DEEP NEURAL NETWORKS

Trends and Applications

Ekapol Chuangsuwanich

July 14, 2017

DNNs (Deep Neural Networks)

- Why deep learning?
- Greatly improved performance in ASR and other tasks (Computer Vision, Robotics, Machine Translation, NLP, etc.)
- Surpassed human performance in many tasks

Task	Previous state-of-the-art	Deep learning (2012)	Deep learning (2017)
TIMIT	24.4%	20.0%	17.0%
Switchboard	23.6%	16.1%	5.5%
Google voice search	16.0%	12.3%	4.9%

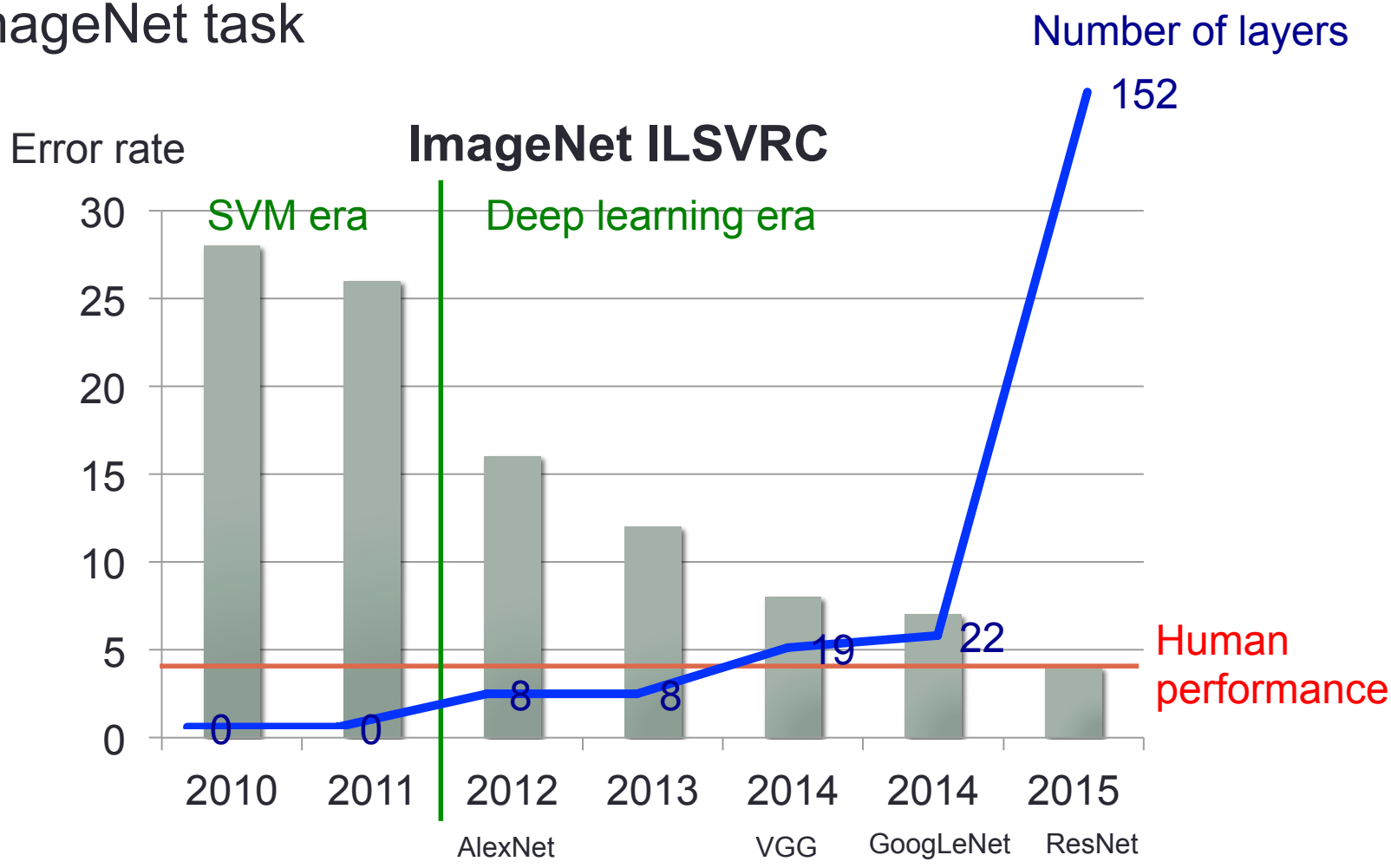
Geoffrey Hinton, Deep Neural Networks for Acoustic Modeling in Speech Recognition, 2012

Why now

- Neural Networks has been around since 1990s
- Big data – DNN can take advantage of large amounts of data better than other models
- GPU – Enable training bigger models possible
- Deep – Easier to avoid local minima when the model is large

Wider and deeper networks

- ImageNet task



Olga Russakovsky, ImageNet Large Scale Visual Recognition Challenge https://arxiv.org/abs/1409.0575

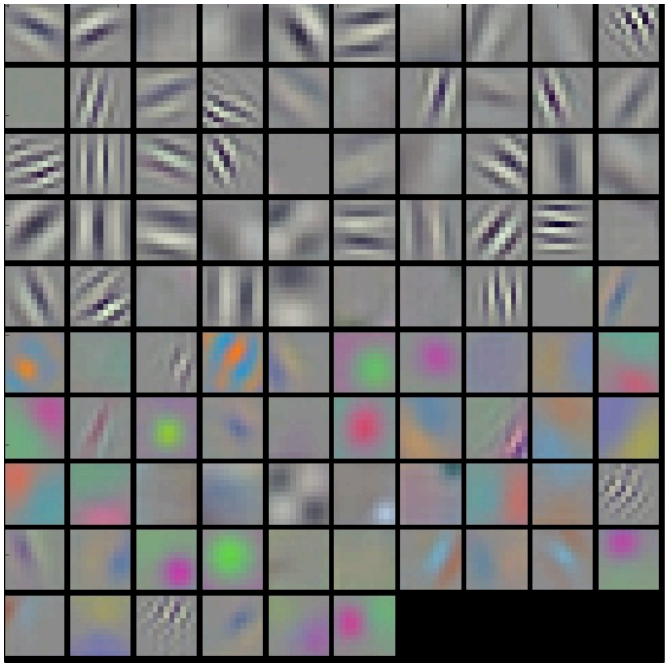
Why is deep learning good

- Traditional machine learning approaches need feature engineering (MFCC, SIFT, HoG, etc.)
  - Features are based on human understanding of the phenomena
  - Have some simplifying assumptions
- Models also has simplifying assumptions

- Human knowledge captures known knowns, and Known unknowns NOT unknown unknowns
- Deep learning has enough parameters and model freedom to automatically learn the unknown unknowns (and everything else)
  - Downside: Needs LOTS of data

Why is deep learning good

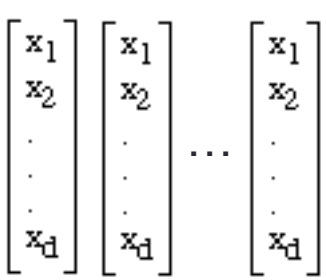
- DNN can handle higher dimensional input feature
- Deep learning also combines features engineering and modeling into one single optimization problem
  - Guarantees the matching between the feature and the model



Features learned from a CNN for a vision task.
Inputs are now raw pixels instead of descriptors.

http://cs231n.github.io/understanding-cnn/

MFCC (Mel Frequency Cepstral Coefficient)

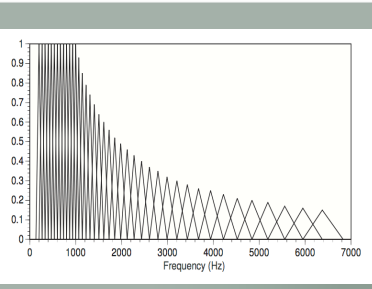


Features for traditional ASR

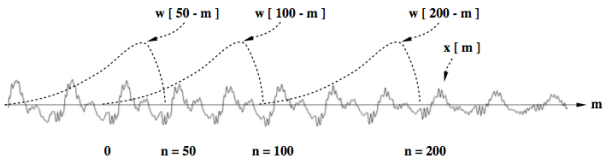
Select lower coefficients

DCT

log

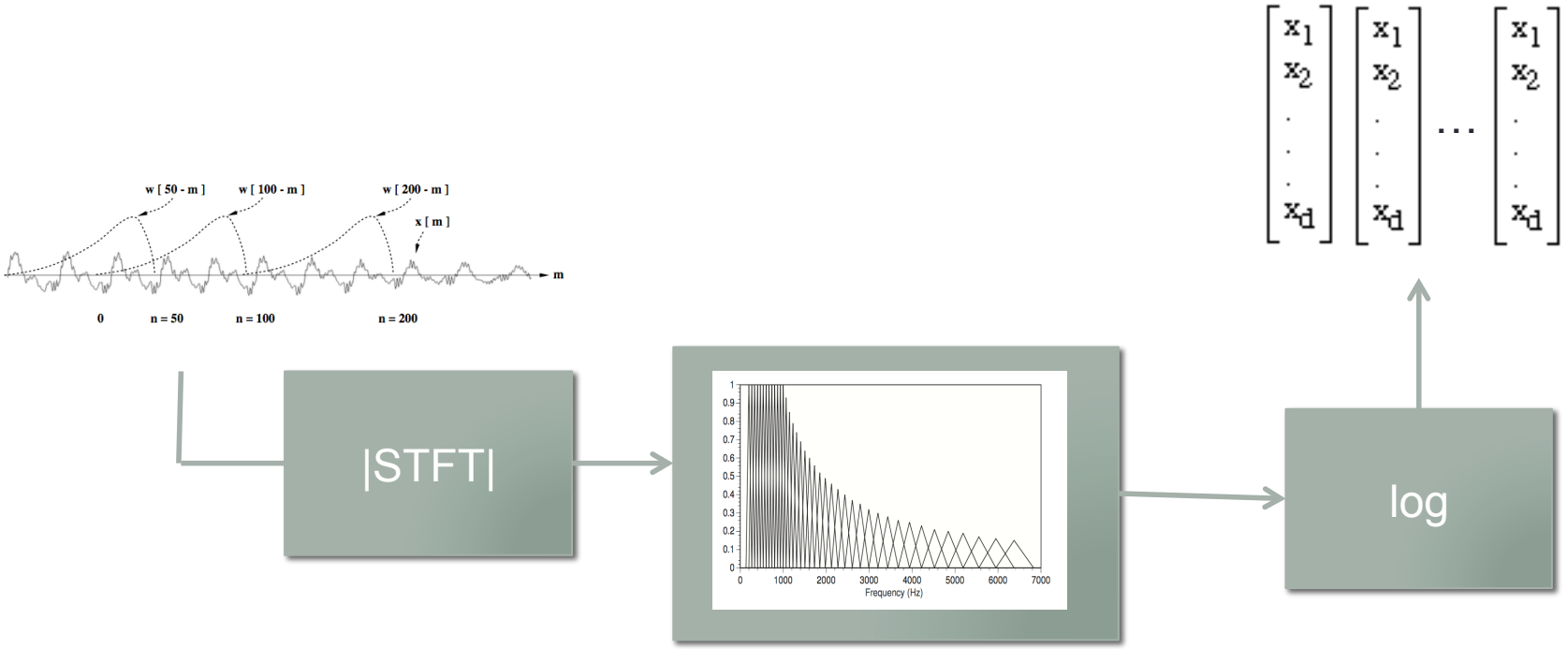


|STFT|

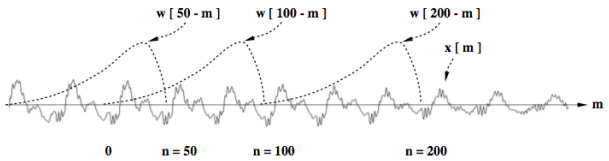


MFCC (Mel Frequency Cepstral Coefficient)

Features for ASR with DNN



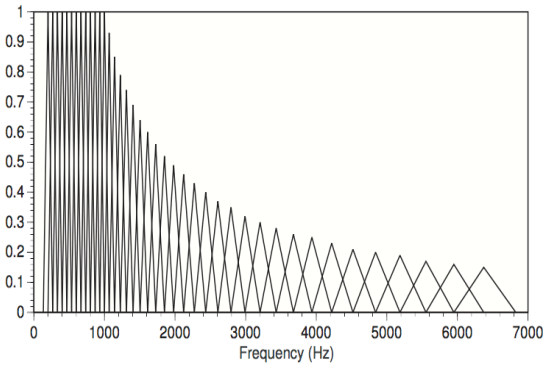
MFCC (Mel Frequency Cepstral Coefficient)



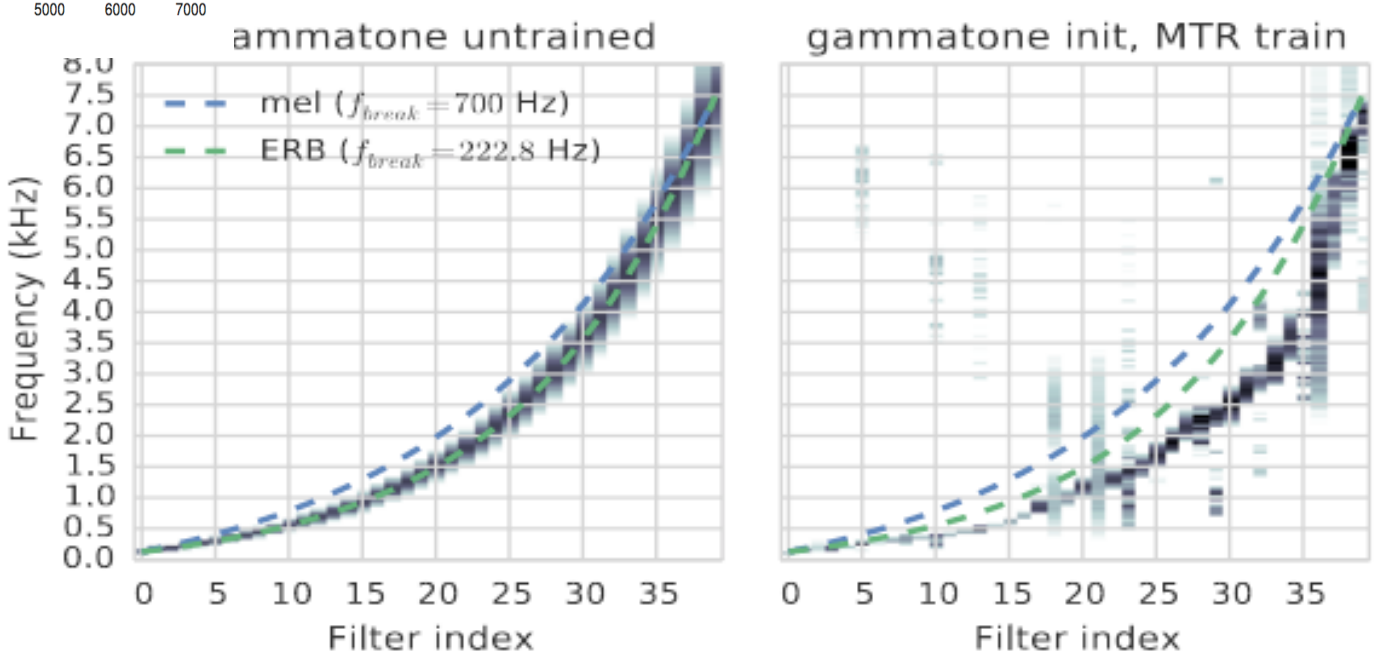
Features for ASR with DNN

$$\begin{bmatrix} x_1 \\ x_2 \\ \cdot \\ \cdot \\ \cdot \\ x_d \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \cdot \\ \cdot \\ \cdot \\ x_d \end{bmatrix} \cdots \begin{bmatrix} x_1 \\ x_2 \\ \cdot \\ \cdot \\ \cdot \\ x_d \end{bmatrix}$$

Learning the Mel filters



Filter type	WER
DNN+Random	16.4
DNN+Gammatone (untrained)	16.4
DNN+Gammatone (trained)	16.2



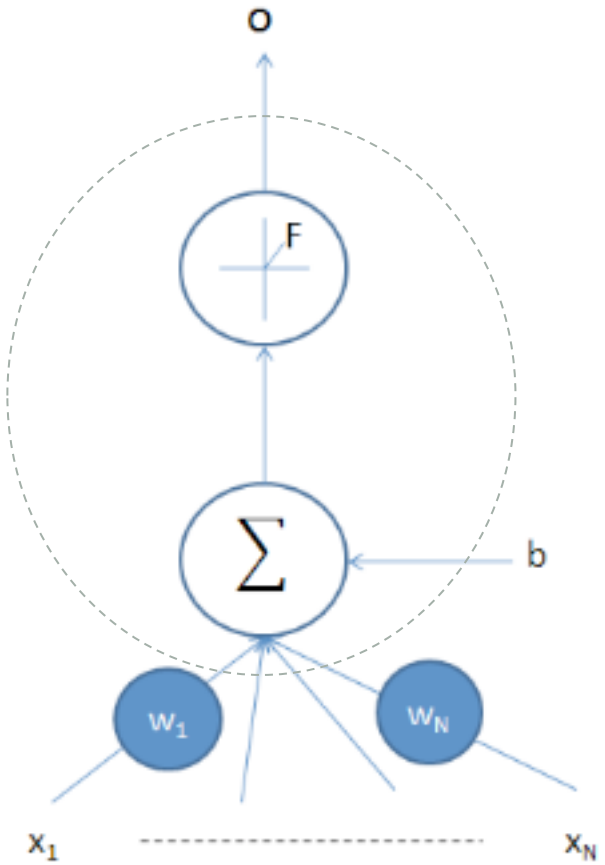
Tara Sainath, Learning the Speech Front-end with Raw Waveform CLDNNs, 2015

DNNs architectures

- Fully connected networks
  - Components in neural networks
  - How is neural networks trained
- Convolutional neural networks
- Recurrent neural networks
- Gated recurrent units
- Long short-term memory networks
- Encoder-Decoder

Fully connected networks

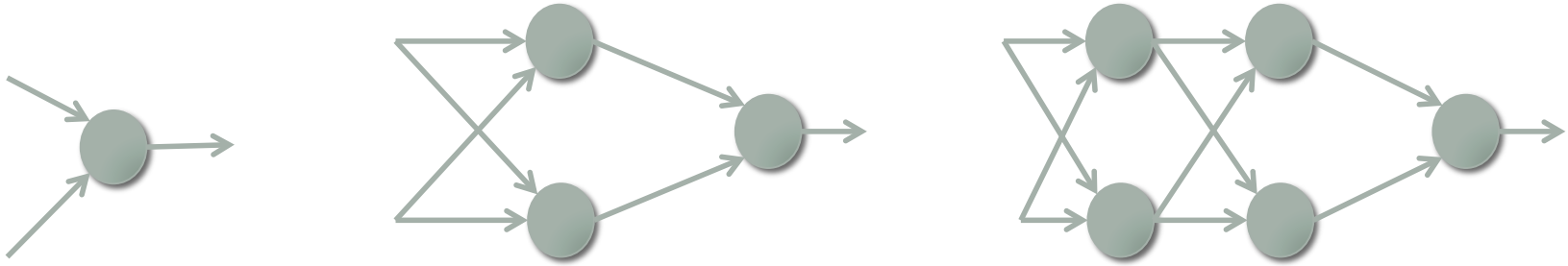
- Many names: feed forward networks or deep neural networks or Multilayer perceptron
- Composed of multiple neurons

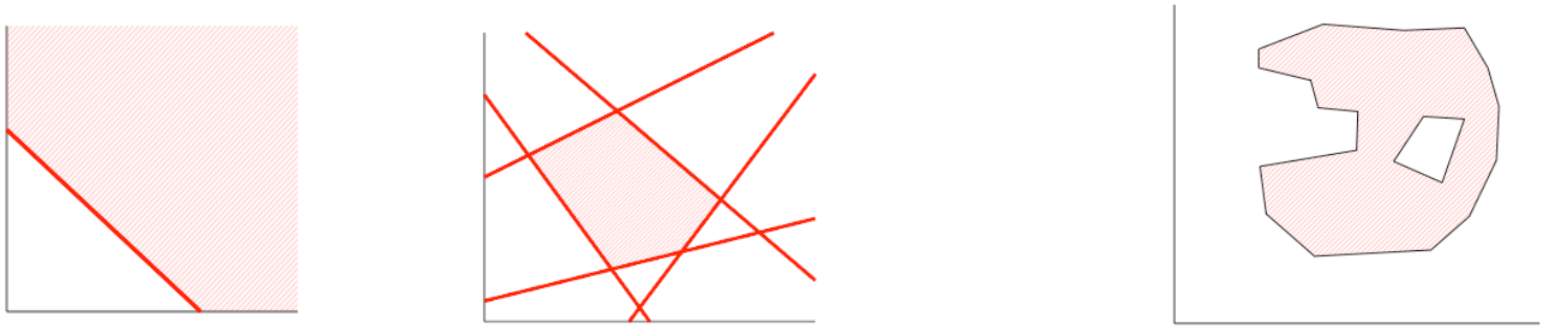


- $o = F(\mathbf{w}^T\mathbf{x} + b)$
- o - activation output
- F - nonlinear function
- **w** - weight vector
- b - bias

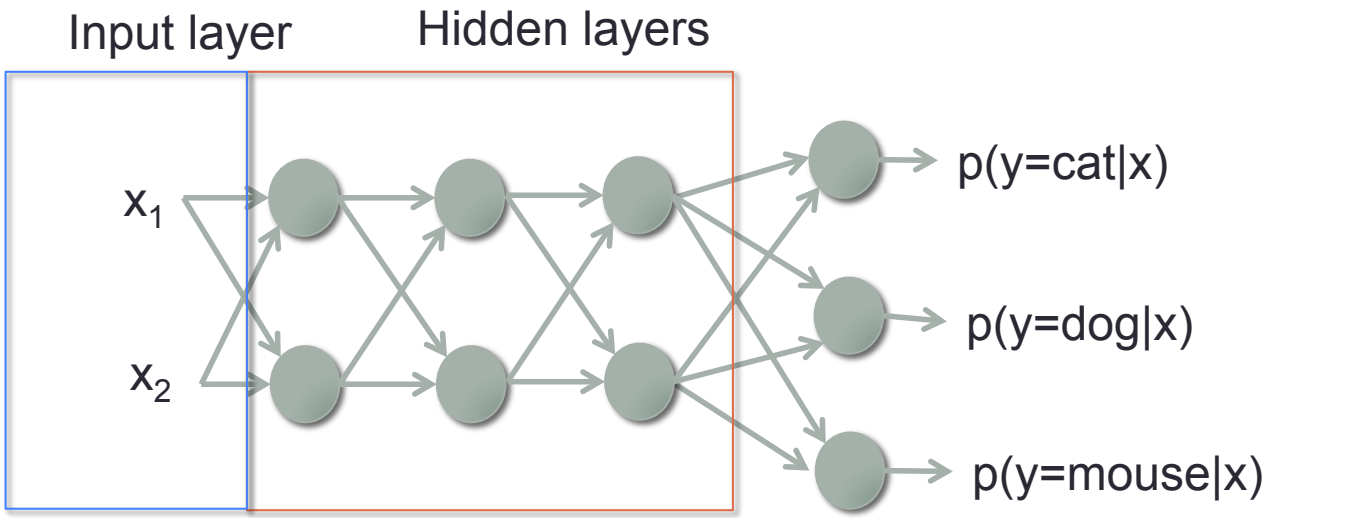
Combining neurons

- Each neuron splits the feature space with a hyperplane
- Stacking neuron creates more complicated decision boundaries
- More powerful but prone to overfitting





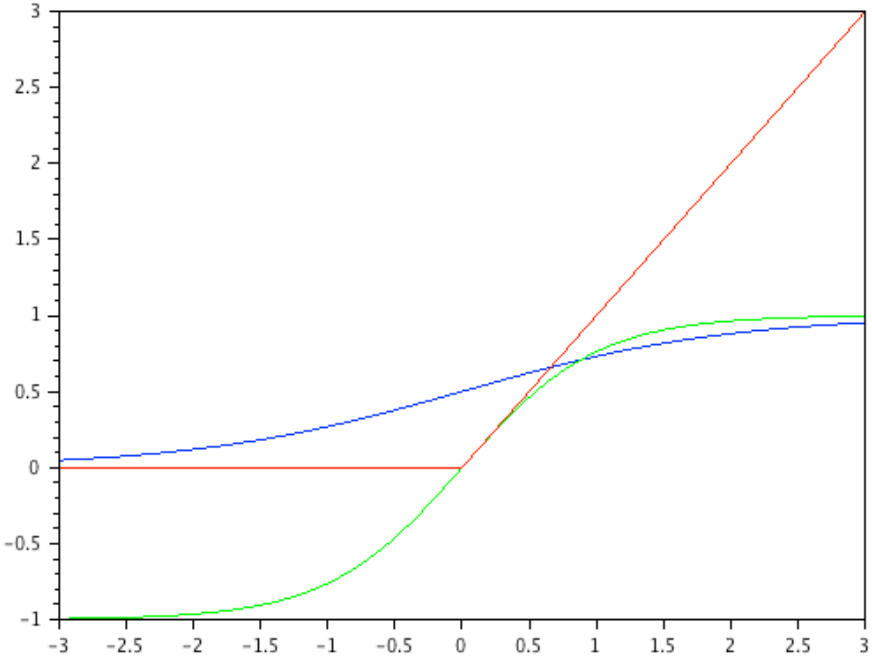
Terminology



Input should be scaled to have zero mean unit variance

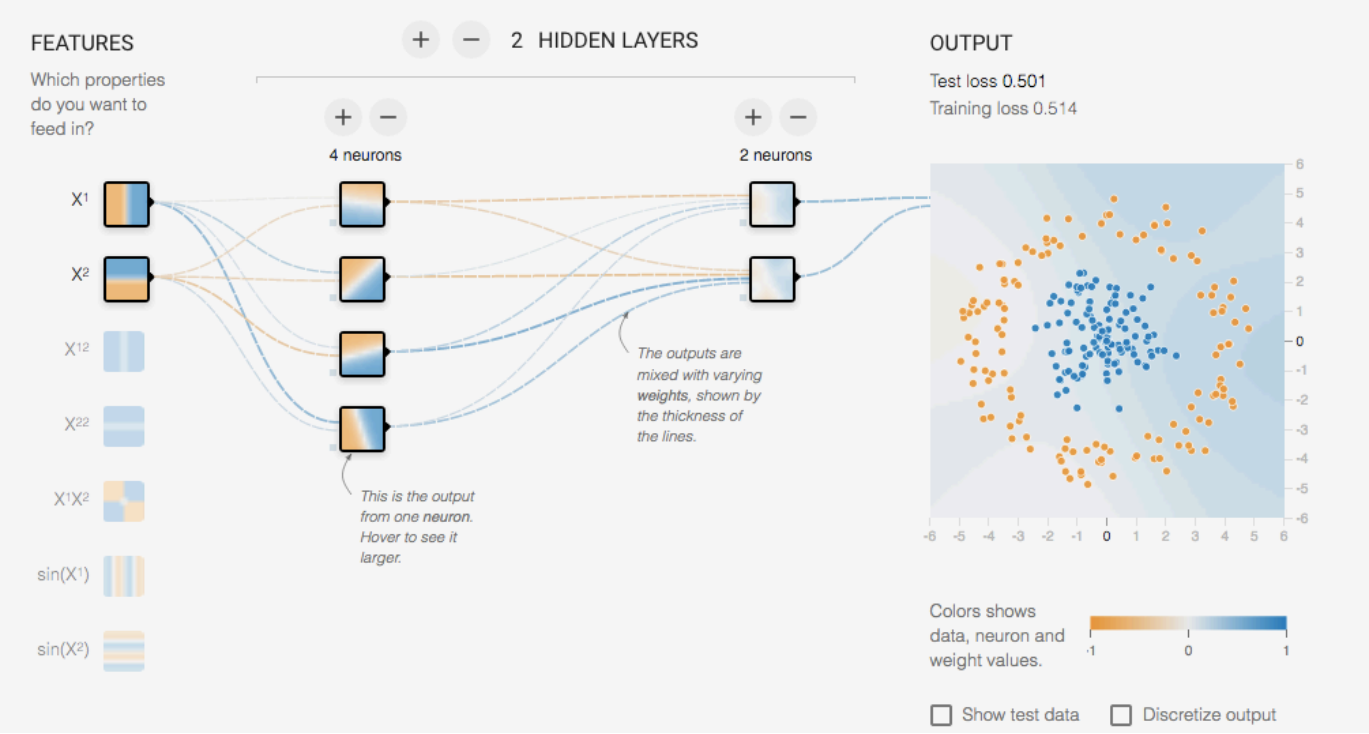
Non-linearity

- The F Non-linearity is important in order to stack neurons
  - If F is linear, a multi layered network can be collapsed as a single layer (by just multiplying weights together)
- Sigmoid
- tanh
- Rectified Linear Unit (ReLU)
- Most popular is ReLU and its variants (Fast to train, and more stable)

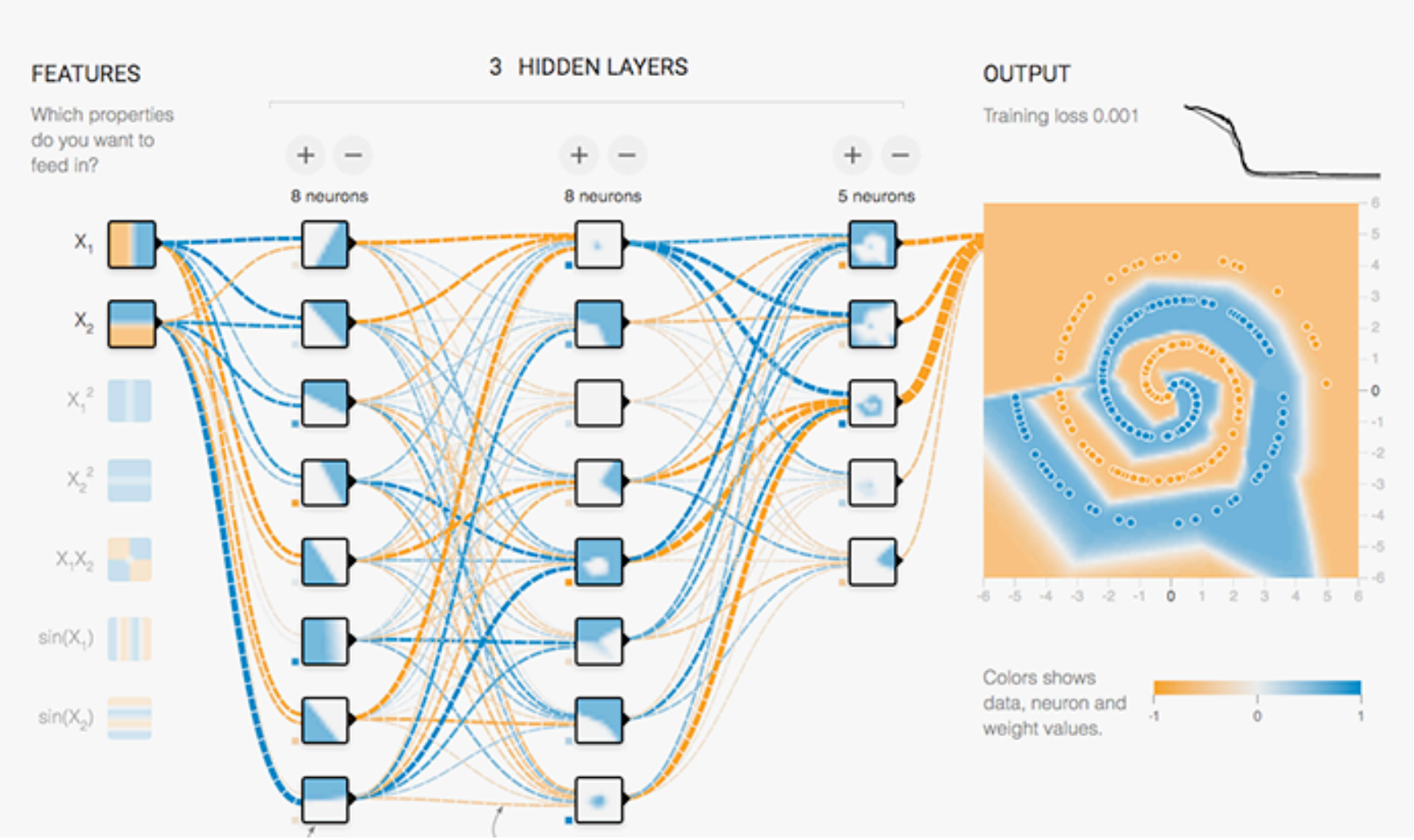


Playground

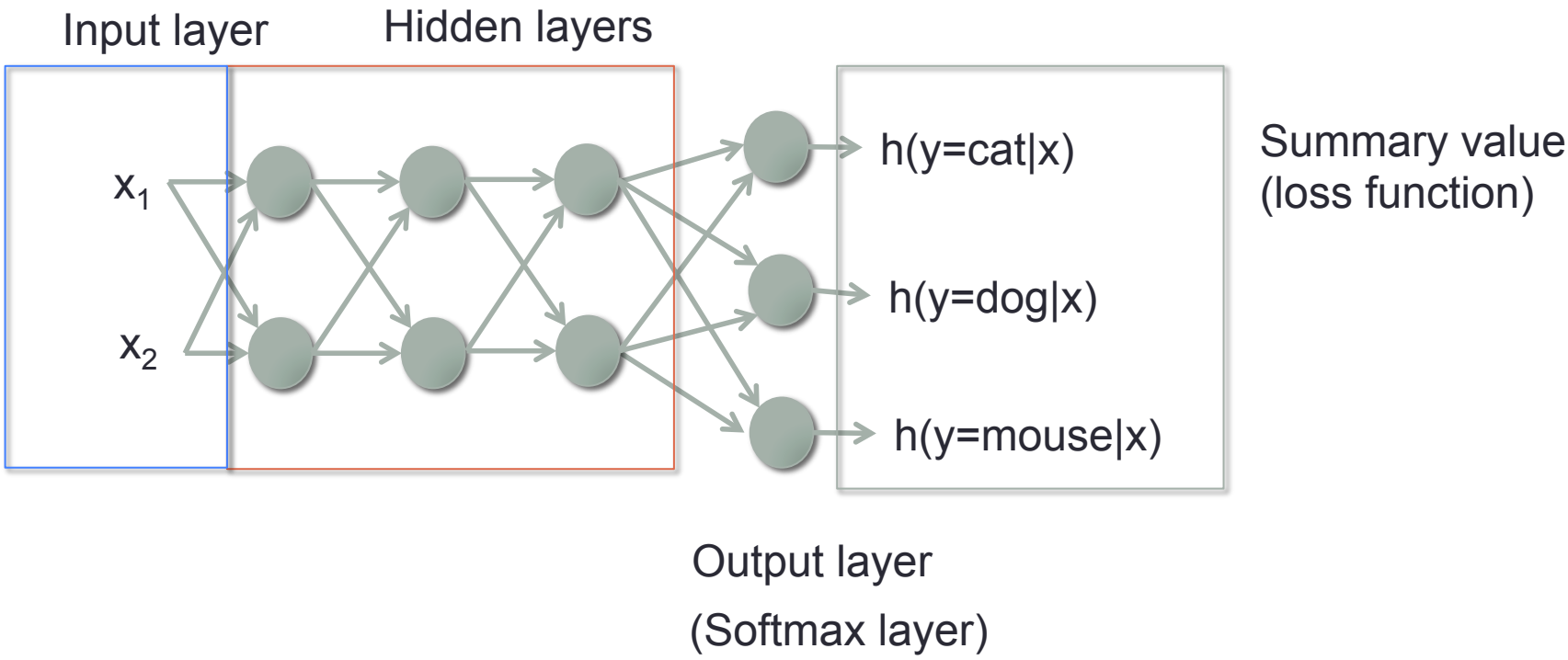
- http://playground.tensorflow.org/



Playground


FEATURES
Which properties do you want to feed in?
3 HIDDEN LAYERS
8 neurons
8 neurons
5 neurons
OUTPUT
Training loss 0.001
Colors shows data, neuron and weight values.

Terminology



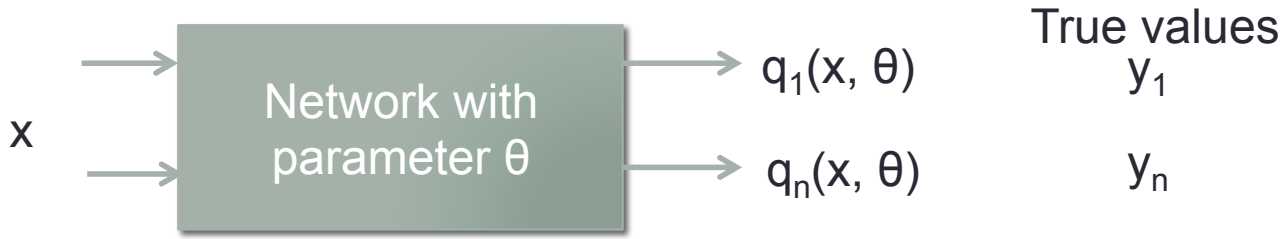
Objective function (Loss function)

- Can be any function that summarizes the performance into a single number
- Cross entropy
  - Used for softmax outputs (probabilities)

$$L = -\Sigma_n y_n log q_n(x, \theta)$$

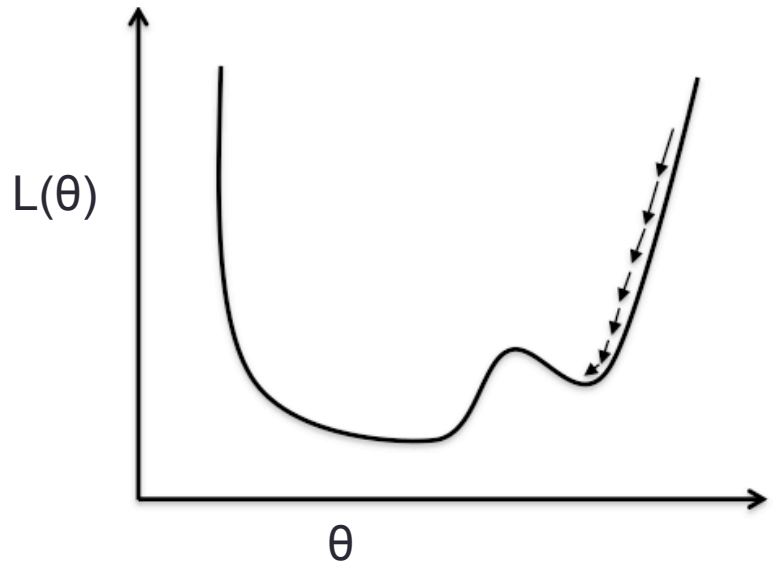
- Sum of square errors
  - Used for any real valued outputs

$$L = -\frac{1}{2}\Sigma_n (y_n - q_n(x, \theta))^2$$



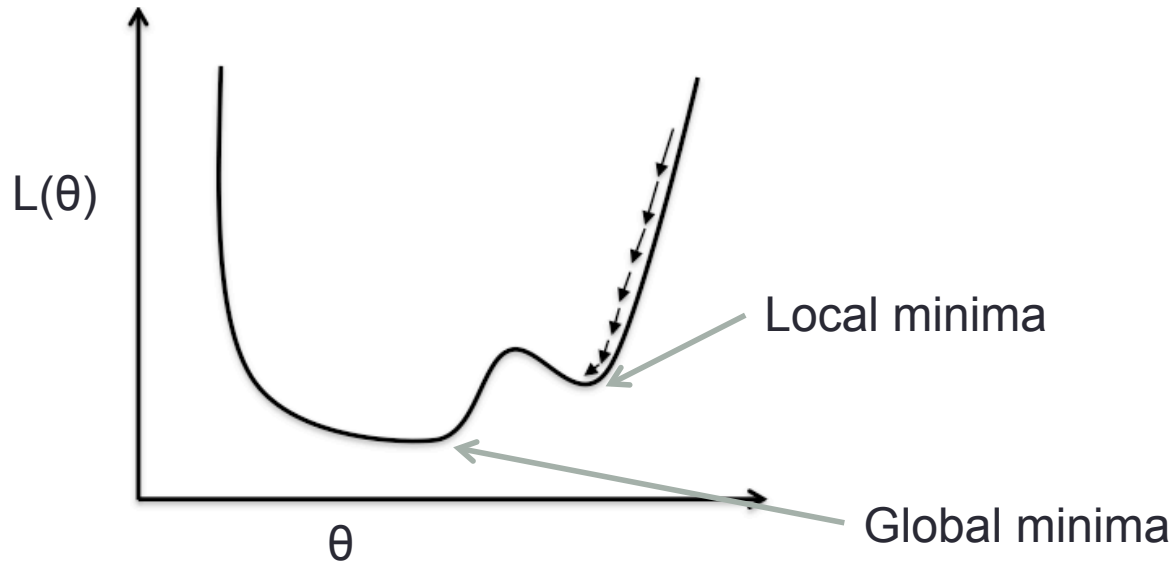
Minimization using gradient descent

- We want to minimize L with respect to θ
  - Differentiate with respect to θ
  - Gradients passes through the network by Back Propagation



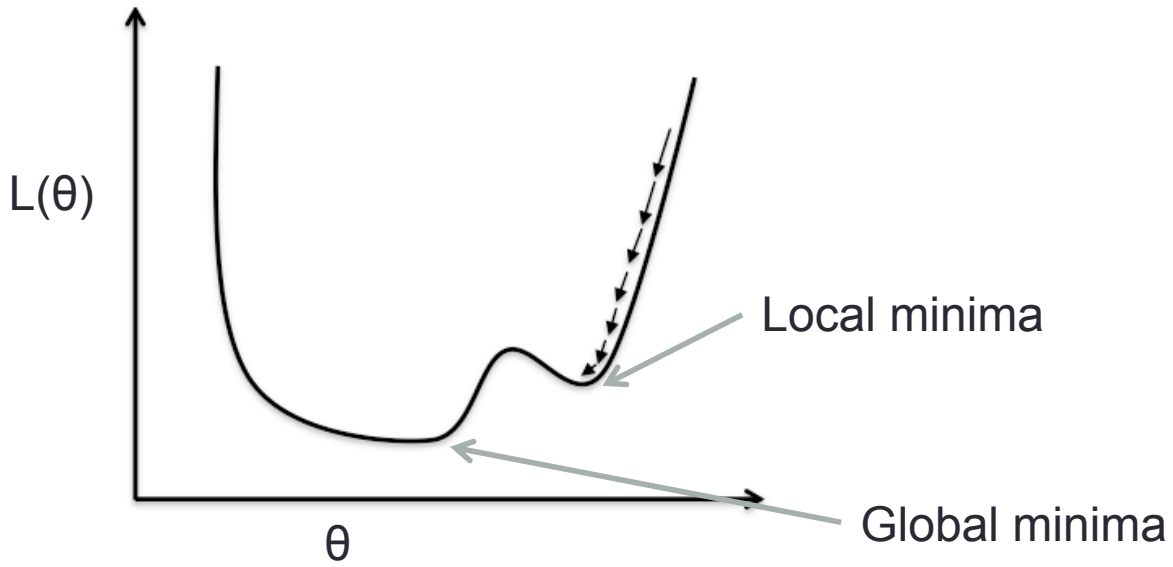
Deep vs Shallow

- The loss function of neural network is non-convex (and non-concave)
  - Local minimas can be avoided with convexity
  - Convexity gives easier training



Deep vs Shallow

- If deep, most local minimas are the global minima!
  - Always a way to lower the loss in the network with millions of paramters
  - Enough parameters to remember every training examples
  - Does not imply anything about generalization



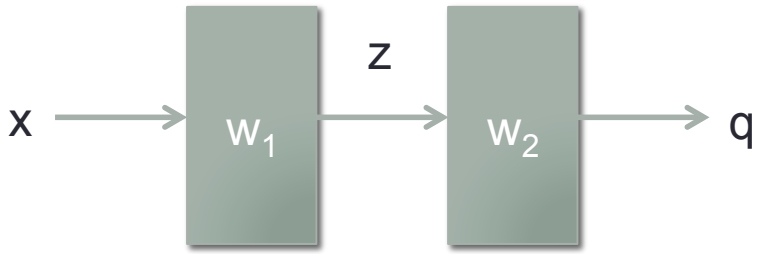
Kenji Kawaguchi,Deep Learning without Poor Local Minima, https://arxiv.org/abs/1605.07110

Stochastic gradient descent (SGD)

- Main method to train a neural network
- Consider you have one million training examples
  - Gradient descent computes the objective function of all samples, then decide direction of descent
    - Takes too long
  - SGD computes the objective function on subsets of samples
    - The subset should not be biased and properly randomized to ensure no correlation between samples
- The subset is called a mini-batch
- Size of the mini-batch determines the training speed and accuracy
  - Usually somewhere between 32-1024 samples per mini-batch

Back propagation

- To update the network, we need to compute the direction (gradient) that minimizes the loss function
  1. Compute the loss function of a mini-batch (forward pass)
  2. Compute the gradient (partial derivative) with respect to each parameter (backward pass)
- Key: Use chain rule



True value: y

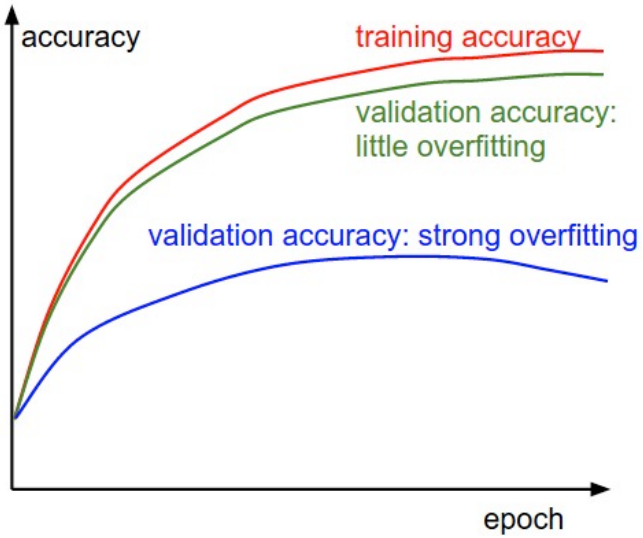
$z = w_1 x \qquad q = w_2 z$
$L = (y-q)^2$

$$\frac{dL}{dw_2} = \frac{dL}{dq}\frac{dq}{dw_2}$$
$$= [2(y-q)(-1)][z]$$

$$\frac{dL}{dw_1} = \frac{dL}{dq}\frac{dq}{dz}\frac{dz}{dw_1}$$
$$= [2(y-q)(-1)][w_2][x]$$

Overfitting

- You can keep doing back propagation forever!
- The training loss will always go down
- But it overfits
- Need to monitor performance on a held out set
- Stop or decrease learning rate when overfit happens



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

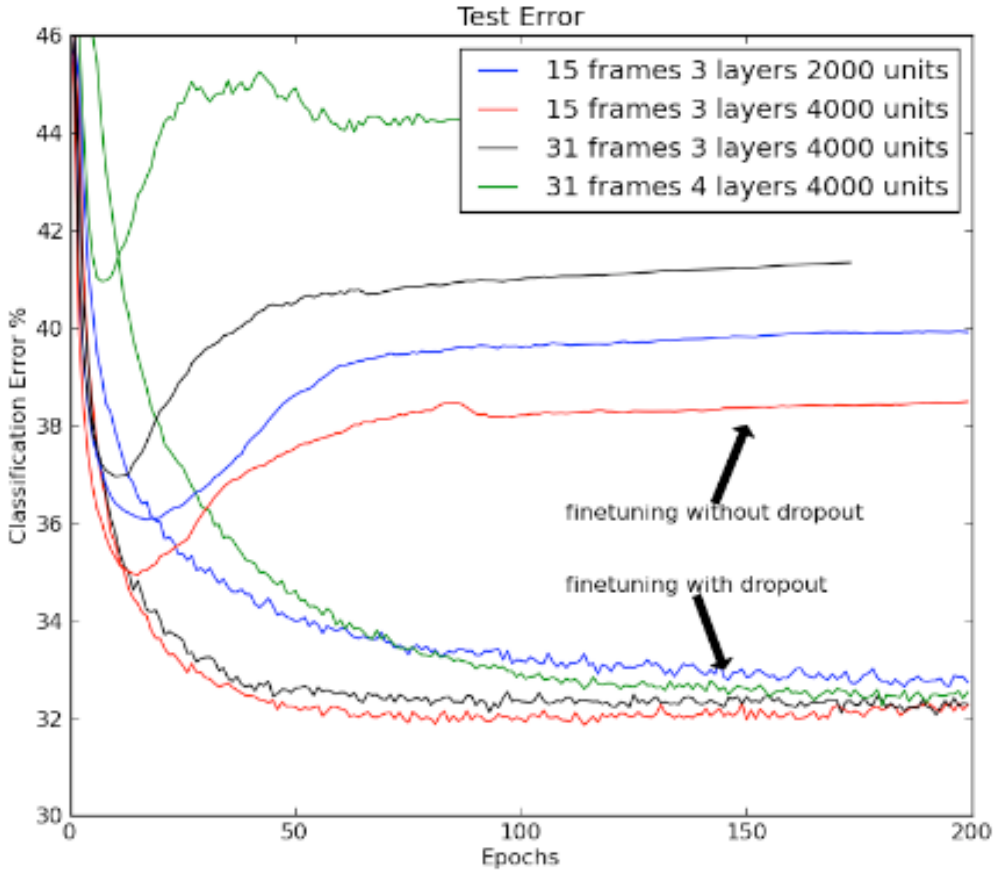
Reducing overfitting - dropout

- A *RECENT* (2012) regularization technique for reducing overfitting
- Randomly turn off different subset of neurons during training
  - Network no longer depend on any particular neuron
  - Force the model to have redundancy – robust to any corruption in input data
  - A form of performing model averaging (assemble of experts)
- Now a standard technique

Geoffrey Hinton, Improving neural networks by preventing co-adaptation of feature detectors, https://arxiv.org/abs/1207.0580

Dropout on TIMIT

- A phoneme recognition task



Geoffrey Hinton, Improving neural networks by preventing co-adaptation of feature detectors, https://arxiv.org/abs/1207.0580

Vanishing/Exploding gradient

- Backprop introduces many multiplications down chain
- The gradient value gets smaller and smaller
  - The deeper the network the smaller the gradient in the lower layers
  - Lower layers changes too slowly (or not at all)
  - Hard to train very deep networks (>6 layers)
- The opposite can also be true. The gradient explodes from repeated multiplication
  - Put a maximum value for the gradient (Gradient clipping)

Residual networks

- Adds a direct connection between layers (bypass non linearity
- Gradient now has two components (direct connection and traditional direction)

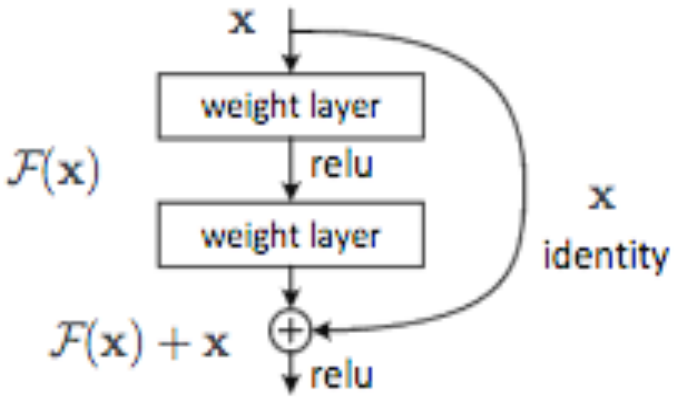


Image recognition task

	plain	ResNet
18 layers	27.94	27.88
34 layers	28.54	**25.03**

Kaiming He, Deep Residual Learning for Image Recognition

Residual networks

ASR task

System	#layers	with overlap	no overlap
LSTMP	3	50.7	41.7
LSTMP	8	52.6	43.8
HLSTMP	3	50.4	41.2
HLSTMP	8	50.7	41.3

Yu Zhang, Highway Long Short-Term Memory RNNs for Distant Speech Recognition

DNNs architectures

- ~~Fully connected networks~~
- Convolutional neural networks
- Recurrent neural networks
- Gated recurrent units
- Long short-term memory networks
- Encoder-decoder

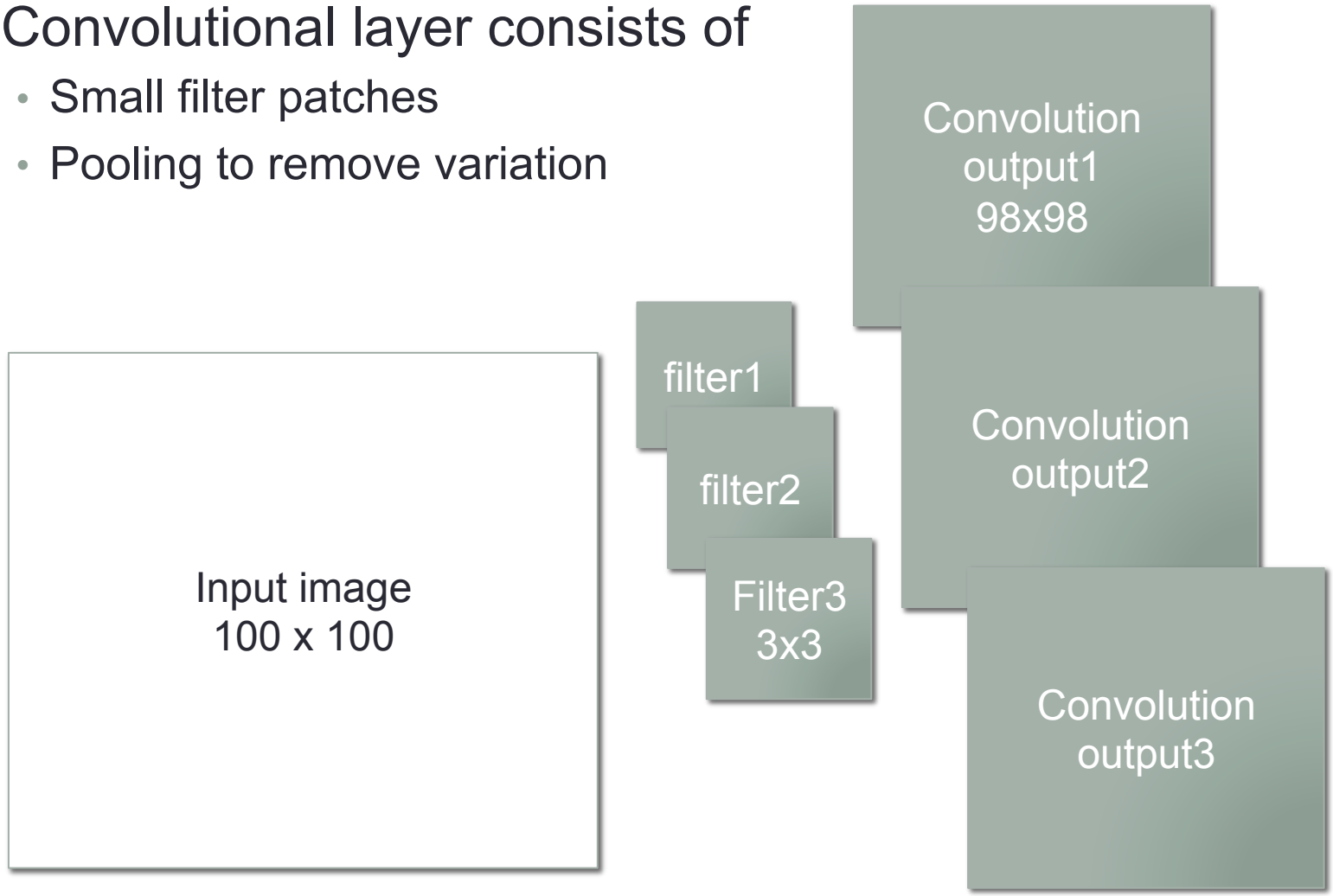
Convolutional Neural Networks (CNNs)

- Consider an image of a cat. DNNs need different neurons to learn every possible location a cat can be

- Can we use the same parameters to learn that a cat exists regardless of location
  - Shift-invariance
- 2 parts: convolutional layer and pooling layer

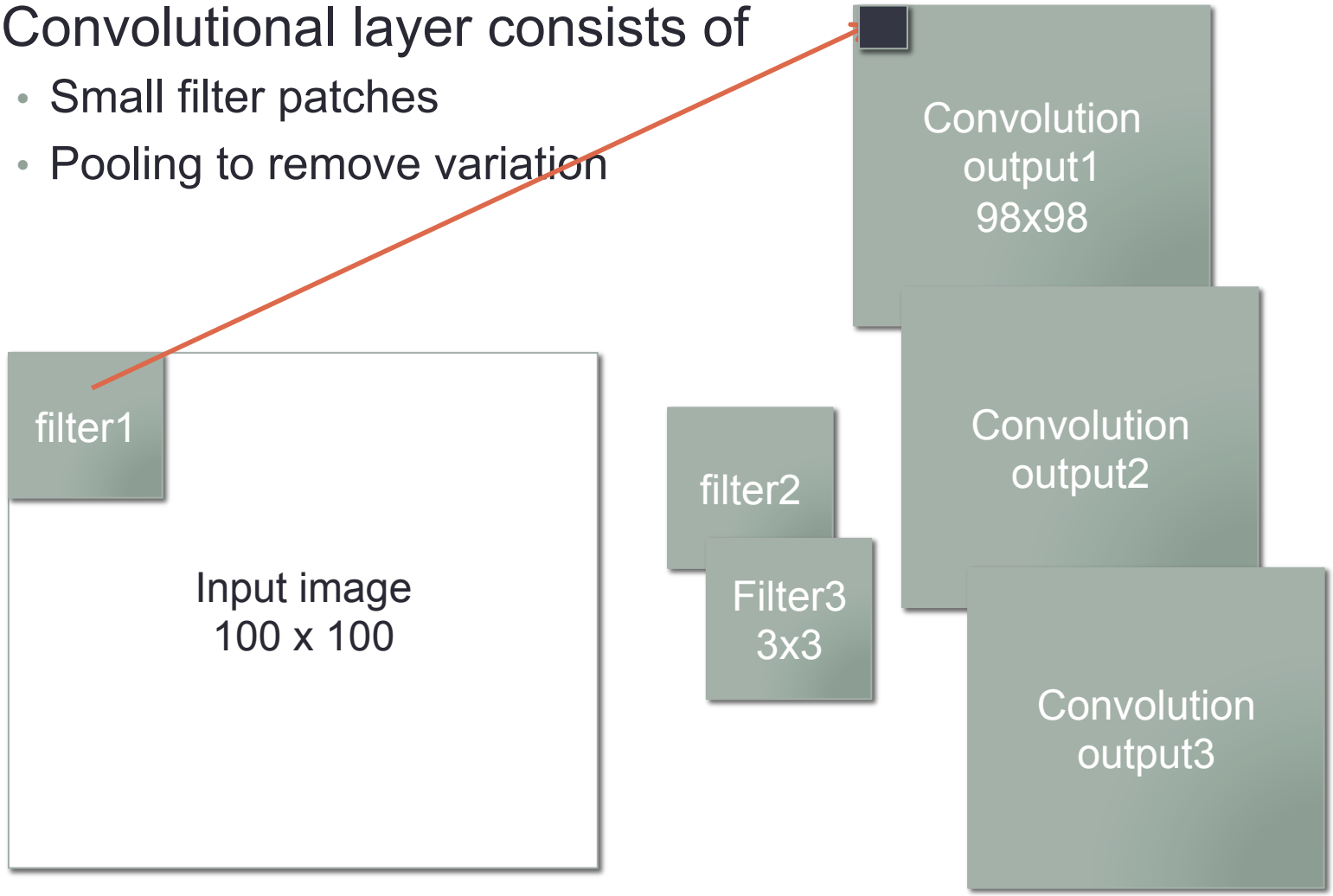
Convolutional filters

- Convolutional layer consists of
  - Small filter patches
  - Pooling to remove variation



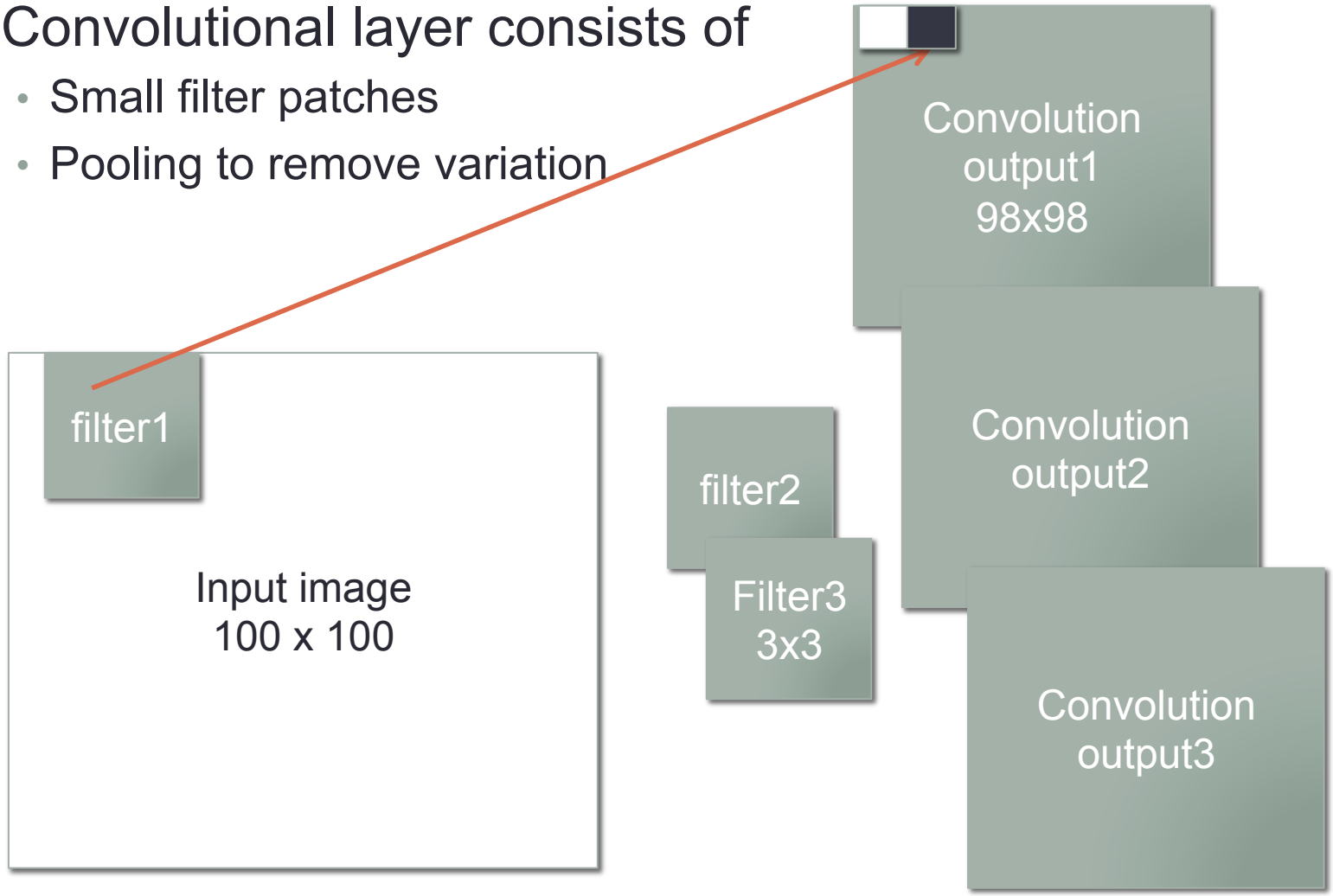
Convolutional filters

- Convolutional layer consists of
  - Small filter patches
  - Pooling to remove variation



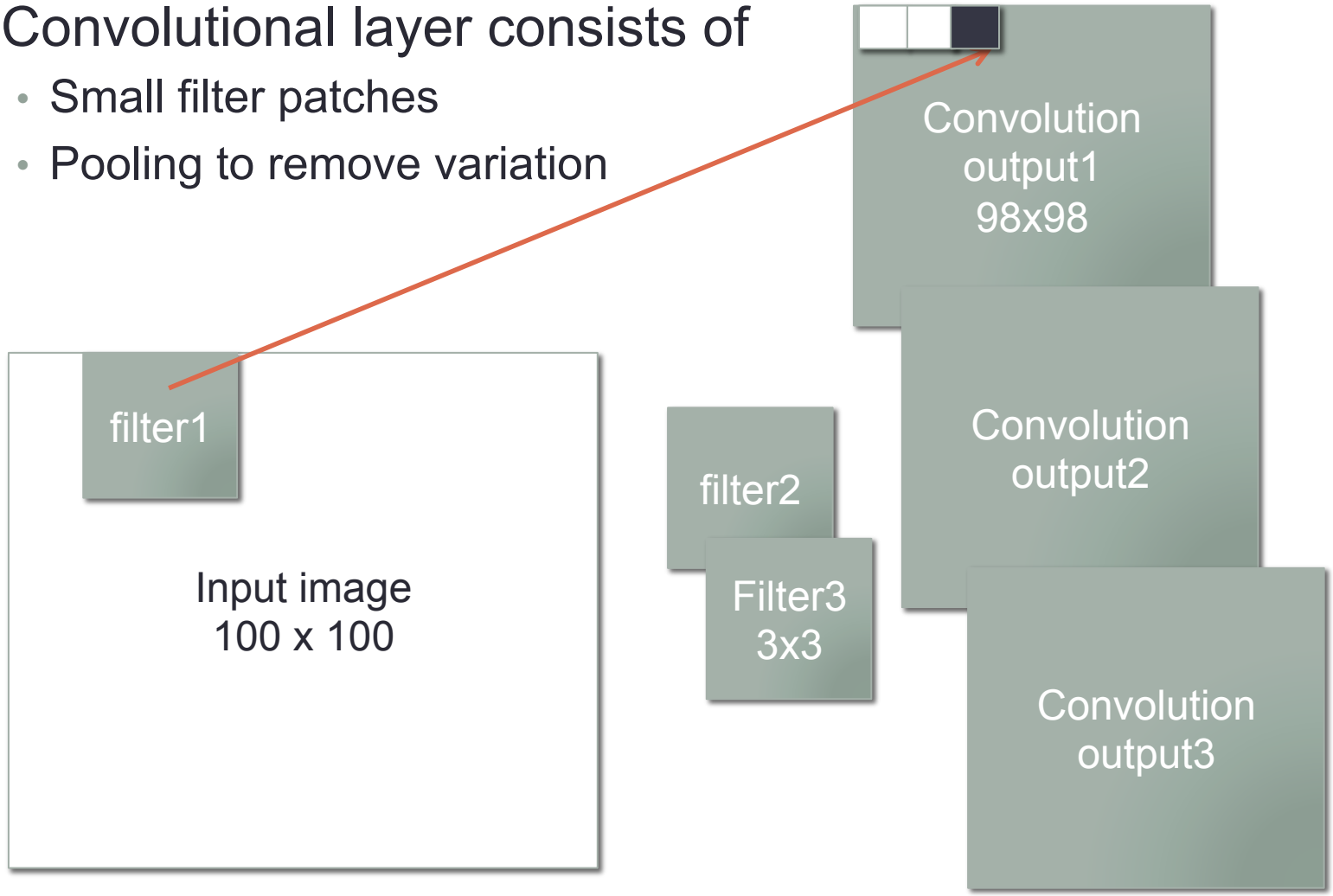
Convolutional filters

- Convolutional layer consists of
  - Small filter patches
  - Pooling to remove variation



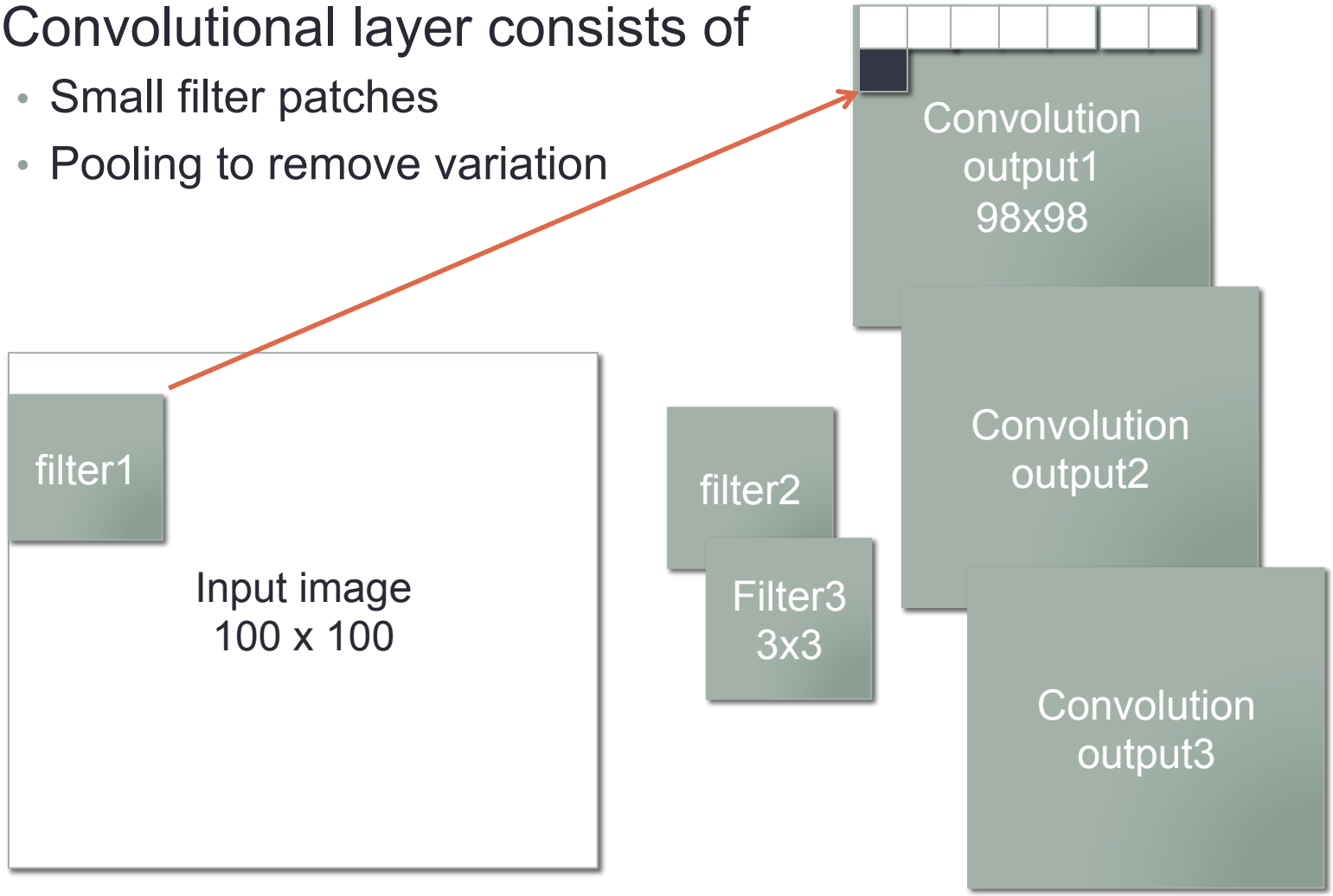
Convolutional filters

- Convolutional layer consists of
  - Small filter patches
  - Pooling to remove variation



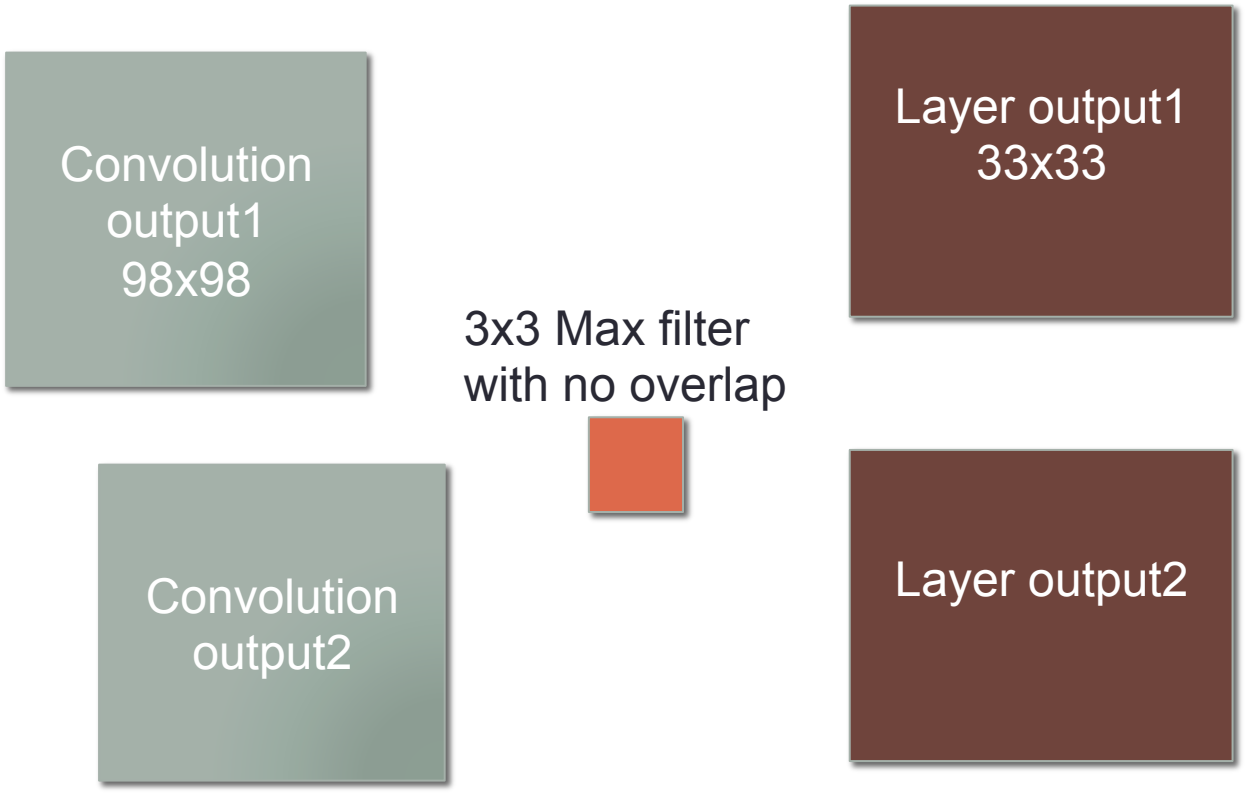
Convolutional filters

- Convolutional layer consists of
  - Small filter patches
  - Pooling to remove variation



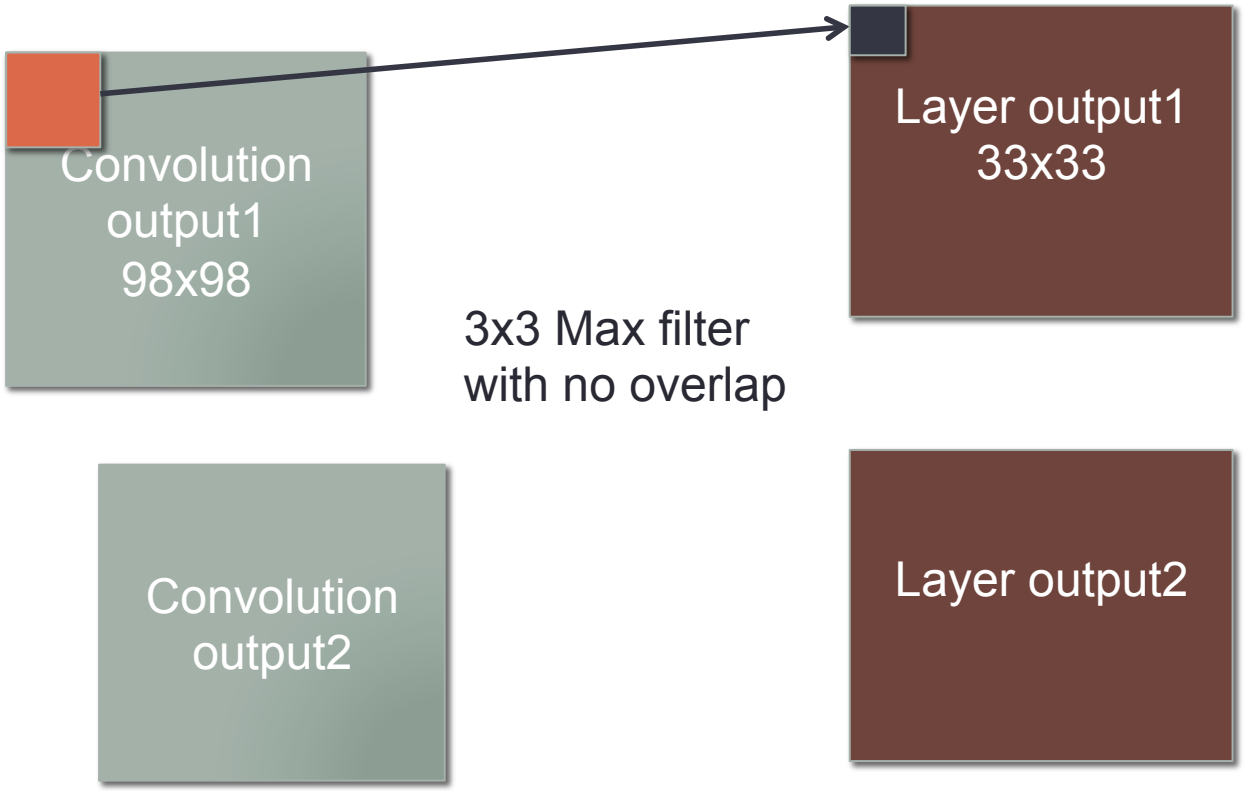
Pooling/subsampling

- Convolutional layer consists of
  - Small filter patches
  - Pooling to remove variation



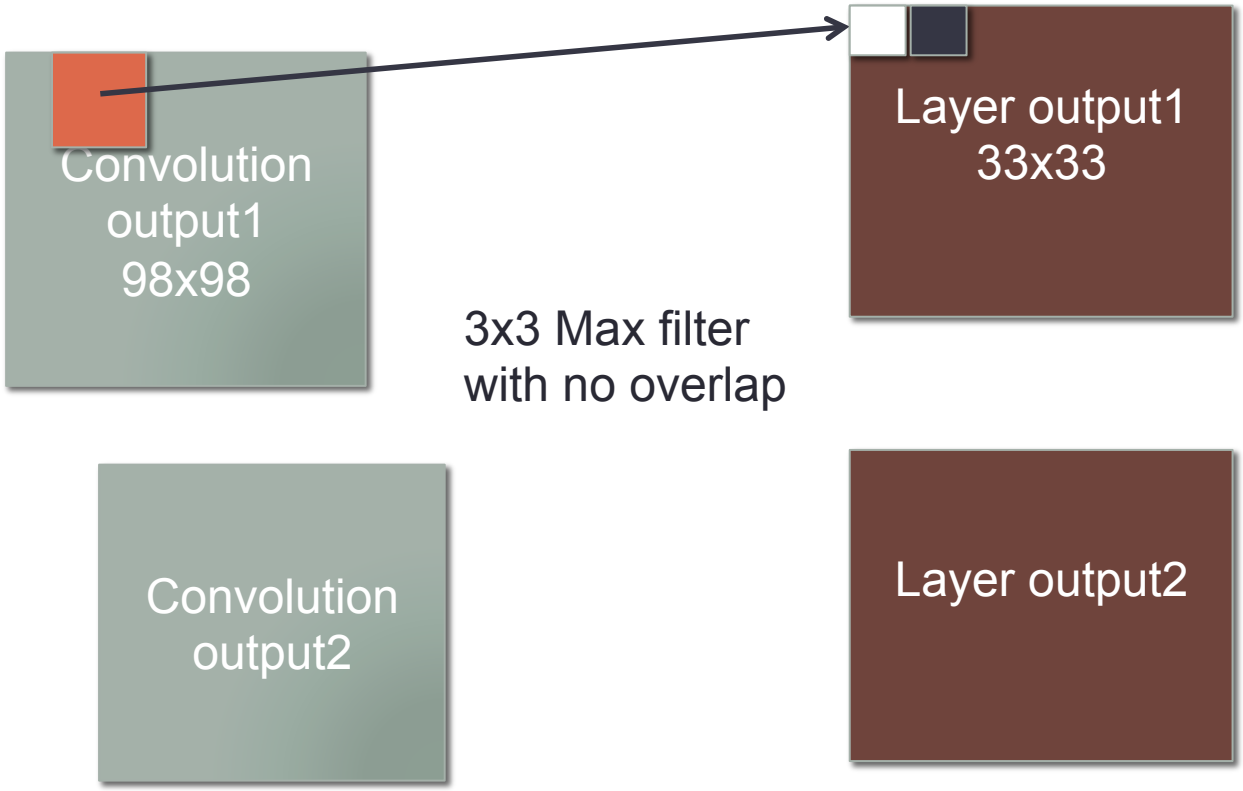
Pooling/subsampling

- Convolutional layer consists of
  - Small filter patches
  - Pooling to remove variation



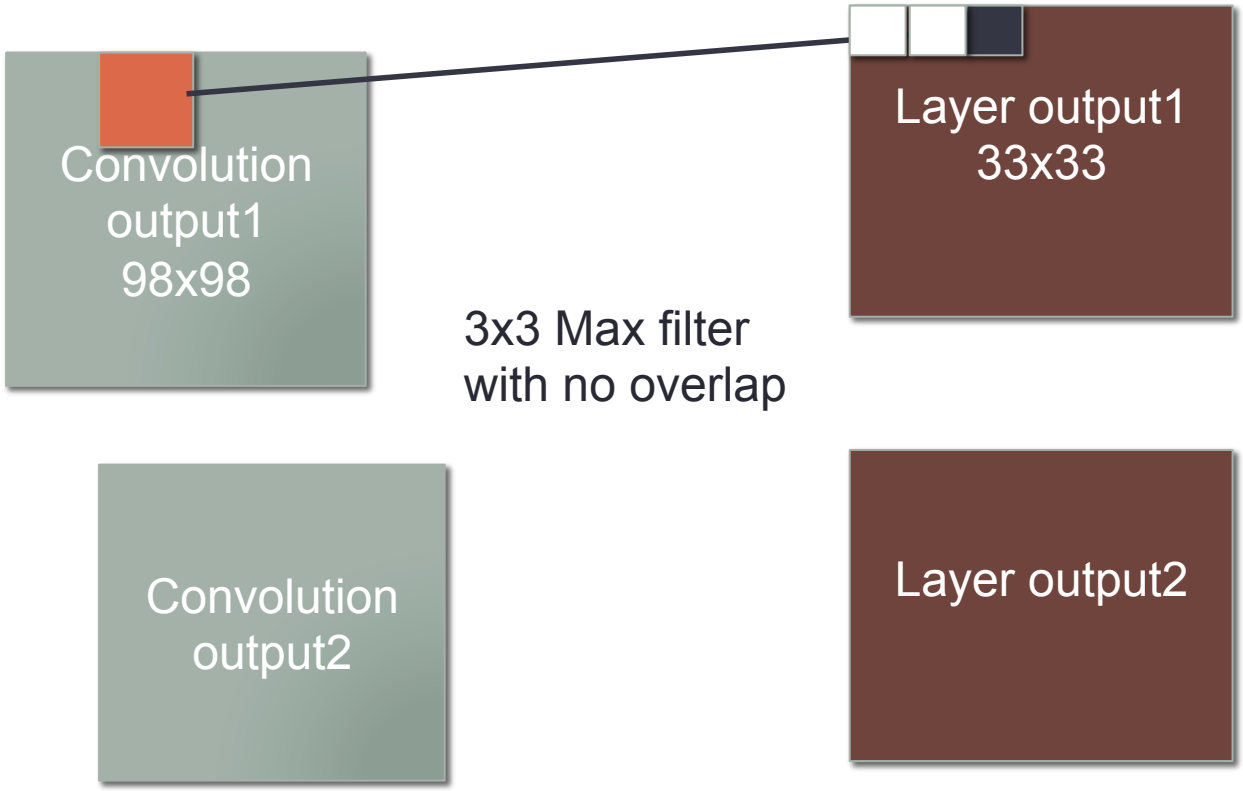
Pooling/subsampling

- Convolutional layer consists of
  - Small filter patches
  - Pooling to remove variation



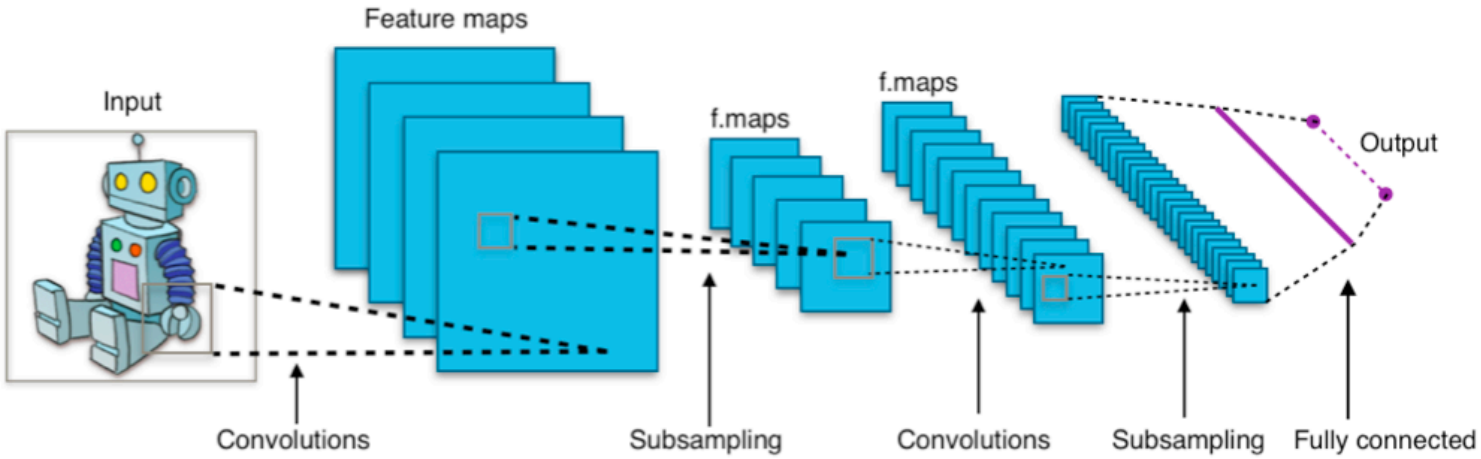
Pooling/subsampling

- Convolutional layer consists of
  - Small filter patches
  - Pooling to remove variation



CNN

- Filter size, number of filters, and pooling rate are all parameters
- Usually followed by a fully connected network at the end
  - CNN is good at learning low level features
  - DNN combines the features into high level features and classify



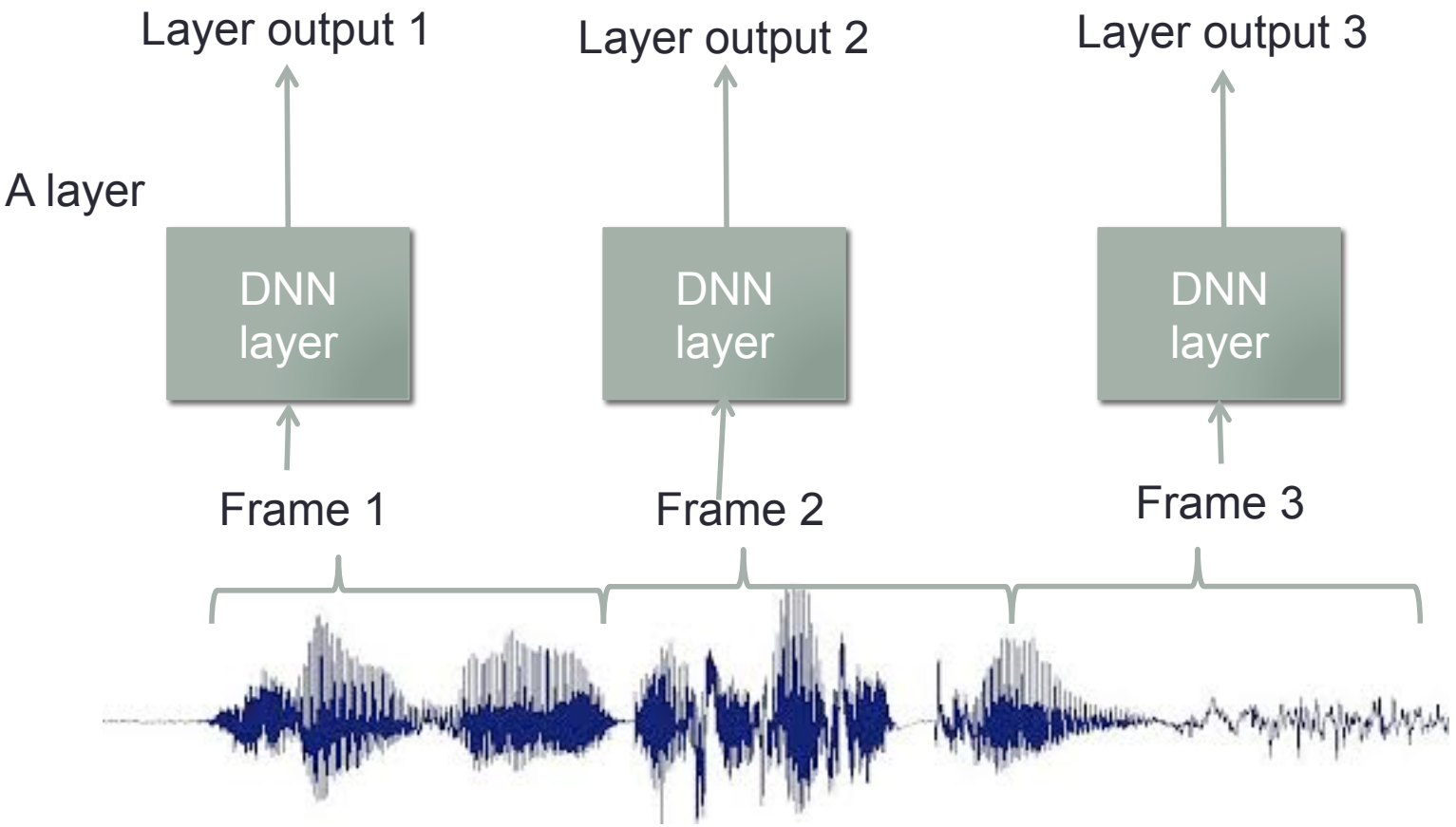
https://en.wikipedia.org/wiki/Convolutional_neural_network#/media/File:Typical_cnn.png

Recurrent neural network (RNN)

- Many tasks has temporal relationship
  - Speech – CVC structure
  - NLP – open bracket should follow by an end bracket
- DNN parameters are fixed (for the forward pass). Thus, it does not have the memory to keep track of these relationships
- RNN try to force the model to take into account of previous inputs and outputs

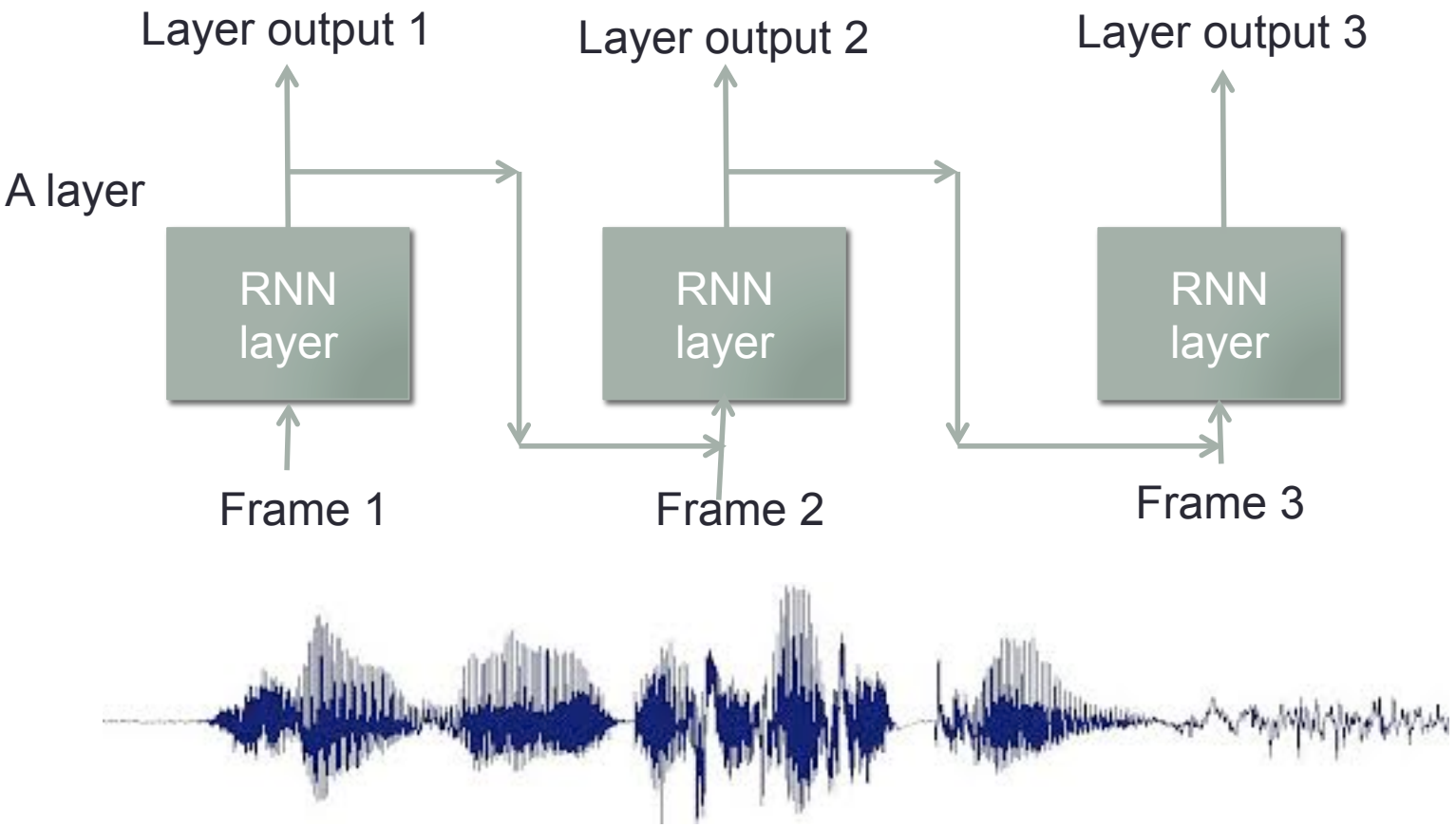
Recurrent neural network (RNN)

- DNN framework



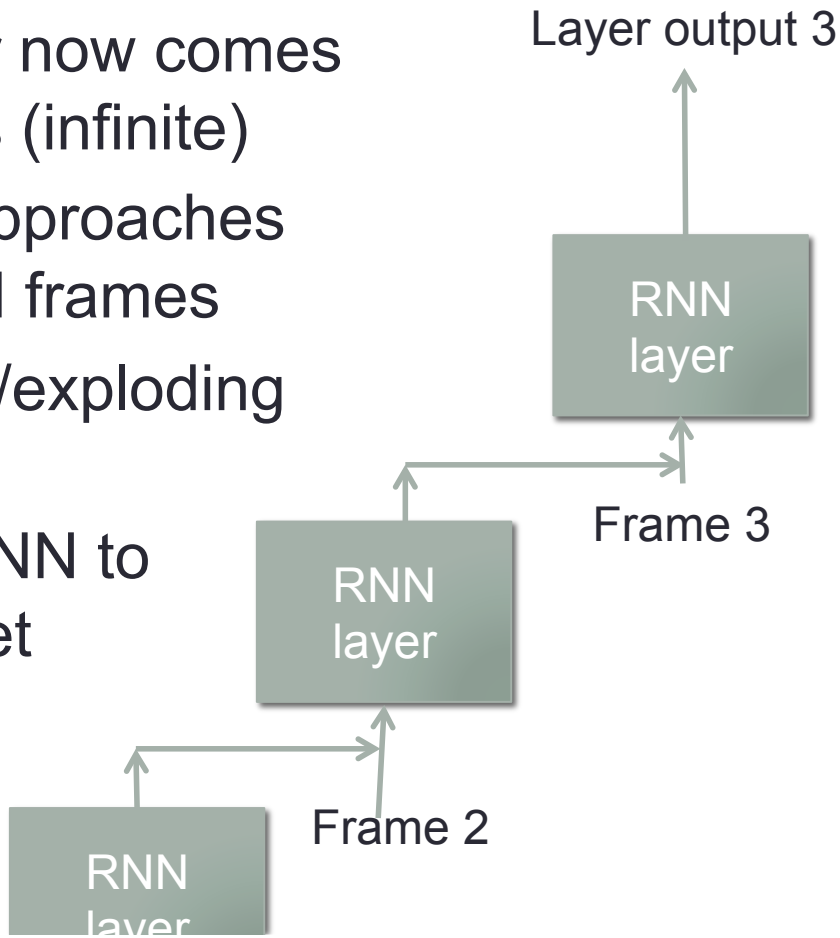
Recurrent neural network (RNN)

- RNN framework



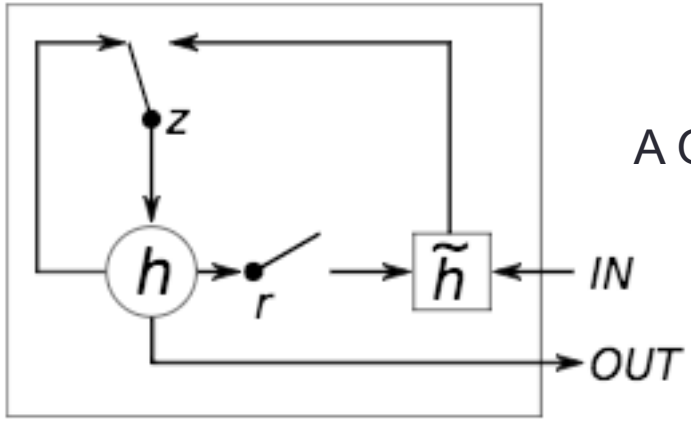
Back propagation through time

- RNN weights are now shared across time. Updating the weights is now trickier
- Updates for a layer now comes from multiple terms (infinite)
- Practical training approaches will truncate after N frames
- Gradient vanishing/exploding problem!
- Needs a way for RNN to know when to forget



Gated recurrent unit (GRU)

- Forms a Gated Recurrent Neural Networks (GRNN)
- Add gates that can choose to reset (r) or update (z)



A GRU layer

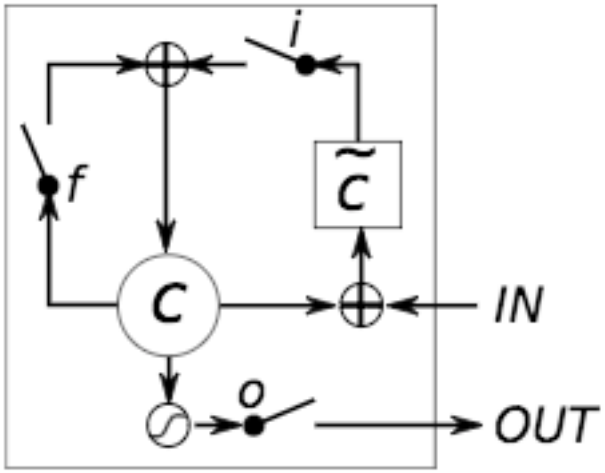
$$z_t = \sigma_g(W_z x_t + U_z h_{t-1} + b_z)$$
$$r_t = \sigma_g(W_r x_t + U_r h_{t-1} + b_r)$$
$$h_t = z_t \circ h_{t-1} + (1 - z_t) \circ \sigma_h(W_h x_t + U_h(r_t \circ h_{t-1}) + b_h)$$

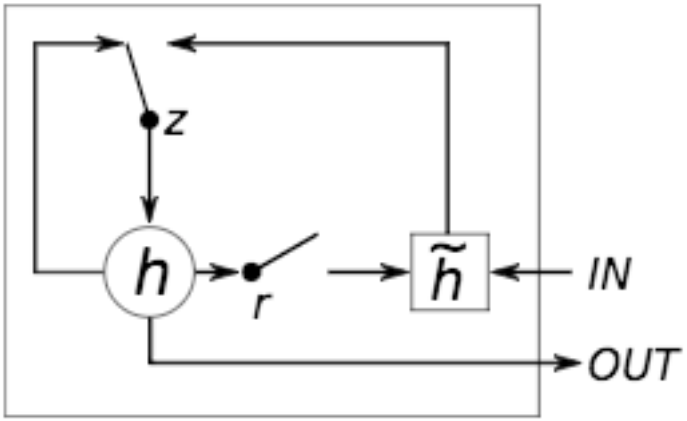
$\tilde{h}$

Junyoung Chung, Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling, https://arxiv.org/abs/1412.3555

LSTMs

- Have 3 gates, forget (f), input (i), output (o)
- Has a memory cell (c) that can be different from the output



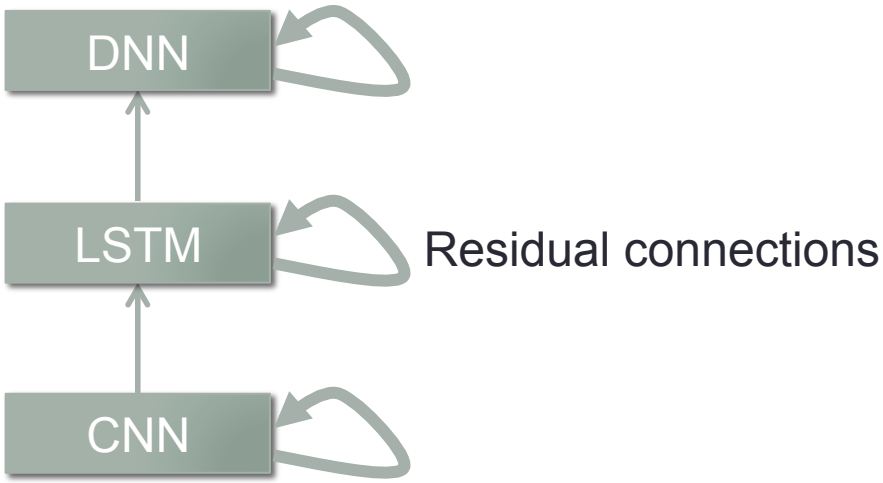


- No clear conclusion whether LSTM or GRU is better yet
- For smaller data set, GRU seems to be better

Junyoung Chung, Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling, https://arxiv.org/abs/1412.3555

LSTMs

- Usually combined with CNN and DNN for real applications



LSTM memories remember meaningful things

Cell sensitive to position in line:

```
The sole importance of the crossing of the Berezina lies in the fact
that it plainly and indubitably proved the fallacy of all the plans for
cutting off the enemy's retreat and the soundness of the only possible
line of action--the one Kutuzov and the general mass of the army
demanded--namely, simply to follow the enemy up. The French crowd fled
at a continually increasing speed and all its energy was directed to
reaching its goal. It fled like a wounded animal and it was impossible
to block its path. This was shown not so much by the arrangements it
made for crossing as by what took place at the bridges. When the bridges
broke down, unarmed soldiers, people from Moscow and women with children
who were with the French transport, all--carried on by vis inertiae--
pressed forward into boats and into the ice-covered water and did not,
surrender.
```

Cell that turns on inside quotes:

```
"You mean to imply that I have nothing to eat out of.... On the
contrary, I can supply you with everything even if you want to give
dinner parties," warmly replied Chichagov, who tried by every word he
spoke to prove his own rectitude and therefore imagined Kutuzov to be
animated by the same desire.

Kutuzov, shrugging his shoulders, replied with his subtle penetrating
smile: "I meant merely to say what I said."
```

Cell that robustly activates inside if statements:

```
static int __dequeue_signal(struct sigpending *pending, sigset_t *mask,
        siginfo_t *info)
{
  int sig = next_signal(pending, mask);
  if (sig) {
    if (current->notifier) {
      if (sigismember(current->notifier_mask, sig)) {
        if (!(current->notifier)(current->notifier_data)) {
          clear_thread_flag(TIF_SIGPENDING);
          return 0;
        }
      }
    }
    collect_signal(sig, pending, info);
  }
  return sig;
}
```

A large portion of cells are not easily interpretable. Here is a typical example:

```
/* Unpack a filter field's string representation from user-space
 * buffer. */
char *audit_unpack_string(void **bufp, size_t *remain, size_t len)
{
  char *str;
  if (!*bufp || (len == 0) || (len > *remain))
    return ERR_PTR(-EINVAL);
  /* Of the currently implemented string fields, PATH_MAX
   * defines the longest valid length.
   */
```

Andrej Karpathy, Visualizing and Understanding Recurrent Networks https://arxiv.org/abs/1506.02078

Cell that turns on inside comments and quotes:

```
/* Duplicate LSM field information.  The lsm_rule is opaque, so
 * re-initialized. */
static inline int audit_dupe_lsm_field(struct audit_field *df,
                                       struct audit_field *sf)
{
  int ret = 0;
  char *lsm_str;
  /* our own copy of lsm_str */
  lsm_str = kstrdup(sf->lsm_str, GFP_KERNEL);
  if (unlikely(!lsm_str))
    return -ENOMEM;
  df->lsm_str = lsm_str;
  /* our own (refreshed) copy of lsm_rule */
  ret = security_audit_rule_init(df->type, df->op, df->lsm_str,
                                 (void **)&df->lsm_rule);
  /* Keep currently invalid fields around in case they
   * become valid after a policy reload. */
  if (ret == -EINVAL) {
    pr_warn("audit rule for LSM \'%s\' is invalid\n",
            df->lsm_str);
    ret = 0;
  }
  return ret;
}
```

Cell that is sensitive to the depth of an expression:

```
#ifdef CONFIG_AUDITSYSCALL
static inline int audit_match_class_bits(int class, u32 *mask)
{
  int i;
  if (classes[class]) {
    for (i = 0; i < AUDIT_BITMASK_SIZE; i++)
      if (mask[i] & classes[class][i])
        return 0;
  }
  return 1;
}
```

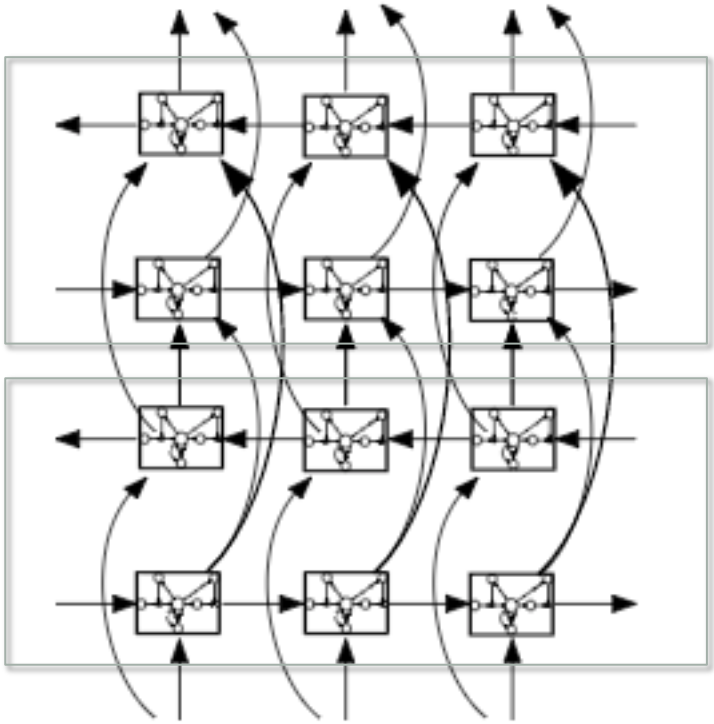
Cell that might be helpful in predicting a new line. Note that it only turns on for some ")":

```
char *audit_unpack_string(void **bufp, size_t *remain, si
{
  char *str;
  if (!*bufp || (len == 0) || (len > *remain))
    return ERR_PTR(-EINVAL);
  /* Of the currently implemented string fields, PATH_MAX
   * defines the longest valid length.
   */
  if (len > PATH_MAX)
    return ERR_PTR(-ENAMETOOLONG);
  str = kmalloc(len + 1, GFP_KERNEL);
  if (unlikely(!str))
    return ERR_PTR(-ENOMEM);
  memcpy(str, *bufp, len);
  str[len] = 0;
  *bufp += len;
  *remain -= len;
  return str;
}
```

Andrej Karpathy, Visualizing and Understanding Recurrent Networks https://arxiv.org/abs/1506.02078

Bi-directional LSTMs

- Uni-directional LSTMs only consider information from the past
- Bi-directional LSTMs consider information from the future and the past
- For each bi-directional layer, there are two LSTMs, one from the past, one from the future.



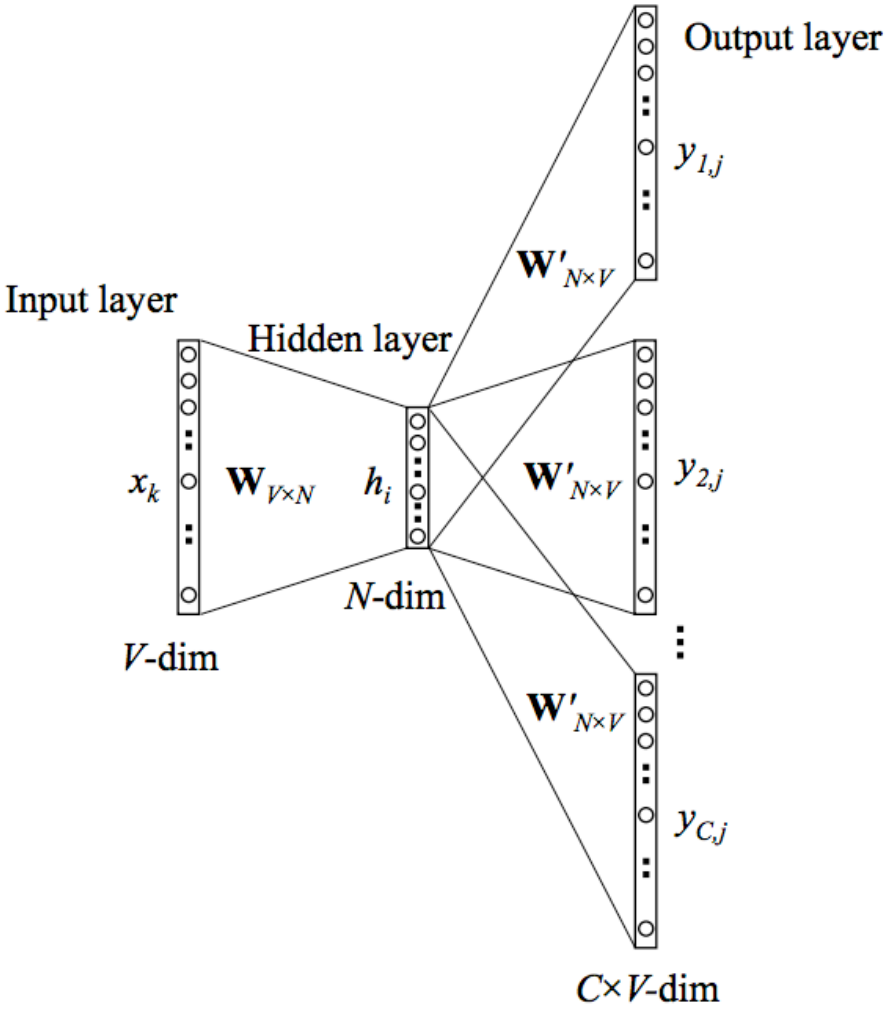
Hasim Sak, Long Short-Term Memory Recurrent Neural Network Architectures for Large Scale Acoustic Modeling, 2014

Encoder-decoder

- We know neural networks can learn representations internally
- Can we use those internal representations?

Word2Vec

- We can understand the meaning of a word from the context
- Train a neural network that predicts what are likely words around an input word
- Use the hidden activations as features



"A quick brown XXX jumps over the fence"

Mikolov, Linguistic Regularities in Continuous Space Word Representations, 2013

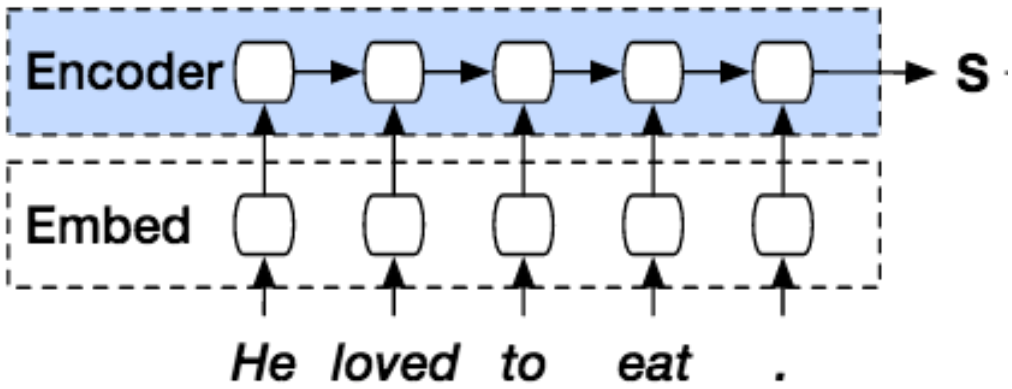
Word2Vec

- The vector representations (embeddings) has interesting properties
- Similar meaning words are near each other
- You can also add and subtract meanings
- Word2vec is now used in most NLP tasks as an input feature

Mikolov, Linguistic Regularities in Continuous Space Word Representations, 2013

Sentence encoder

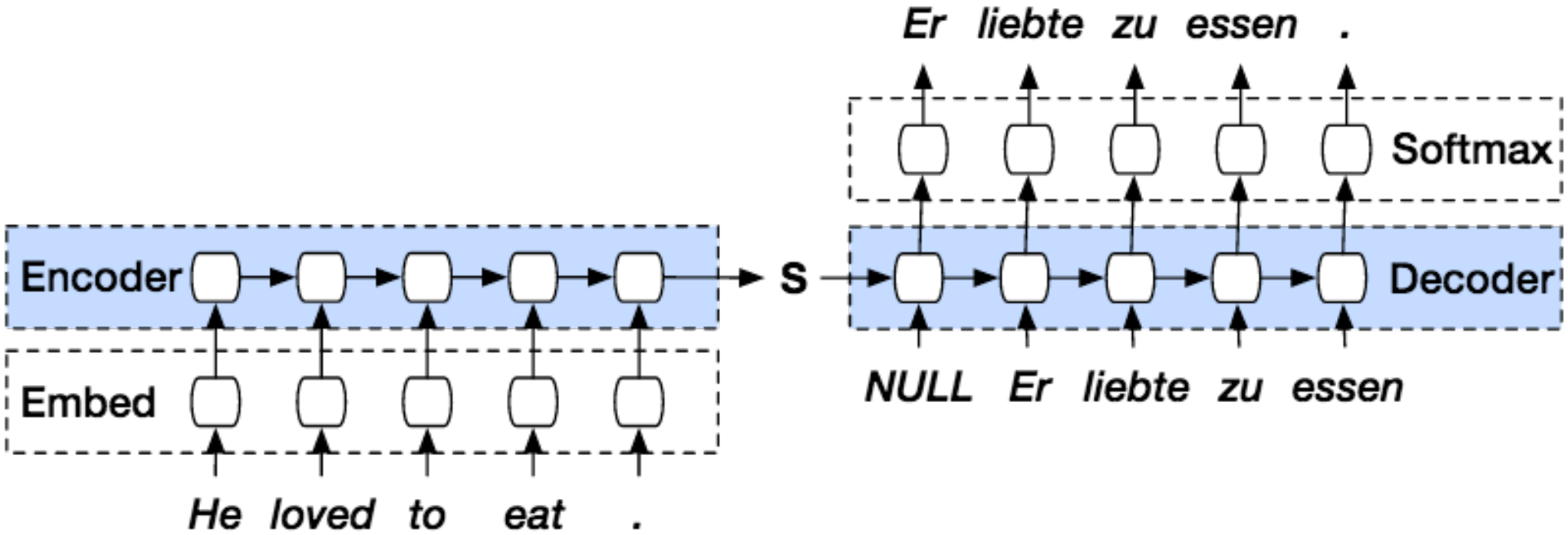
- How can we represent sentences?
- Similarly, use the internal states as the representation
  - LSTMs/GRUs are good for sequence task
  - Use LSTMs/GRUs memory cells outputs as representation



https://smerity.com/articles/2016/google_nmt_arch.html

Machine translation using encoder-decoder

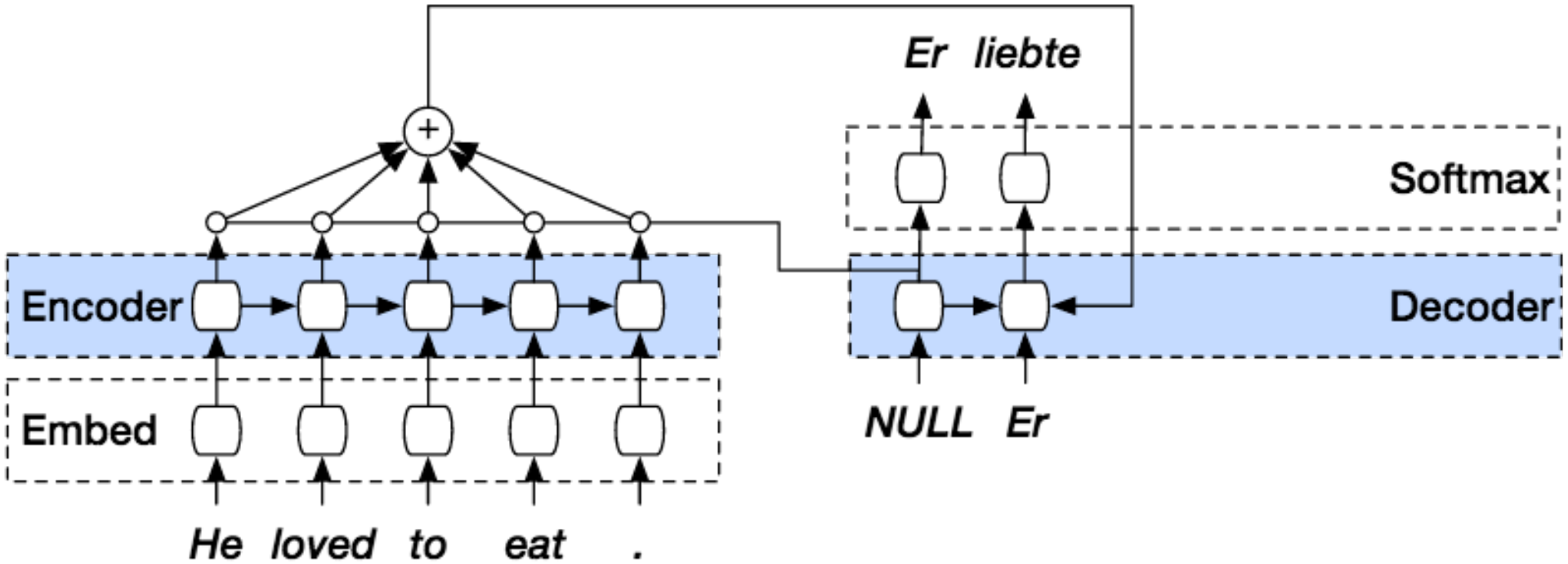
- Use another LSTM as the decoder
- Input to the LSTM is the encoded sentence and the previously output word



Ilya Sutskever, Sequence to Sequence Learning with Neural Networks, https://arxiv.org/abs/1409.3215

Attention model

- If the sentence is long, the model cannot remember long enough
  - Also problem of vanishing gradients
- Let the model pick where to focus



Dzmitry Bahdanau, Neural Machine Translation by Jointly Learning to Align and Translate, https://arxiv.org/abs/1409.0473

Attention model for summarization

- Attention model can help with long context sentences for NLP

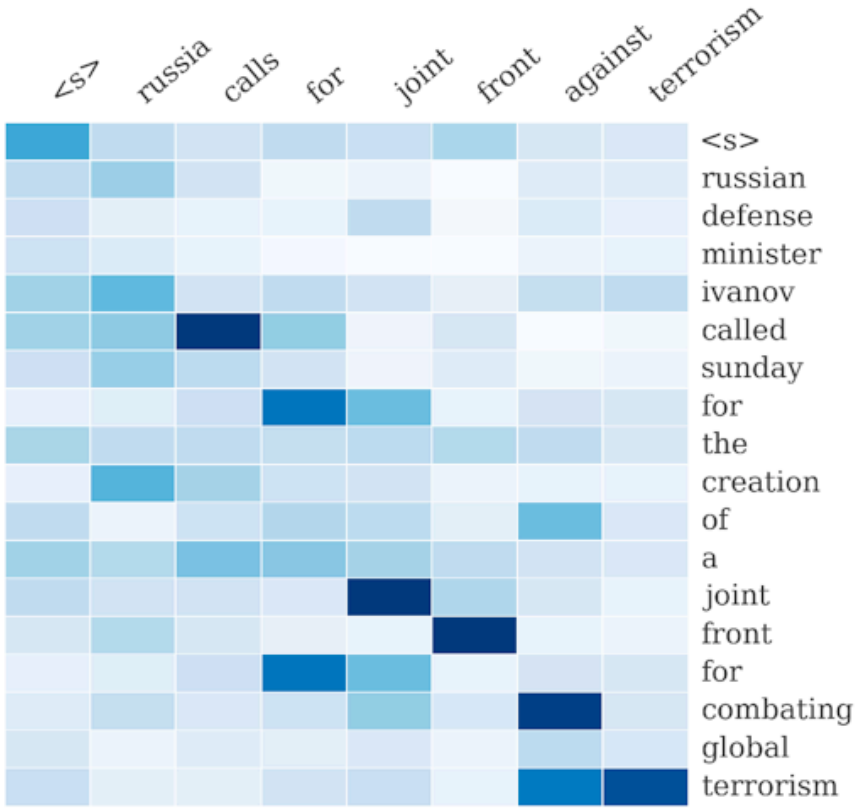


Figure 1: Example output of the attention-based summarization (ABS) system. The heatmap represents a soft alignment between the input (right) and the generated summary (top). The columns represent the distribution over the input after generating each word.

Alexander Rush, A Neural Attention Model for Sentence Summarization, https://arxiv.org/abs/1509.00685

Transfer Learning

- Neural networks learns useful information about the data
  - Hierarchal features
- These features are useful for other tasks

- Learn on a big data set and apply on small data set
- Many pre-trained models available

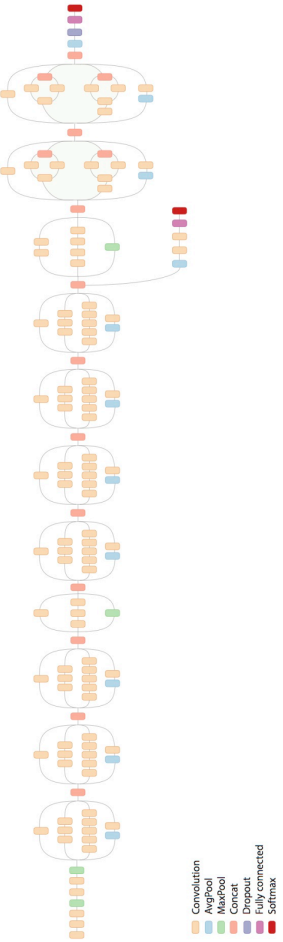
- Replace softmax layer with your task

Transfer Learning

Imagenet : millions of pictures, 1000+ object classes

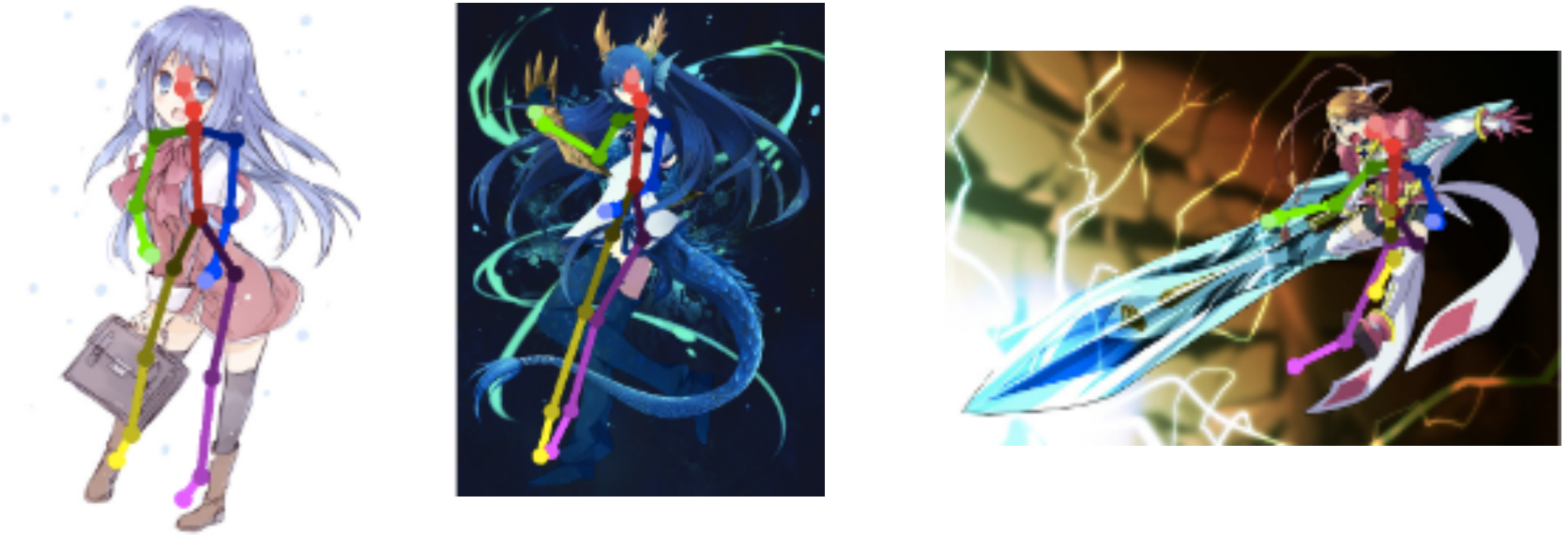
http://image-net.org/

Inception model



https://github.com/tensorflow/models/tree/master/inception

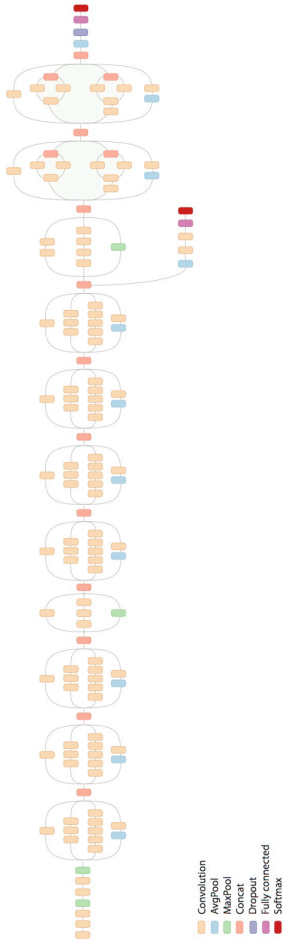
Pose Estimation of Anime Characters



Pramook Khungurn, Pose estimation of anime/manga characters: a case for synthetic data

Transfer Learning Note

- Remove more layers at the top if you believe lower level representations are more relevant to your task
- If you believe there is change in sensors, add layers at the bottom
  - Using a different microphone from the original set

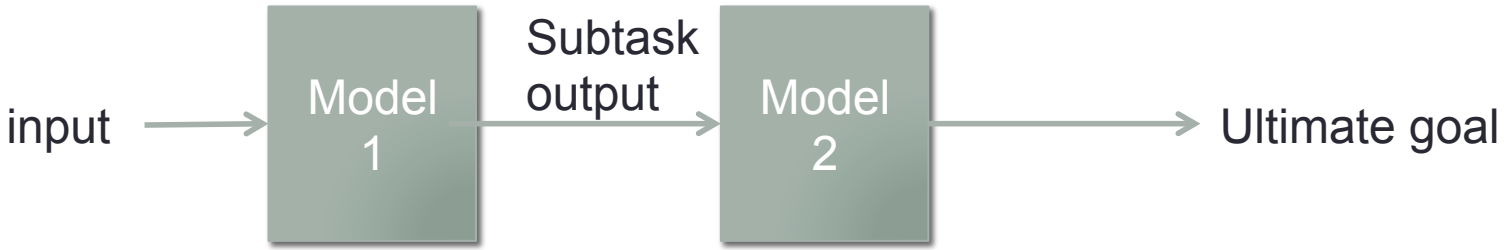


Trends in Deep learning

- End-to-end models
- Multimodels (General purpose model)
- Speeding up simulations
- Reinforcement learning
- One button machines

End-to-End Models

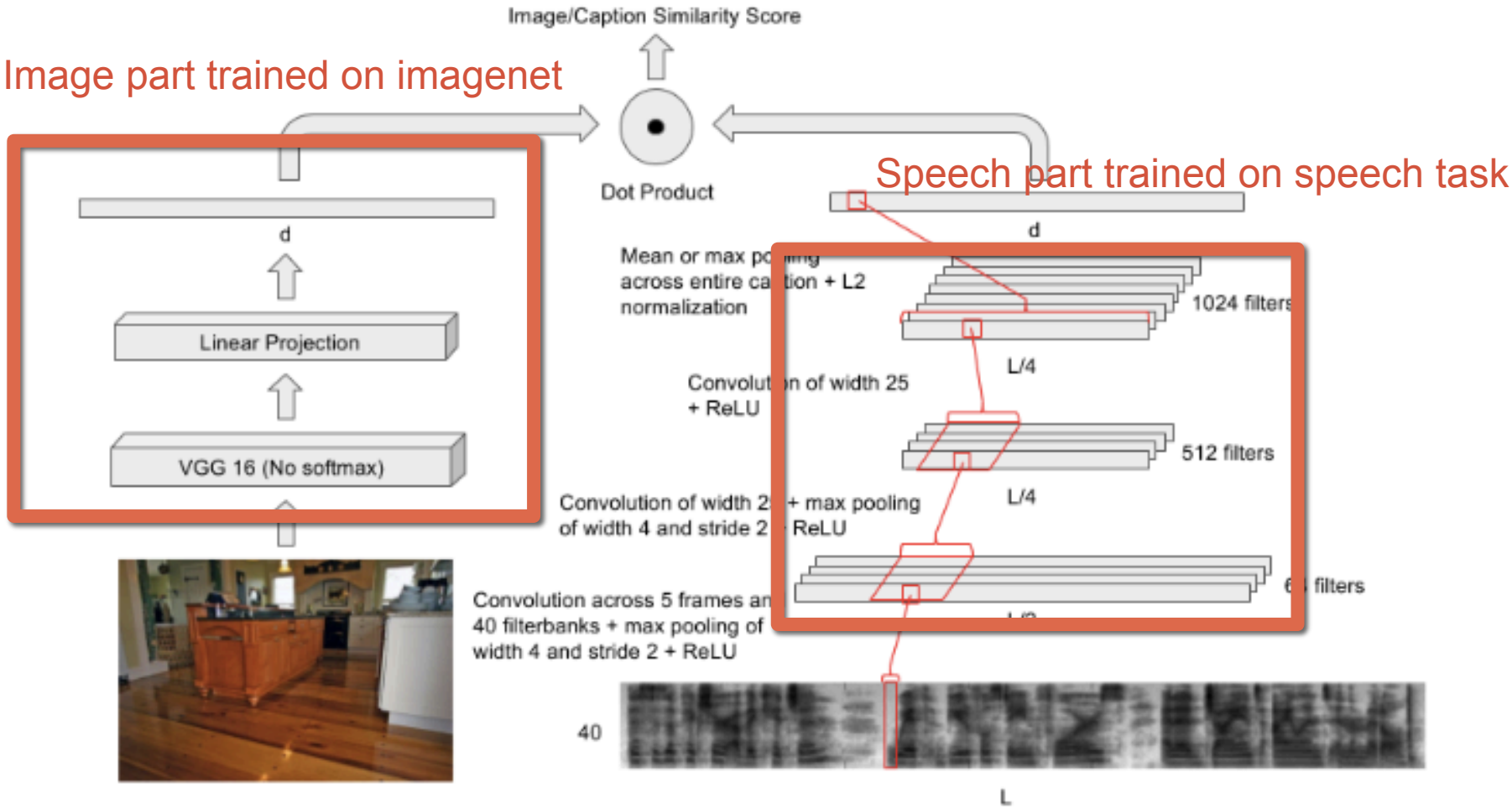
- Traditional machine learning divides the main task into smaller sub-tasks
  - Speech-to-speech translation
    - L1 Speech -> L1 Text -> L2 Text -> L2 Speech
  - Text caption generation
    - Image segmentation -> objection recognition -> NLP
- Errors propagate
- Each part are not jointly optimized



End-to-End Models

- Traditional machine learning divides the main task into smaller sub-tasks
  - Speech-to-speech translation
    - L1 Speech -> L1 Text -> L2 Text -> L2 Speech
  - Text caption generation
    - Image segmentation -> objection recognition -> NLP
- Errors propagate
- Each part are not jointly optimized

Speech description for image querying



Whole model trained on a smaller parallel corpus

David Harwath, Unsupervised Learning of Spoken Language with Visual Context

Pix2Code

- Given image of UI, outputs platform specific code

input

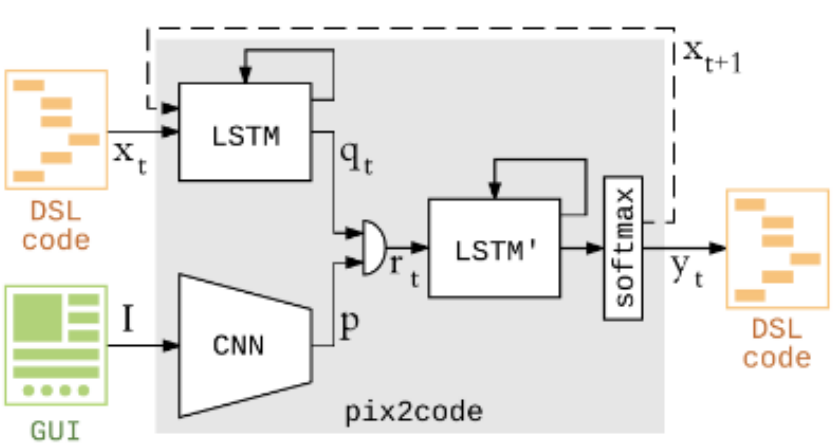
```
stack {
    row {
        label, switch
    }
    row {
        label, btn-add
    }
    row {
        label, slider, label
    }
    row {
        img, label
    }
}
footer {
    btn-more, btn-contact, btn-search, btn-download
}
```

output

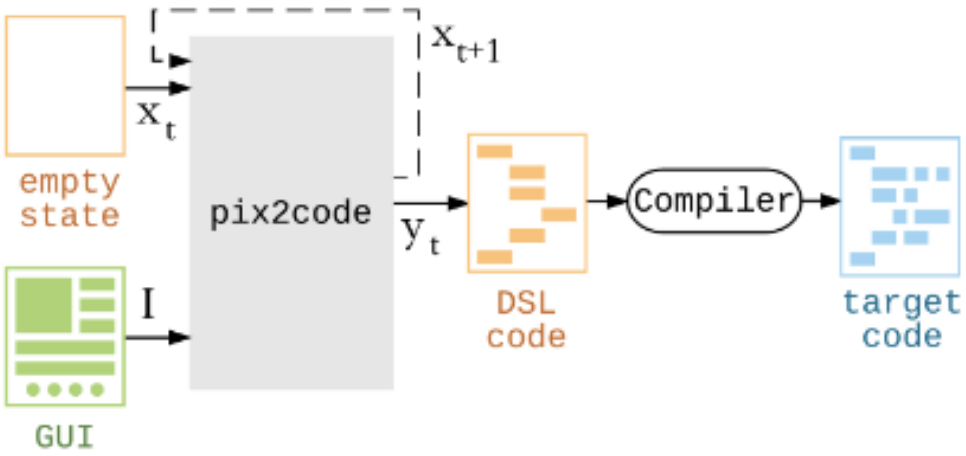
Tony Beltramelli, pix2code: Generating Code from a Graphical User Interface Screenshot, https://arxiv.org/abs/1705.07962

Pix2Code

- Similar to the machine translation encoder-decoder but with image recognition as the encoder



Training

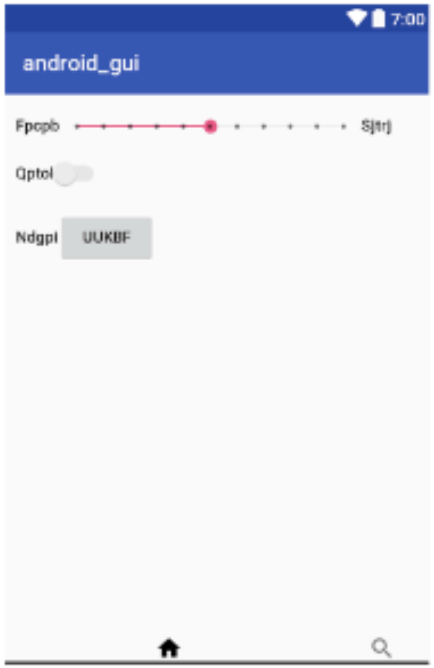


Testing

Tony Beltramelli, pix2code: Generating Code from a Graphical User Interface Screenshot, https://arxiv.org/abs/1705.07962

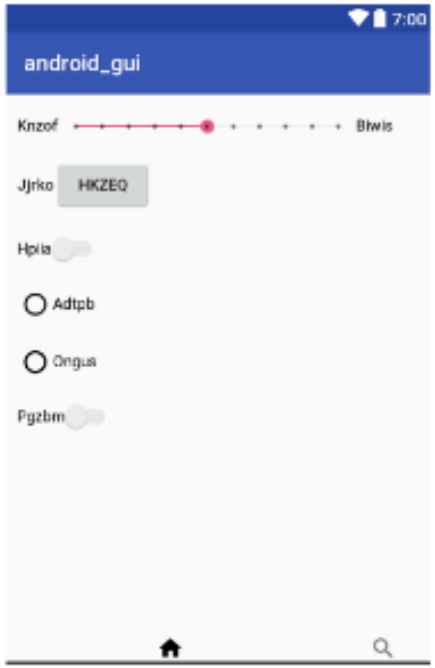
Pix2Code

Input picture



Generated UI

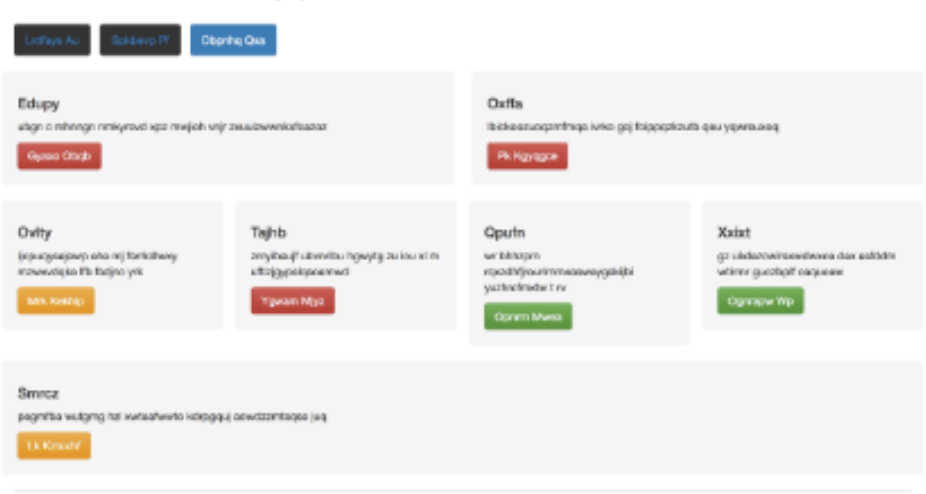
Input picture



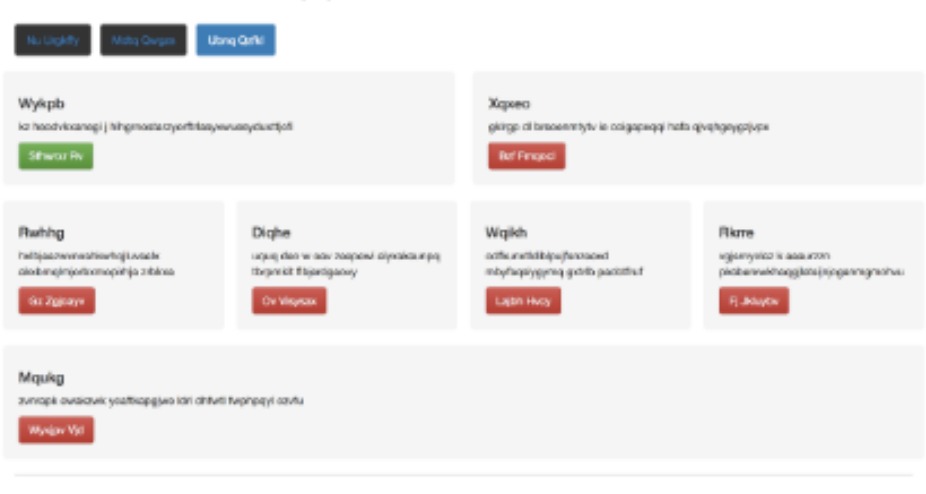
Generated UI

Tony Beltramelli, pix2code: Generating Code from a Graphical User Interface Screenshot, https://arxiv.org/abs/1705.07962

Pix2Code



Input picture

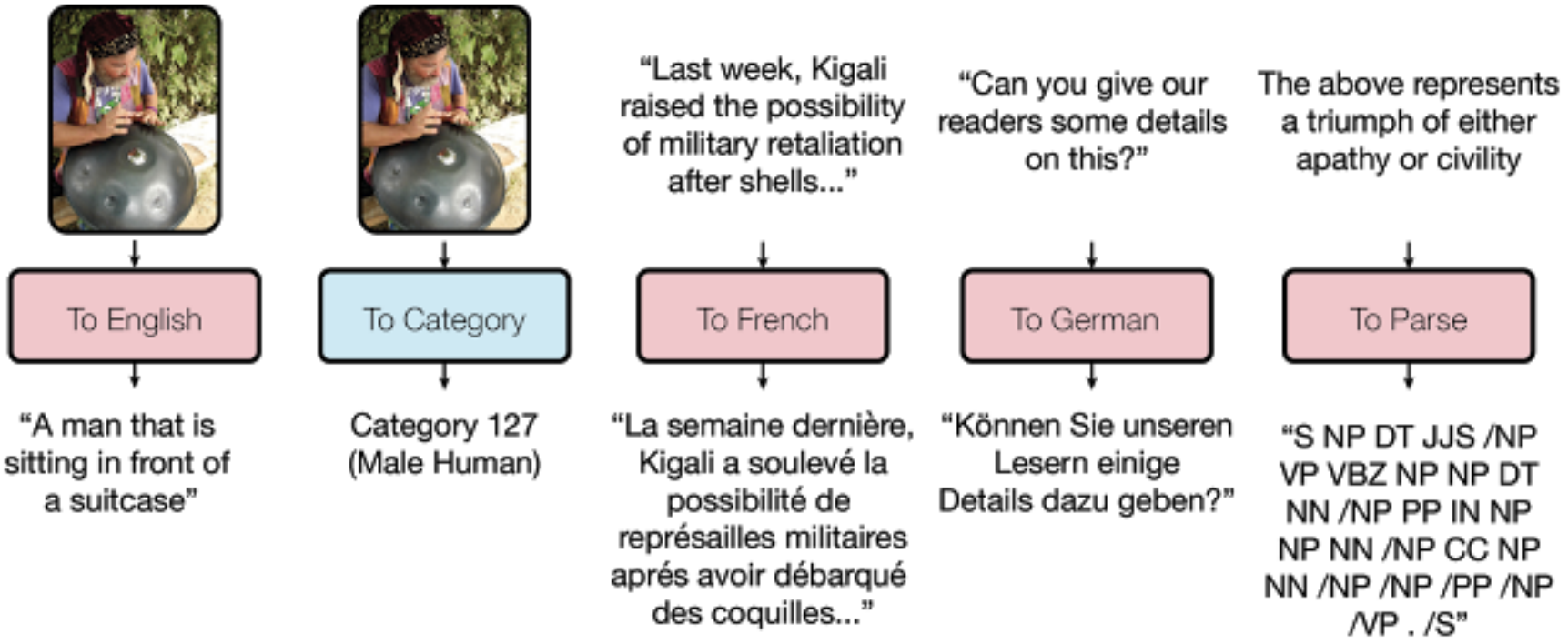


Generated UI

Tony Beltramelli, pix2code: Generating Code from a Graphical User Interface Screenshot, https://arxiv.org/abs/1705.07962

Multimodels

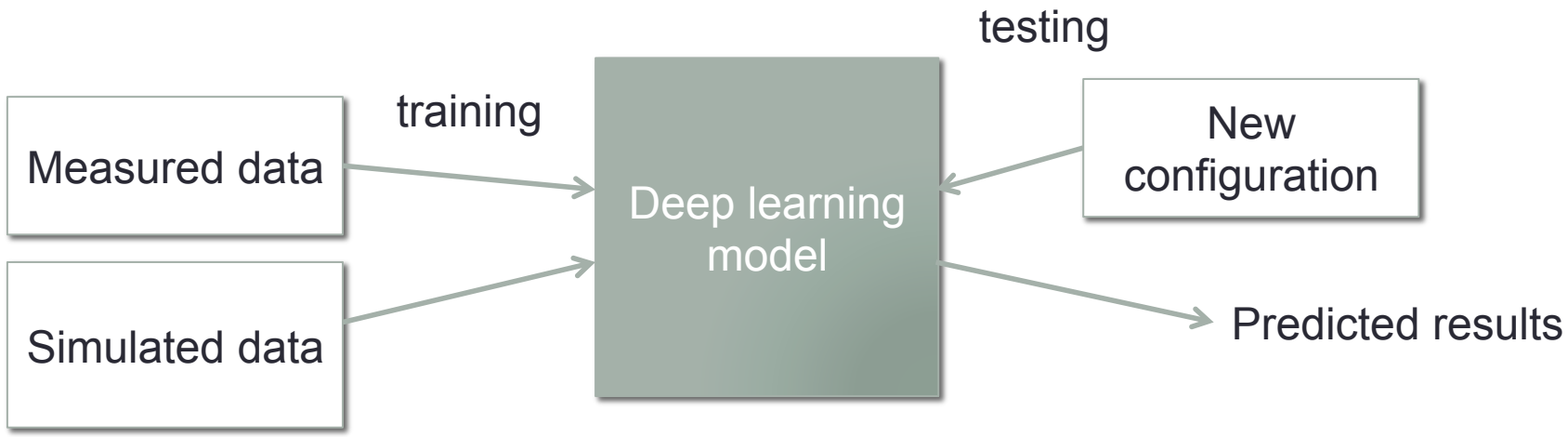
- A single model that can handle different kinds of inputs
- Joint representation of meaning improves performance



https://research.googleblog.com/2017/06/multimodel-multi-task-machine-learning.html

Speeding up simulations in physical science

- Several phenomena can be predicted by existing models in physics/chemistry/…
- Some models are slow
- Let deep learning learns the equations and give faster predictions at test time



Nongnuch Artrith, An implementation of artificial neural-network potentials for atomistic materials simulations: Performance for TiO2

Accelerating Eulerian Fluid Simulation with Convolutional Networks

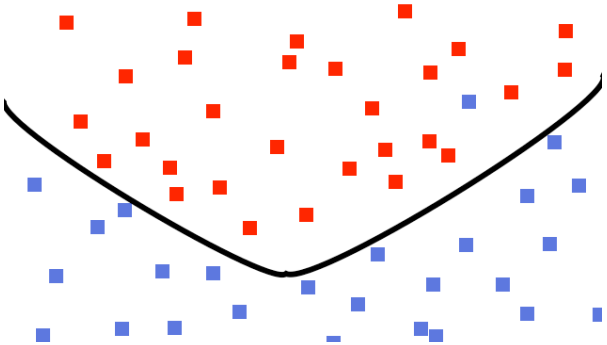
- https://www.youtube.com/watch?v=9pXubxPsIJw

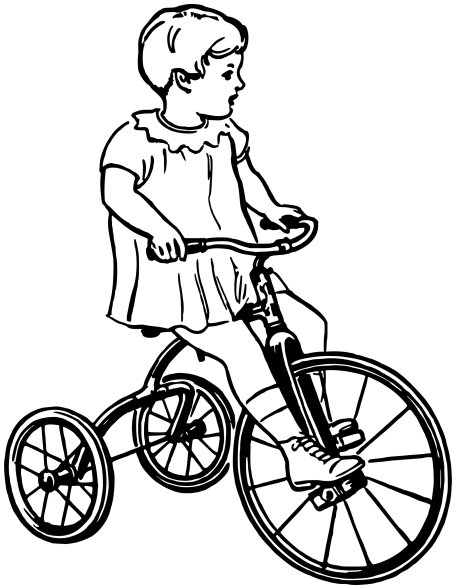
Jonathan Tompson, Accelerating Eulerian Fluid Simulation With Convolutional Networks, https://arxiv.org/abs/1607.03597

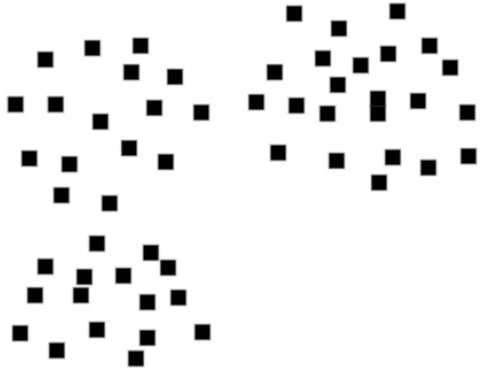
Reinforcement learning

- Learning from experience

Reinforcement learning



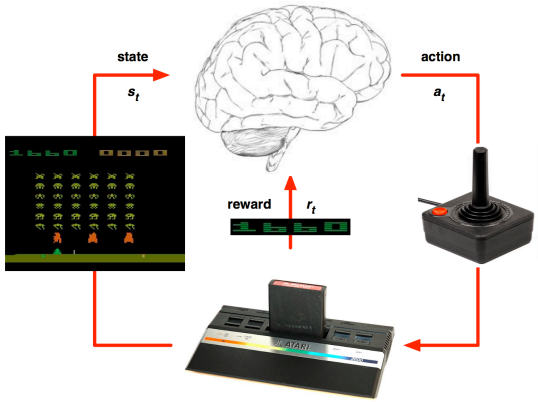




Supervised Learning

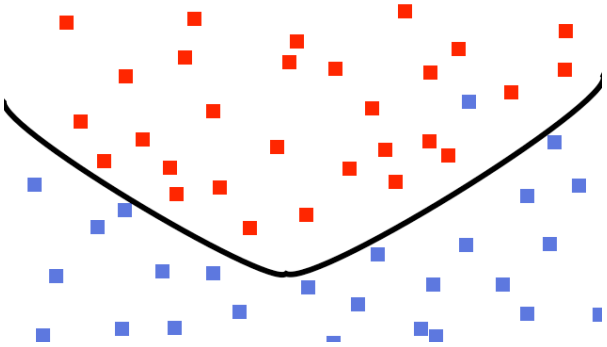
Reinforcement Learning

Unsupervised Learning

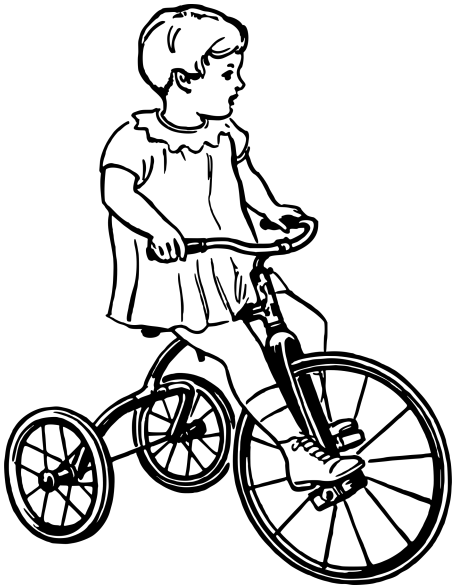


https://keon.io/deep-q-learning/

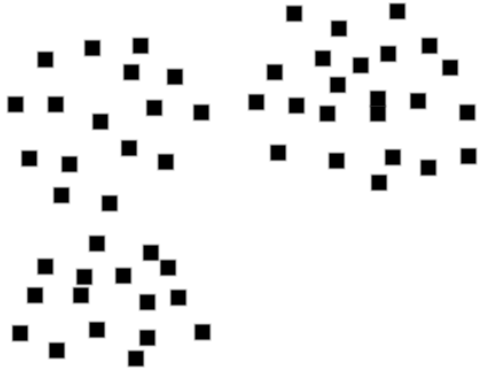
Reinforcement learning



Supervised Learning



Reinforcement Learning



Unsupervised Learning

- Reinforcement learning tries to learn the effect of each action and how it effect the outcome
- Reinforcement learning has been around for a long time
- Deep learning simplifies the learning process

Reinforcement learning for robotics

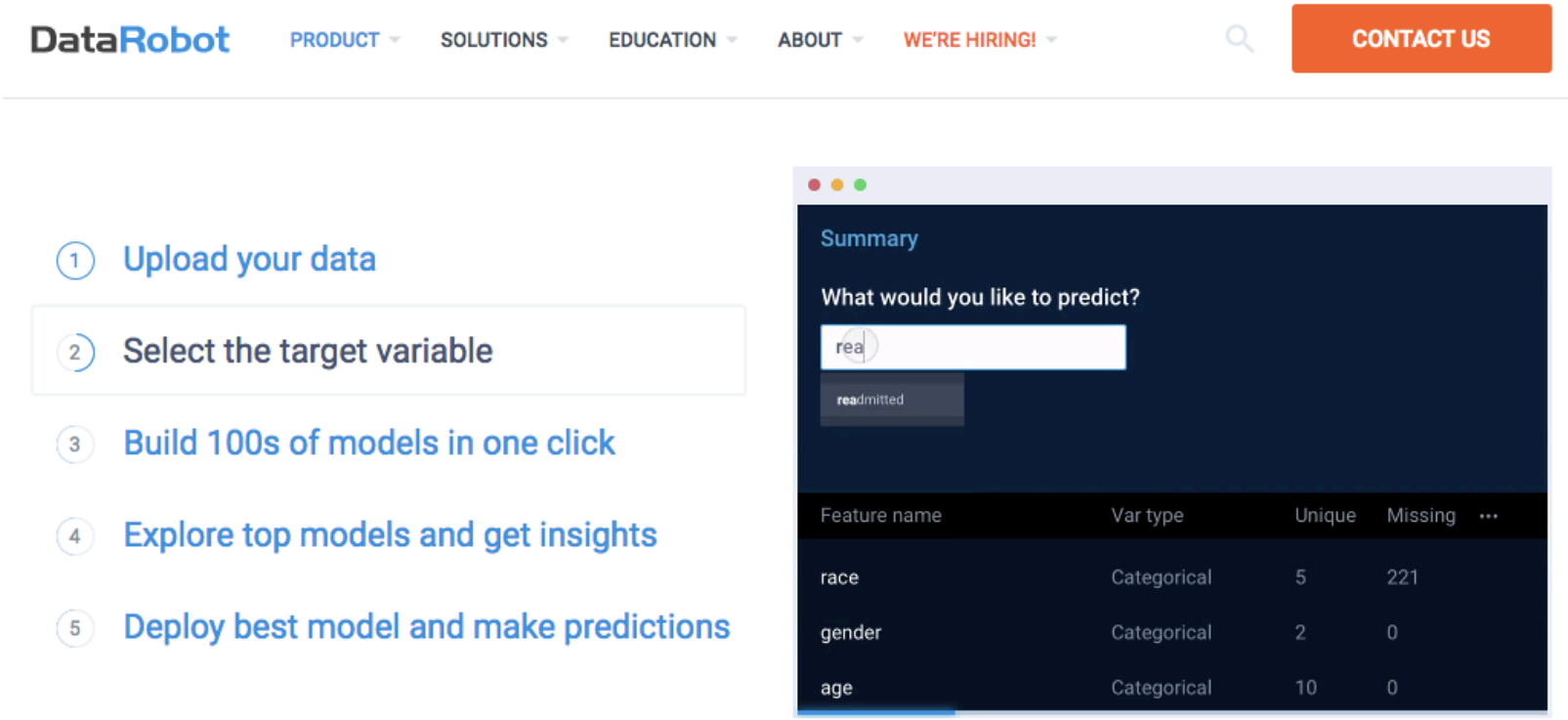
- https://www.youtube.com/watch?v=HbHqC8Himol

- Or use human supervision to help guide

Dhiraj Gandhi, Learning to Fly by Crashing, https://arxiv.org/abs/1704.05588

One button machines

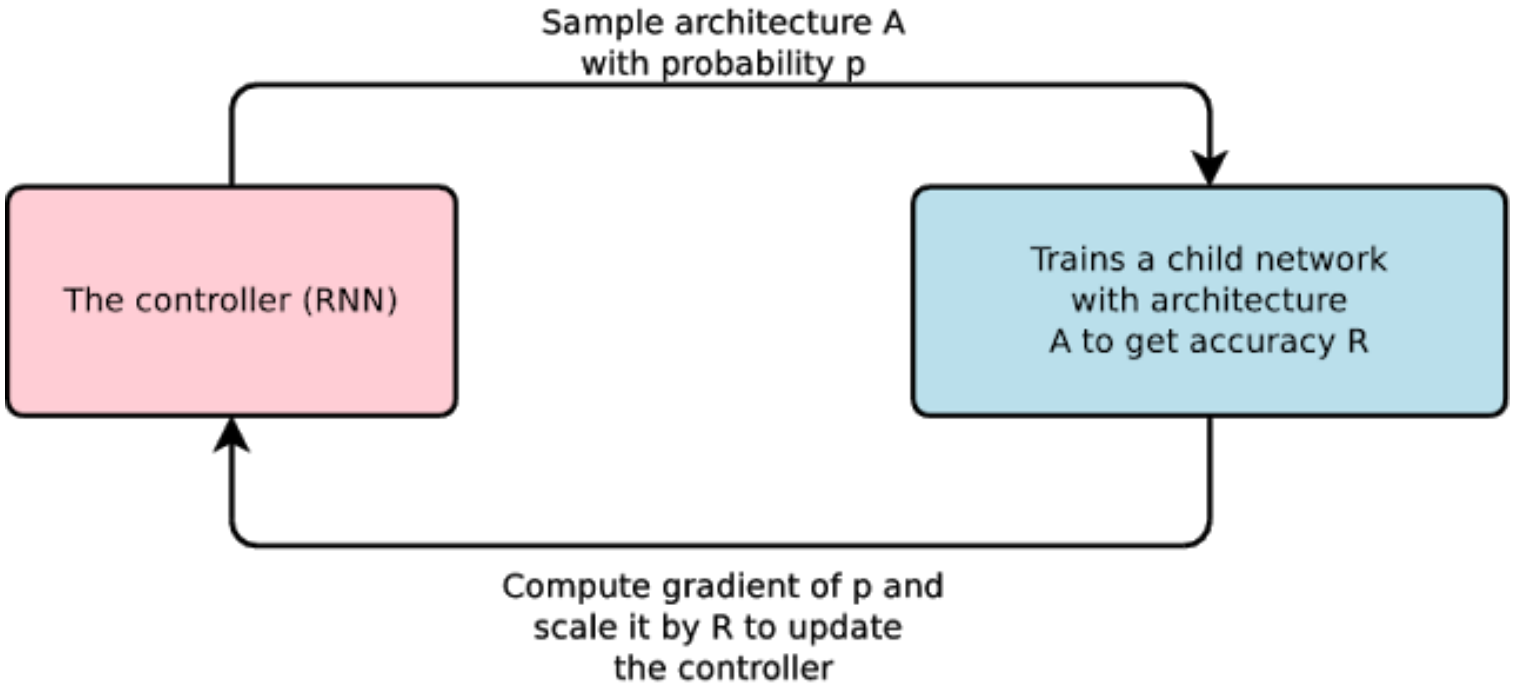
- Machine learning as a tool for non-experts
- Can a non-expert just provide the data and let the machine decides how to proceed



https://www.datarobot.com

Reinforcement Learning for Model Selection

- Tuning a network takes time
- Let machine learning learns how to tune a network
- Matches or outperforms ML experts performance



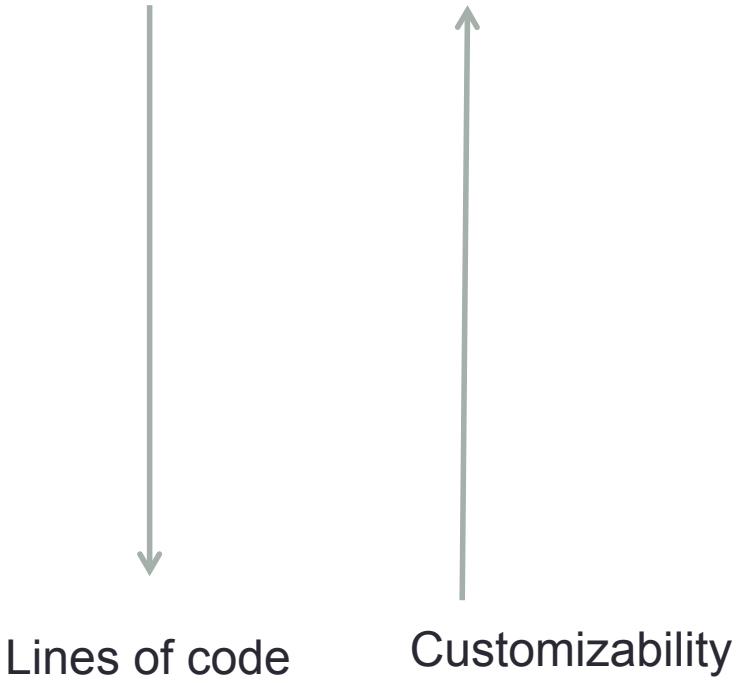
https://research.googleblog.com/2017/05/using-machine-learning-to-explore.html

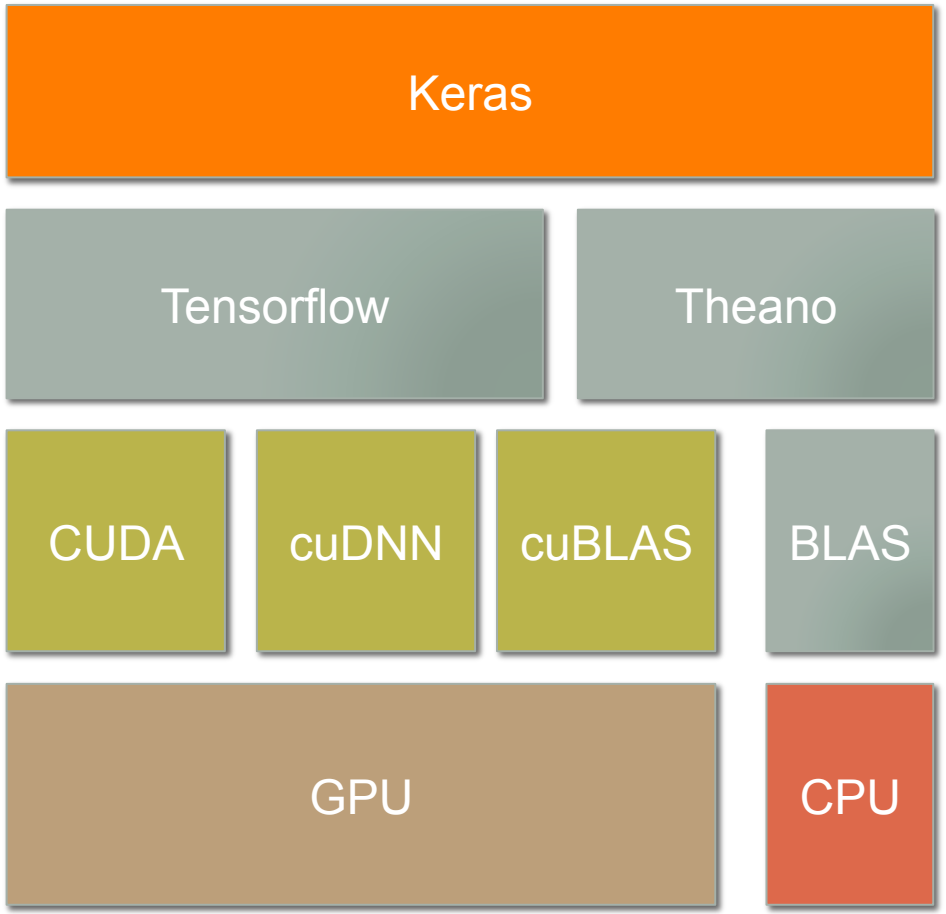
What's a good candidate for DNN applications

- Anything human can complete in 1 second (Andrew Ng)
- Biggest bottleneck is still labeled data
  - Transfer learning
  - Multimodel learning
  - Unsupervised training (auto-encoders)
  - Crowd sourcing

What toolkit

- Tensorflow – lower level
- Keras – higher level

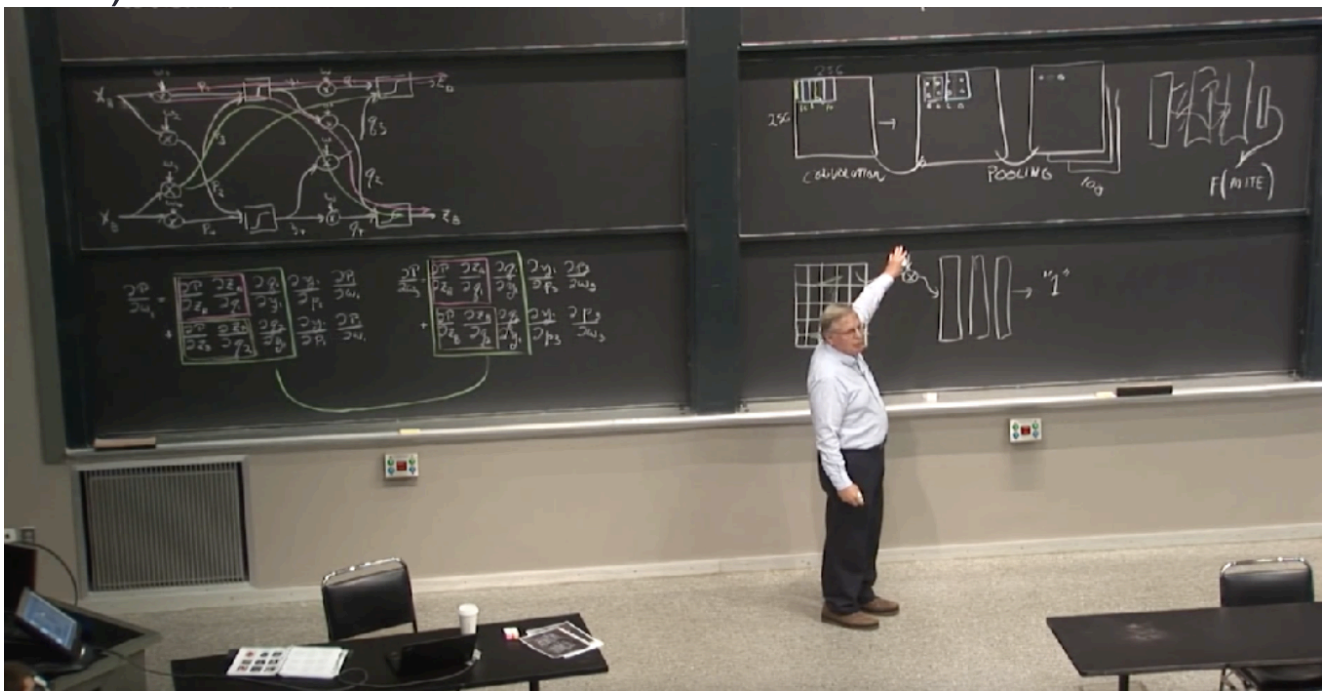




Where to learn more

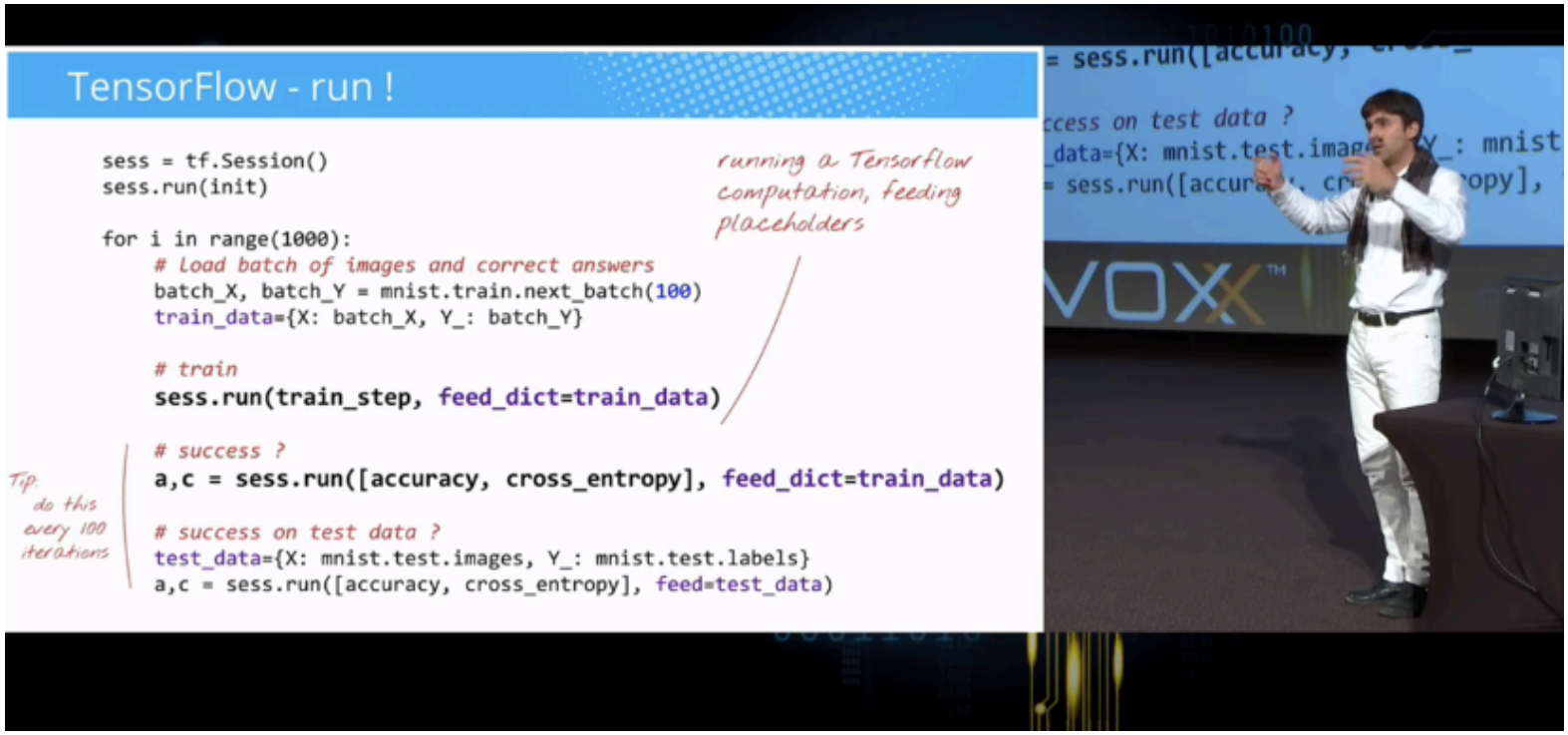
- Time commitment 2 hours:
  - Artificial Intelligence Complete Lectures (01-23)
  - The math in neural networks

http://artificialbrain.xyz/artificial-intelligence-complete-lectures-01-23/
(watch lecture 12a 12b)



Where to learn more

- Time commitment 3 hours:
  - Tensorflow and deep learning - without a PhD by Martin Görner
  - https://www.youtube.com/watch?v=vq2nnJ4g6N0

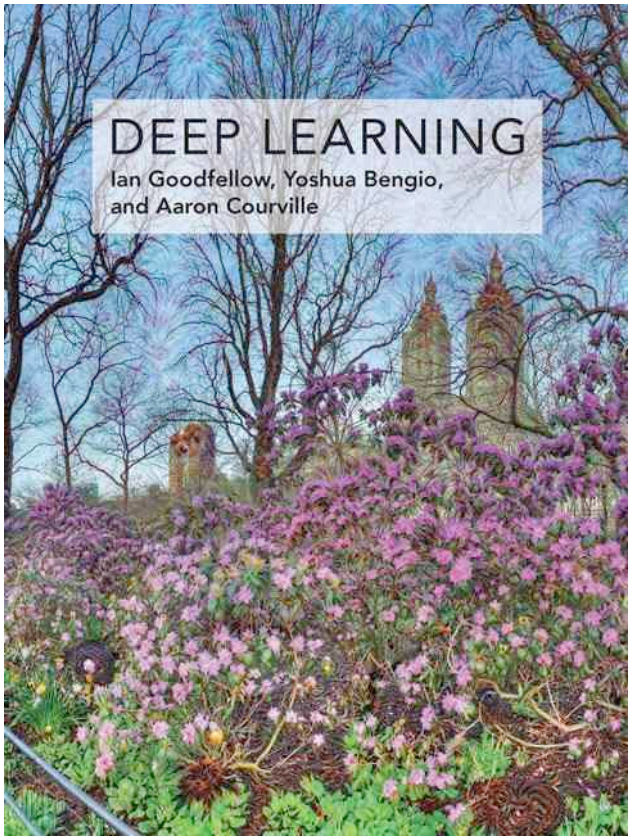


Where to learn more

- Deep learning school
  - 2 day workshop with many talks and tutorial (videos on Youtube)
  - https://www.bayareadlschool.org/
- Fei-Fei Li and Andrew Karpathy's Computer Vision class (Stanford cs231n 2015/2016)
  - Full course about deep learning (vision oriented)
  - http://cs231n.stanford.edu/
- Richard Socher's Deep Learning and Natural Language Processing (Standard cs224d)
  - Full course about deep learning (NLP oriented)
  - http://cs224d.stanford.edu/

Where to learn more

- Hinton Neural networks course
  - If you like lots of math and really want to learn the theory
  - https://www.coursera.org/learn/neural-networks
- http://www.deeplearningbook.org/
  - The only deep learning book (currently)
  - Good coverage, quite terse



Where to learn more

- For the latest developments subscribe to arxiv
  - Send email to cs@arxiv.org
  - Title: subscribe <name>
  - Content: add AI
- Reading group: TBA

Demo

- Jupyter http://jupyter.org/
- https://github.com/aymericdamien/TensorFlow-Examples
- Picasso https://medium.com/merantix/picasso-a-free-open-source-visualizer-for-cnns-d8ed3a35cfc5