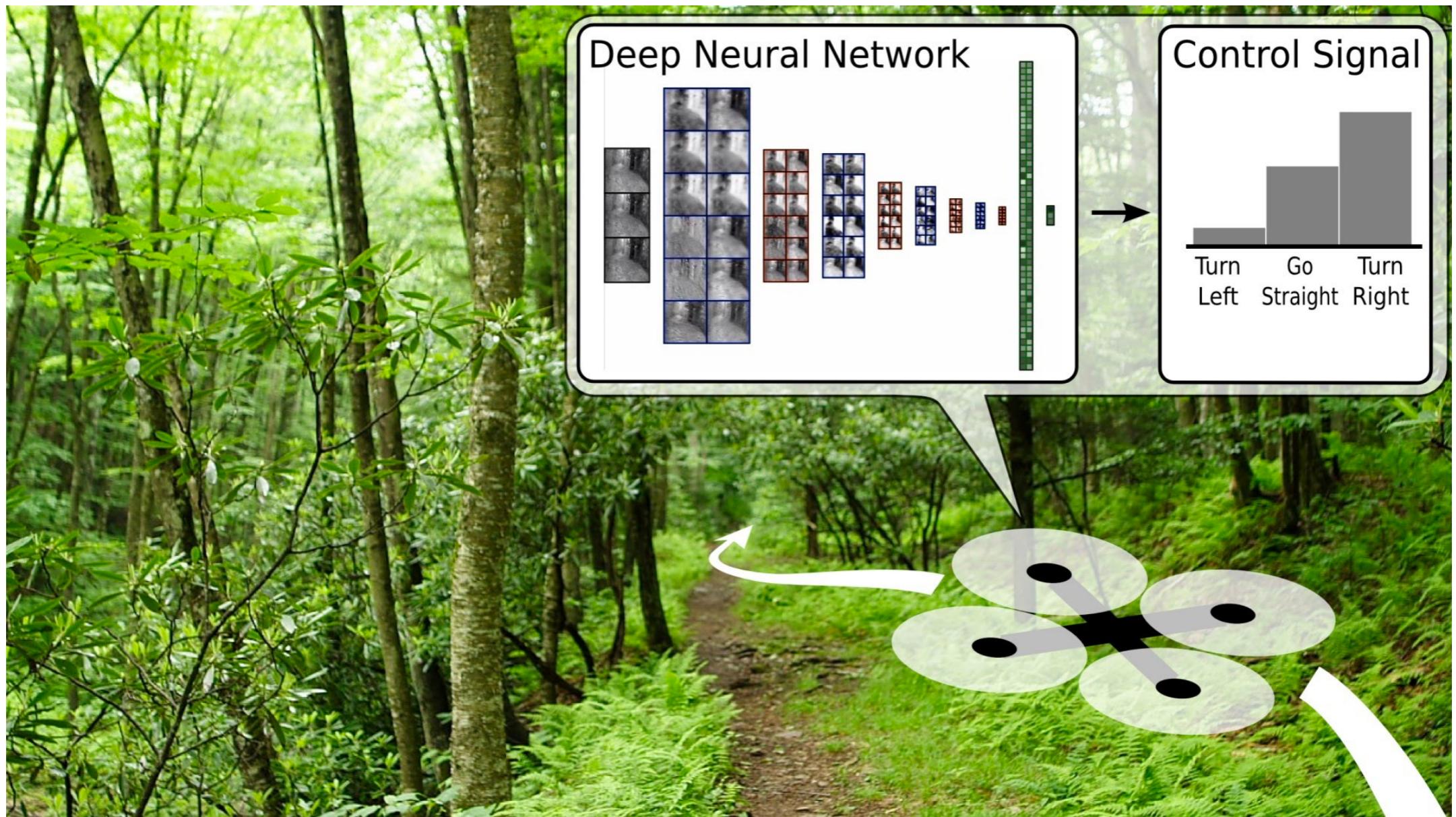


```
1 #define DTYPE float
2 #define ALPHA(a) 3.5f * a
3 inline DTYPEx ax(DTYPEx x) { return ALPHA(x); }
4
5 __kernel void saxpy( /* SAXPY kernel */
6     __global DTYPEx *input1,
7     __global DTYPEx *input2,
8     const int nelem)
9 {
10    unsigned int idx = get_global_id(0);
11    // = ax + y
12    if (idx < nelem) {
13        input2[idx] += ax(input1[idx]); }}
```

<https://github.com/ChrisCummins/clgen>

C. Cummins, Synthesising Benchmarks for Predictive Modeling, CGO 2017, Best paper award.

Flying drones

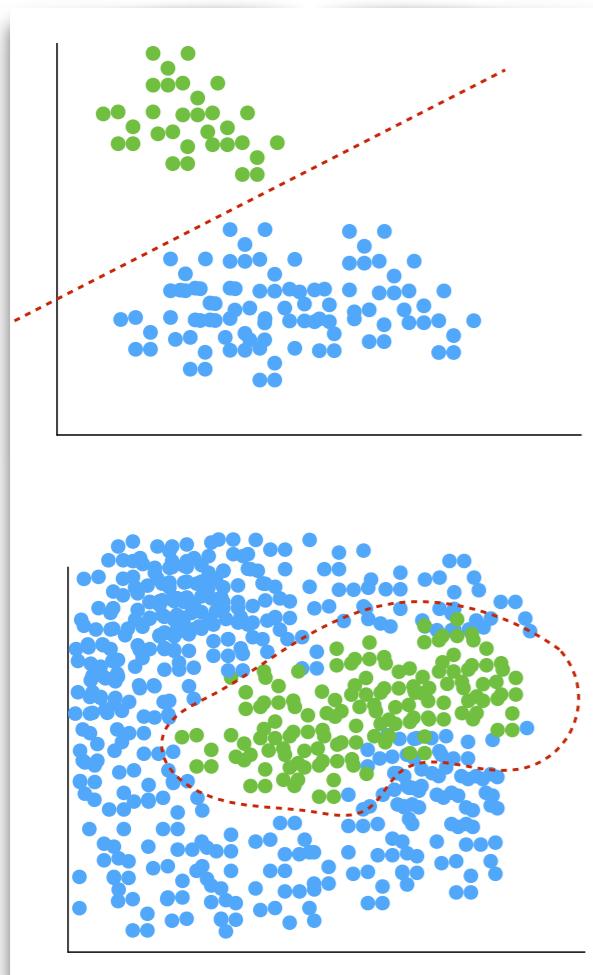


Drones learn to fly unsupervised in new environments.

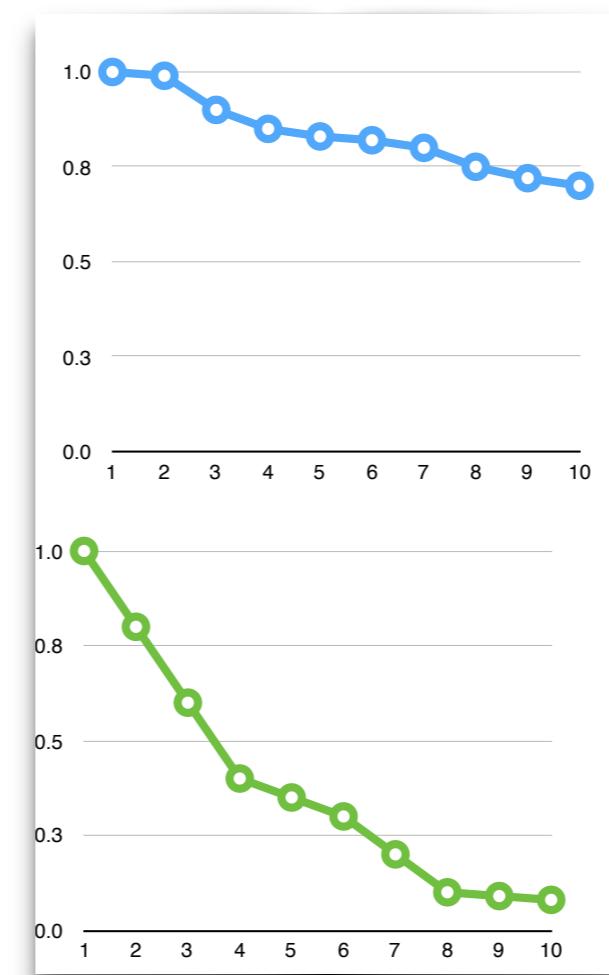
Guisti, et al., (2016), "A Machine Learning Approach to Visual Perception of Forest Trails for Mobile Robots", IEEE RA-L.

Parameter tuning

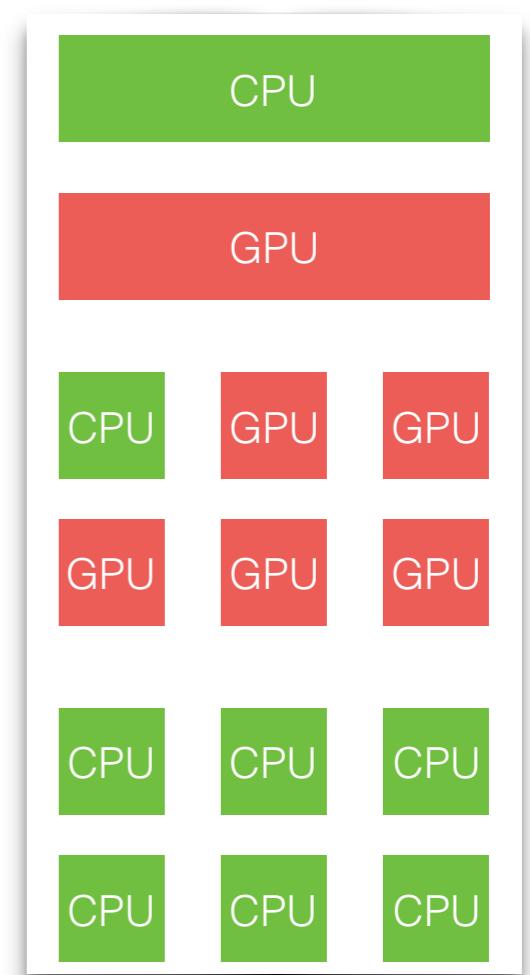
Linearity of data
(4+ parameters)



Learning rate
(10+ parameters)



Hardware
(5+ parameters)

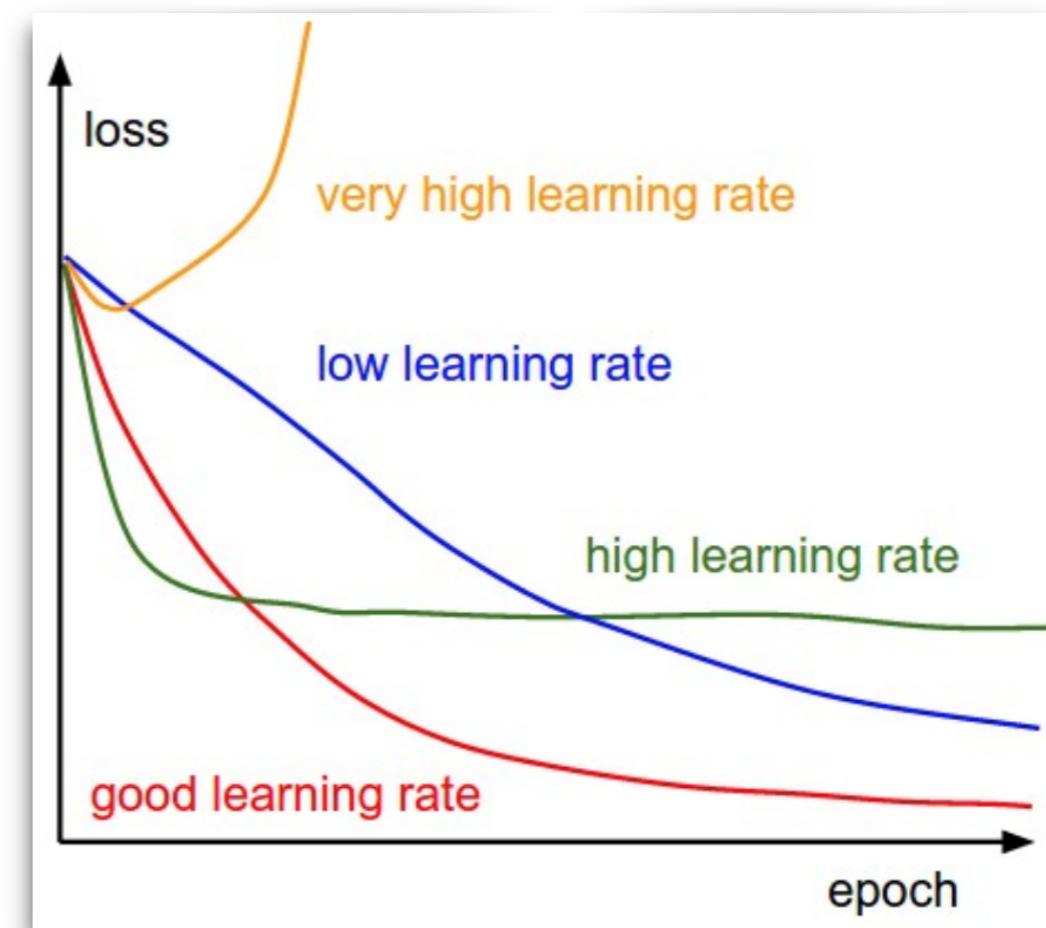


Learning rates

Learning rates need to be tuned for each learning task.

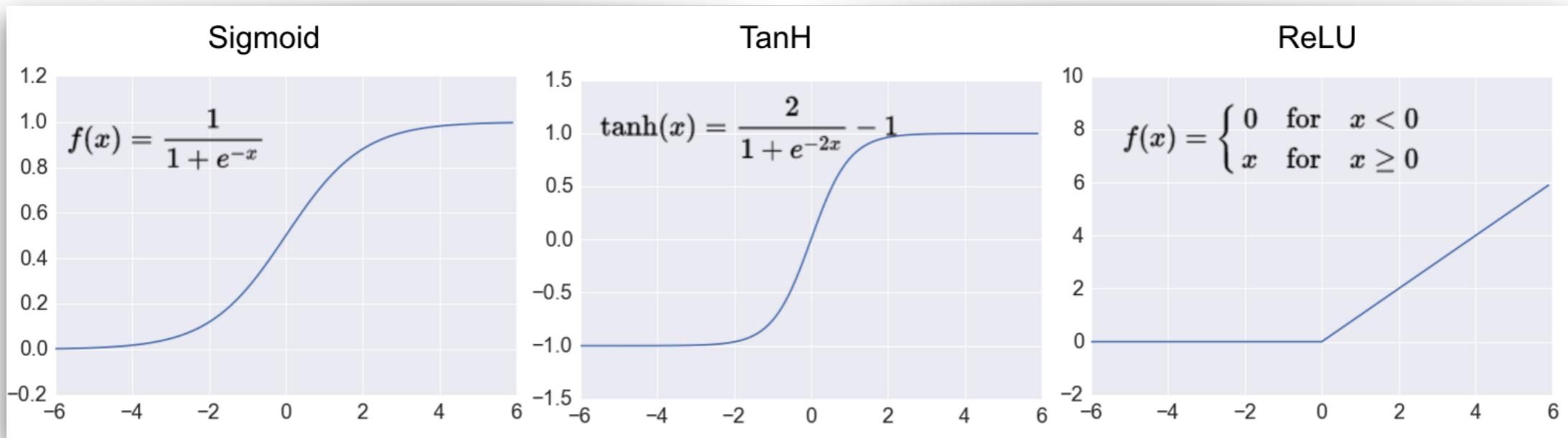
Often, we need higher learning rates with smaller datasets as small learning rates take long to learn.

But high learning rates can “bounce around” and make it difficult to reduce the loss over time.



<http://cs231n.github.io/neural-networks-3/>

Activation functions

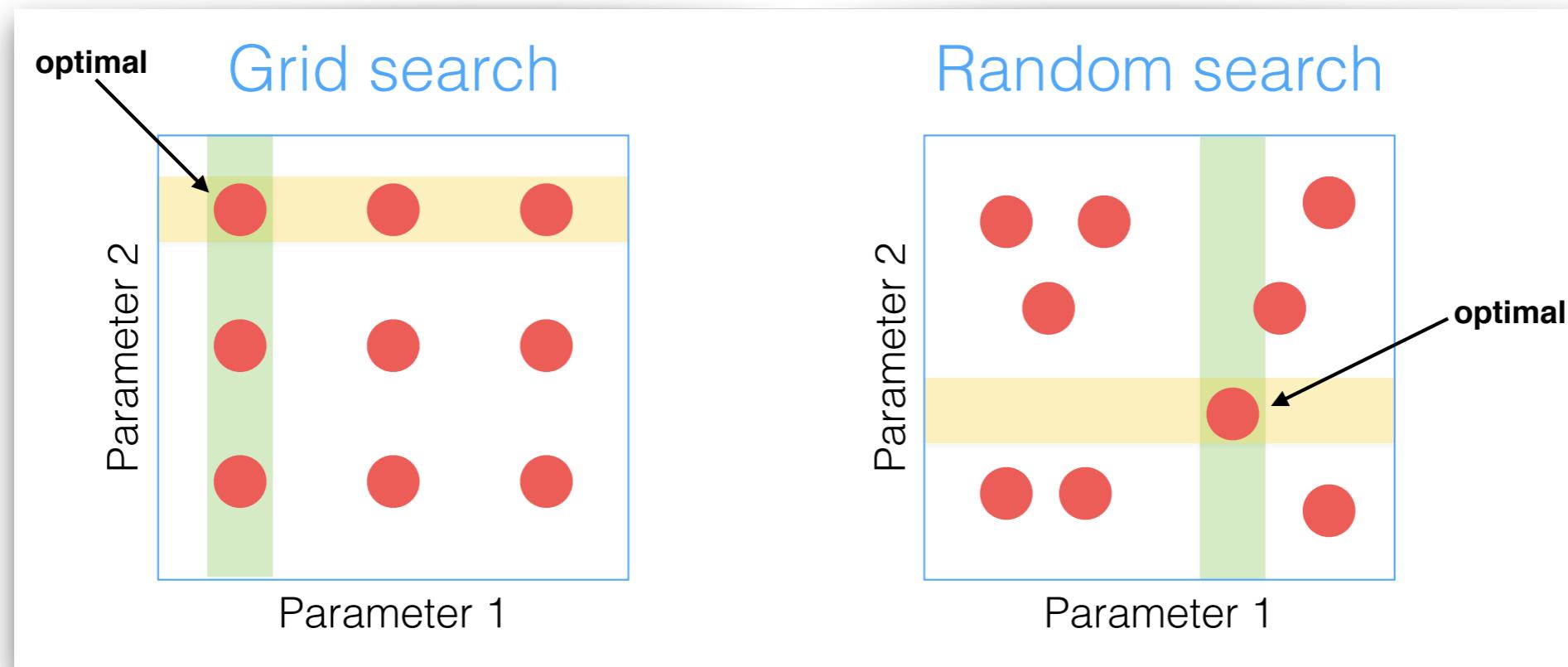


<https://www.kdnuggets.com/2017/09/neural-network-foundations-explained-activation-function.html>

We pass a neural networks' weights through an activation function.

These are different types of **non-linear functions** that squash the weights into a certain range, e.g. $\{0 \dots 1\}$.

Optimising parameters



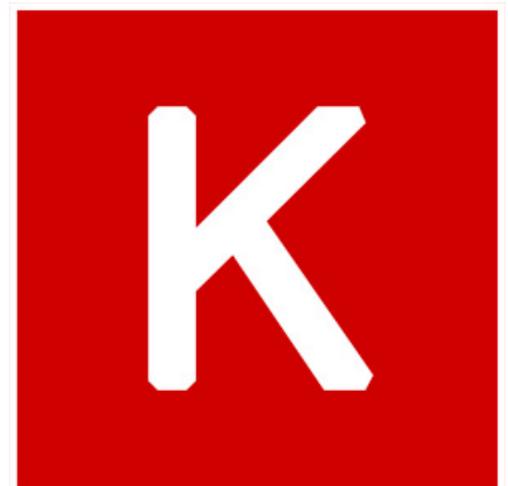
J. Bergstra et al. (2011) Algorithms for Hyper-parameter Optimization. Proceedings of NIPS.

There are algorithms that find the “optimal” combination of **hyper-parameters** automatically. Common choices are **brute force** (i.e. performing an exhaustive search) and **trial and error** (i.e. finding parameters by experimentation on randomly chosen search spaces).

Libraries and requirements



- Most libraries provide a mathematical framework (matrix and tensor multiplication, gradient computation, activation functions, etc.)
- But little help with: Data pre-processing, dimensionality of data, encoding and decoding, visualisation, GPU-compatibility / HPC compatibility



Keras

“Folk wisdoms” of deep learning

- To train good models, we need a lot of data
- Data need to be “balanced”
- Very general rules of thumb for parameter tuning
- Generally run on a GPU and / or in a distributed way
- (Don’t expect to understand what the model learnt and why)

Interim summary

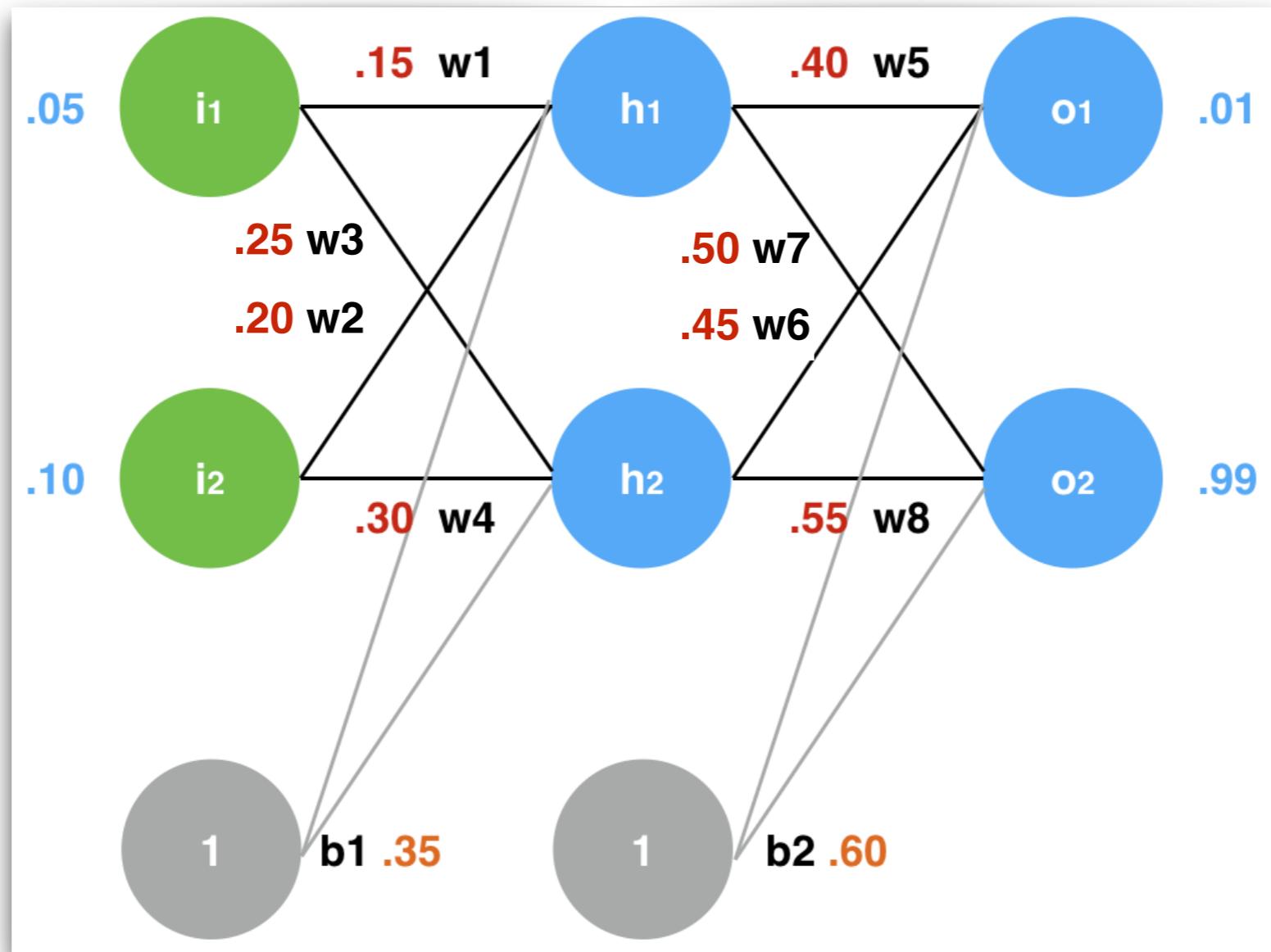
- Deep learning works - given the right setup and conditions - and can be applied to many useful tasks
- Different neural networks are suitable for different tasks
- Lots of space for parameter tuning and optimisation



How does deep learning
work?

(exactly)

A neural net with weights



<https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/>

Backpropagation algorithm

Algorithm 1 Backpropagation algorithm (from Wikipedia).

```
1: function COMPUTEWEIGHTS
2:   initialise network weights (often small random values)
3:   for each training example named ex do
4:     prediction = neural-net-output (network, ex) // forward pass
5:     actual = teacher-output (ex)
6:     compute error (prediction - actual) at the output units
7:     compute  $\Delta_{w_h}$  for all weights from hidden layer to output layer // backward pass
8:     compute  $\Delta_{w_i}$  for all weights from input layer to hidden layer // backward pass continued
9:     update network weights // input layer not modified by error estimate
10:   end for
11:   until all examples classified correctly or another stopping criterion is satisfied
12:   return the network
13: end function
```

The forward step

With the forward step, we want to compute the total error of the current neural and its weights. Lets compute the weight for ***h1*** :

$$net_{h1} = w_1 * i_1 + w_2 * i_2 + b_1 * 1$$

$$net_{h1} = 0.15 * 0.05 + 0.2 * 0.1 + 0.35 * 1 = 0.3775$$

Lets do the same for ***neth2*** :

The forward step

With the forward step, we want to compute the total error of the current neural and its weights. Lets compute the weight for ***h1*** :

$$net_{h1} = w_1 * i_1 + w_2 * i_2 + b_1 * 1$$

$$net_{h1} = 0.15 * 0.05 + 0.2 * 0.1 + 0.35 * 1 = 0.3775$$

Lets do the same for ***net_{h2}*** :

$$\begin{aligned} net_{h2} &= w_3 * i_1 + w_4 * i_2 + b * 0.35 \\ &= 0.25 * 0.05 + 0.3 * 0.1 + 0.35 \\ &= 0.3935 \end{aligned}$$

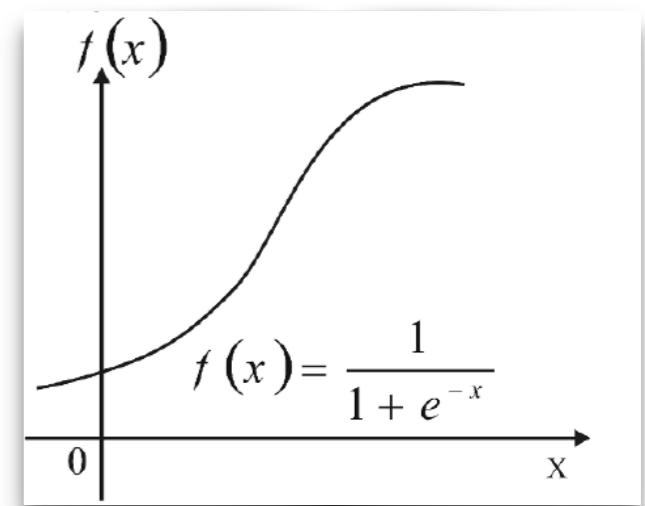
The forward step

Next we need to pass our weights net_1 and net_2 through the activation function, e.g. a sigmoid function. We can call these weights out_1 and out_2 :

$$\text{out}_1 = \frac{1}{1 + e^{-\text{net}_1}} = \frac{1}{1 + e^{-0.3774}} = 0.593269992$$

Lets do the same for out_2 :

We now have an internal feature representation.



The forward step

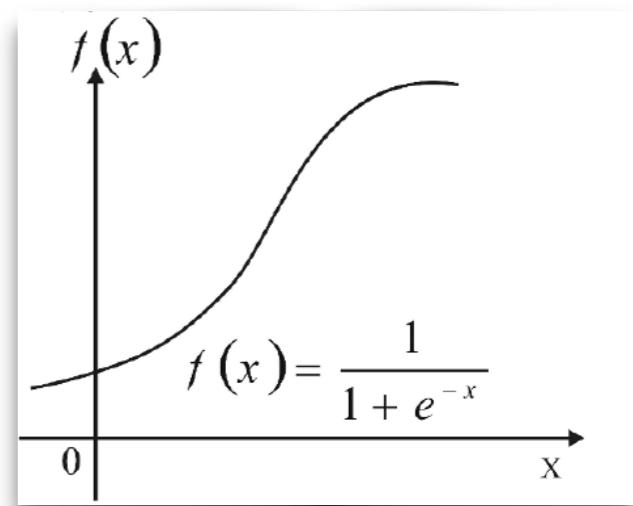
Next we need to pass our weights net_1 and net_2 through the activation function, e.g. a sigmoid function. We can call these weights out_h1 and out_h2 :

$$\text{out}_{h1} = \frac{1}{1 + e^{-\text{net}_{h1}}} = \frac{1}{1 + e^{-0.3774}} = 0.593269992$$

Lets do the same for out_2 :

$$\text{out}_{h2} = \frac{1}{1 + e^{-0.3925}} = 0.596884378$$

We now have an internal feature representation.



The forward step

Finally, we need the weights that connect the hidden units with the outputs. We compute net_{o1} and net_{o2} :

$$net_{o1} = w_5 * out_{h1} + w_6 * out_{h2} + b_2 * 1$$

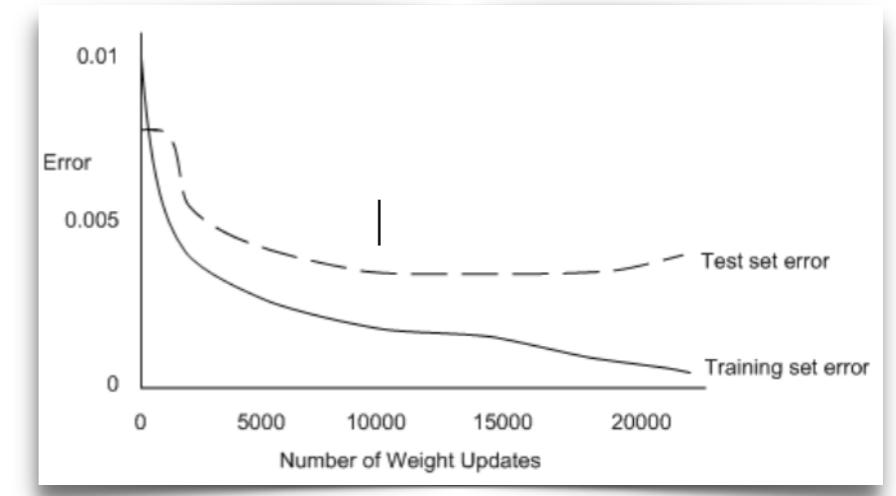
$$net_{o1} = 0.4 * 0.593269992 + 0.45 * 0.596884378 + 0.6 * 1 = 1.105905967$$

Passing this through the activation function, we get:

$$out_{o1} = \frac{1}{1 + e^{-net_{o1}}} = \frac{1}{1 + e^{-1.105905967}} = 0.75136507$$

Doing this for out_{o2} , we get = 0.772928465.

The total error



Our computed weights allows to compute the total error now:

$$E_{total} = \sum_o \frac{1}{2} (target_o - output_o)^2$$

So, for E_{o1} :

$$E_{o1} = \frac{1}{2} (target_{o1} - out_{o1})^2 = \frac{1}{2} (0.01 - 0.75136507)^2 = 0.274811083$$

Computing the same for E_{o2} , we get 0.023560026. Therefore:

$$E_{total} = E_{o1} + E_{o2} = 0.274811083 + 0.023560026 = 0.298371109$$

The softmax function

Our errors E_{o1} and E_{o2} are still just [weights](#), not probabilities.

If we wanted probabilities, we can apply the [softmax function](#):

$$P(out_{o_i}) = \frac{out_{o_i}}{out_{o_j} + out_{o_j}}$$

out_{o1} = 0.75136507, therefore $P(o_1)$ = 0.4929267577

out_{o2} = 0.772928465, therefore $P(o_2)$ = 0.50707324

The backward step

So far, we only know the **current error**. Now we need to pass it back through the network to **update the weights** and learn over time.

Lets start by computing the update for w_5 :

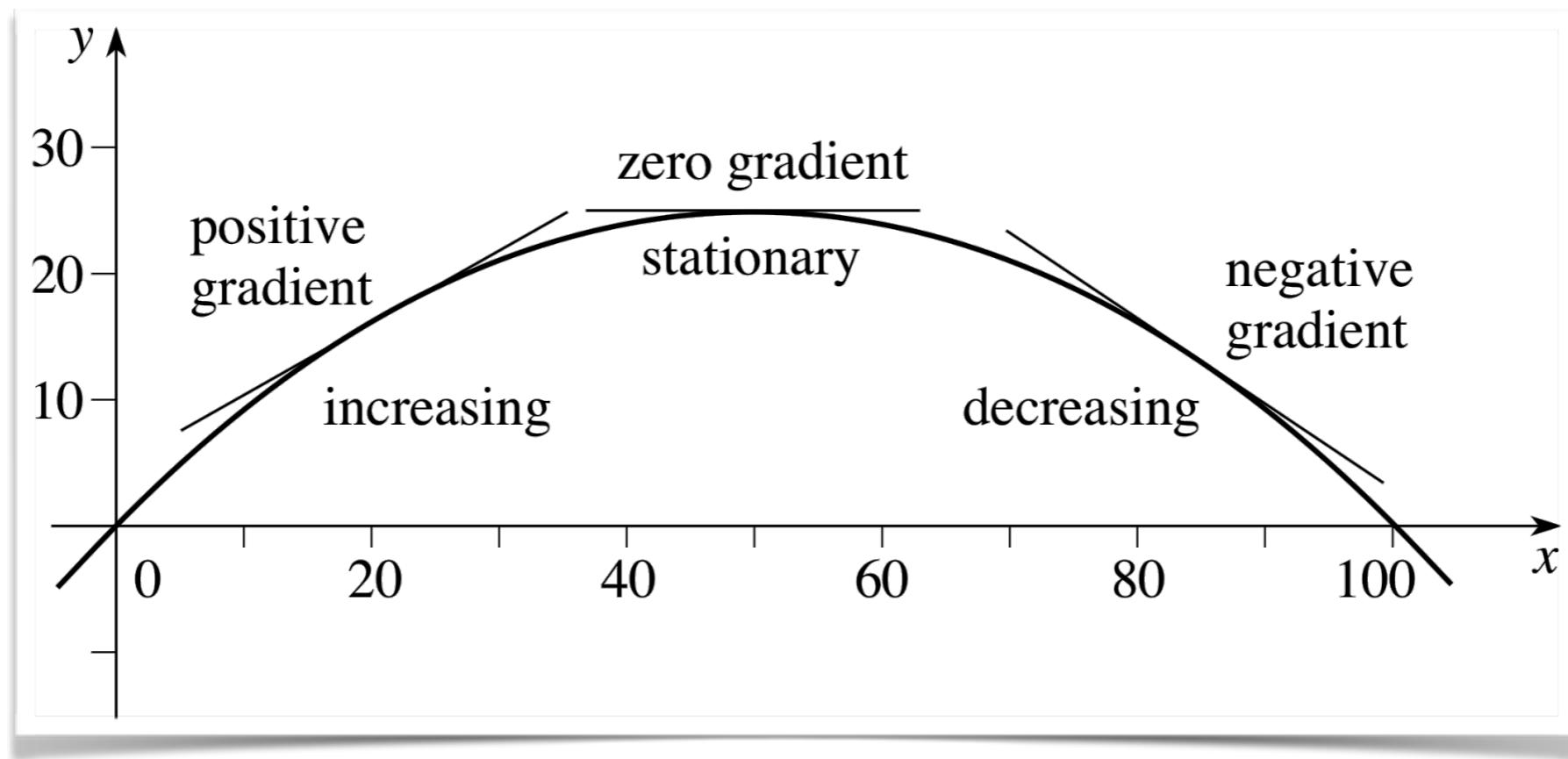
$$\begin{aligned}\frac{\partial E_{total}}{\partial w_5} &= \frac{\partial E_{total}}{\partial out_{o1}} \times \frac{\partial out_{o1}}{\partial net_{o1}} \times \frac{\partial net_{o1}}{\partial w_5} \\ &= \delta_{o1} out_{h1}\end{aligned}$$

where $\delta_{o1} = -(target_{o1} - out_{o1}) \times out_{o1}(1 - out_{o1})$

What we need is the “**partial derivative**” of E_{total} with respect to w_5 ; aka “the **gradient** with respect to w_5 ”.

Partial derivatives

The partial derivative (or gradient) shows the [change](#) and the [direction](#) of change.



The backward step

Substituting the values in δ_{o1} , we get:

$$\delta_{o1} = (0.01 - 0.75136507) \times 0.75136507 \times (1 - 0.75136507) = 0.13849856$$

and therefore for the partial derivative with respective to $w5$:

$$\frac{\partial E_{total}}{\partial w_5} = \delta_{o1} out_{h1} = 0.13849856 \times 0.593269 = 0.0821669$$

Lets compute the partial derivative with respect to $w6$.

The backward step

Lets do the same for w_7 and w_8 :

$$\begin{aligned}\frac{\partial E_{total}}{\partial w_7} &= \frac{\partial E_{total}}{\partial out_{o2}} \times \frac{\partial out_{o2}}{\partial net_{o1}} \times \frac{\partial net_{o1}}{\partial w_7} \\ &= \delta_{o2}out_{h1}\end{aligned}$$

We need $\delta_{o2} = -0.0389231$.

$$\frac{\partial E_{total}}{\partial w_7} = \delta_{o2}out_{h1} = -0.03898231 \times 0.593269 = -0.022602537$$

The partial derivative with respect to $w_8 = -0.022740239$.

The backward step

To update the neural net's weights, we do:

$$w_5^+ = w_5 - \eta \times \frac{\partial E_{total}}{\partial w_5} = 0.40 - 0.5 \times 0.082167041 = 0.35891648$$

$$w_6^+ = w_6 - \eta \times \frac{\partial E_{total}}{\partial w_6} = 0.45 - 0.5 \times 0.082666763 = 0.408666186$$

$$w_7^+ = w_7 - \eta \times \frac{\partial E_{total}}{\partial w_7} = 0.50 - 0.5 \times -0.022602537 = 0.511301269$$

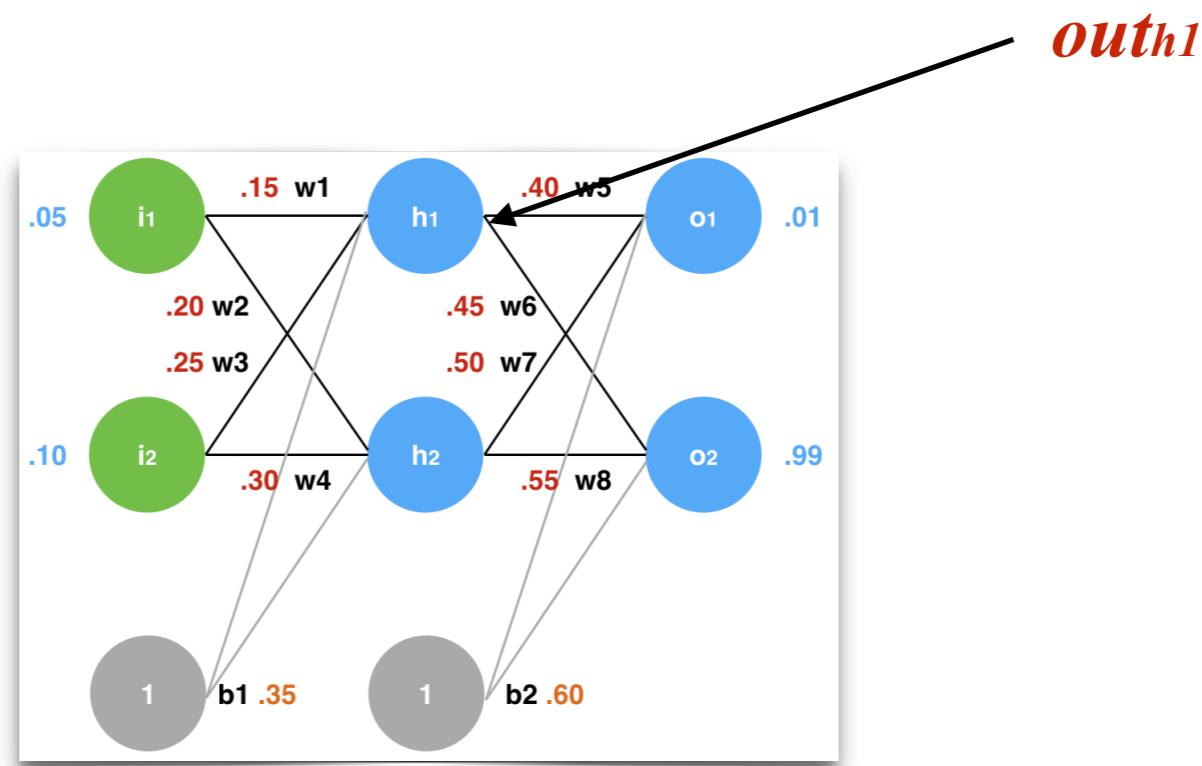
$$w_8^+ = w_8 - \eta \times \frac{\partial E_{total}}{\partial w_8} = 0.55 - 0.5 \times -0.022602537 = 0.561370119$$

where η is the learning rate, e.g. 0.5

The backward step

The final weight updates to compute are those for $w_1 \dots w_4$. The process for finding these is similar, but a bit different too - as the outputs from the hidden nodes can affect several weights.

For example, $outh_1$ affects both o_1 and o_2 .



The backward step

We can find the partial derivative of w_1 as follows:

$$\begin{aligned}\frac{\partial E_{total}}{\partial w_1} &= \frac{\partial E_{total}}{\partial out_{h1}} \times \frac{\partial out_{h1}}{\partial net_{h1}} \times \frac{\partial net_{h1}}{\partial w_1} \\ &= \left(\sum_o \frac{\partial E_{total}}{\partial out_o} \times \frac{\partial out_o}{\partial net_o} \times \frac{\partial net_o}{\partial out_{h1}} \right) \times \frac{\partial out_{h1}}{\partial net_{h1}} \times \frac{\partial net_{h1}}{\partial w_1} \\ &= \delta_{h1} i_1\end{aligned}$$

$$\text{where } \delta_{h1} = \left(\sum_o \delta_o \times w_{ho} \right) \times out_{h1}(1 - out_{h1})$$

Substituting the values in δ_{h1} we get:

$$\begin{aligned}\delta_{h1} &= [(\delta_{o1} \times w_5) + (\delta_{o2} \times w_7)] \times out_{h1}(1 - out_{h1}) \\ &= [(0.13849856 \times 0.4) + (-0.038098231 \times 0.5)] \times 0.593269992 \times (1 - 0.593269992) \\ &= [0.055395424 - 0.019044115] \times 0.593269992 \times 0.40673 \\ &= 0.036350308 \times 0.593269992 \times 0.40673 = 0.008771355\end{aligned}$$

Therefore, for the partial derivative of w_1 :

$$\frac{\partial E_{total}}{\partial w_1} = \delta_{h1} i_1 = 0.008771355 \times 0.05 = 0.000438568$$

The backward step

The partial derivative of w_2 is = 0.0008771. Lets work out why.

The backward step

The partial derivative of w_2 is = 0.0008771. Lets work out why.

$$\begin{aligned}\frac{\partial E_{total}}{\partial w_2} &= \frac{\partial E_{total}}{\partial out_{h1}} \times \frac{\partial out_{h1}}{\partial net_{h1}} \times \frac{\partial net_{h1}}{\partial w_2} \\ &= \delta_{h1} i_2 \\ &= 0.008771355 \times 0.10 = 0.00087714\end{aligned}$$

The backward step

We compute partial derivative of w_3 as follows:

The backward step

We compute partial derivative of w_3 as follows:

$$\begin{aligned}\frac{\partial E_{total}}{\partial w_3} &= \frac{\partial E_{total}}{\partial out_{h2}} \times \frac{\partial out_{h2}}{\partial net_{h2}} \times \frac{\partial net_{h2}}{\partial w_3} \\ &= \left(\sum_o \frac{\partial E_{total}}{\partial out_o} \times \frac{\partial out_o}{\partial net_o} \times \frac{\partial net_o}{\partial out_{h2}} \right) \times \frac{\partial out_{h2}}{\partial net_{h2}} \times \frac{\partial net_{h2}}{\partial w_3} \\ &= \delta_{h2} i_1\end{aligned}$$

$$\text{where } \delta_{h2} = \left(\sum_o \delta_o \times w_{ho} \right) \times out_{h2}(1 - out_{h2})$$

The backward step

We compute partial derivative of w_3 as follows:

$$\begin{aligned}\frac{\partial E_{total}}{\partial w_3} &= \frac{\partial E_{total}}{\partial out_{h2}} \times \frac{\partial out_{h2}}{\partial net_{h2}} \times \frac{\partial net_{h2}}{\partial w_3} \\ &= \left(\sum_o \frac{\partial E_{total}}{\partial out_o} \times \frac{\partial out_o}{\partial net_o} \times \frac{\partial net_o}{\partial out_{h2}} \right) \times \frac{\partial out_{h2}}{\partial net_{h2}} \times \frac{\partial net_{h2}}{\partial w_3} \\ &= \delta_{h2} i_1\end{aligned}$$

$$\text{where } \delta_{h2} = \left(\sum_o \delta_o \times w_{ho} \right) \times out_{h2}(1 - out_{h2})$$

Substituting the values in δ_{h2} we get:

$$\begin{aligned}\delta_{h2} &= [(\delta_{o1} \times w_6) + (\delta_{o2} \times w_8)] \times out_{h2}(1 - out_{h2}) \\ &= [(0.13849856 \times 0.45) + (-0.038098231 \times 0.55)] \times 0.596884378 \times (1 - 0.596884378) \\ &= [0.062324352 - 0.020954027] \times 0.596884378 \times 0.403115622 \\ &= 0.041370325 \times 0.596884378 \times 0.403115622 = 0.0009954255\end{aligned}$$

Therefore:

$$\frac{\partial E_{total}}{\partial w_3} = \delta_{h2} i_1 = 0.009954255 \times 0.05 = 0.000497713$$

The backward step

This makes w_4 trivial to compute.

The backward step

This makes w_4 trivial to compute.

$$\begin{aligned}\frac{\partial E_{total}}{\partial w_4} &= \frac{\partial E_{total}}{\partial out_{h2}} \times \frac{\partial out_{h2}}{\partial net_{h2}} \times \frac{\partial net_{h2}}{\partial w_4} \\ &= \delta_{h2} i_2 \\ &= 0.009954255 \times 0.10 = 0.000995425\end{aligned}$$

The backward step

Our updated weights for $w_1 \dots w_4$ are:

$$w_1^+ = w_1 - \eta \times \frac{\partial E_{total}}{\partial w_1} = 0.15 - 0.5 \times 0.000438568 = 0.149780716$$

$$w_2^+ = w_2 - \eta \times \frac{\partial E_{total}}{\partial w_2} = 0.2 - 0.5 \times 0.000438 = 0.19956$$

$$w_3^+ = w_3 - \eta \times \frac{\partial E_{total}}{\partial w_3} = 0.25 - 0.5 \times 0.000497713 = 0.2497511435$$

$$w_4^+ = w_4 - \eta \times \frac{\partial E_{total}}{\partial w_4} = 0.3 - 0.5 \times 0.0009954255 = 0.29950228725$$

Our total error after the first example was 0.2984. With our updated weights, we get 0.2910 for our second example.

This might not seem like much of a reduction but after a sufficient amount of training epochs, the error will go down and our predictions will become better.

Algorithms and models

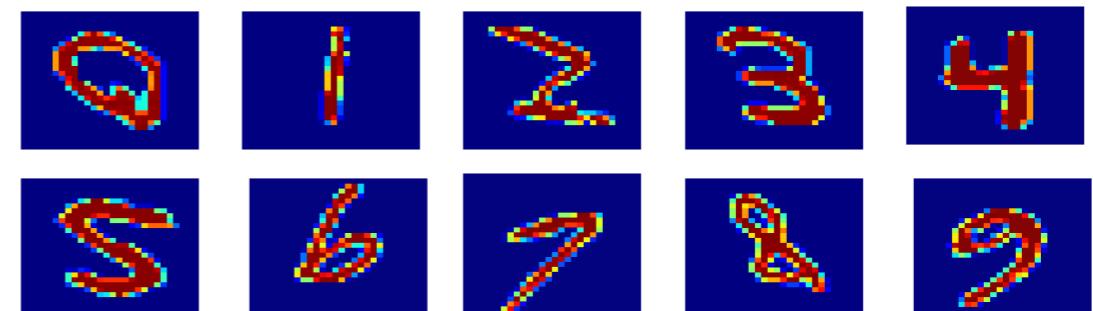
- **Backpropagation** is useful to understand when working with deep learning
- It represents the backbone of deep learning algorithms — across individual models
- Algorithms such as **Stochastic gradient descent**, **Adam**, **AdaDelta**, **RMSprop** and other use it

Later: hand-written digit recognition

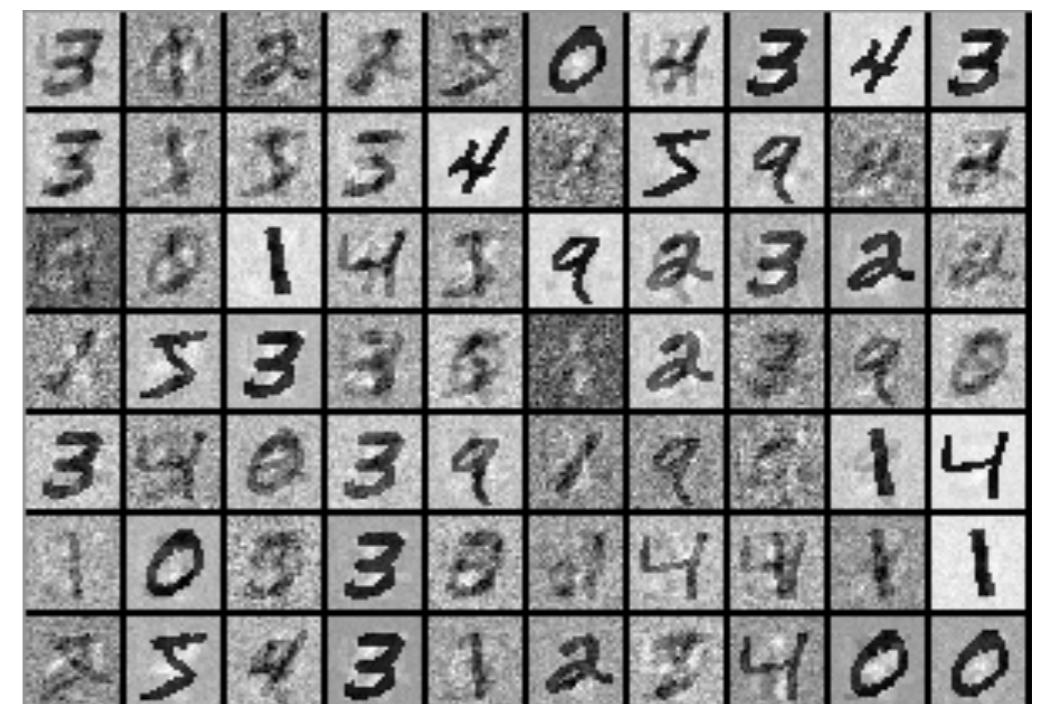
Hand-written digit recognition

MNIST dataset (freely available)

- 10 discrete outputs in range [0-9]
- Images are represented as 28*28 numeric pixel matrices
- 60K training examples and 10K test examples - sufficient size

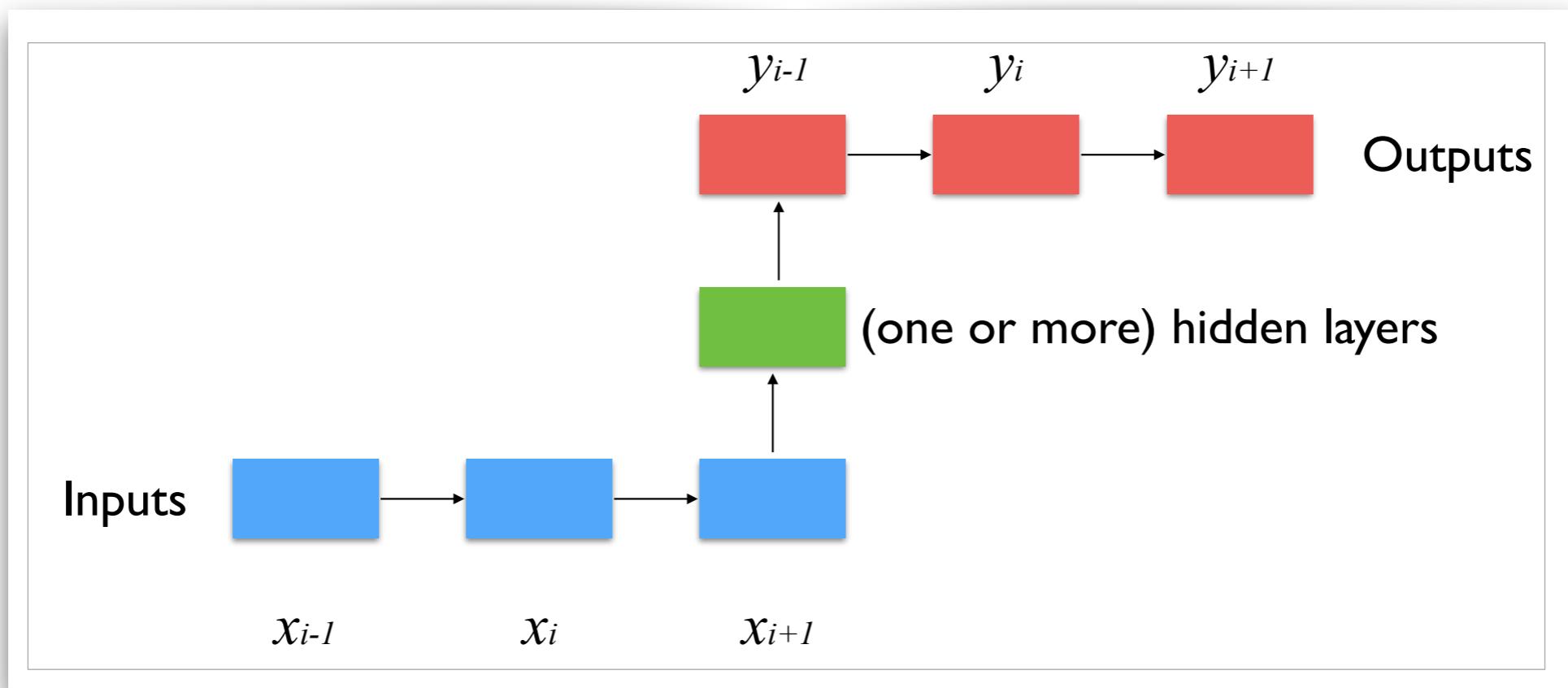


Reconstruction from original



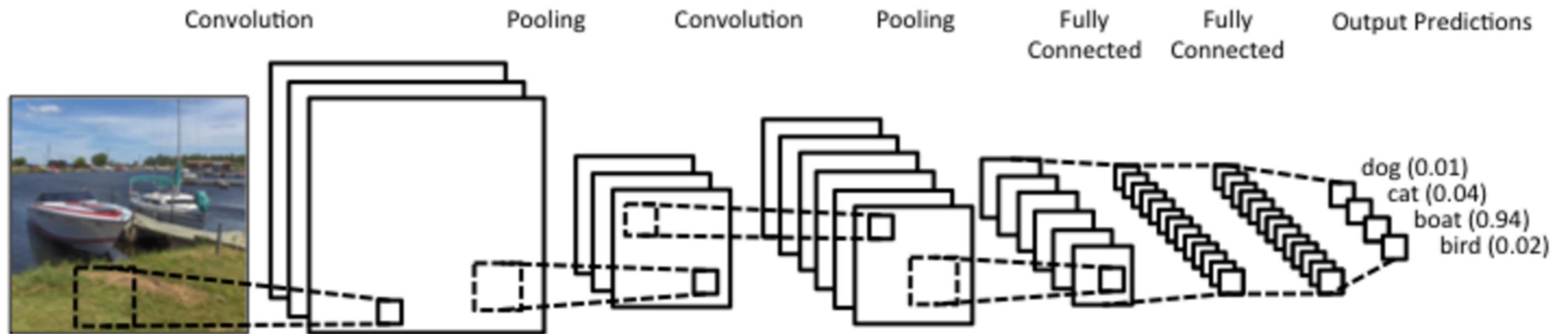
Reconstruction from learnt weights

Recurrent neural nets



A sequence-to-sequence recurrent neural network

Convolutional neural nets



<http://www.wildml.com/2015/11/understanding-convolutional-neural-networks-for-nlp/>

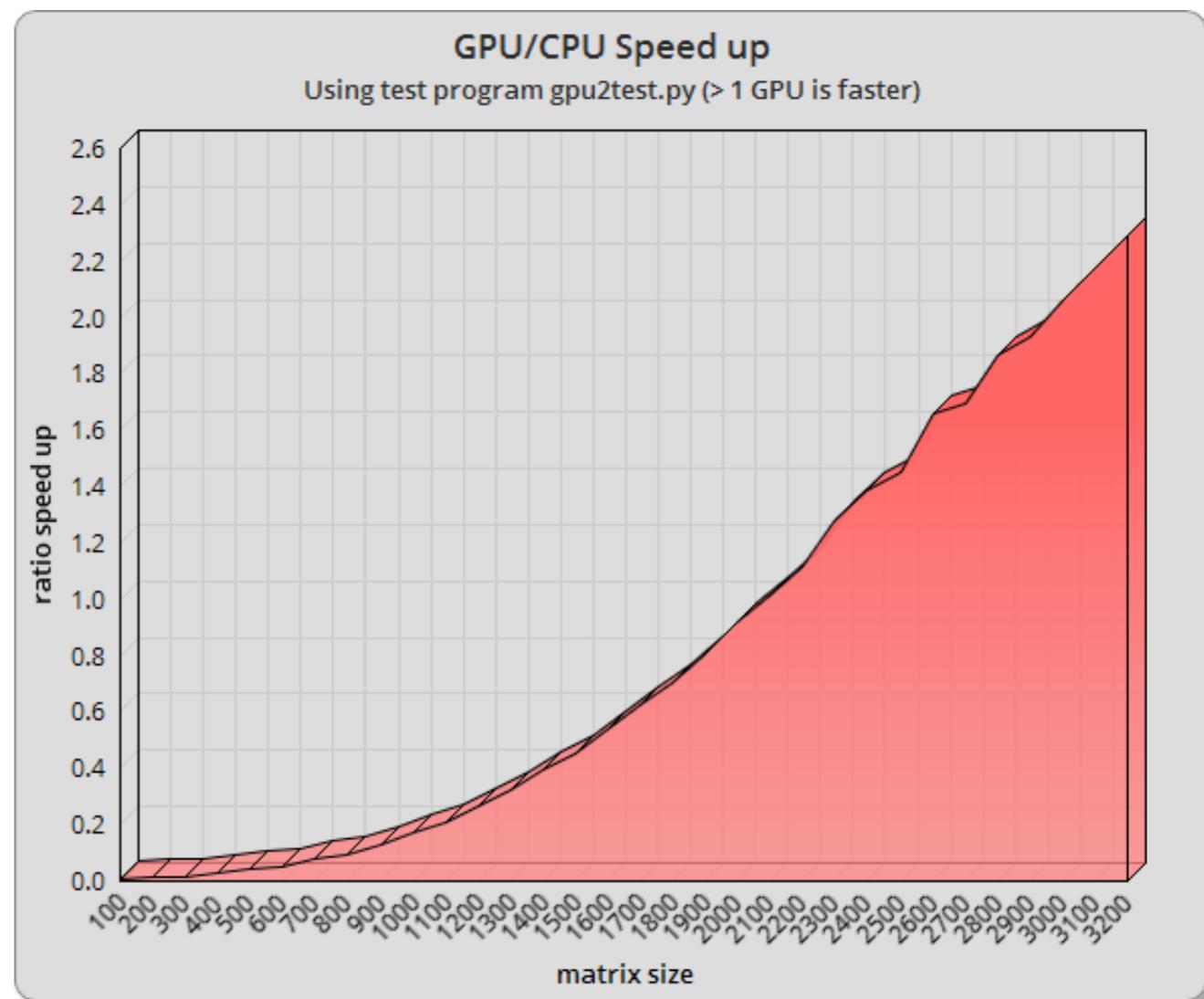
A convolutional neural network with convolutional, max pooling and fully-connect dense layer.

Libraries and tools

- Various libraries for deep learning available: best known are TensorFlow, Theano, Keras, Caffe, Torch.
- Very good Python libraries:
 - `numpy` = mathematical functions and matrix operations
 - `sklearn` = machine learning
 - `matplotlib` = plots and visualisation
 - `pandas` = data manipulation and analysis

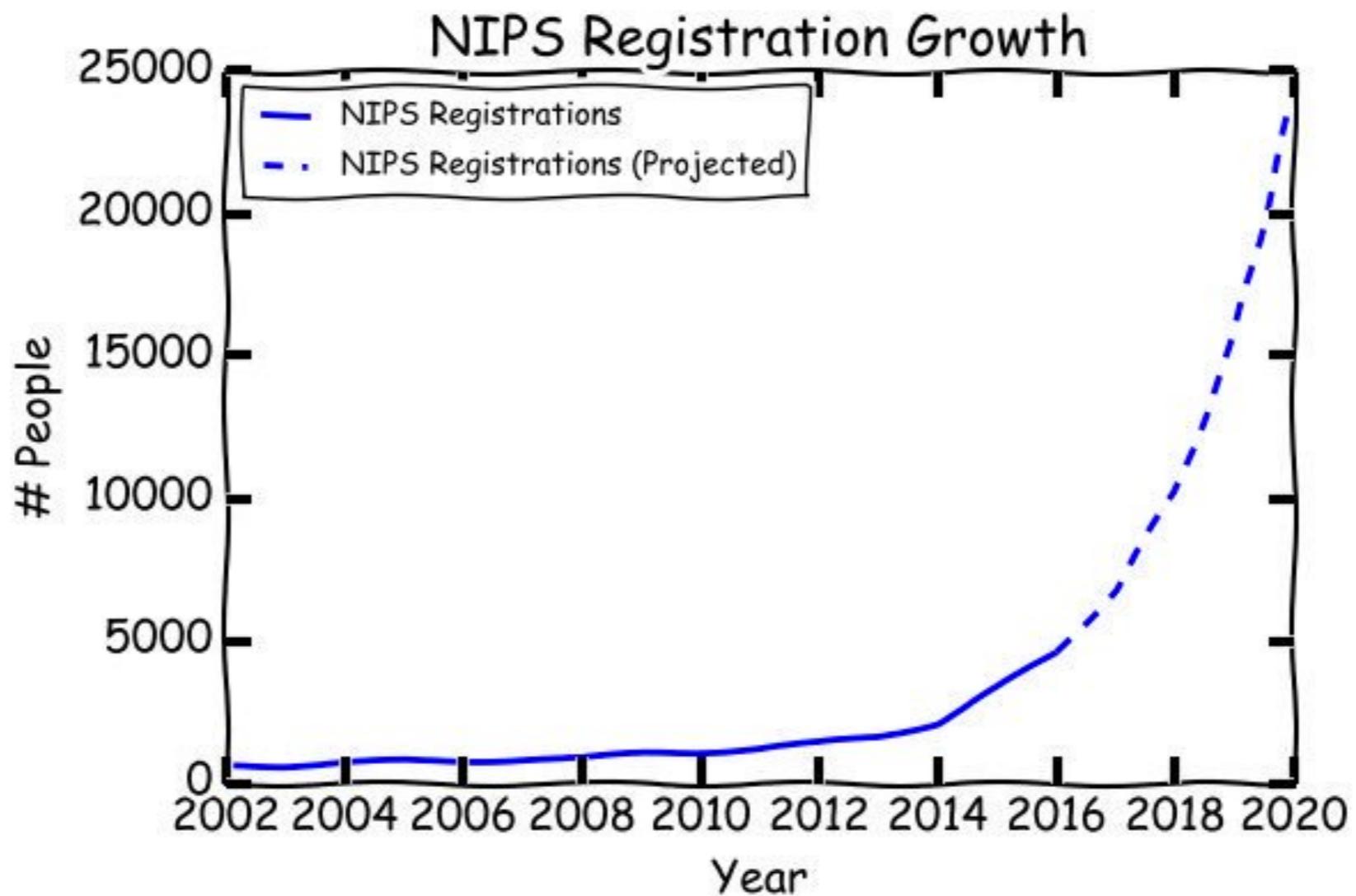
HPC and GPUs

- Deep learning involves matrix multiplication, so possible speed-ups with GPUs
- Nvidia, Intel, Google make specific hardware for deep learning
- Depends on matrix size, though - GPUs are not faster by default



from Darren Bird

Why now?



NIPS (Neural Information Processing Systems) is the leading conference on deep learning and neural networks.

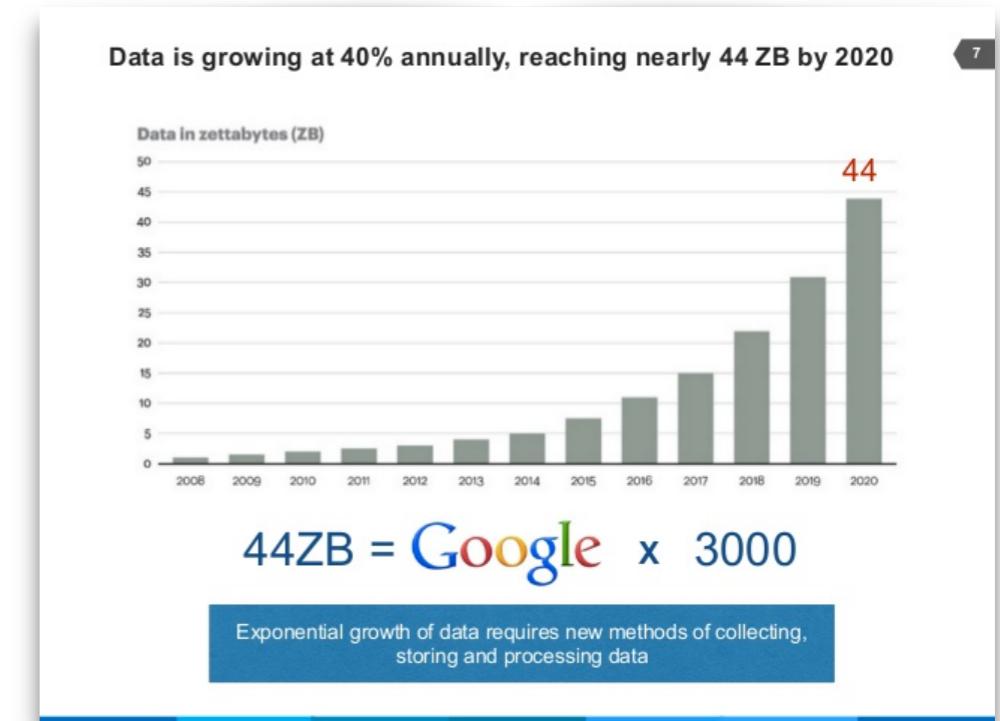
Why now?

- Deep learning is not new - it's been around since the 1960s but was unsuccessful before 2006.
- What has changed in terms of algorithms?
 - New methods for unsupervised pre-training
 - More efficient parameter estimation methods
 - Better understanding of model regularisation

Why now?

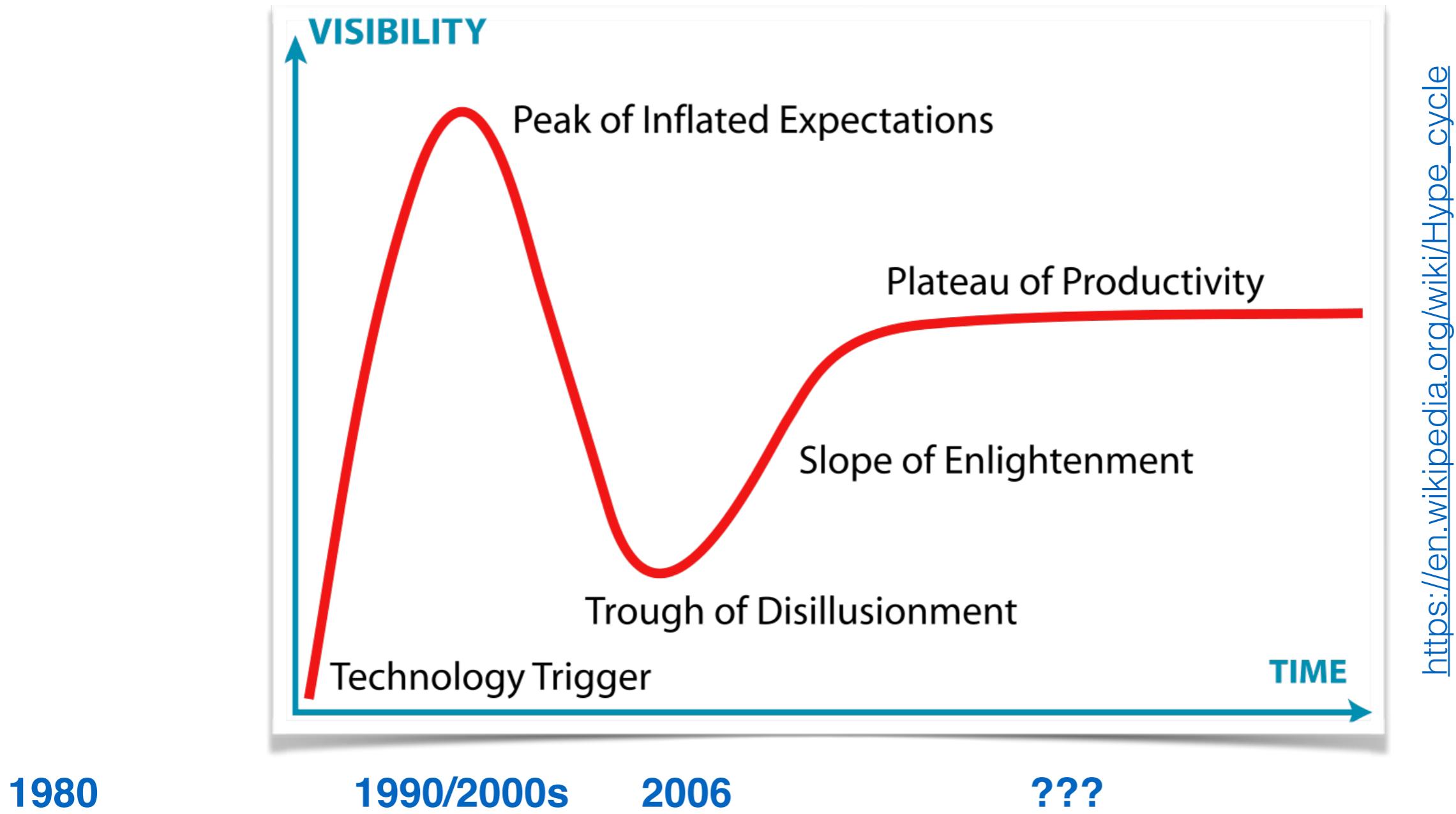
- Deep learning is not new - it's been around since the 1960s but was unsuccessful before 2006.
- What has changed in terms of hardware / resources?

- Better hardware and compute power
 - GPUs and large-scale CPU clusters
 - More data available and easier to store and process



<https://itsvit.com/blog/what-does-big-data-mean-for-your-business/>

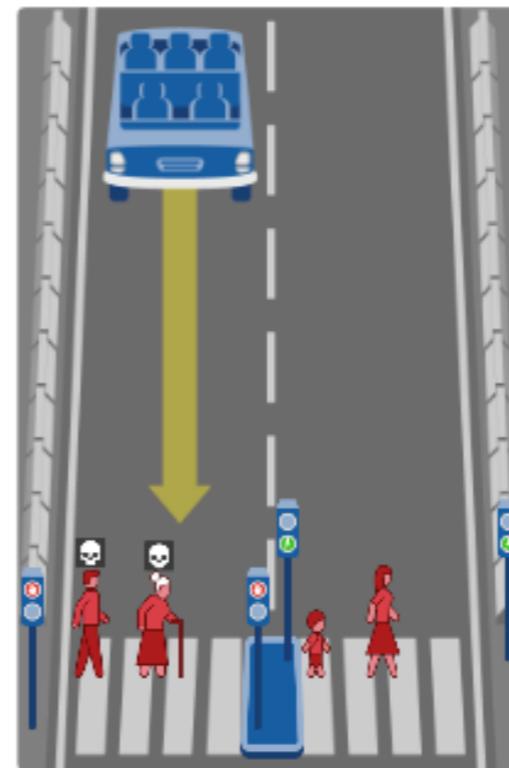
Why now?



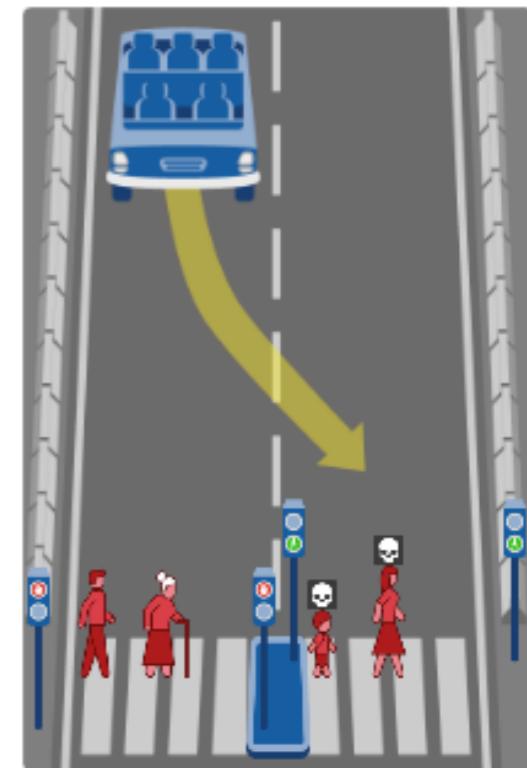
New problems

- Neural nets provide **no rationales** for their decisions
- This is **problematic** for safety and health-critical and ethical tasks

What should the self-driving car do?



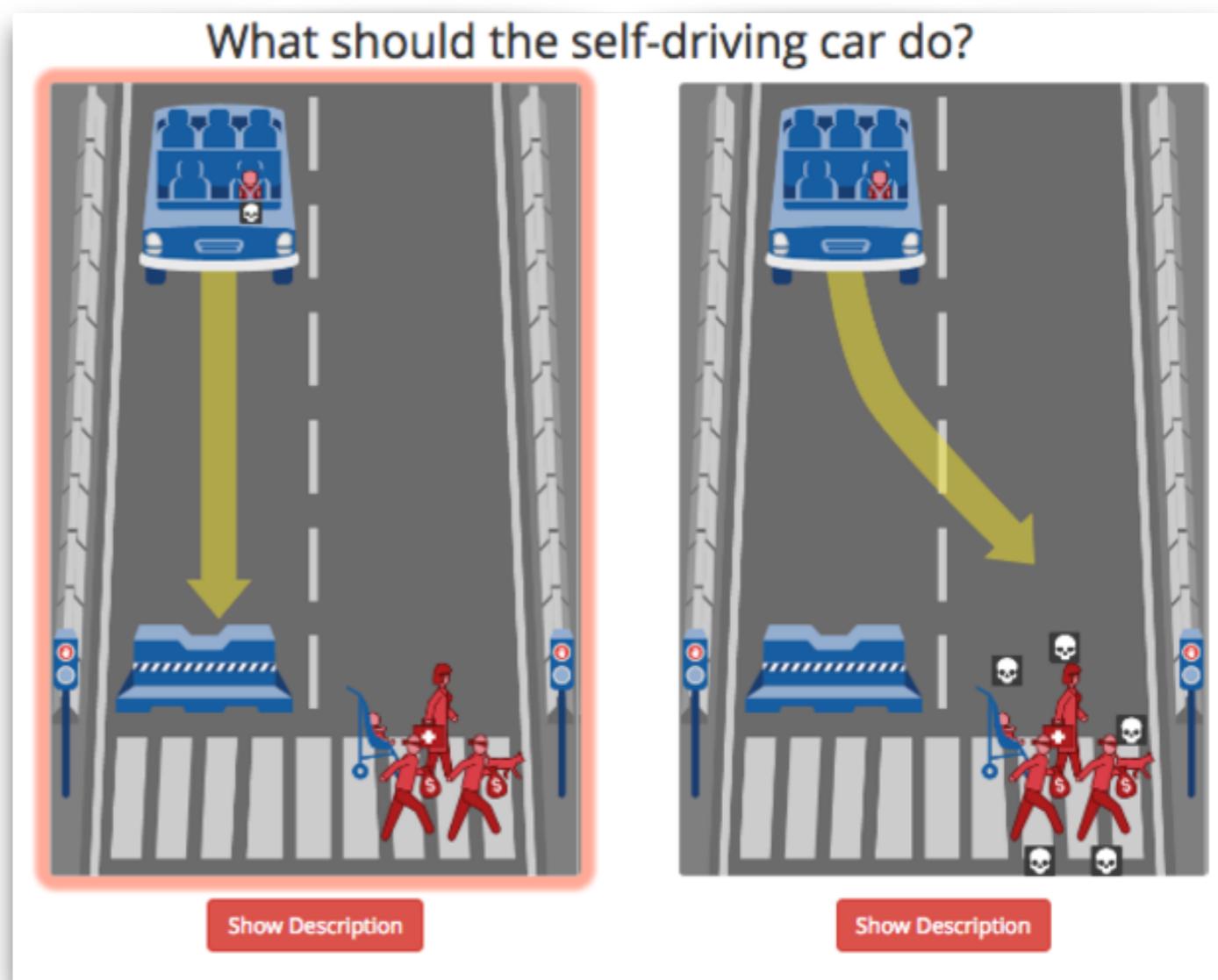
Show Description



Show Description

New problems

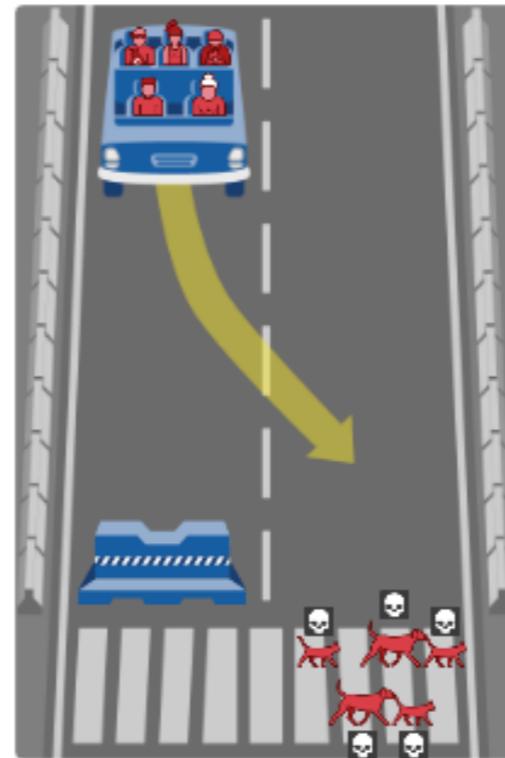
- Neural nets provide no rationales for their decisions
- This is problematic for safety and health-critical and ethical tasks



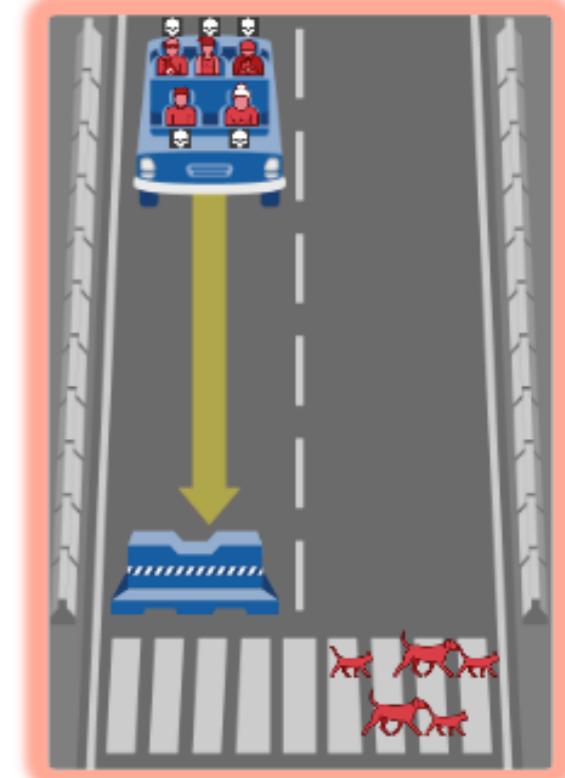
New problems

- Neural nets provide no rationales for their decisions
- This is problematic for safety and health-critical and ethical tasks

What should the self-driving car do?

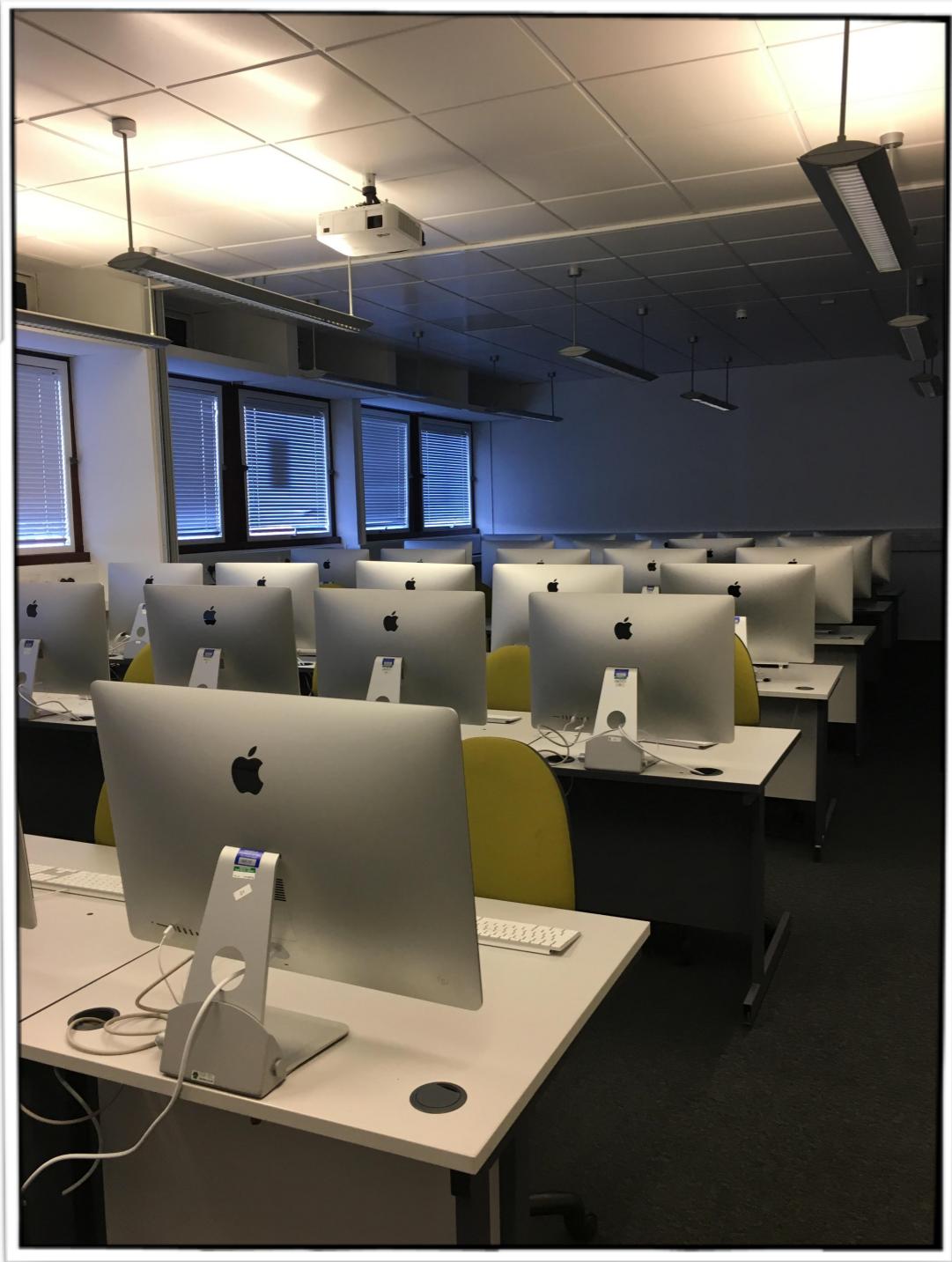


Show Description



Show Description

Tutorials - lab sessions



- 3 lab sessions planned:
- MNIST with Keras
 - MNIST with TensorFlow
 - Image Description generation
(Backpropagation)

Tutorials - lab sessions

3 lab sessions planned:

- MNIST with Keras
- MNIST with TensorFlow
- Image Description generation
(Backpropagation)



Invited seminar

Dr Yannis Konstas, Heriot-Watt University, Edinburgh

Building Adaptable and Scalable Natural Language Generation Systems

Traditionally, computers communicate with humans by converting computer-readable input to human-interpretable output, for example via graphical user interfaces. My research focuses on building programs that automatically generate textual output from computer-readable input. The majority of existing Natural Language Generation (NLG) systems use hard-wired rules or templates in order to capture the input for every different application, and rely on small manually annotated corpora.

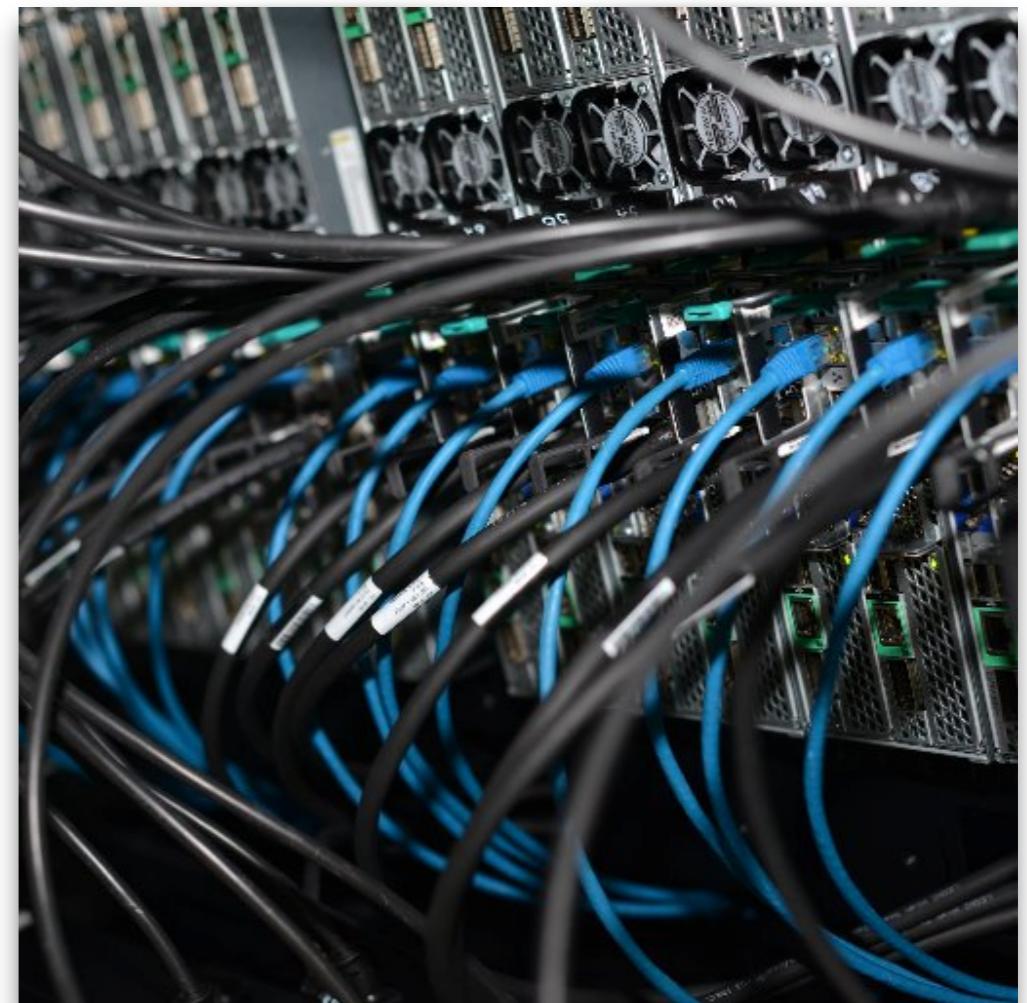
In this talk, I will present a framework for building NLG systems using Neural Network architectures. The approach makes no domain-specific modifications to the input and benefits from training on very large unannotated corpora. It achieves state-of-the-art performance on a number of tasks, including generating text from meaning representations and source code. Such a system can have direct applications to intelligent conversation agents, source code assistant tools and semantic-based Machine Translation.



Viper HPC

Viper tours on Day 2 for visitors during lunch break.

UoH members can arrange for a separate visit.



Keras (MLP)

Hidden nodes - effect on performance and speed?

Activation functions - effect on performance?

Dropout rate - difference between high and low rate?

Layers - effect on performance and speed?

Learning rate - effect of size on performance / speed?

Keras (CNN)

Type of pooling - effect on performance?

Kernel size - effect on speed and performance?

Flipped images - effect on performance?

TensorFlow (MLP)

Change of optimiser - any difference?

Change of cost function - any difference?

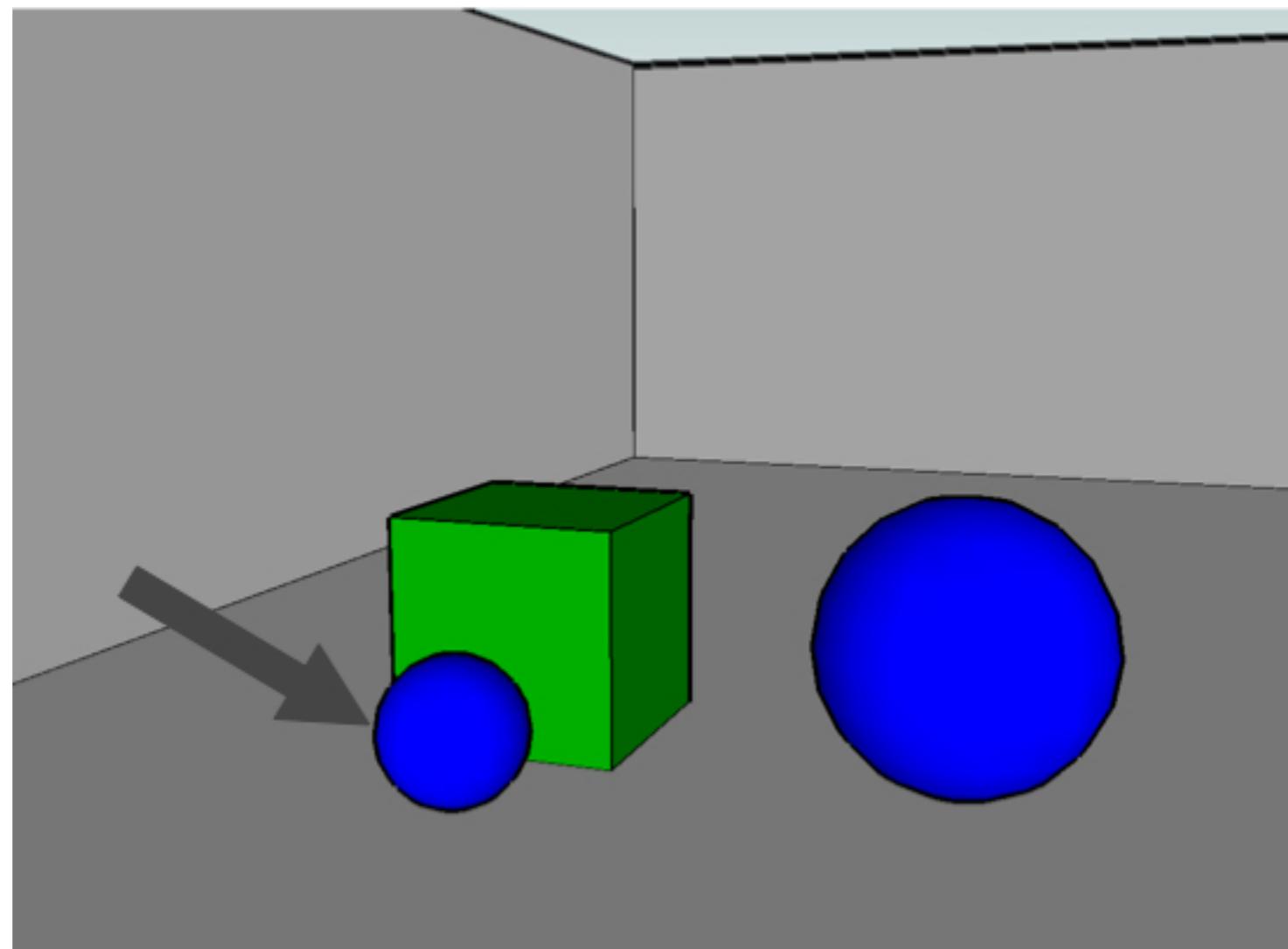
CNN or RNN in TensorFlow?

Projects

Most of tomorrow...

- 5 people per group (can be less but not more)
- One deep learning task
 - need to identify / collect data
 - read it into Keras / TensorFlow
 - Configure parameters
 - Short 5 minute presentation (with slides)
- Prizes (voted by participants)

Image description generation



Could be one project...

Group-chosen project

?

<https://archive.ics.uci.edu/ml/datasets.html>

Groups

Group1 : Ryan, Albert, Rebecca, Lucy

Group 2: Brian, George, Luke, Annika, Charlene

Group 3: Gabi, Alexander Jesse, Alex

Group 4: Lydia, Adam, Aidan, Dan

References and pointers are at the end of the tutorials or linked from slides.

Thanks to the Ferens Education Trust for sponsoring this event.