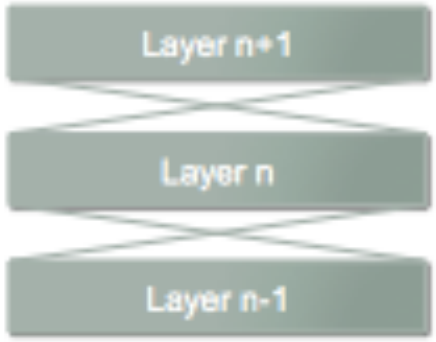
RECENT ADVANCES IN DNN

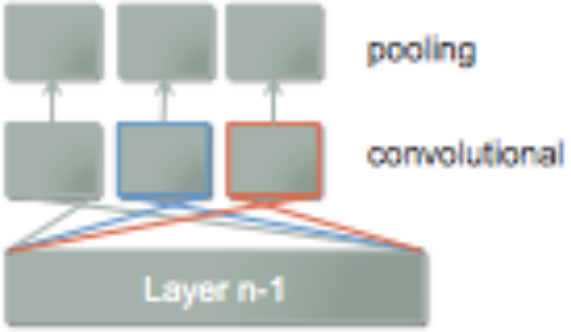
Neural networks

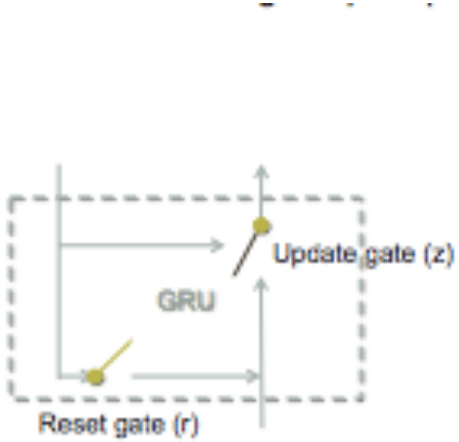
- Fully connected networks
  - Neuron
  - Non-linearity
  - Softmax layer
- DNN training
  - Loss function and regularization
  - SGD and backprop
  - Learning rate
  - Overfitting – dropout, batchnorm

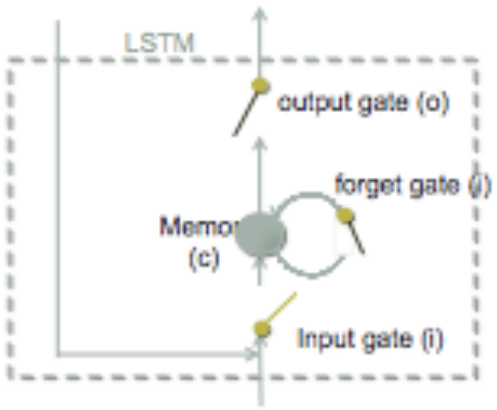
CNNs & RNNs

- CNNs
  - Matched filters
  - Convolution layer
  - Subsampling layer
    - Maxout, 1x1 convolution
- RNNs
  - Backpropagation Through time
  - GRUs (2 Gates)
  - LSTMs (3 Gates, explicite memory cell)









Loose Ends from HW4

- Embedding layer

- Categorical representation
  - One hot encoding

- Words in a vocabulary, characters in Thai language

Apple -> 1 -> [1, 0, 0, 0, …]

Bird -> 2 -> [0, 1, 0, 0, …]

Cat -> 3 -> [0, 0, 1, 0, …]

- Sparse representation
  - Spare means most dimension are zero

One hot encoding

- Sparse – but lots of dimension
  - Need to have many parameters at the input layer
  - Curse of dimensionality
- Does not represent meaning.

Apple -> 1 -> [1, 0, 0, 0, …]

Bird -> 2 -> [0, 1, 0, 0, …]

Cat -> 3 -> [0, 0, 1, 0, …]

|Apple – Bird| = |Bird – Cat|

Getting meaning into the feature vectors

- You can add back meanings by hand-crafted rules
- Old-school NLP is all about feature engineering
- HW4 example:
  - Cluster Numbers
  - Cluster letters
- Concatenate them
- 1 = [0 0 0 0 1 0 0 0, 1, 0]
- ก = [0 0 0 1 0 0 0 0, 0, 1]
- า = [1 0 0 0 0 0 0 0, 0, 2]
- Which rules to use?
  - Try as many as you can think of, and do feature selection or use models that can do feature selection

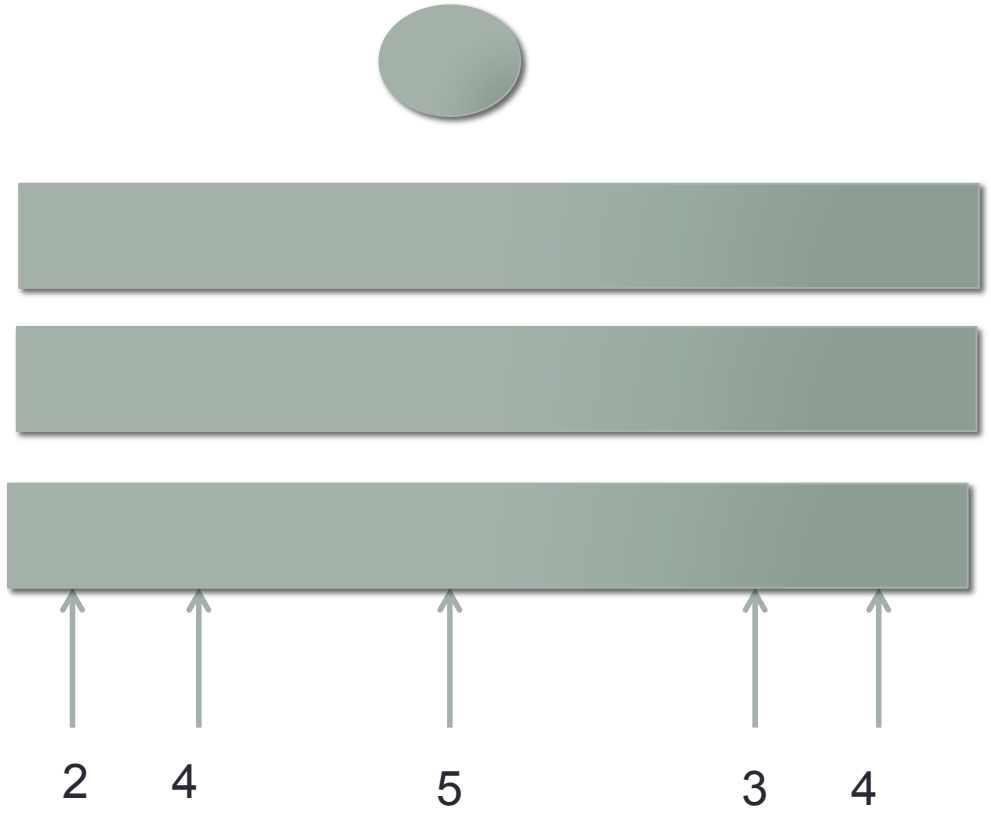
Dense representation

- In PCA, we learn we can encode data in a lower dimensionality space.
- We use a linear transform for PCA
  - v = Px, F: R^N -> R^M, where N>M
- Similarly, we can encode one hot vectors into lower dimensional dense vectors.

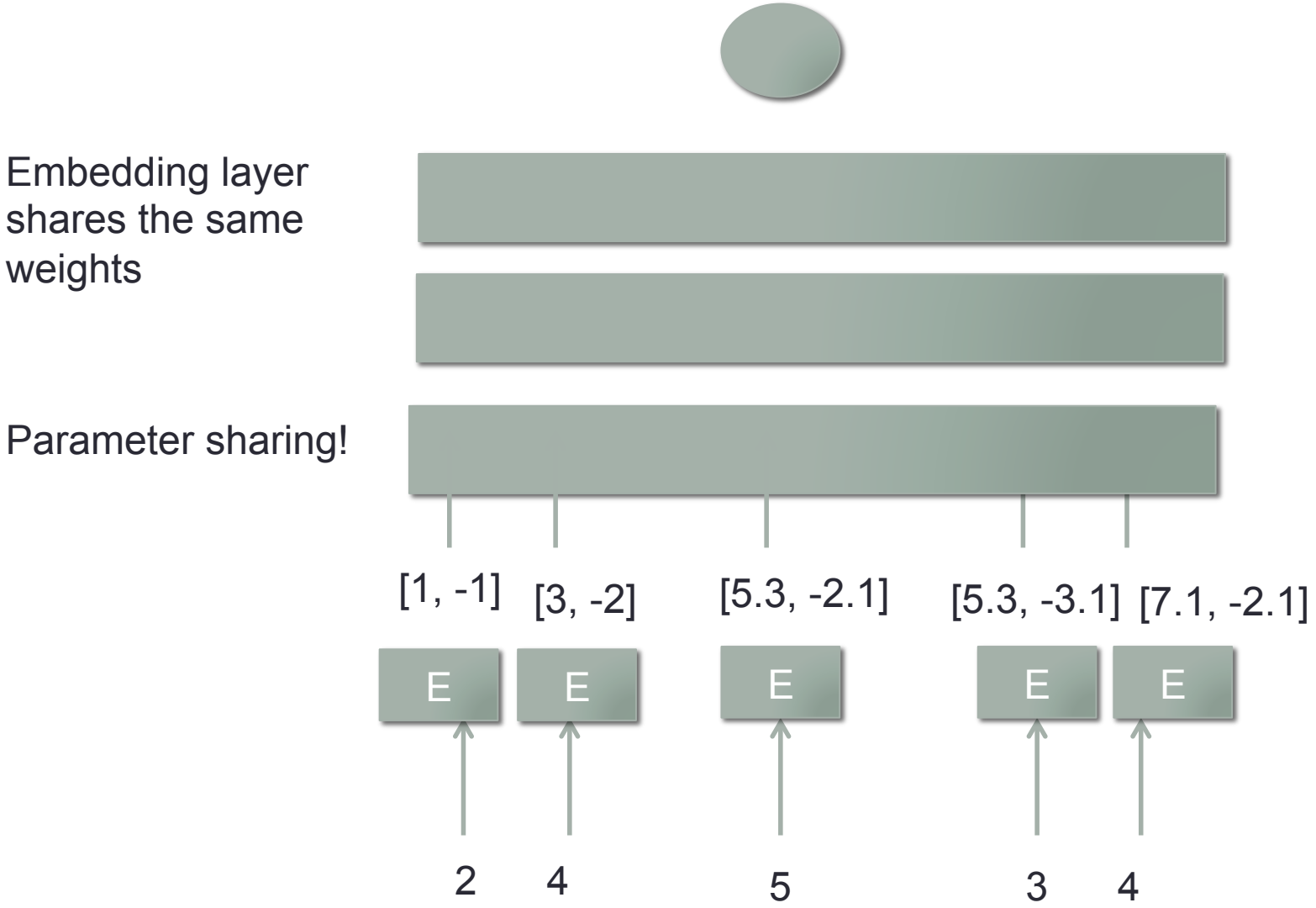
Apple -> 1 -> [1, 0, 0, 0, …] -> [2.3, 1.2]
Bird -> 2 -> [0, 1, 0, 0, …] -> [-1.0, 2.4]
Cat -> 3 -> [0, 0, 1, 0, …] -> [-3.0, 4.0]

- In HW4, we can do this by using an embedding layer

HW4 fully connected



HW4 fully connected with embedding



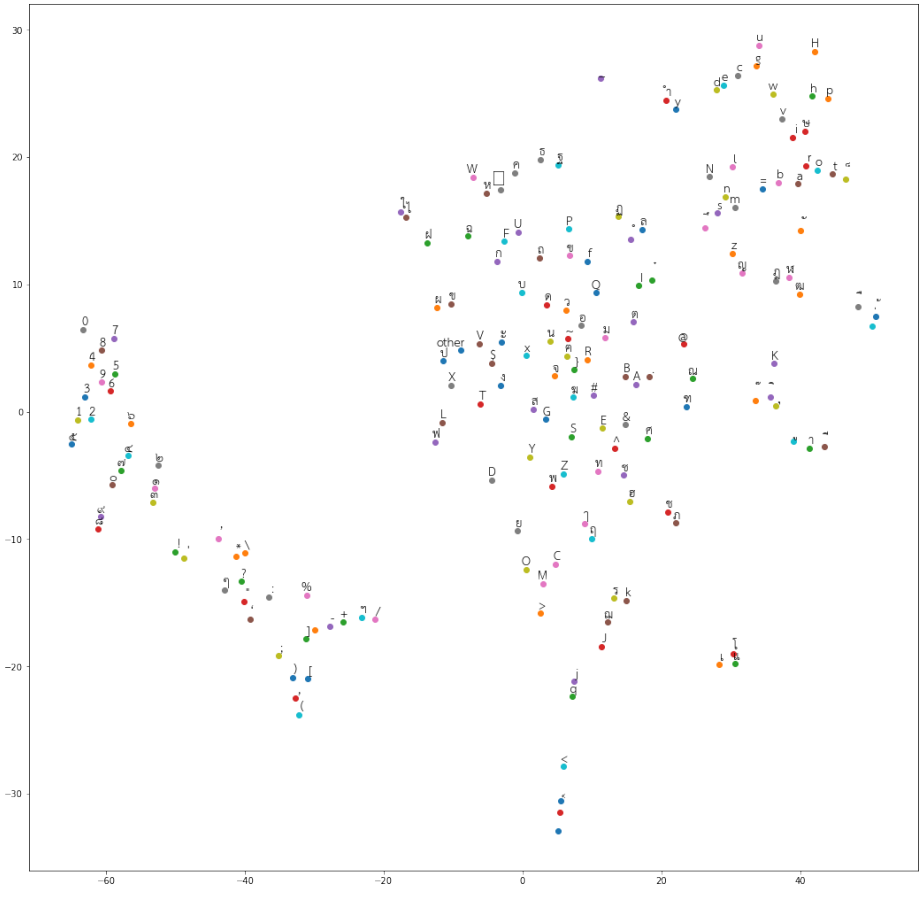
Embedding vs PCA

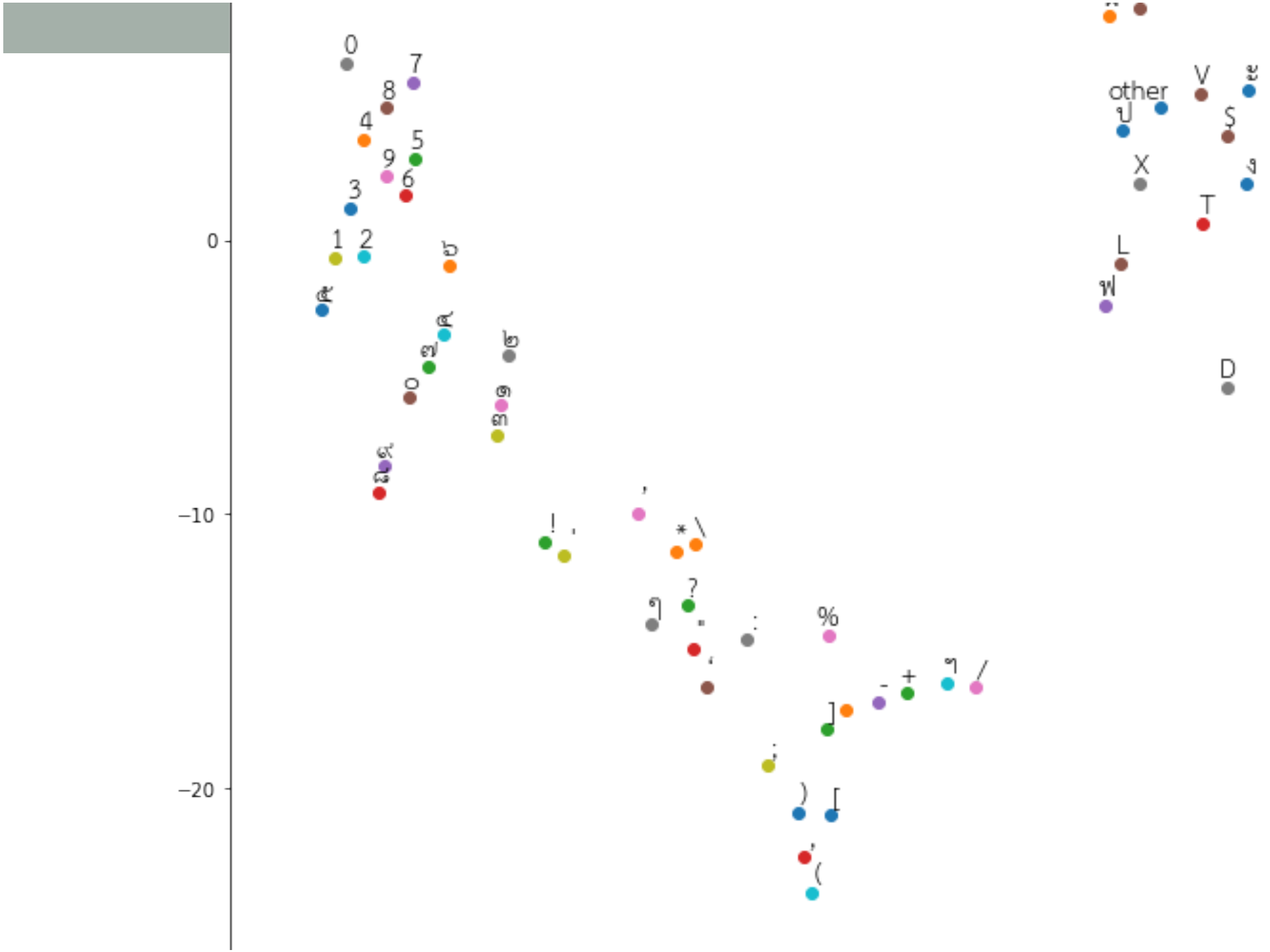
- PCA – unsupervised method
- Embedding – train supervised with the task
  - Embedding should be superior with the task
- PCA – linear
- Embedding – potentially non-linear
  - Should be more powerful

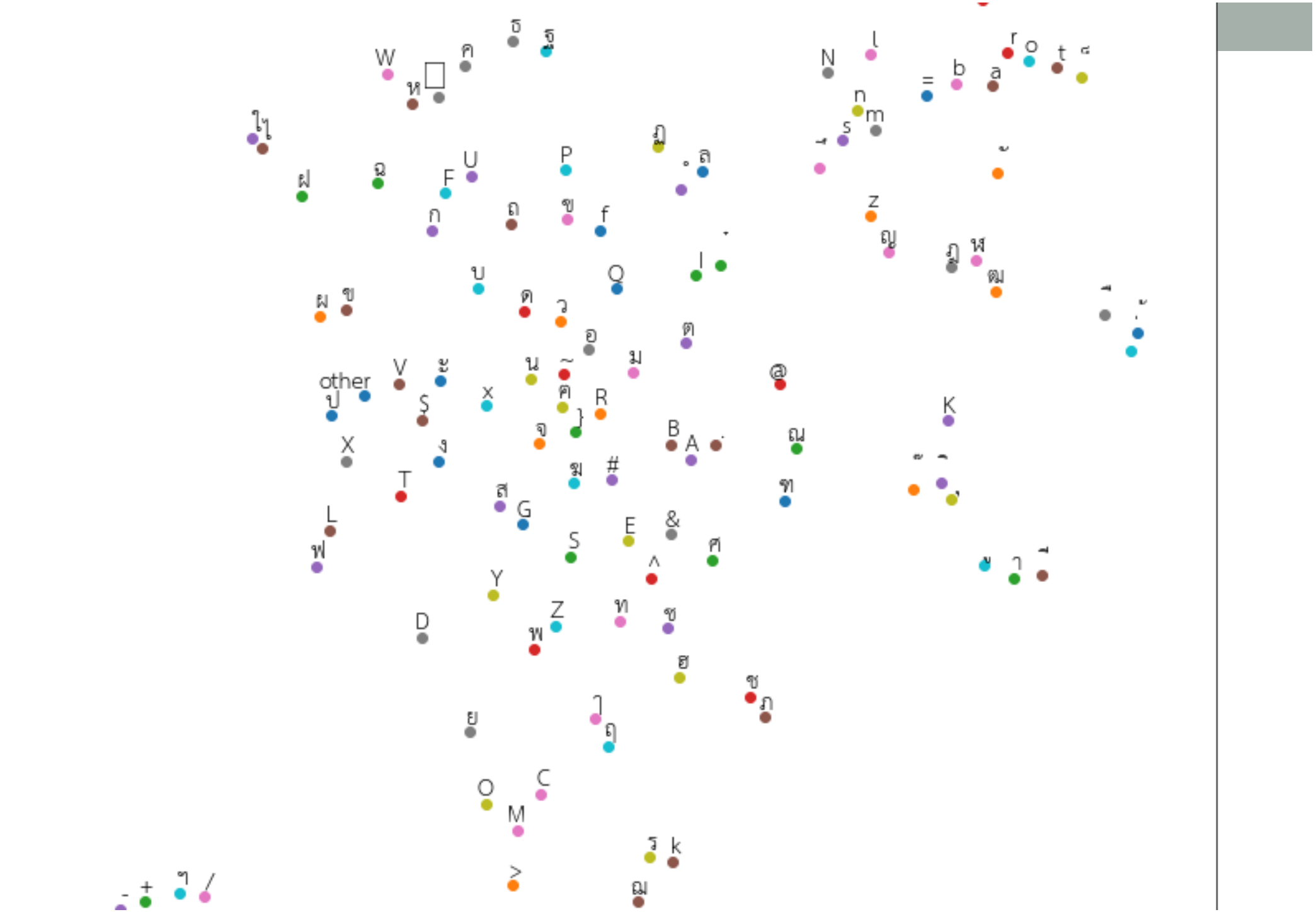
- You can learn an embedding on one task (with lots of data) and use it on another task
  - Embedding learns meaningful feature representations

Embedding and meaning (semantics)

- Meaning is inferred from the task
- Embedding of 32 dimensions -> t-SNE into 2 dimension for visualization
- Automatically!

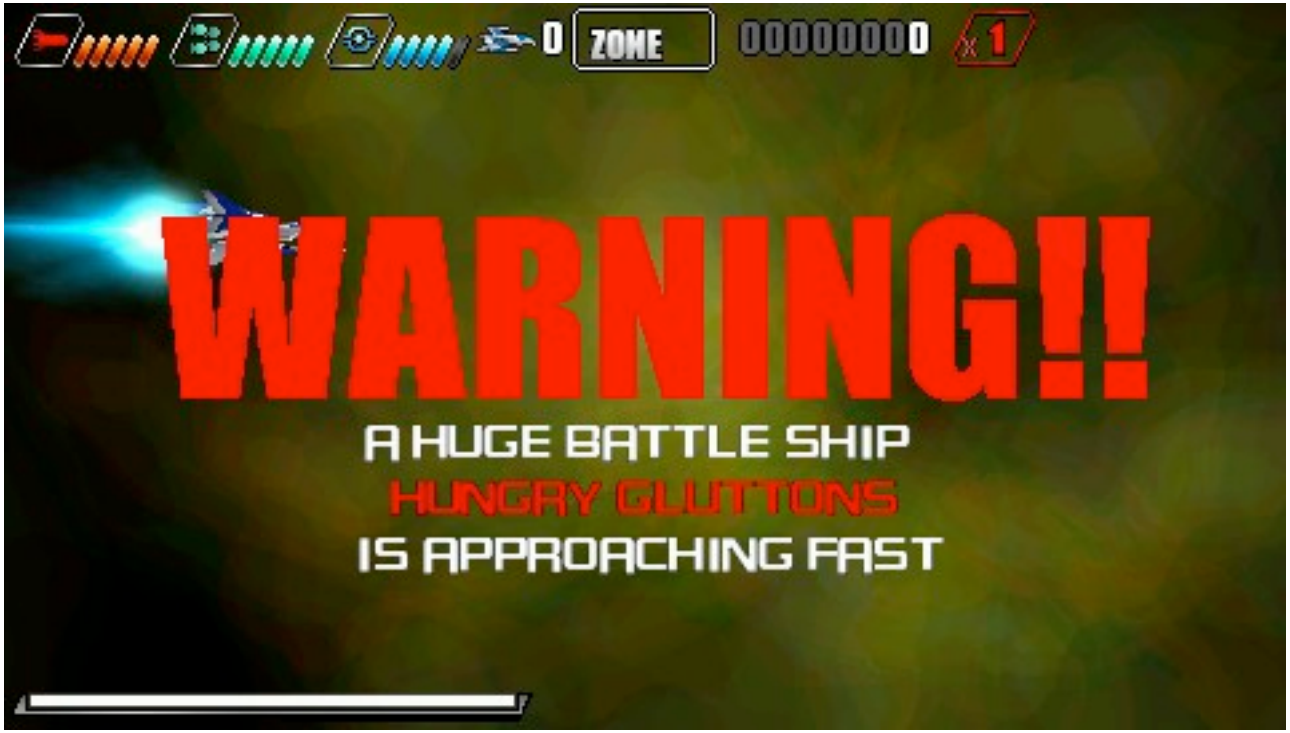



0
−10
−20
other


other

other

- You are not expected to understand any of the following parts in this lecture.
- This lecture serves as an overview of things going on with deep learning.
- If you are curious about any terms that gets thrown out there, see the relevant papers.



Misc topics

- Residual networks
- Encoder-Decoder
  - Information bottleneck
  - Word2Vec
- Transfer Learning, Multi-task models, Multi-models
- Attention modeling
- Faster R-CNN
- Alternatives to Backprop
  - 2nd order methods
  - Synthetic gradients
  - Genetic Algo

- Reinforcement Learning <- Next lecture!
- GAN <- Next Next lecture!
  - Adversarial inputs

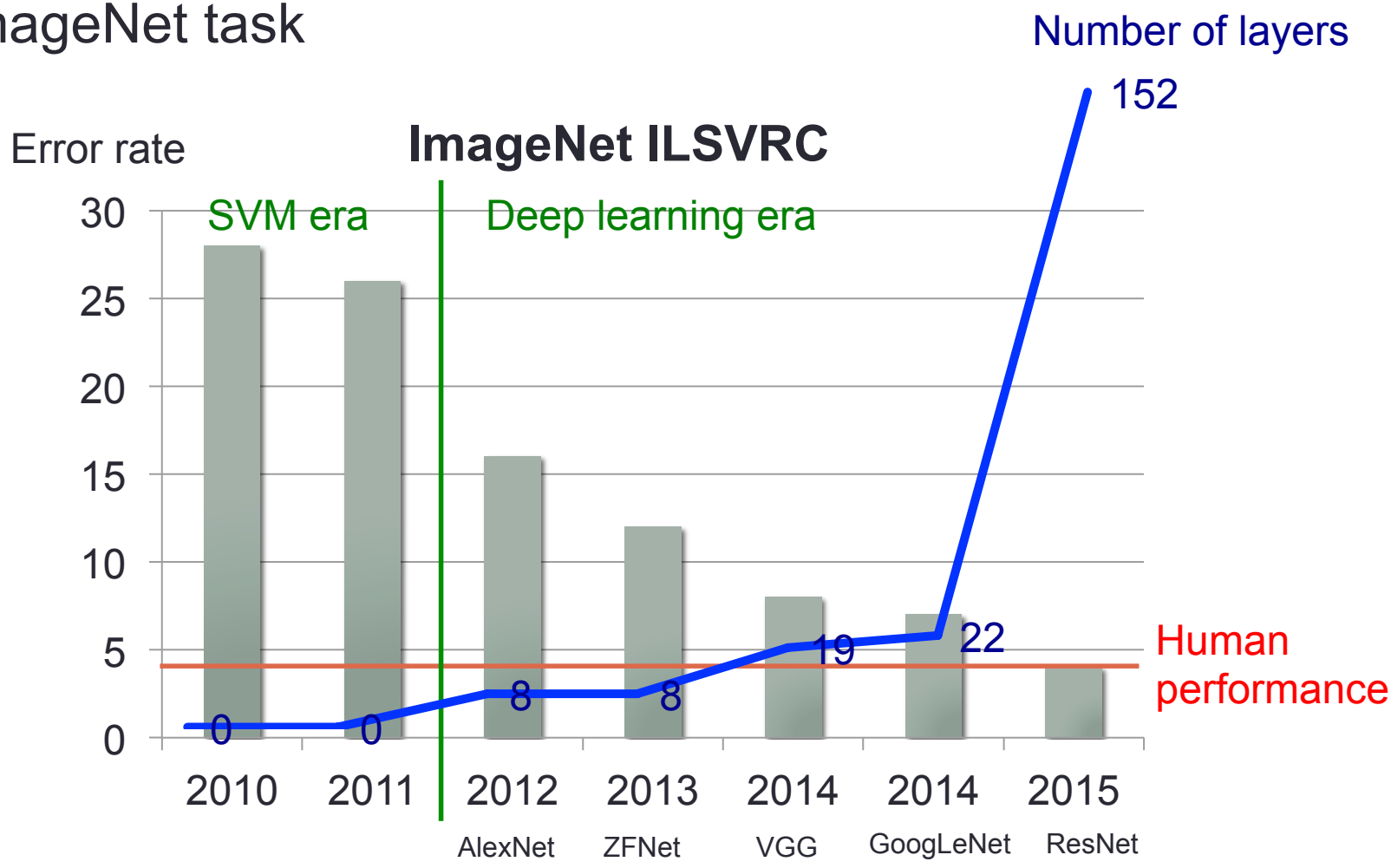
Residual networks

- Super deep networks

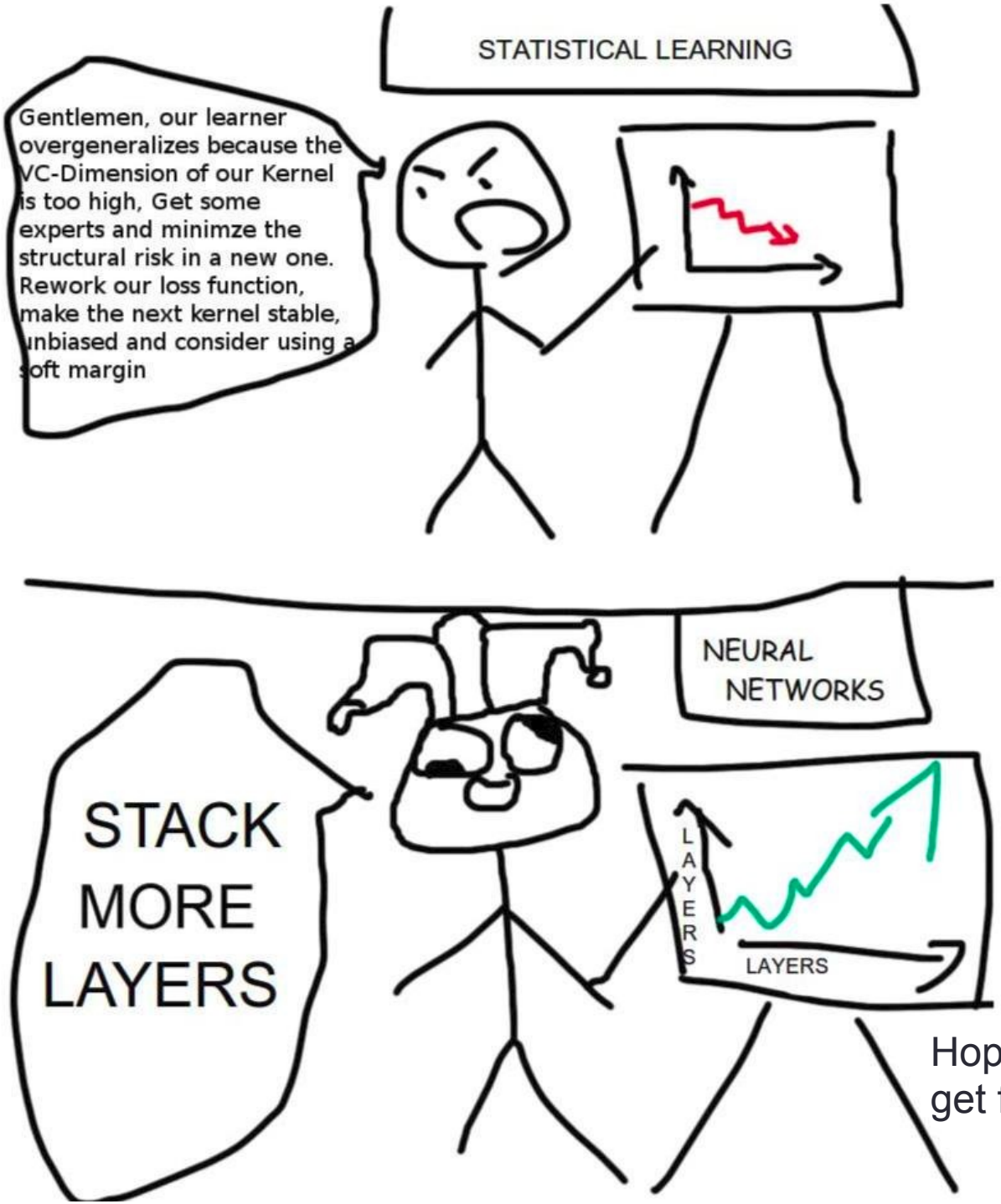
https://arxiv.org/pdf/1512.03385.pdf

Wider and deeper networks

- ImageNet task



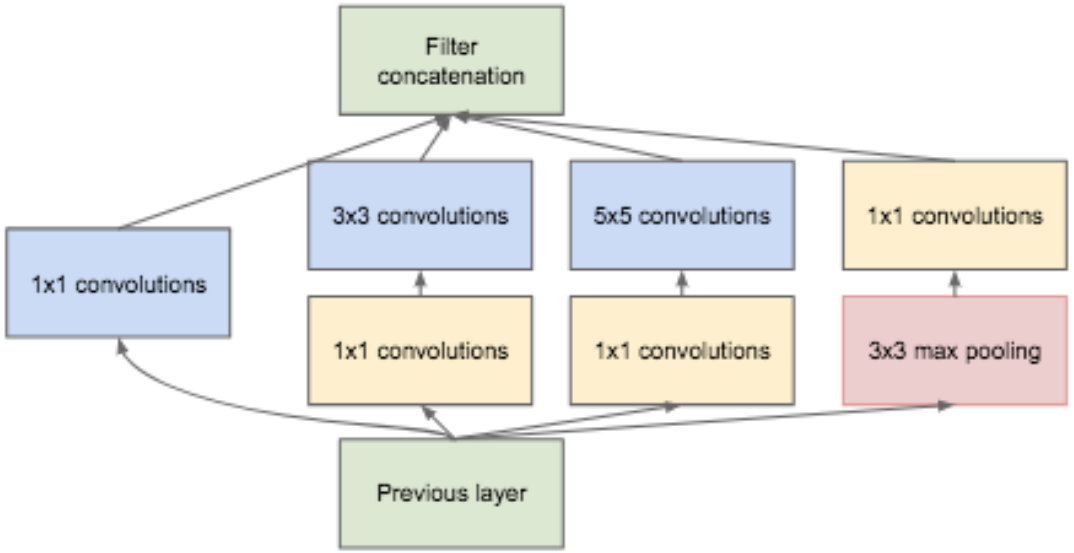
Olga Russakovsky, ImageNet Large Scale Visual Recognition Challenge, 2014 https://arxiv.org/abs/1409.0575



Hopefully this won't be all you get from this class

ImageNET prize winner

- AlexNet – first deep model based on LeNET (simple CNN from 1990s)
- ZFNet – AlexNET just deeper, and better tuned hyperparameters
- GoogLeNet (Inception Model)

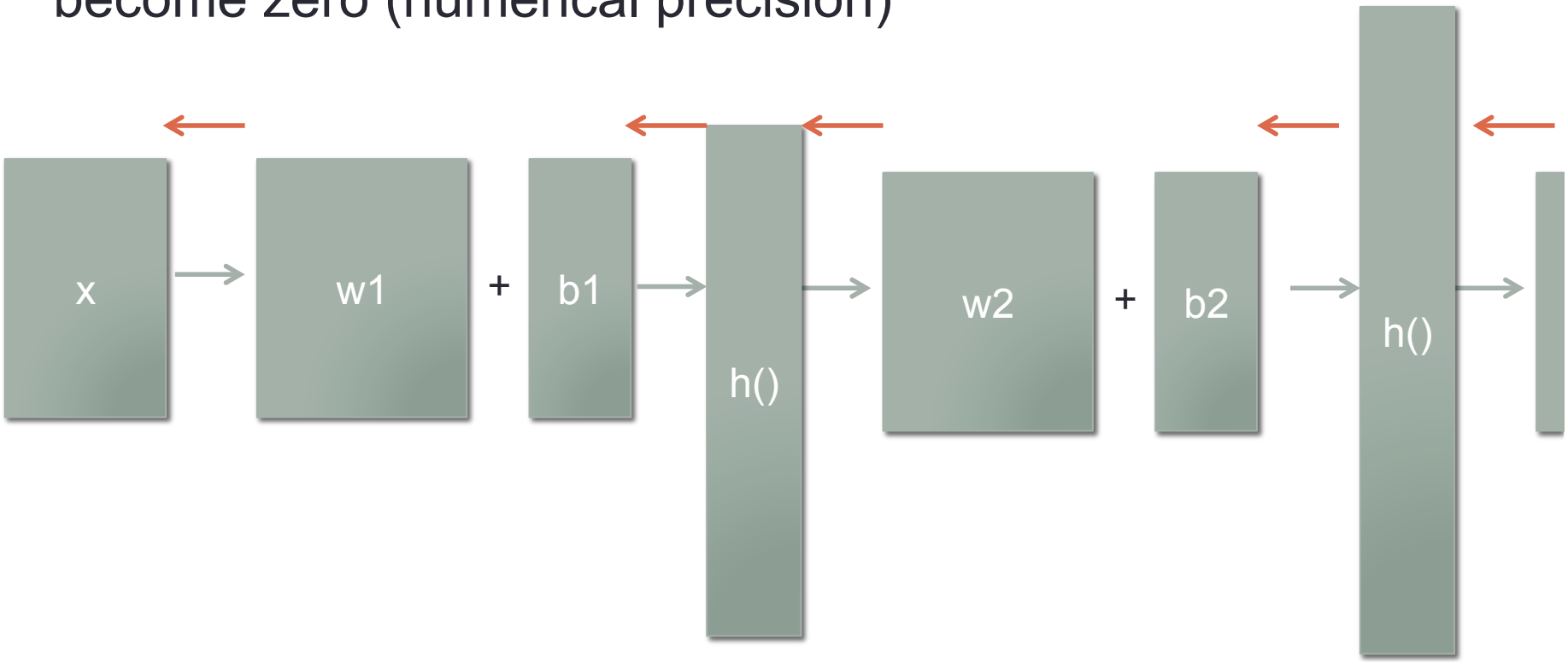


ImageNET prize winner 2

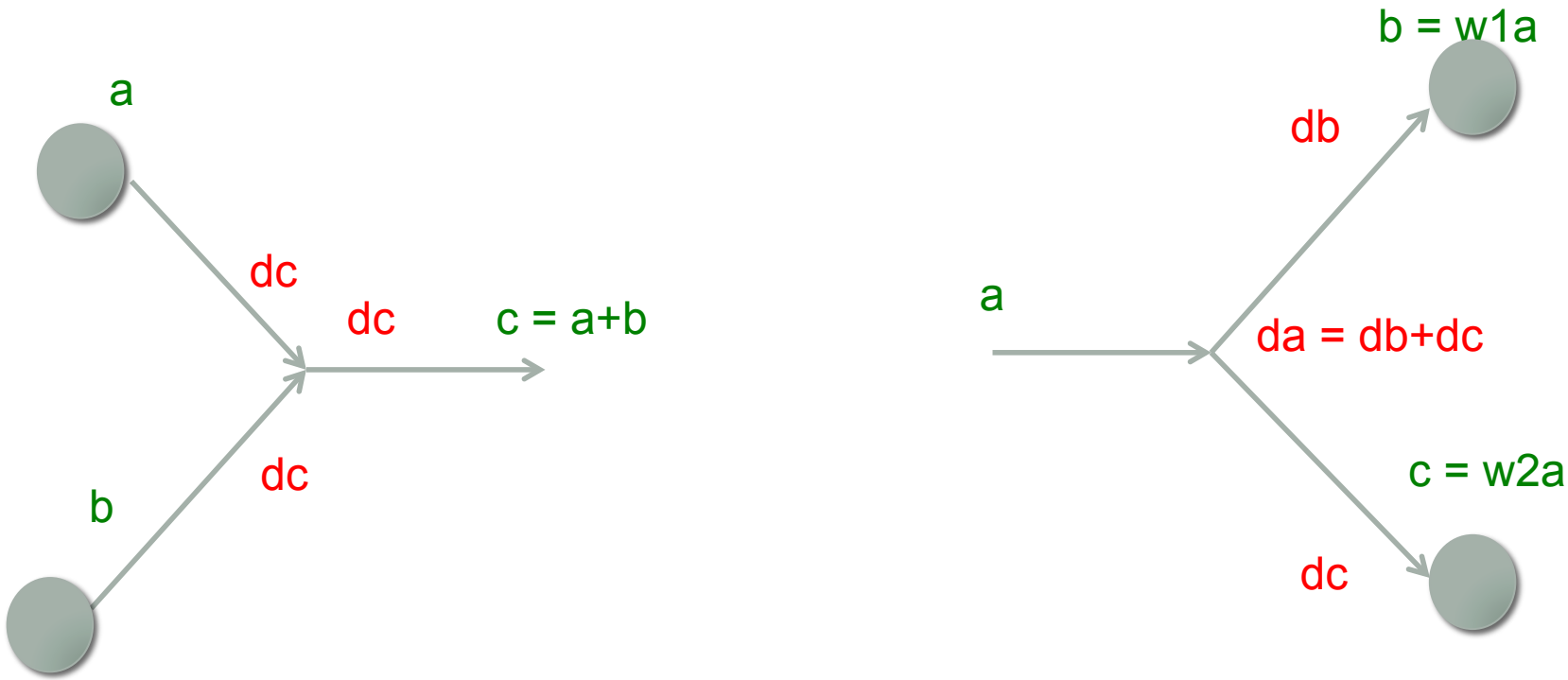
- VGGNet - just a bigger network. Notable mention because model available
- ResNet <- We are here

Vanishing gradients

- Lower layers are hard to train because the gradients become zero (numerical precision)



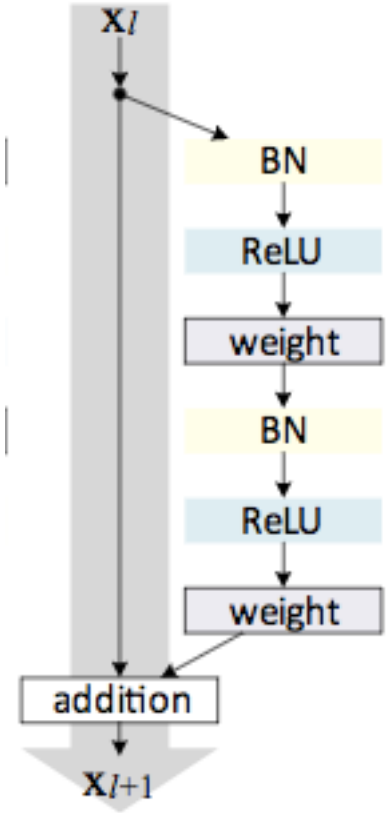
Gradient flow at forks



Forward and backward pass acts differently at forks
When the fork fans out, the gradient adds in the backward pass

Residual networks

- A fork connection (identity) that keep gradients > 1



https://arxiv.org/pdf/1603.05027.pdf

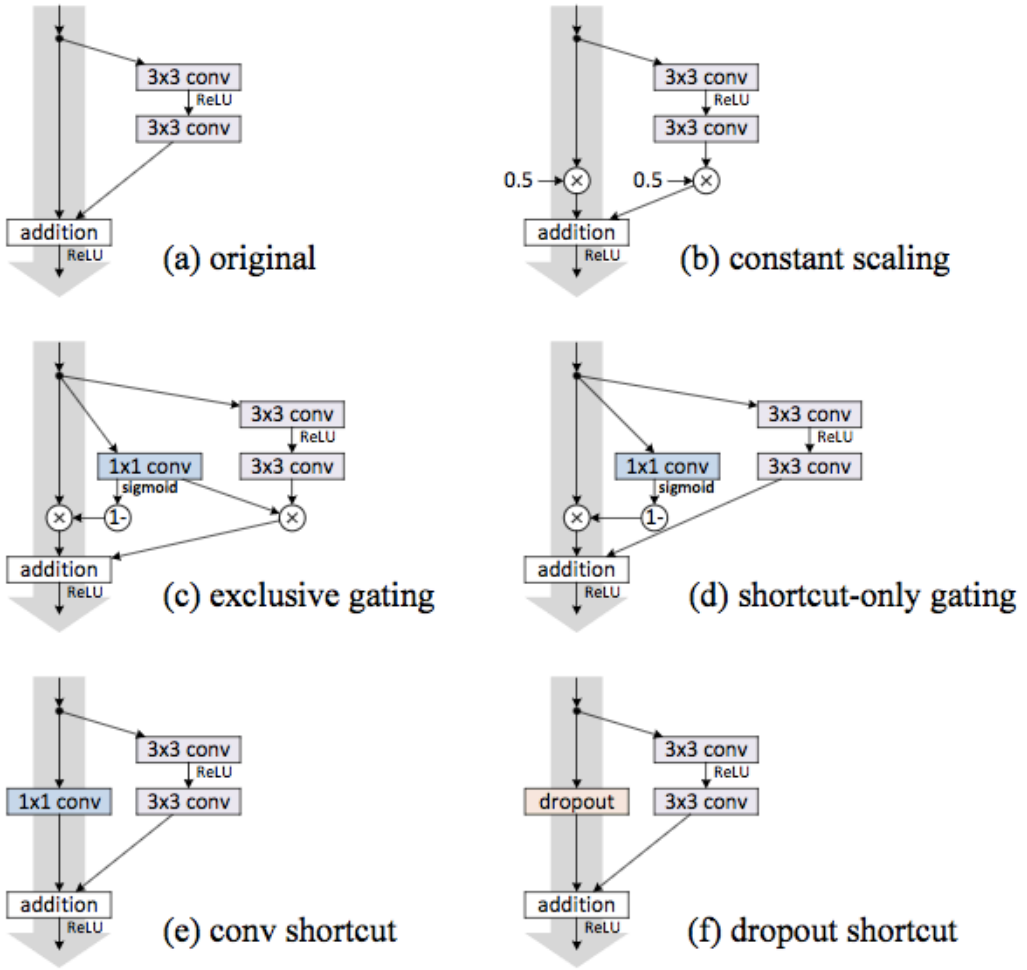
method			error (%)
Maxout [10]			9.38
NIN [25]			8.81
DSN [24]			8.22
	# layers	# params	
FitNet [35]	19	2.5M	8.39
Highway [42, 43]	19	2.3M	7.54 (7.72±0.16)
Highway [42, 43]	32	1.25M	8.80
ResNet	20	0.27M	8.75
ResNet	32	0.46M	7.51
ResNet	44	0.66M	7.17
ResNet	56	0.85M	6.97
ResNet	110	1.7M	**6.43** (6.61±0.16)
ResNet	1202	19.4M	7.93

Table 6. Classification error on the **CIFAR-10** test set. All methods are with data augmentation. For ResNet-110, we run it 5 times and show "best (mean±std)" as in [43].

https://arxiv.org/pdf/1512.03385.pdf

Highway networks

- A generalization of ResNets
  - The identity connection is no longer identity
- People found identity connections still the best



https://arxiv.org/pdf/1603.05027.pdf

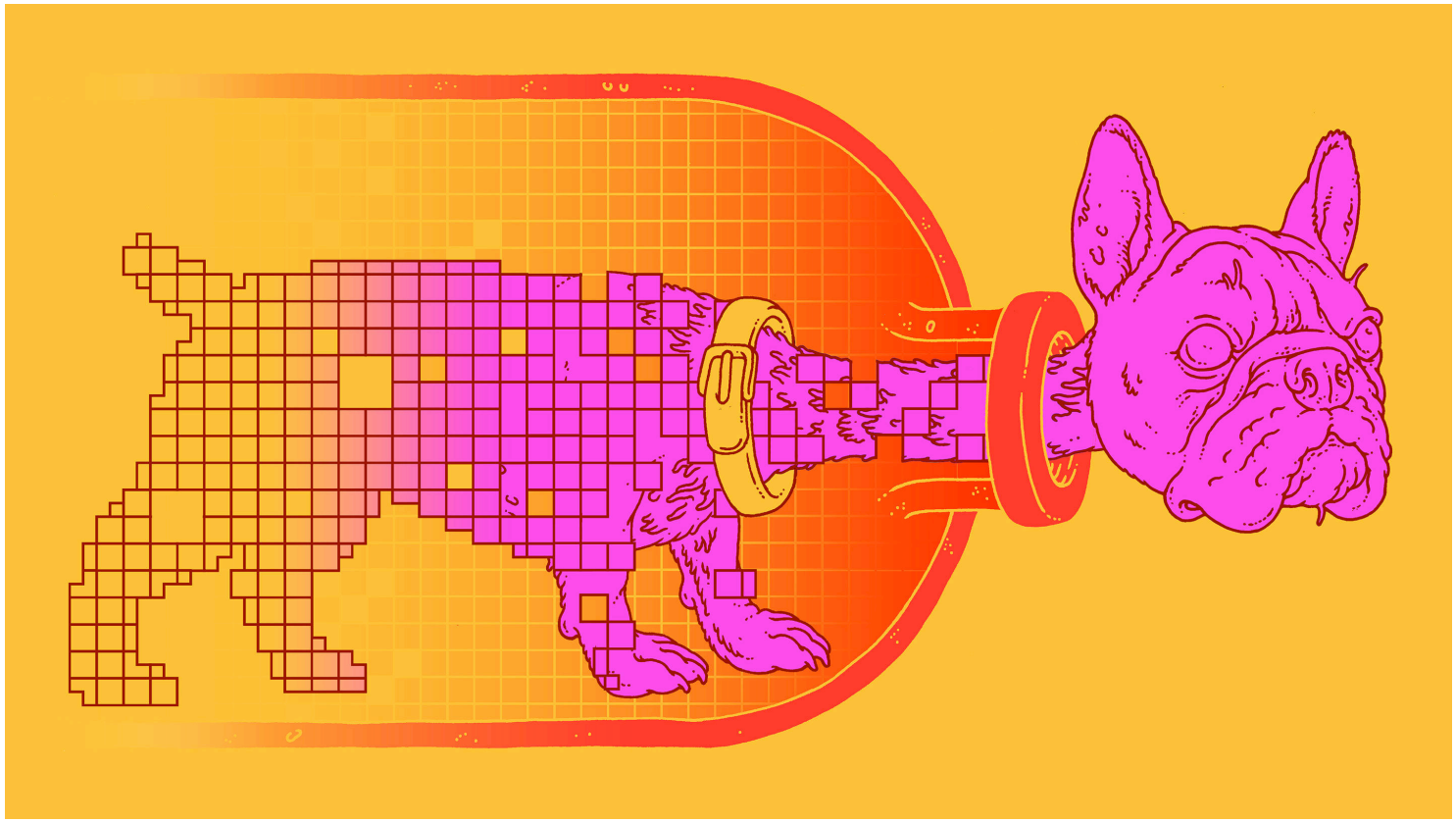
Misc topics

- Residual networks
- Encoder-Decoder
  - Information bottleneck
  - Word2Vec
- Transfer Learning, Multi-task models, Multi-models
- Attention modeling
- Faster R-CNN
- Alternatives to Backprop
  - 2nd order methods
  - Synthetic gradients
  - Genetic Algo

- Reinforcement Learning <- Next lecture!
- GAN <- Next Next lecture!
  - Adversarial inputs

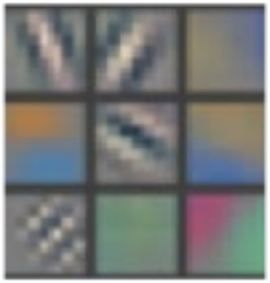
Information bottlenecks

- PCA, and embeddings try to explain the input in a lower dimensional space.

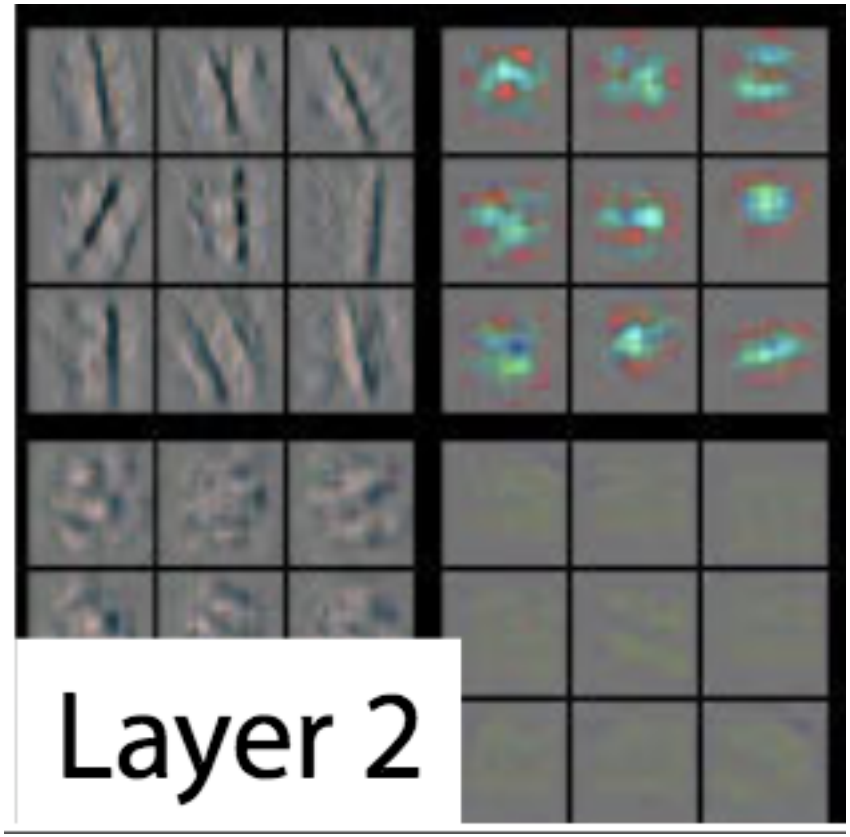


https://www.quantamagazine.org/new-theory-cracks-open-the-black-box-of-deep-learning-20170921/

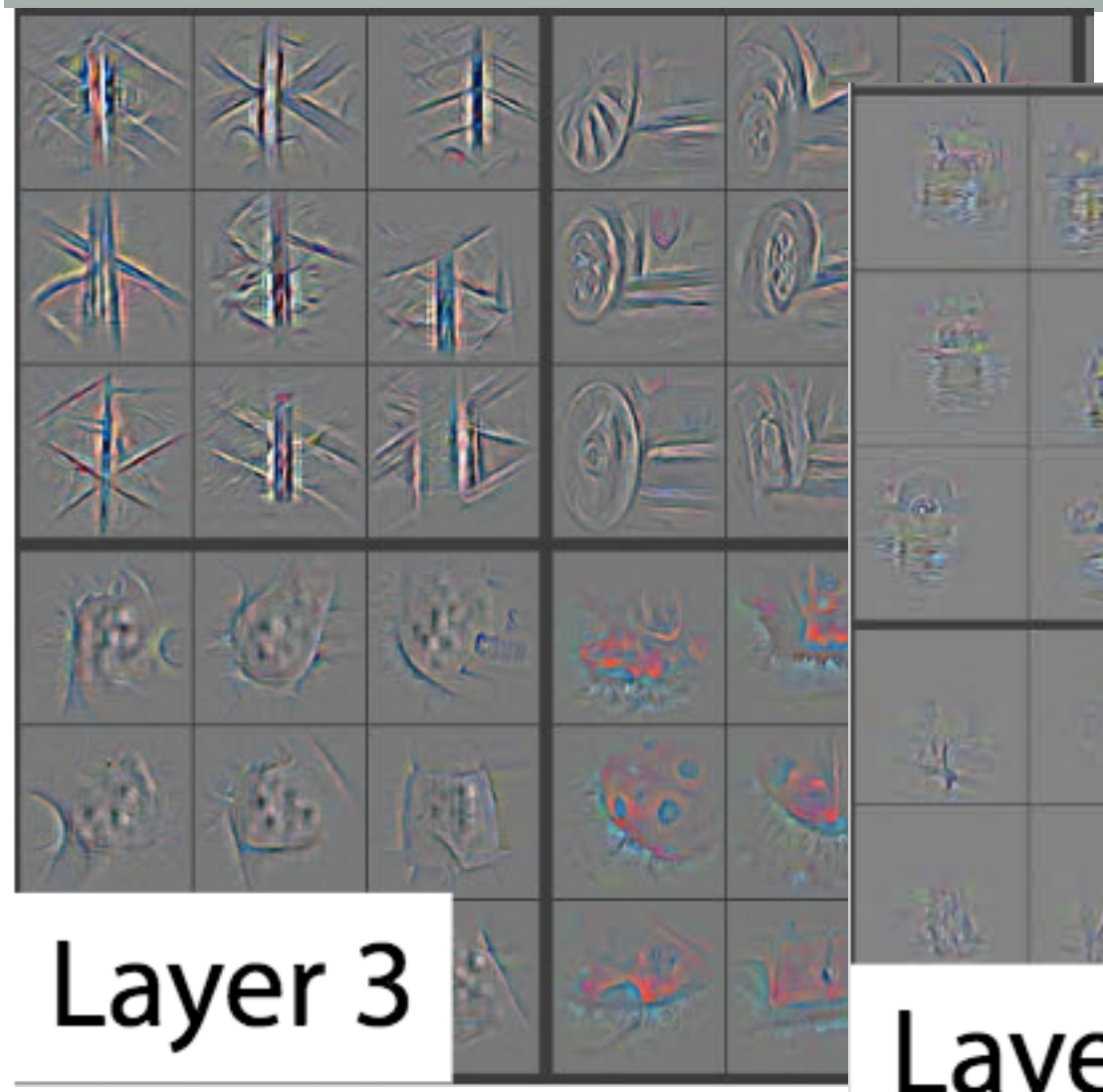
Visualizing neural networks

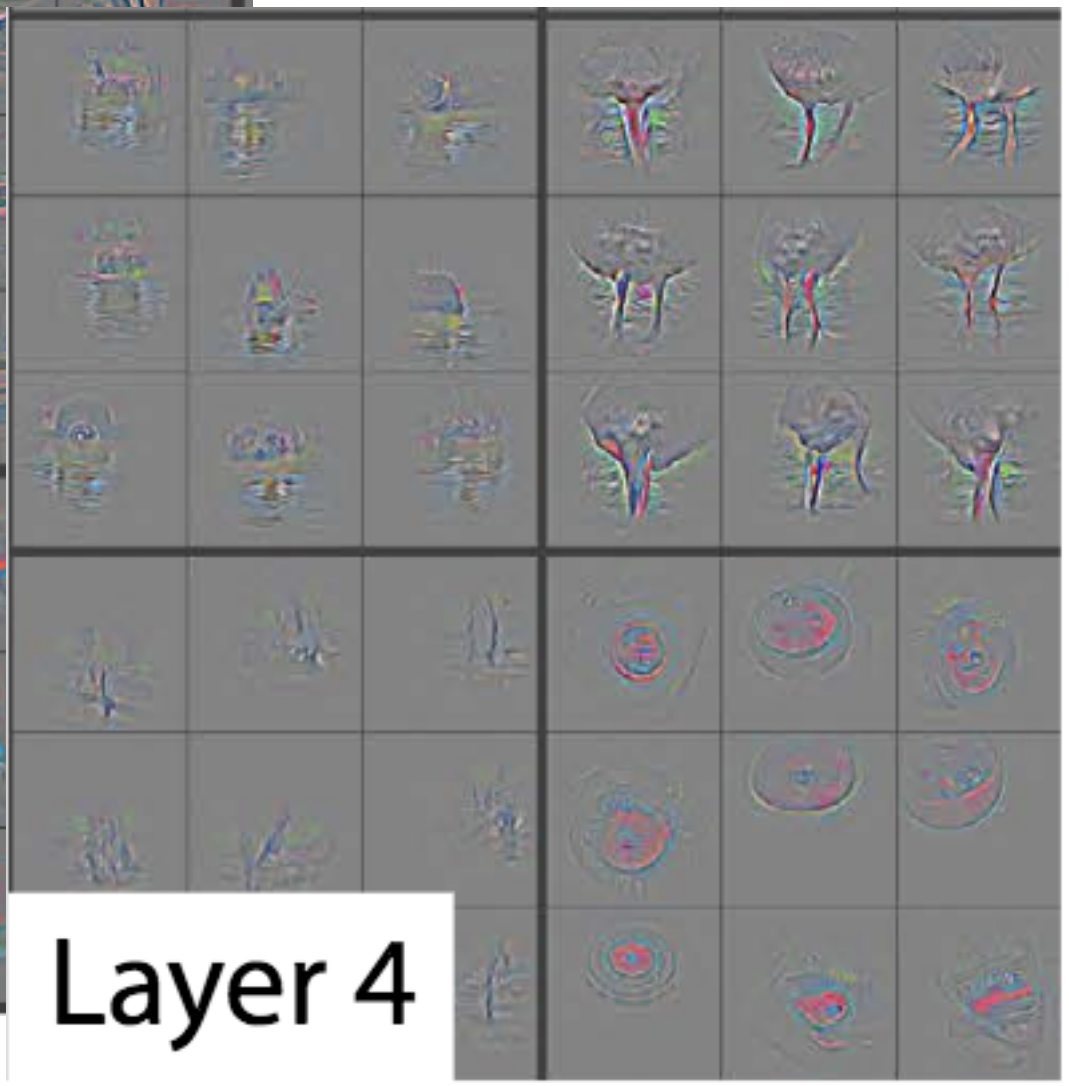


Layer 1



https://arxiv.org/pdf/1311.2901.pdf


Layer 3


Layer 4

Some information theory basics

- The entropy H(X)
  - Amount of information needed in order to determine a sample X
  - Or randomness of X

$$-\sum_{i=1}^{n} \mathrm{P}(x_i) \log_b \mathrm{P}(x_i)$$

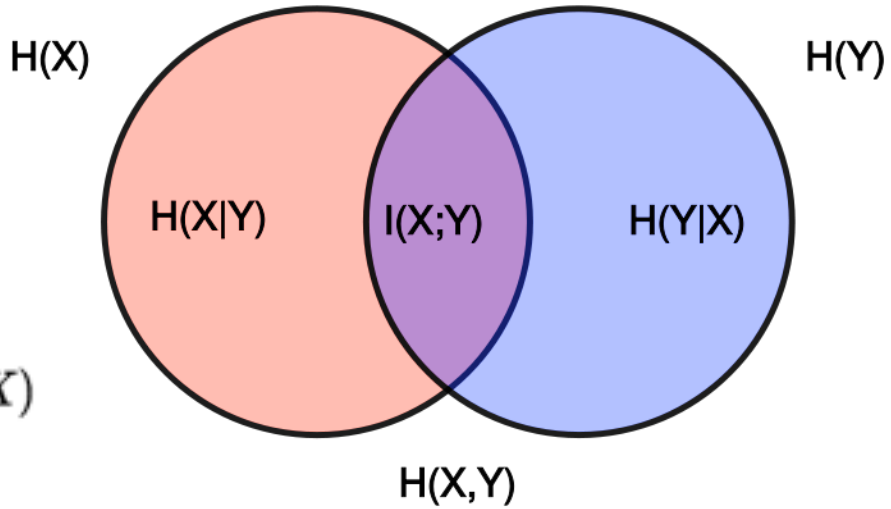
- Example: Four possible letters ABCT
  - If I want to send a string CAT, how many bits do I need?
    - Assume each letter is equally likely, I need 2 bits for each letter
    - Need to send 2x3 = 6 bits
  - H(string of length 3) = 6

Some information theory basics

- The mutual information
  - How one random variables tell us about the other, and vice versa
    - Independence means the mutual information is 0
  - For X Y random variables

$$I(X;Y) = \sum_{y \in Y} \sum_{x \in X} p(x,y) \log \left(\frac{p(x,y)}{p(x)\,p(y)} \right)$$

$$\begin{aligned} I(X;Y) &\equiv \mathrm{H}(X) - \mathrm{H}(X|Y) \\ &\equiv \mathrm{H}(Y) - \mathrm{H}(Y|X) \\ &\equiv \mathrm{H}(X) + \mathrm{H}(Y) - \mathrm{H}(X,Y) \\ &\equiv \mathrm{H}(X,Y) - \mathrm{H}(X|Y) - \mathrm{H}(Y|X) \end{aligned}$$



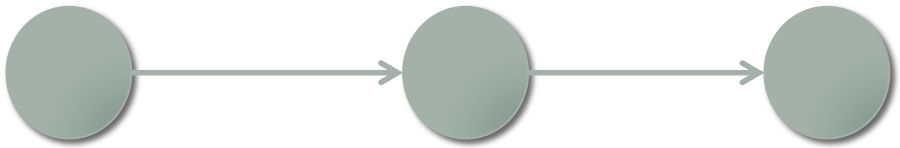
https://en.wikipedia.org/wiki/Mutual_information

Markov chain (HW2… if you can still remember)

Markov property: current event only depends on the immediate past given the whole history

day1 day2 day3 day4

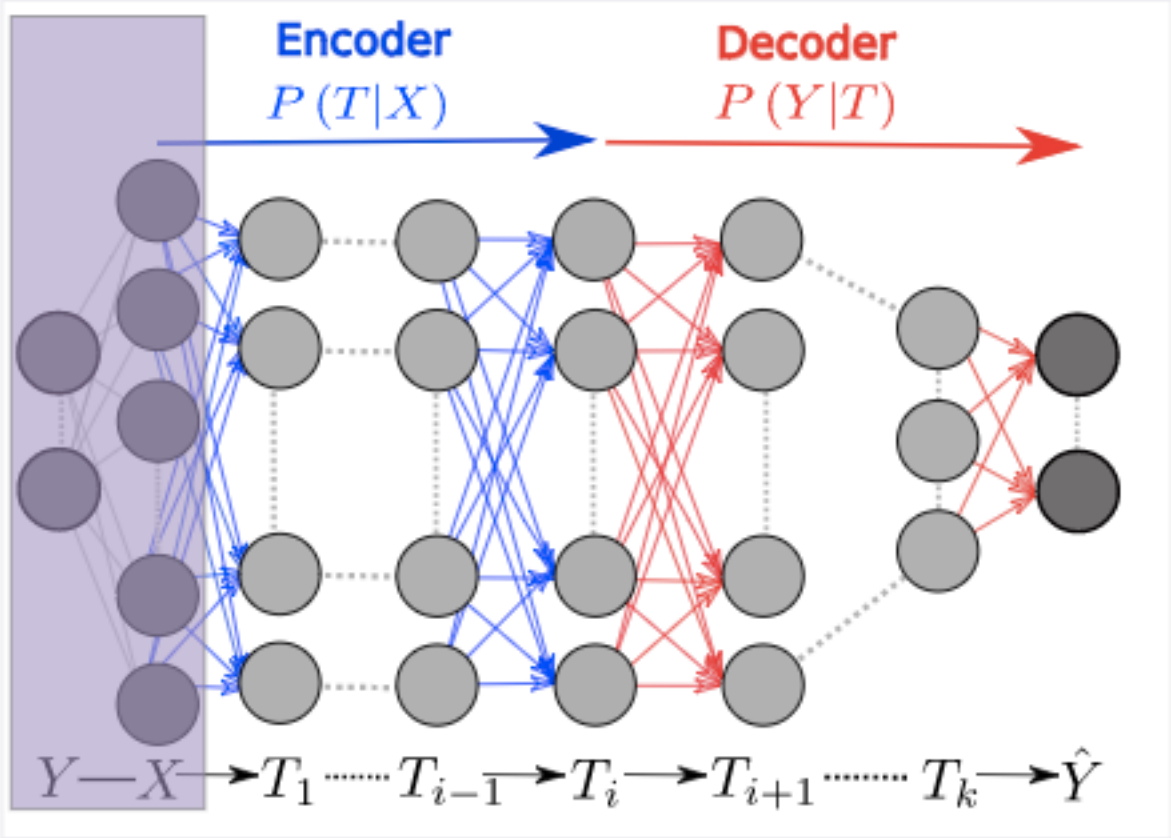
X -> Y -> Z



X Y Z

Data processing inequality
I(X;Y) >= I(X;Z)

Neural networks as a markov chain



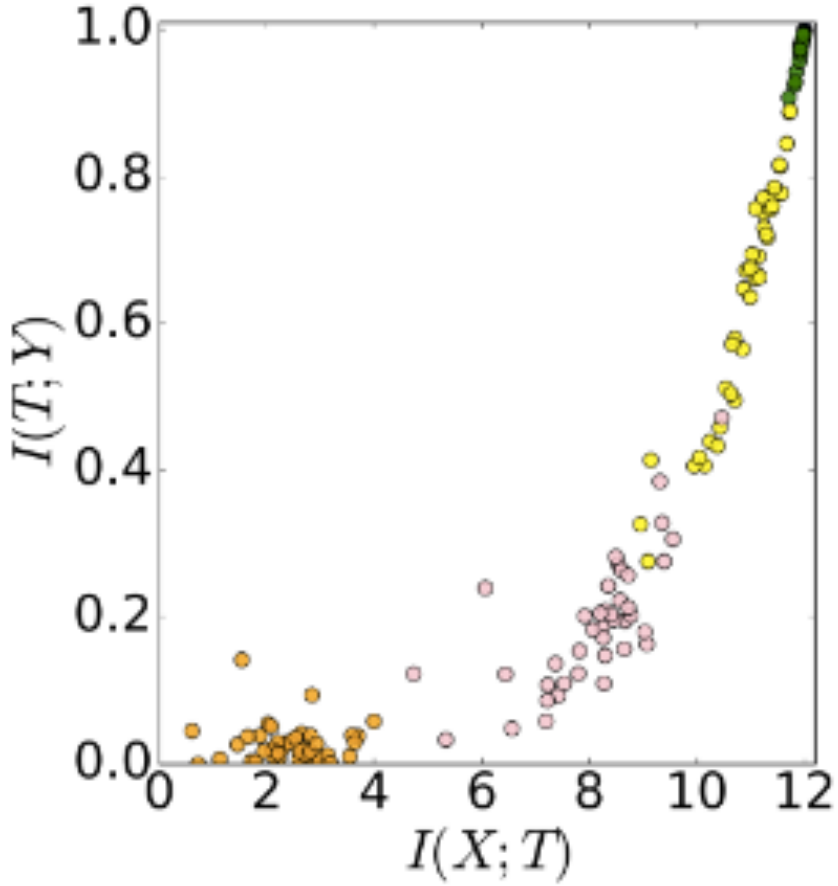
$$I(X;Y) \geq I(T_1;Y) \geq I(T_2;Y) \geq \ldots \geq I(T_k;Y) \geq I(\hat{Y};Y)$$

$$H(X) \geq I(X;T_1) \geq I(X;T_2) \geq \ldots \geq I(X;T_k) \geq I(X;\hat{Y})$$

https://arxiv.org/pdf/1703.00810.pdf

Snapshots – Before training

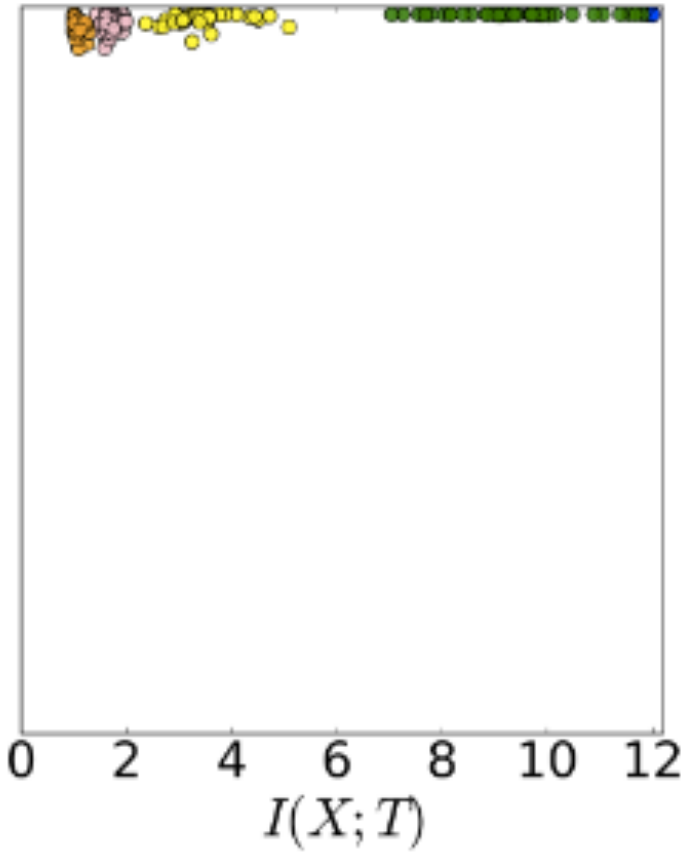
Each point is the mutual information of the layer and X or Y
Color is the layer depth



Snapshots – After training

Higher layers loses information about X but retain information about Y

Dimensionality reduction while still keeping relevant info about the task

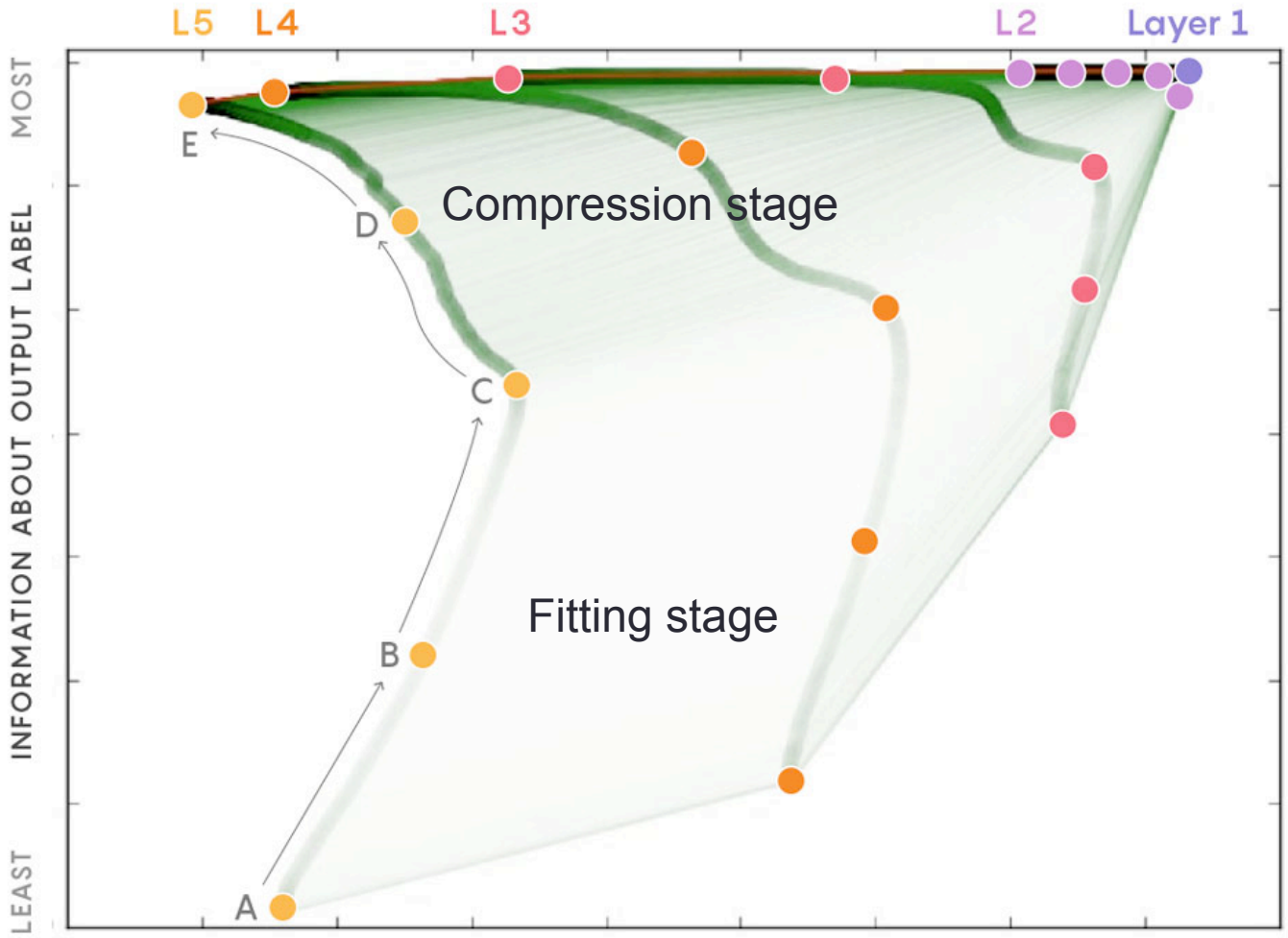


Video

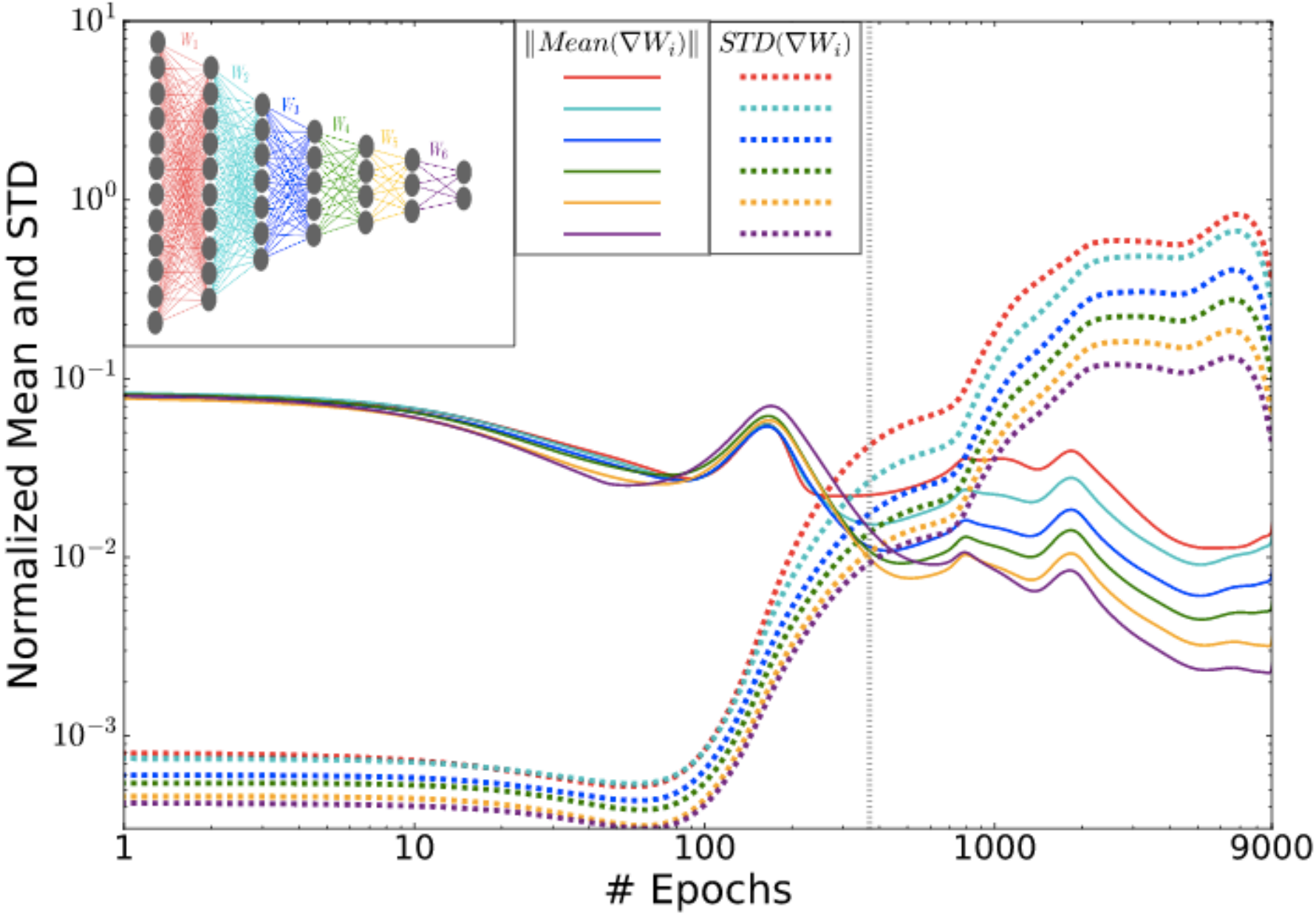
- https://youtu.be/bLqJHjXihK8?t=13m47s

Inside Deep Learning

New experiments reveal how deep neural networks evolve as they learn.



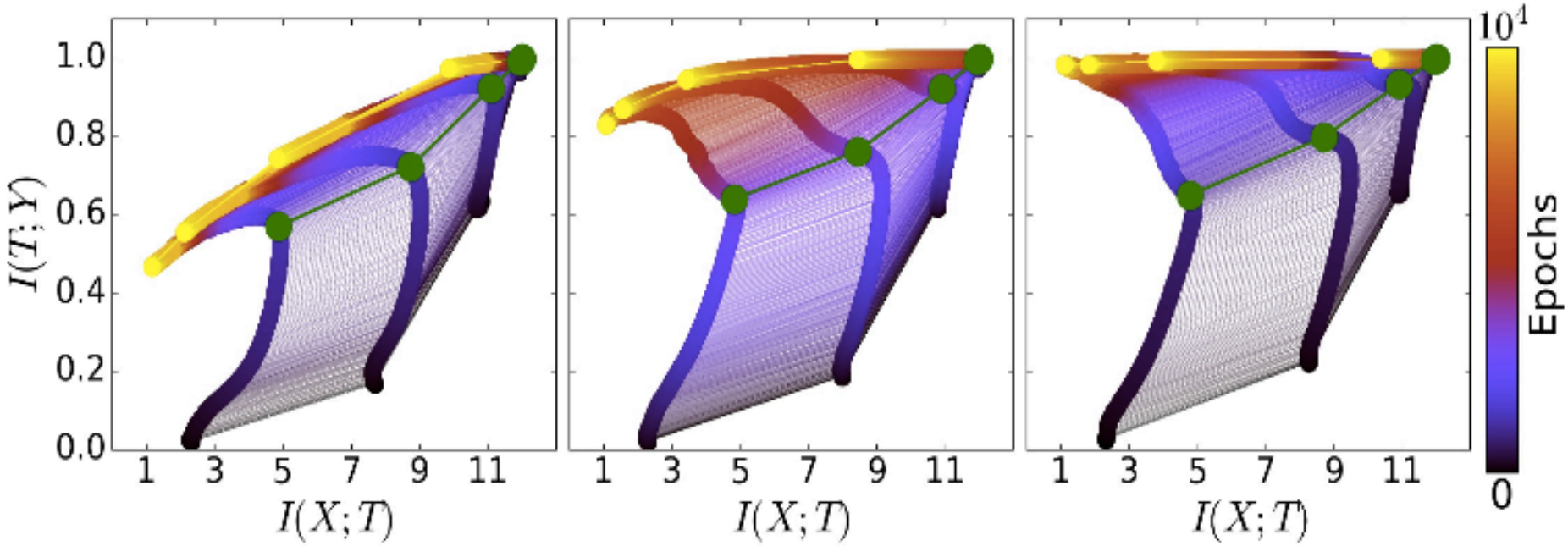
https://www.quantamagazine.org/new-theory-cracks-open-the-black-box-of-deep-learning-20170921


10^1
10^0
10^{-1}
10^{-2}
10^{-3}
Normalized Mean and STD
1
10
100
1000
9000
Epochs
$\|Mean(\nabla W_i)\|$
$STD(\nabla W_i)$
W_1
W_2
W_3
W_4
W_5
W_6

Compression and amount of training

- This compression alleviates overfitting
  - Usually we say more parameters -> easy to overfit
    - We also say more data -> harder to overfit
  - Why neural network doesn't overfit (as much) with millions of parameters?
  - Neural network compression by k bits
    - As if you gain a factor of 2^k training examples

Overfitting



5% training data

45% training data

85% training data

Information bottleneck summary

- Neural network encodes information about the output at the upper layers and about the input in the lower layers.

- Is this useful?
  - It tells us what happens during training.
  - Someone used this to build better networks

DEEP VARIATIONAL INFORMATION BOTTLENECK

- X input, Z hidden layer, Y output
- In information bottleneck, it tries to maximizes
  - I(Z;Y) – BI(Z;X)
  - B is some weight factor
  - Encodes X while keeping Z info
- This will be our new objective function
  - How to maximize this requires lots of prob and math

https://arxiv.org/pdf/1612.00410.pdf

VIB results

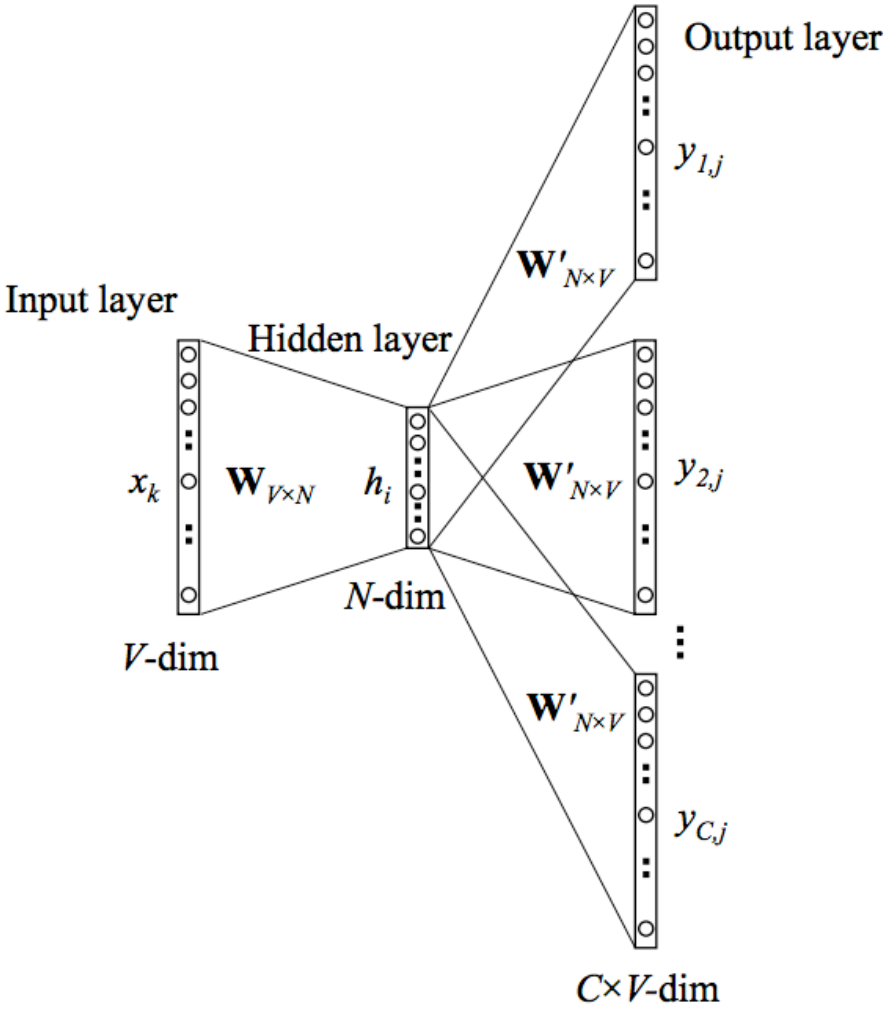
- MNIST results

Model	error
Baseline	1.38%
Dropout	1.34%
Dropout (Pereyra et al., 2017)	1.40%
Confidence Penalty	1.36%
Confidence Penalty (Pereyra et al., 2017)	1.17%
Label Smoothing	1.40%
Label Smoothing (Pereyra et al., 2017)	1.23%
VIB ($\beta = 10^{-3}$)	**1.13%**

- Better compression rate for ImageNET and better tolerance to adversarial attacks.

Word2Vec

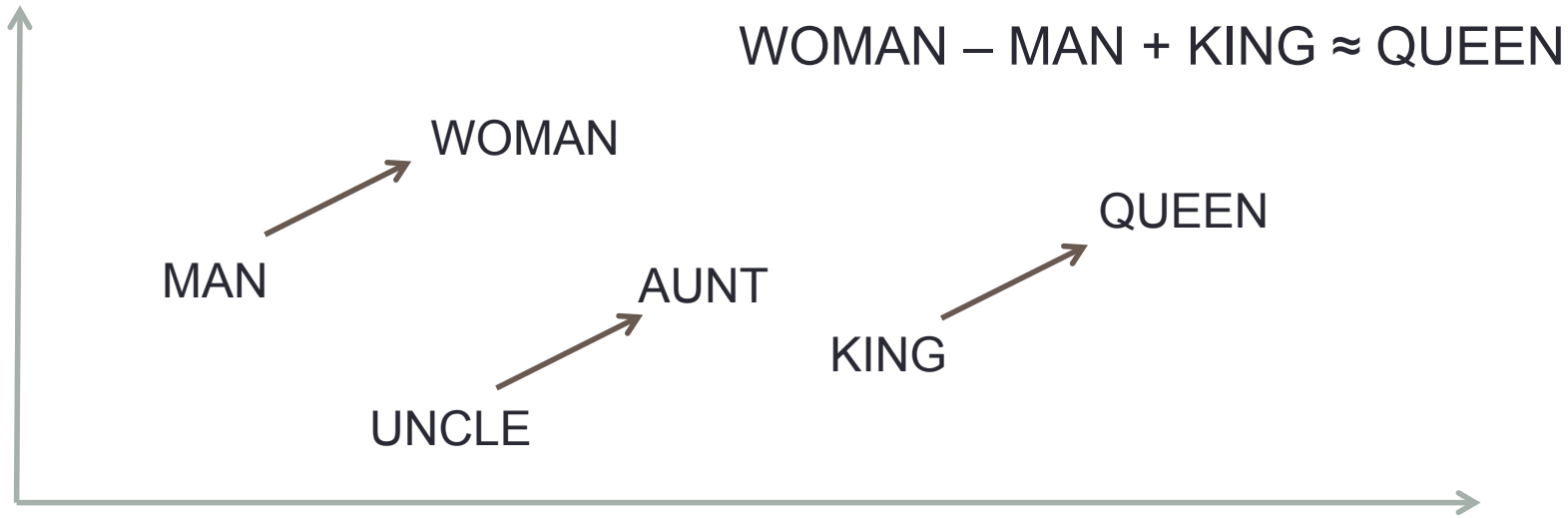
- Maps a word to a vector representation using neural networks
- Input word, predicts word around it
  - Words meaning can be captured by the context words
- Mary ___ an apple.



Mikolov, Linguistic Regularities in Continuous Space Word Representations, 2013

Word2Vec

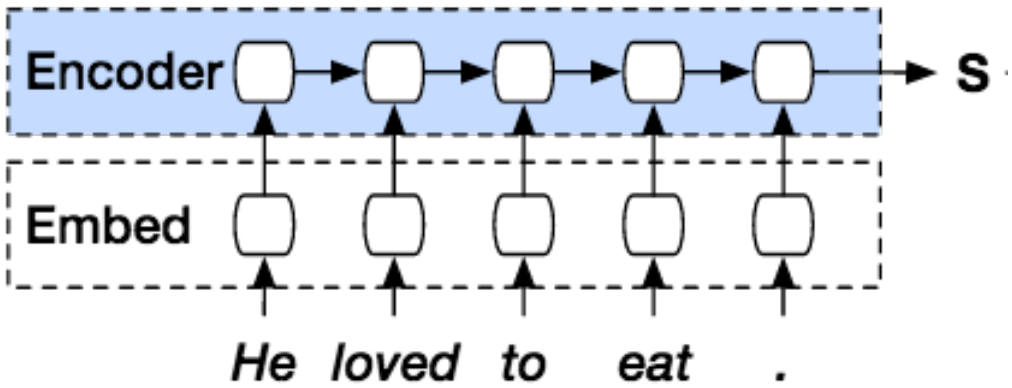
- Vectors from similar words are near each other
- You can also add and subtract meanings just like numbers!



Mikolov, Linguistic Regularities in Continuous Space Word Representations, 2013

Sentence encoder

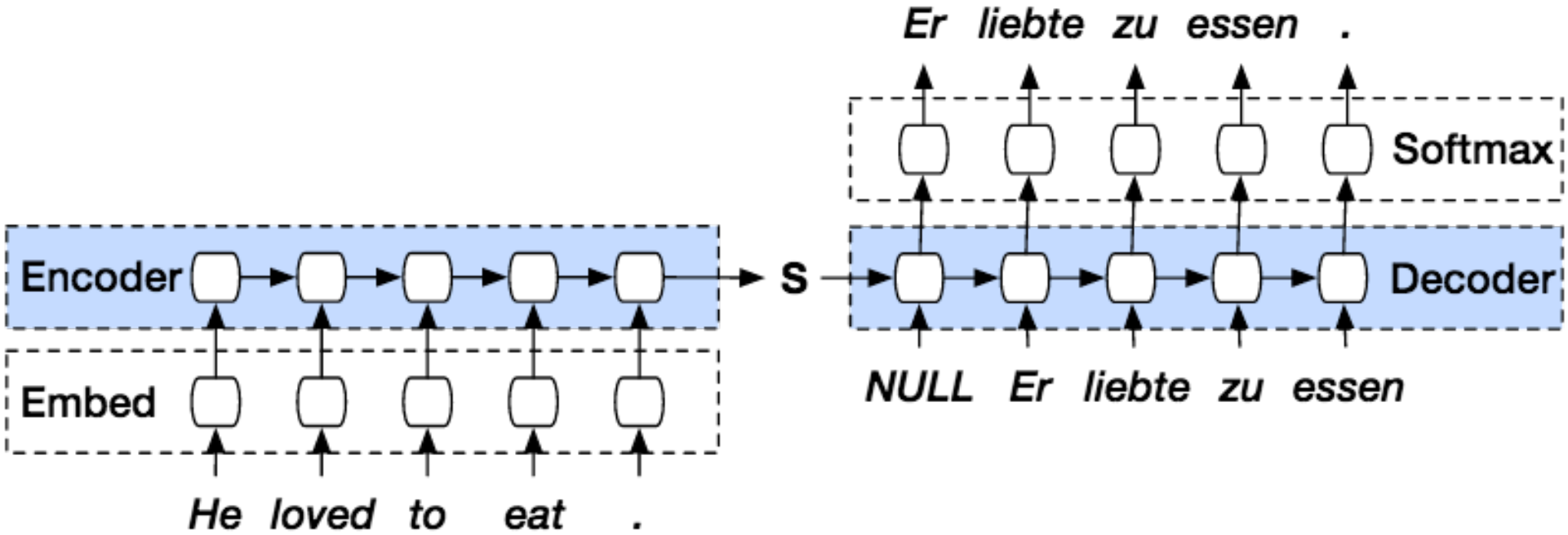
- How can we represent sentences?
- Similarly, use the internal states as the representation
  - LSTMs/GRUs are good for sequence task
  - Use LSTMs/GRUs memory cells outputs as representation



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

Misc topics

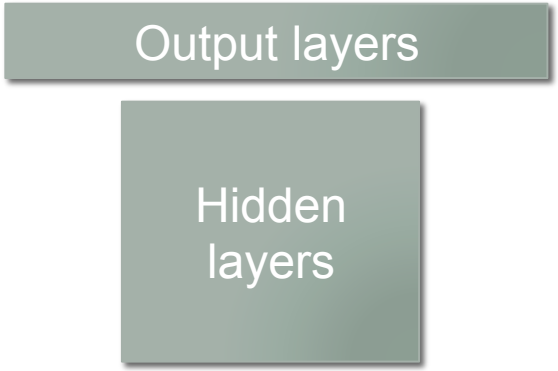
- Residual networks
- Encoder-Decoder
  - Information bottleneck
  - Word2Vec
- Transfer Learning, Multi-task models, Multi-models
- Attention modeling
- Faster R-CNN
- Alternatives to Backprop
  - 2^{nd} order methods
  - Synthetic gradients
  - Genetic Algo

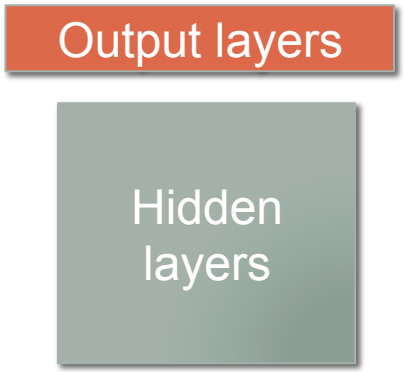
- Reinforcement Learning <- Next lecture!
- GAN <- Next Next lecture!
  - Adversarial inputs

Transfer learning

- We know networks captures good representations
- Can we use it for other tasks?

- Use trained networks to initialize a new network for a different task.
- Re-train the network using SGD on new data.





Notes on re-training

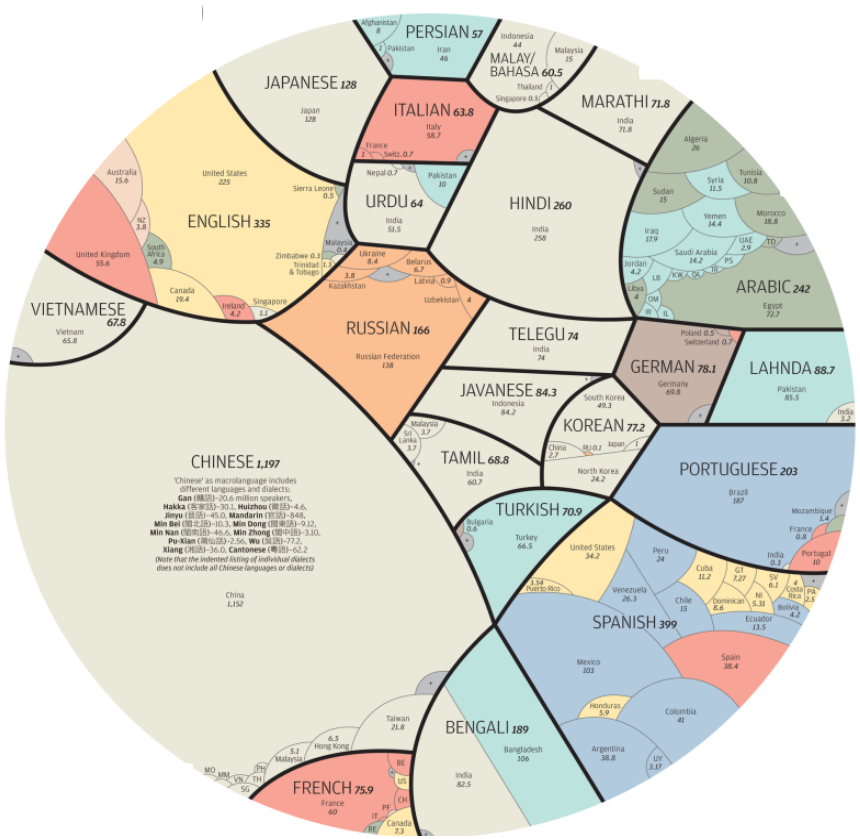
- Freeze the old layers, train the new weights for a bit
- Then, if new data is “large enough”, train the whole new network.

- Which layers to replace? And how many?
  - Depends on your belief of how close the tasks are

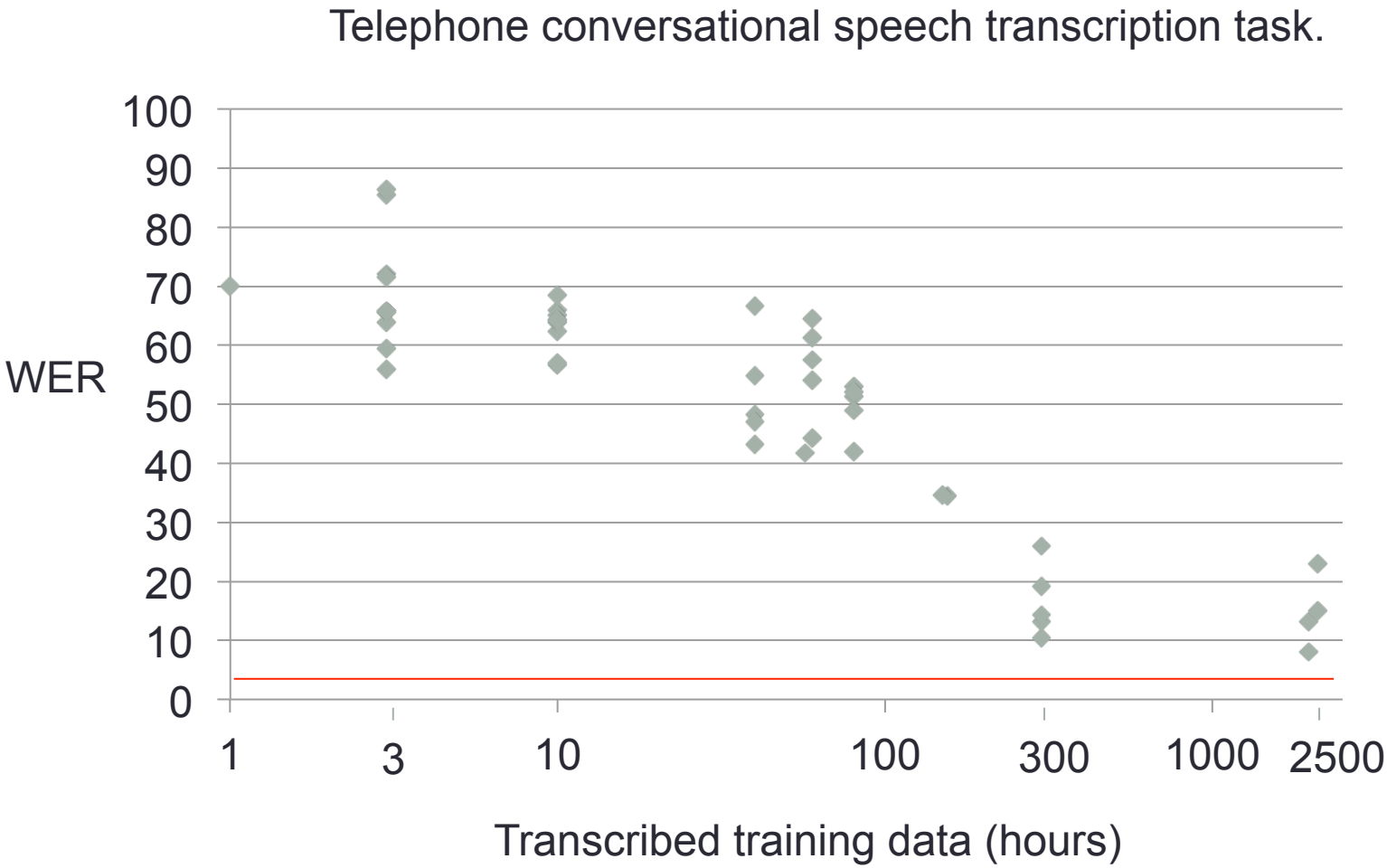
NEURAL NETWORKS TRANSFER LEARNING AND ASR

Language landscape

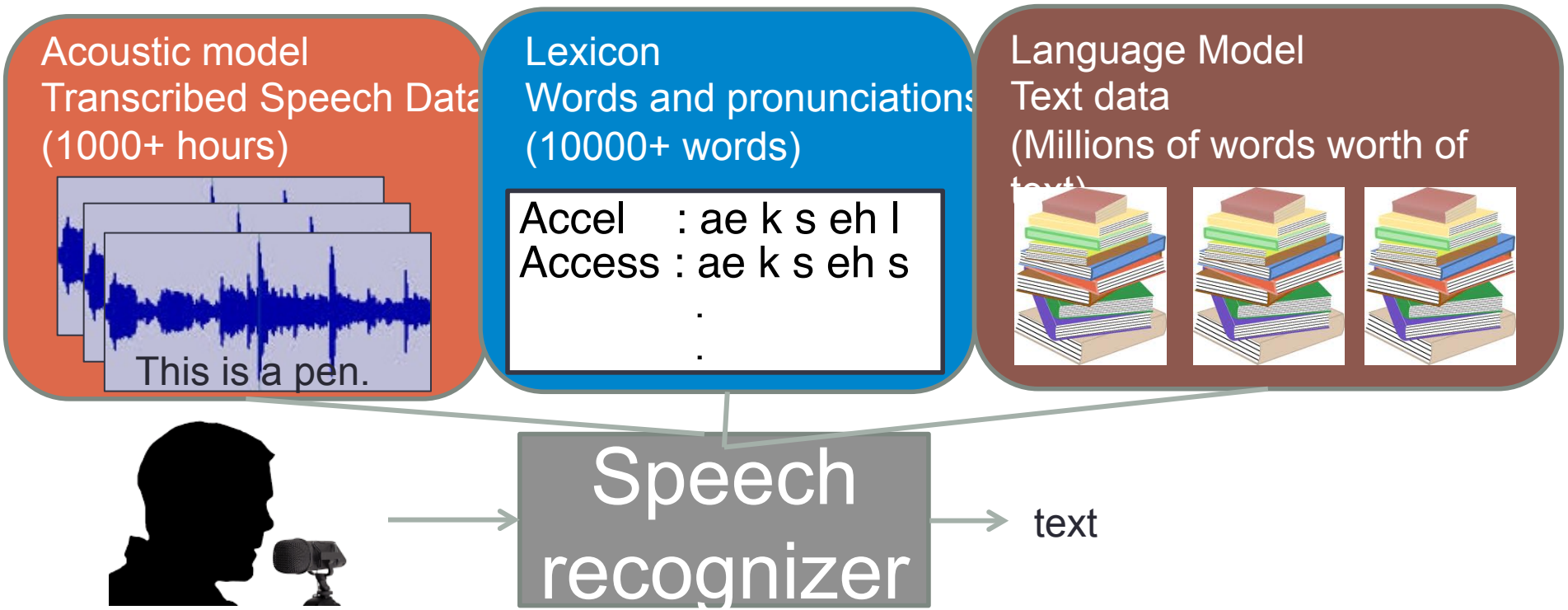
- Lots of languages in the world (7000)
  - Only ~100 has ASR



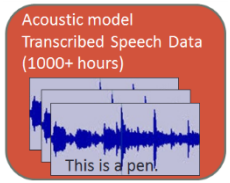
Effect of acoustic training data on ASR performance



The speech recognizer



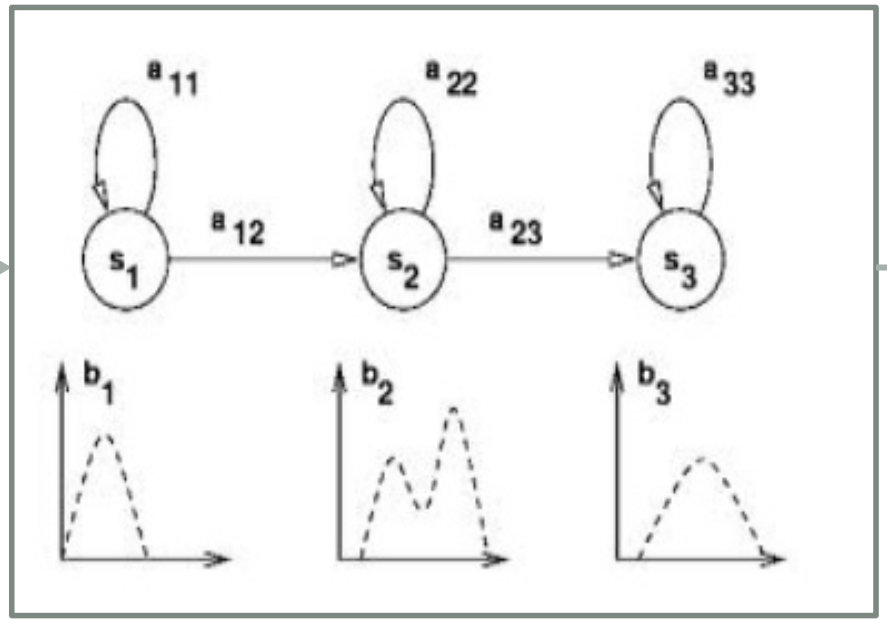
The Acoustic Model



PXS

Feature extraction

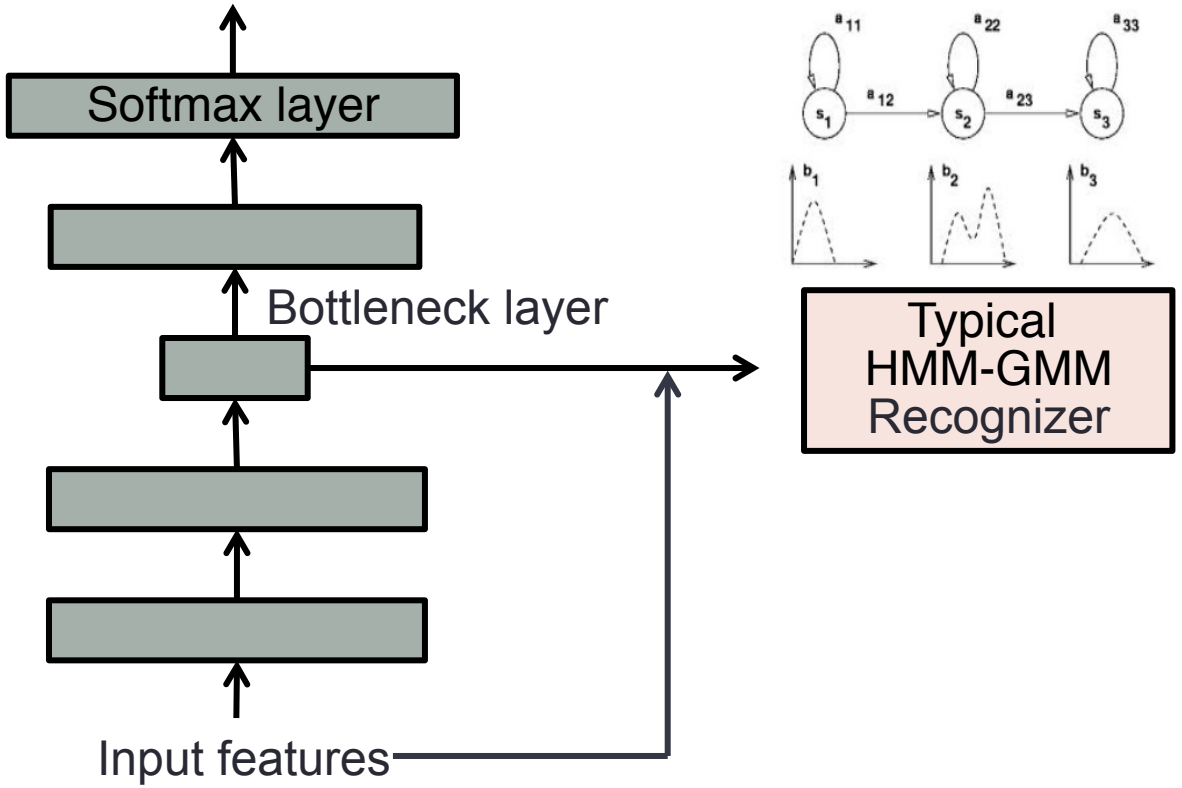
X



PXS

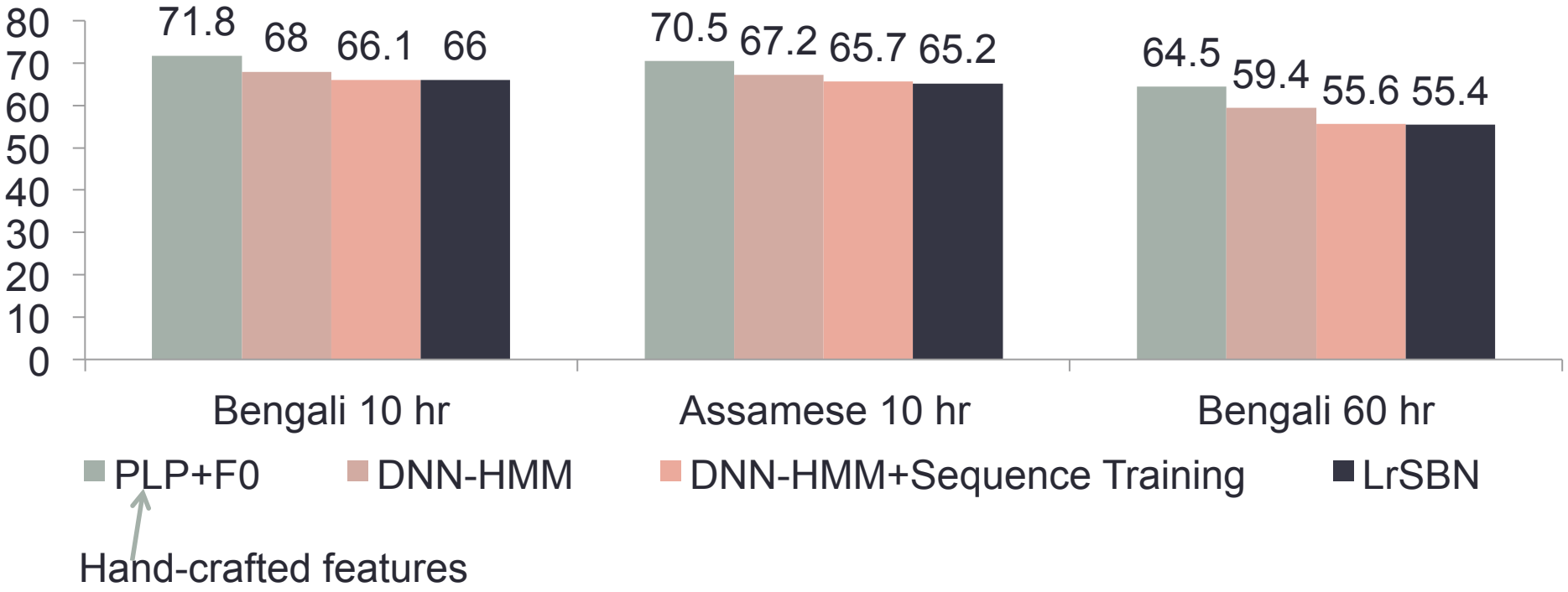
Tandem approach

- Use the DNN to generate good features to feed into the general GMM-HMM framework.
- Typically done by placing a narrow hidden layer in the network.



LrSBN results

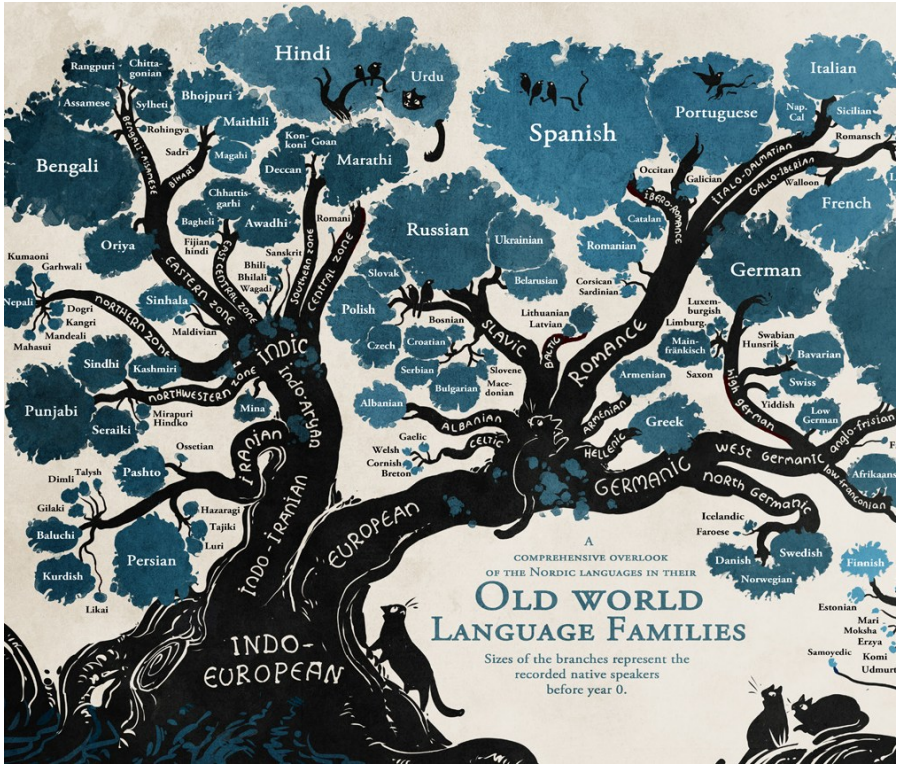
- LrSBN models outperform hybrid DNN-HMM models on all tasks.



https://groups.csail.mit.edu/sls/publications/2014/zhang-icassp14.pdf

Leveraging multilingual and out of domain data

- We have existing resources from other domains or languages
- Languages are related
  - Sounds produced by human
  - Languages share phonemes
- DNN features offer a great way to share this information



A case study in Assamese and Bengali

- Both spoken in India
- Same writing script, with some shared vocabulary
- Bengali : 33 consonants 21 vowels
- Assamese : 30 consonants 20 vowels
- 24-28 shared consonants 15 shared vowels

Bengali

Assamese

Transfer learning using bottleneck features

- BN features learns from DNN offers somewhat language independent features
- Use the DNN learned on Assamese to extract features for Bengali. Train GMM-HMM on Bengali data only.

Bengali	WER
PLP+F0 (10 hr Bengali)	71.8%
PLP+F0 (60 hr Bengali)	64.5%
LrSBN (10 hr Assamese -> 10 hr Bengali)	66.0%
LrSBN (60 hr Assamese -> 10 hr Bengali)	64.6%
+ Adaptation	63.7%
LrSBN (60 hr Bengali -> 10 hr Bengali)	61.6%

Results on more language pairs

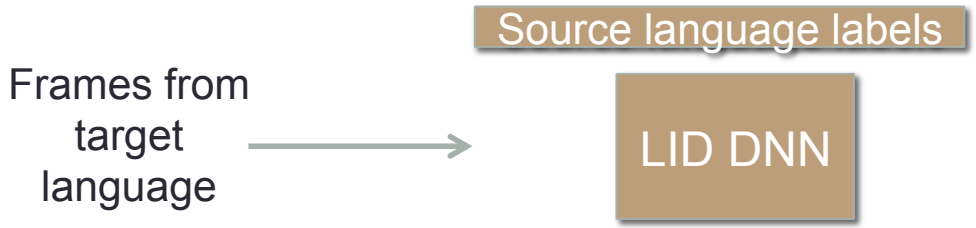
- Learn LrSBN on 60 hours of data, and apply on languages with 10 hours of data
- Numbers in blue is for 10->10 hours (baseline)

		Learn LrSBN on			
		Bengali	Assamese	Lao	Turkish
Use LrSBN on	Bengali	66.0	63.8	65.1	64.2
	Assamese	61.2	65.2	62.9	62.1
	Lao	59.8	60.1	62.3	60.0
	Turkish	61.8	63.1	63.3	63.9

- Transfer learning always improves performance.
- Similar languages perform better on transfer learning
- Can we identify this automatically?

Language Identification for data selection

- Train a classifier (LID) on source languages to predict the language given input frames
- Compute posteriors of the target language data using that classifier
- The best language for the target language should have the highest average posterior score



Predicting the best language

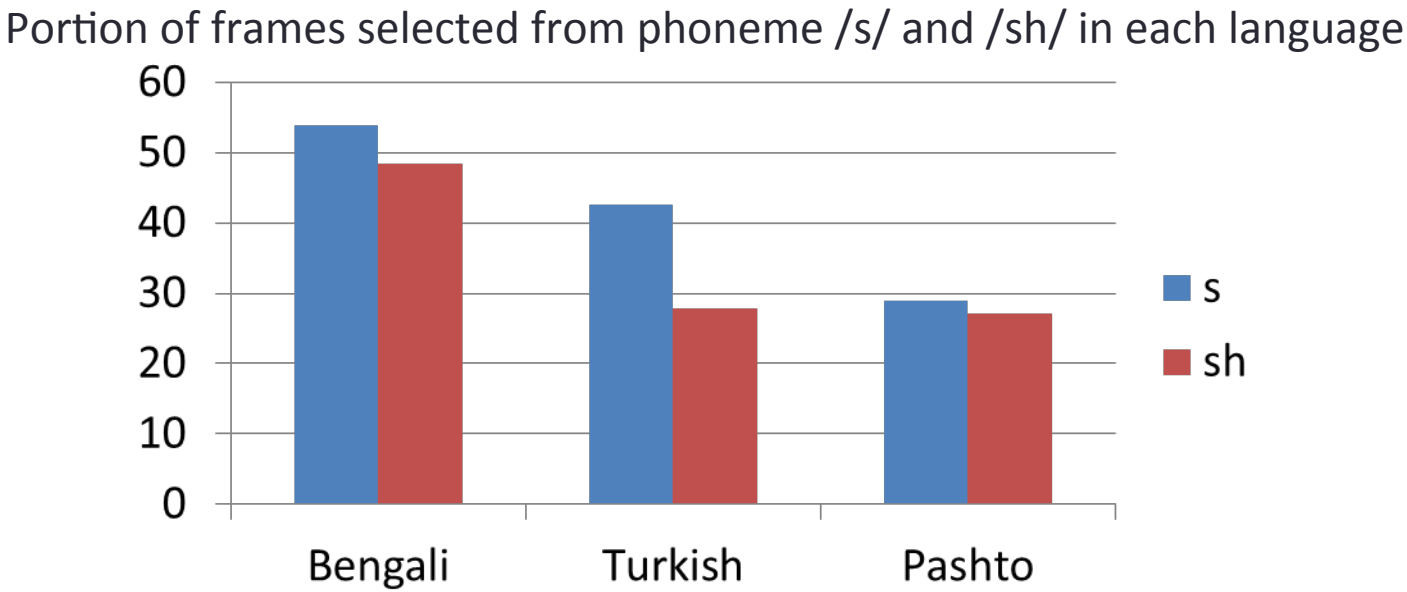
		Learn LrSBN on			
		Bengali	Assamese	Lao	Turkish
Use LrSBN on	Bengali	66.0	63.8	65.1	64.2
	Assamese	61.2	65.2	62.9	62.1
	Lao	59.8	60.1	62.3	60.0
	Turkish	61.8	63.1	63.3	63.9

		Averaged predicted posterior			
		Bengali	Assamese	Lao	Turkish
Input frames	Bengali	0.57	0.21	0.09	0.13
	Assamese	0.21	0.57	0.11	0.11
	Lao	0.08	0.11	0.71	0.10
	Turkish	0.13	0.12	0.10	0.65

The LID scores correspond to the best language to use most of the time.

LID for selecting meaningful parts

- Languages have different phonemes
- Select meaningful parts automatically
- Cebuano has /s/ but not /sh/
- High scoring frames for Cebuano favor /s/ over /sh/



Data selection for transfer learning

- Selecting parts of many language vs selecting the best language

System (LrSBNs)	Cebuano (3hr)	Telugu (3hr)	Swahili (3hr)
	MTWV	MTWV	MTWV
All language (500 hr)	0.2259	0.1269	0.3983
Closest language (50 hr)	0.2526	0.1682	0.4225
Parts (100 hr)	0.2513	0.1711	**0.4244**
Parts (200 hr)	**0.2531**	**0.1756**	0.4233

https://groups.csail.mit.edu/sls/publications/2016/Ekapol_ICASSP-16.pdf

Transfer learning in ASR summary

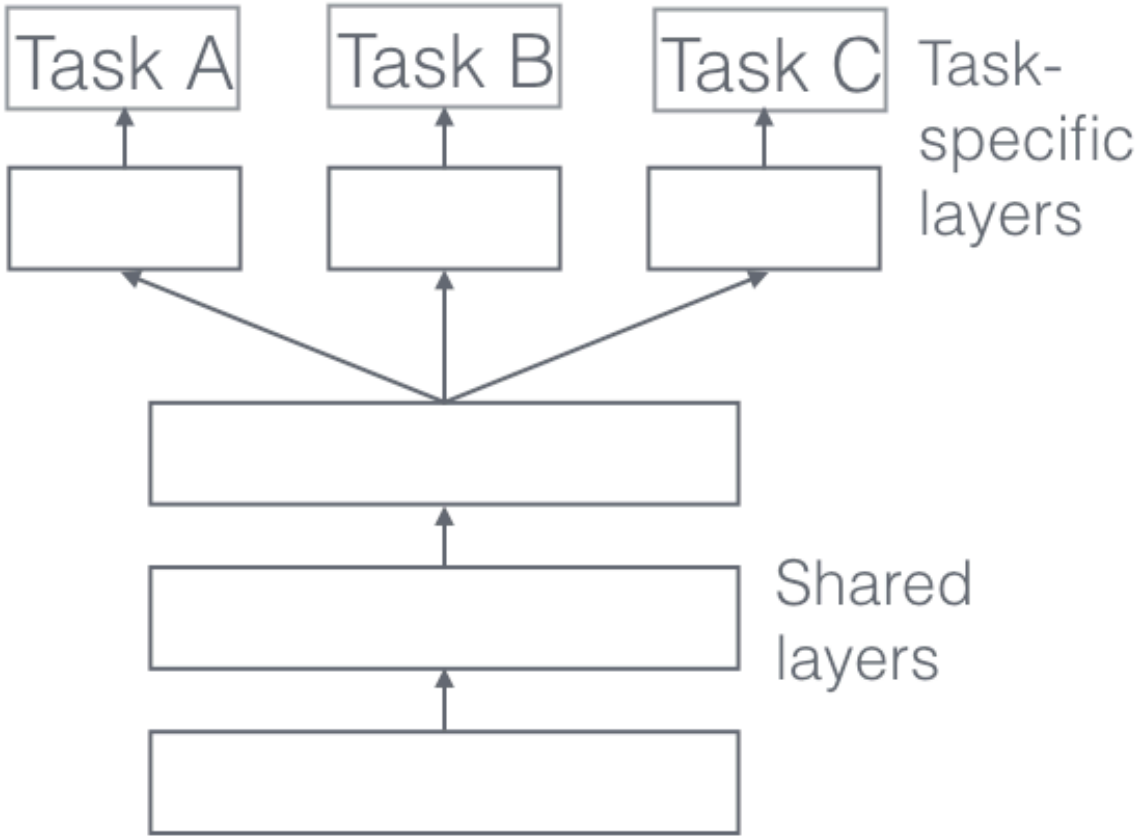
- DNN can learn cross-language features for ASR
- Transfer learning can benefit from data selection

- These features for ASR has been used successfully for other task
  - Language ID
  - Speaker ID

Multi-task training (Joint models)

- Training a neural network on multiple tasks at the same time can help the network distinguish between noise and important information
- Helps partition the output space
- Example: Input speech, output text vs Input speech, output text and language.

Multi-task learning (simplest form)

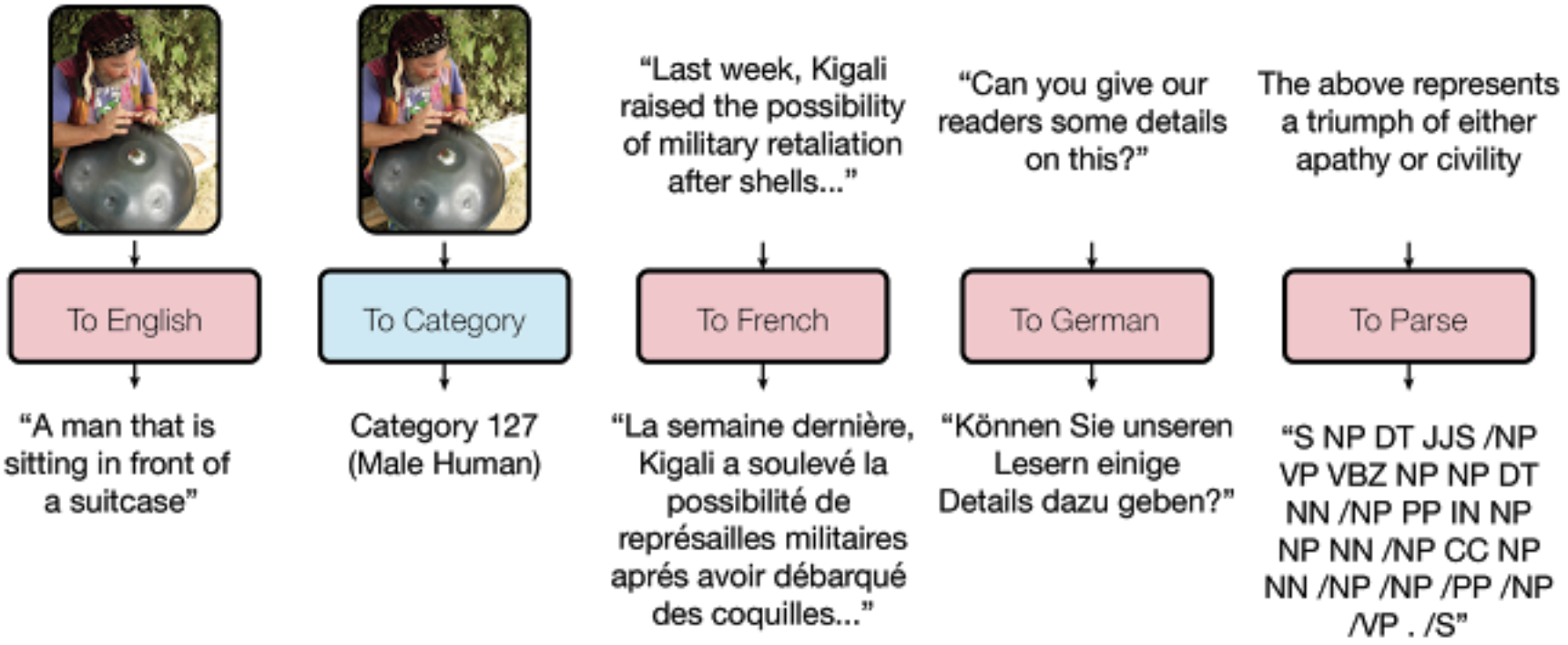


Total Loss = W0 Loss A + W1 Loss B + W2 Loss C

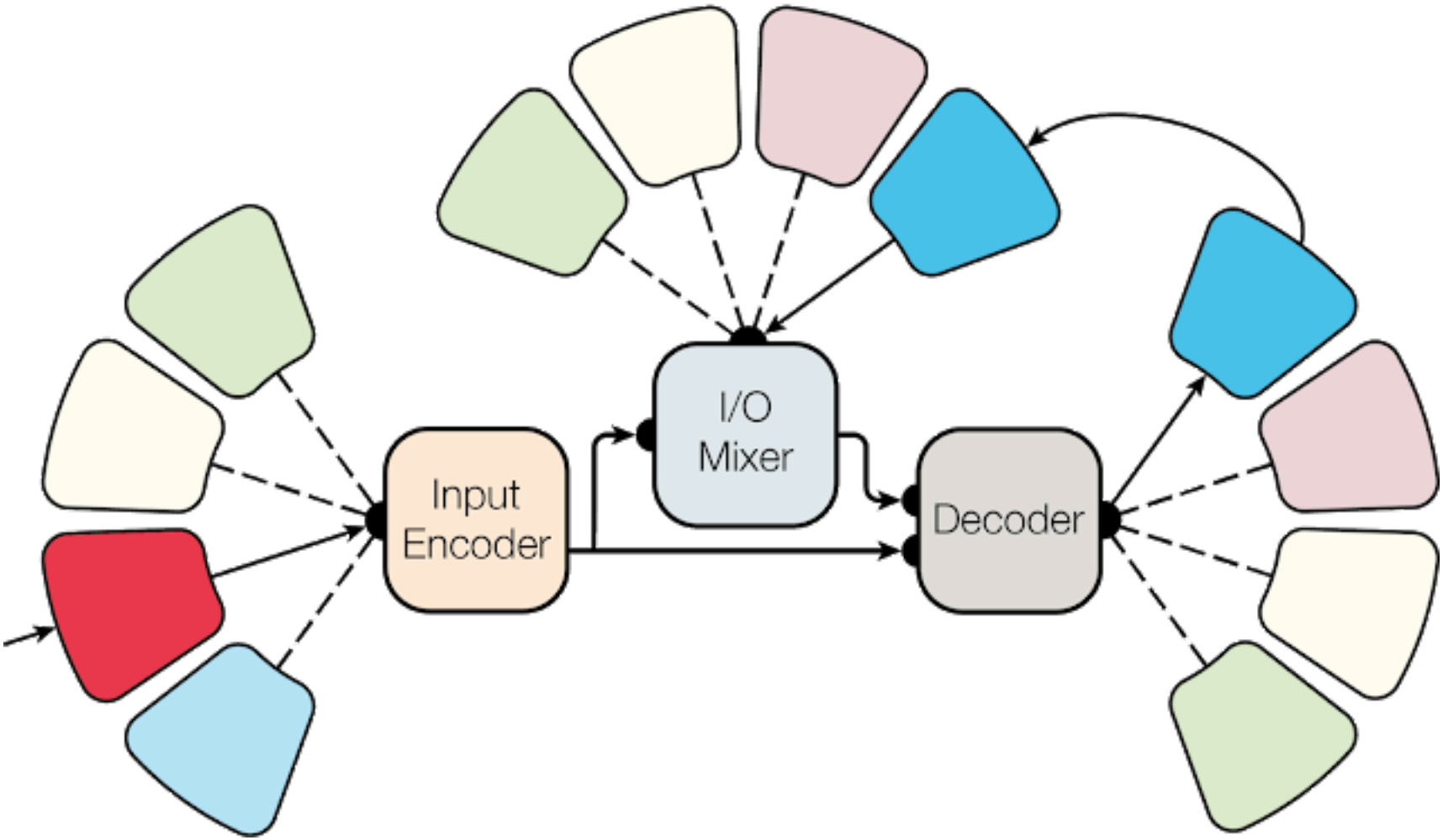
http://ruder.io/multi-task/

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


I/O
Mixer
Input
Encoder
Decoder

Misc topics

- Residual networks
- Encoder-Decoder
  - Information bottleneck
  - Word2Vec
- Transfer Learning, Multi-task models, Multi-models
- Attention modeling
- Faster R-CNN
- Alternatives to Backprop
  - 2^{nd} order methods
  - Synthetic gradients
  - Genetic Algo

- Reinforcement Learning <- Next lecture!
- GAN <- Next Next lecture!
  - Adversarial inputs

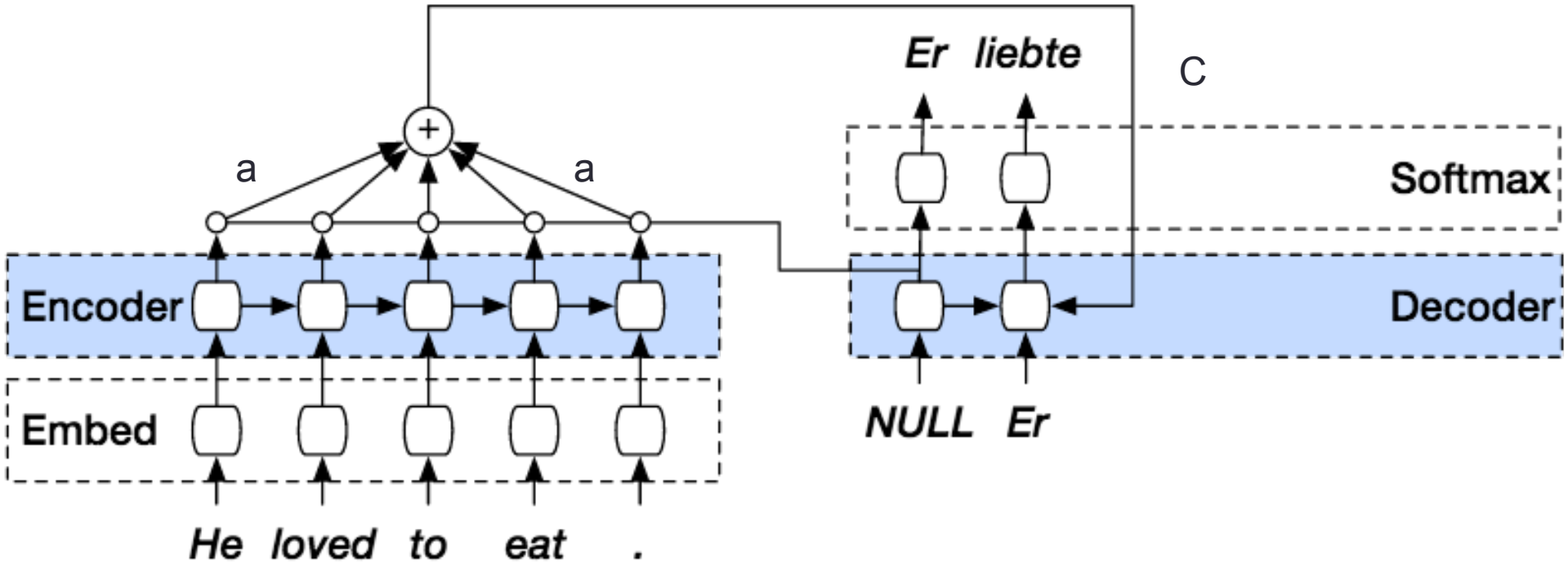
The need for attention

- Recurrent neural networks cannot handle long inputs
- LSTM/GRUs can learn to forget, so it can handle longer inputs.
- Still not long enough

- Need another mechanism to tell where the network will focus

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

Context vector

- In attention we want to find a score $\mathbf{a_t}$ at decoding step t that gives weights to each encoder memory $\mathbf{h_l}$ at time step l
- LSTM decoder memory $\mathbf{y_t}$
- $a_{tl} = \text{softmax}(F(\mathbf{h_l}, \mathbf{y_{t-1}}))$
- $\mathbf{c_t} = \text{Sum}_l\ a_{tl}\mathbf{h_l}$
- F can be a feed forward network or just a weight matrix

https://arxiv.org/pdf/1409.0473.pdf

http://ruder.io/deep-learning-nlp-best-practices/index.html#a

Attention model for summarization

- Attention model can help with long context sentences for NLP

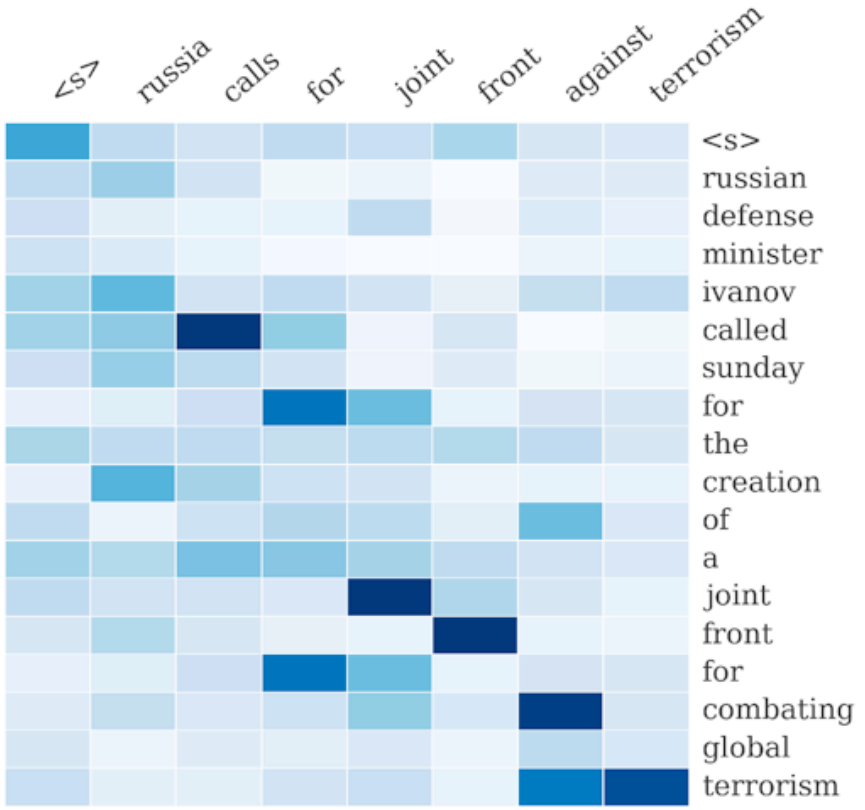


Figure 1: Example output of the attention-based summarization (ABS) system. The heatmap represents a soft alignment between the input (right) and the generated summary (top). The columns represent the distribution over the input after generating each word.

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

Self attention

- In attention we want to find a score $\mathbf{a_t}$ at decoding step t that gives weights to each encoder memory $\mathbf{h_l}$ at time step l
- LSTM decoder memory $\mathbf{y_t}$
- a_{tl} = softmax(F($\mathbf{h_l}$,$\mathbf{y_{t-1}}$)) -> a_{tl} = softmax(F($\mathbf{h_l}$))
- $\mathbf{c_t}$ = Sum$_l$ $a_{tl}\mathbf{h_l}$
- F can be a feed forward network or just a weight matrix

- No need for outside task to summarize the vector

https://arxiv.org/pdf/1703.03130.pdf

Attention is all you need

- Claims you only need attention
  - No need for CNN and RNN
- Tried on various NLP tasks: Translation, Parsing

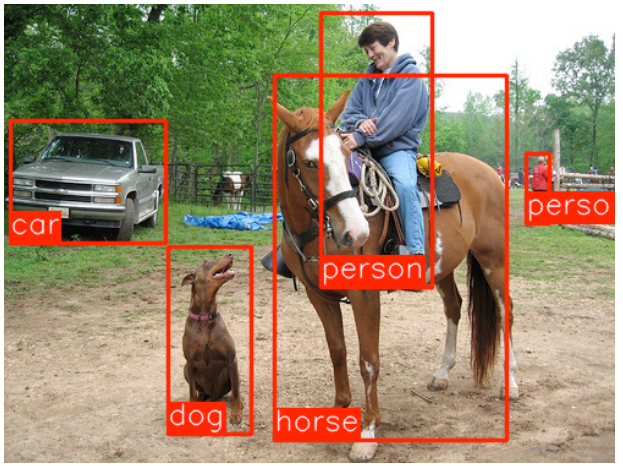
https://arxiv.org/pdf/1706.03762.pdf

Misc topics

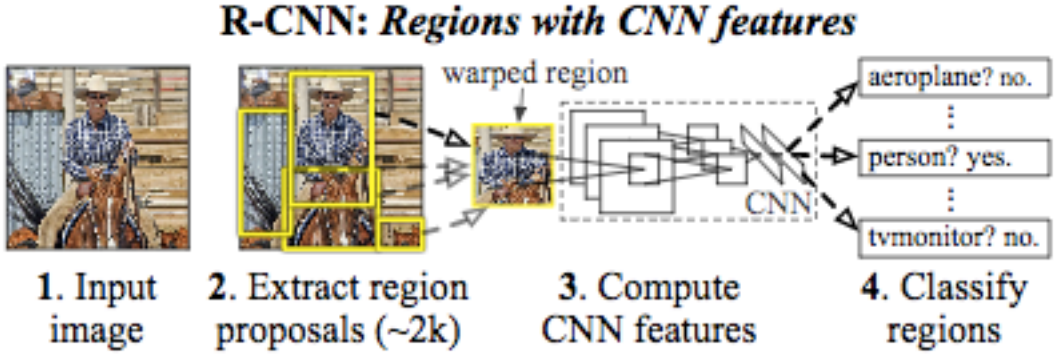
- Residual networks
- Encoder-Decoder
  - Information bottleneck
  - Word2Vec
- Transfer Learning, Multi-task models, Multi-models
- Attention modeling
- Faster R-CNN
- Alternatives to Backprop
  - 2^{nd} order methods
  - Synthetic gradients
  - Genetic Algo

- Reinforcement Learning <- Next lecture!
- GAN <- Next Next lecture!
  - Adversarial inputs

Object detection



- Identify all objects in an image
- Two sub-task
  - Determine what objects
  - Determine where objects are
- YOLO: do both at the same time by having the network output both scores
- R-CNN (Region CNN): Propose possible locations for objects, then classify



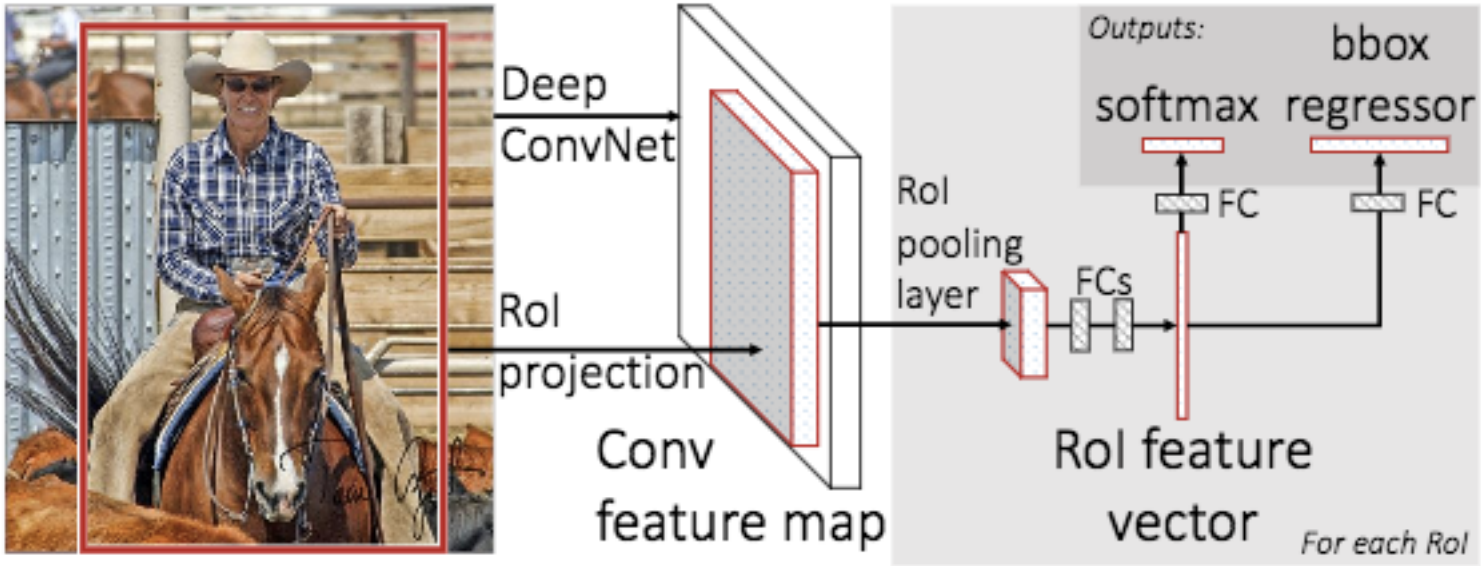
https://arxiv.org/abs/1311.2524.

R-CNN drawbacks

- Computes object classification over multiple regions. Some overlaps.
- Also does region refinements. Regression task that crops the bounding box to better fit the image.
  - Two classifiers: one for object label, one for refinement

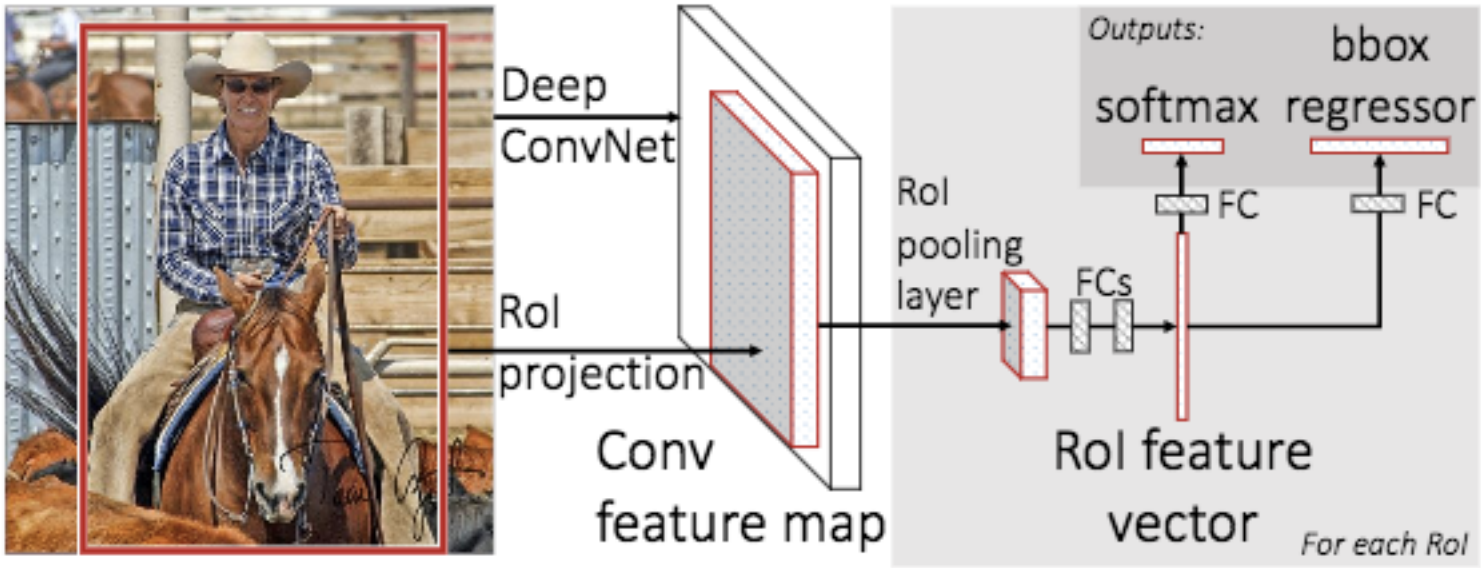
Fast R-CNN

- Insight: We compute the same forward pass over multiple overlapping segments
- Region of Interest Pooling (RoIPool)



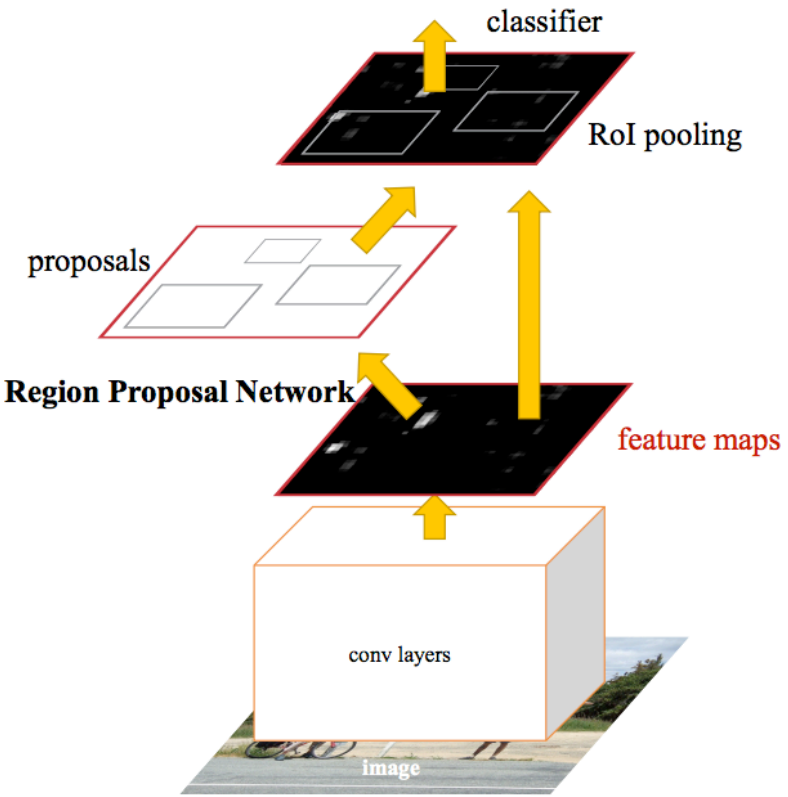
RoIPool

- Put whole image through a convnet
- Divide the output convolutional layer. Max pool over each RoI.
- Do joint classification and bounding box regression



Faster R-CNN

- Fast R-CNN still rely on RoI proposal to extract maxpool features.
  - This is separated from the classification
- Let's combine the two.



https://arxiv.org/pdf/1506.01497.pdf

Misc topics

- Residual networks
- Encoder-Decoder
  - Information bottleneck
  - Word2Vec
- Transfer Learning, Multi-task models, Multi-models
- Attention modeling
- Faster R-CNN
- Alternatives to Backprop
  - 2^{nd} order methods
  - Synthetic gradients
  - Genetic Algo

- Reinforcement Learning <- Next lecture!
- GAN <- Next Next lecture!
  - Adversarial inputs

2nd order methods

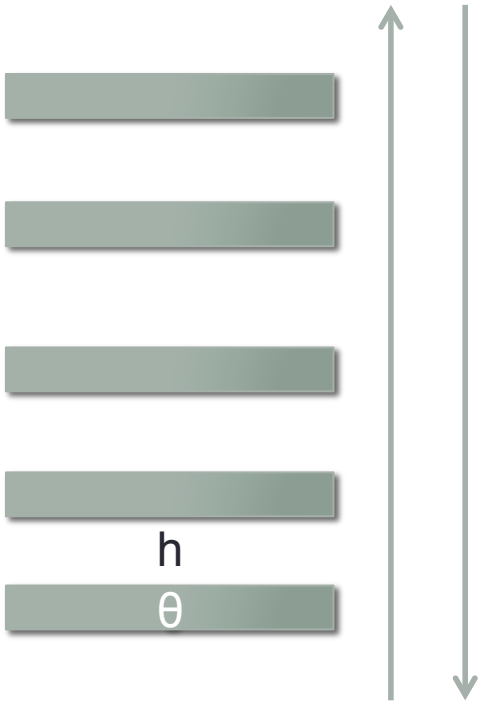
- How to minimize a function
  - Differentiation, set to 0
    - Hard for real problems
  - SGD of the differentiation
    - Needs step size because 1st derivative is local in that point
  - Use 2nd derivative information with 1st derivative.
    - 1st tells direction
    - 2nd tells how that direction would change as we move along.
    - Can be more sure about stepping and somewhat avoid local minima/ inflection points
  - Other name: Newton method, Hessian method
  - Hessian – 2nd derivative metrix

2nd order methods and neural networks

- In theory it should be better than SGD
- Doesn’t really work…yet
  - Active area of research

Synthetic gradients

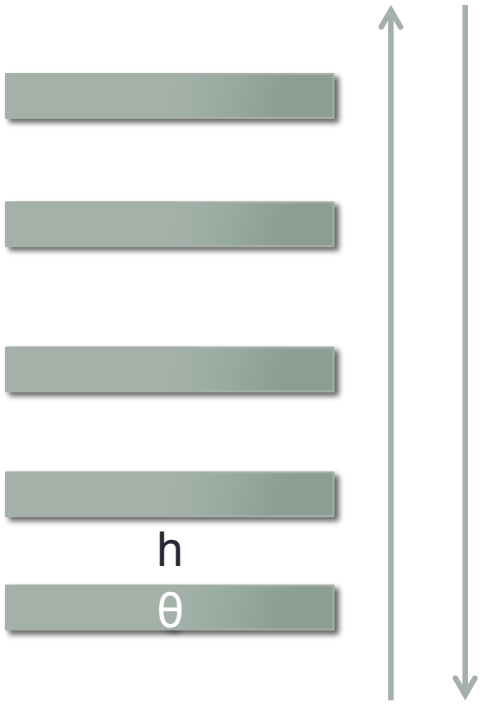
- Deep networks can take time in doing forward and backward pass



$$\theta \leftarrow \theta - \alpha \, dL/d\theta$$

Synthetic gradients

- To speed things up we can do early compute



$\theta \leftarrow \theta - \alpha \, dL/dh \, dh/d\theta$

$dh/d\theta$ we can compute right away

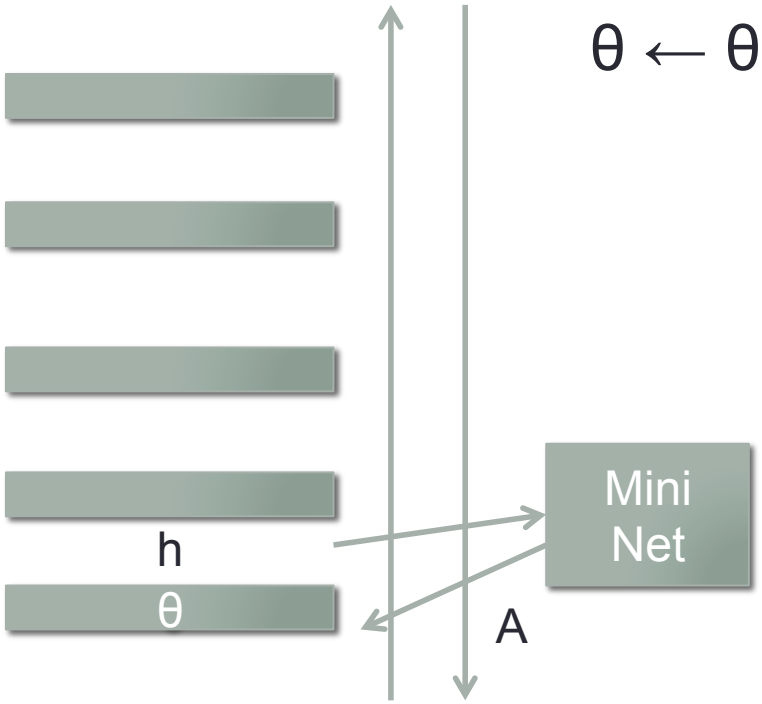
Use an estimate of $dL/dh = A$

$\theta \leftarrow \theta - \alpha \, A \, dh/d\theta$

Can compute update right away

Estimating A

- A can be a estimated small neural network
- This network gets updated as the true gradients come in.



$$\theta \leftarrow \theta - \alpha A \, dh/d\theta$$

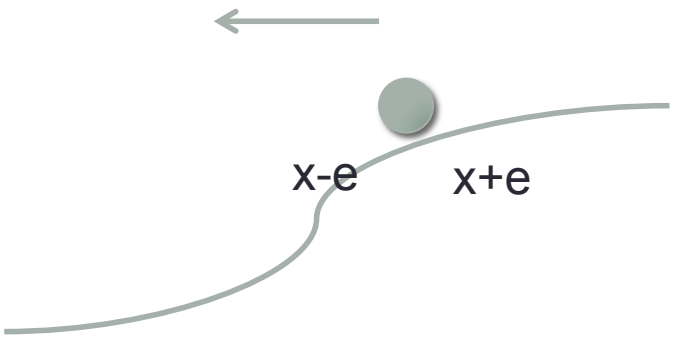
https://arxiv.org/pdf/1608.05343.pdf

Synthetic gradients

- Probably doesn't help if you're not google/facebook/etc. with large scale multi-gpu training
- But networks in networks are cool ☺

Genetic Algo for training neural networks

- How to minimize a function
  - Find direction
    - Find gradient by taking derivation
    - Try different directions, direction that goes down is the derivative!
      - Brute force solution



Genetic Algo for training neural networks

- Initialize network
- Spawns N networks with small perturbations
- Evaluate the loss, pick the best N
  - Update: network <- network + r * best network

- Works well for models that has tricky gradients
  - Reinforcement learning

https://arxiv.org/pdf/1703.03864.pdf

Open research

- Unsupervised learning <- next next lecture
- More reasoning (understanding deep learning)
- Language understanding
- Training-test mis-match (Maximum likelihood criterion) and maybe an alternative to current training strategies

Misc topics

- Residual networks
- Encoder-Decoder
  - Information bottleneck
  - Word2Vec
- Transfer Learning, Multi-task models, Multi-models
- Attention modeling
- Faster R-CNN
- Alternatives to Backprop
  - 2nd order methods
  - Synthetic gradients
  - Genetic Algo

- Reinforcement Learning <- Next lecture! Guest lecture!
- GAN <- Next Next lecture!
  - Adversarial inputs

Project

- Some people don’t have a team yet

- Fill project form in courseville by tonight.