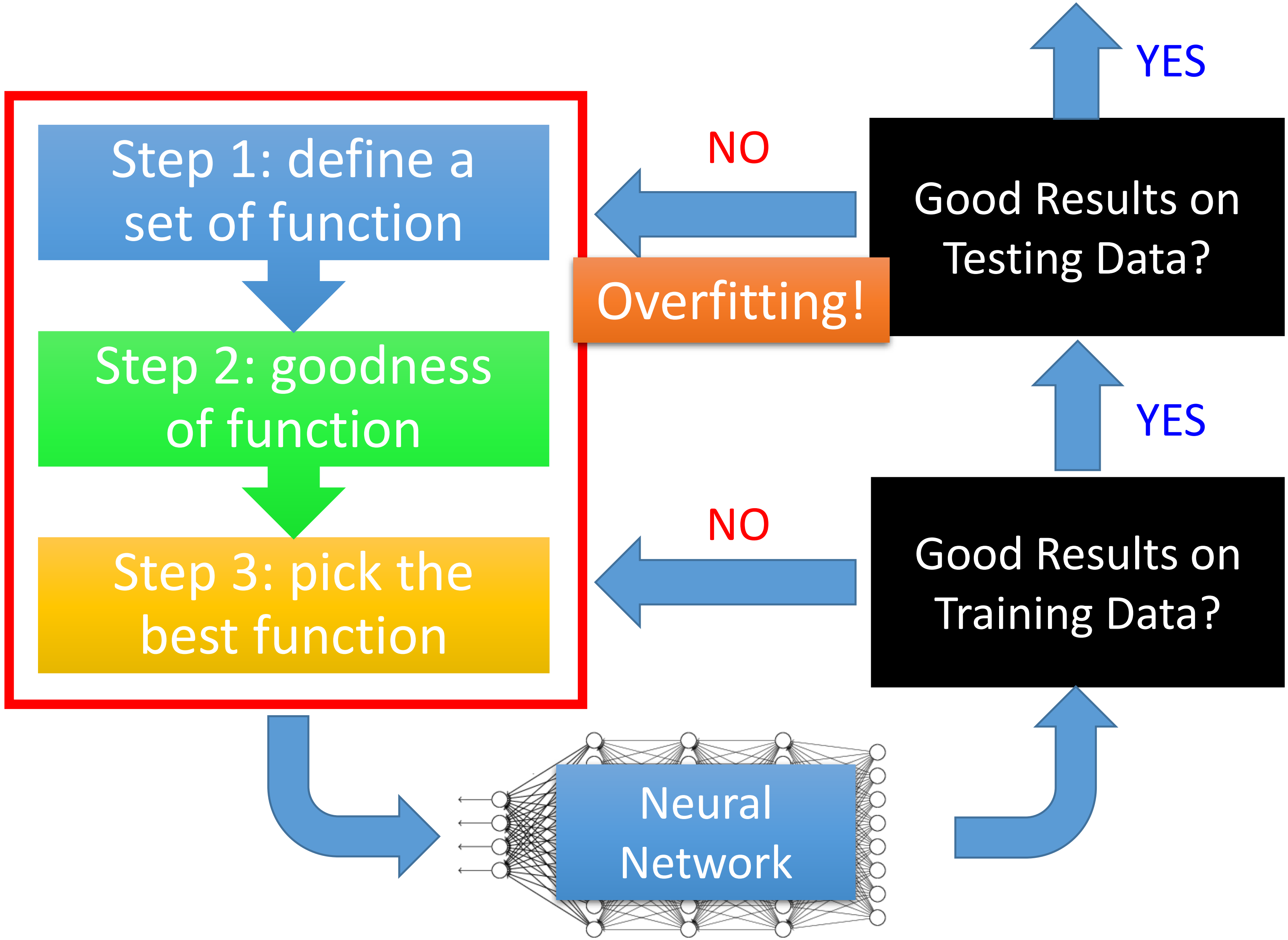
Advanced Tips for Deep Learning

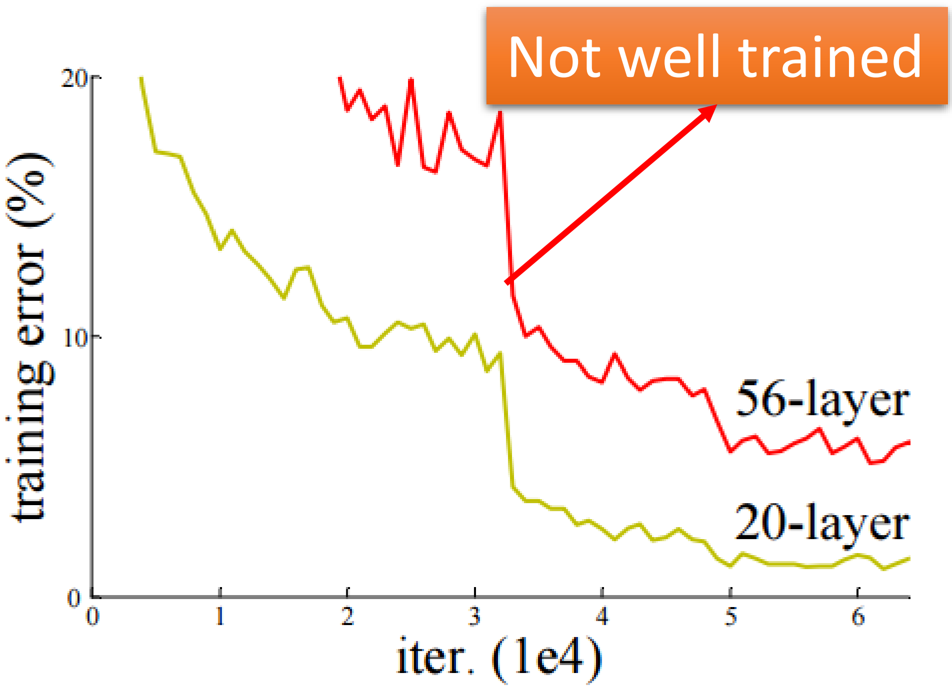
Hung-yi Lee

Prerequisite: https://www.youtube.com/watch?v=xki61j7z-30

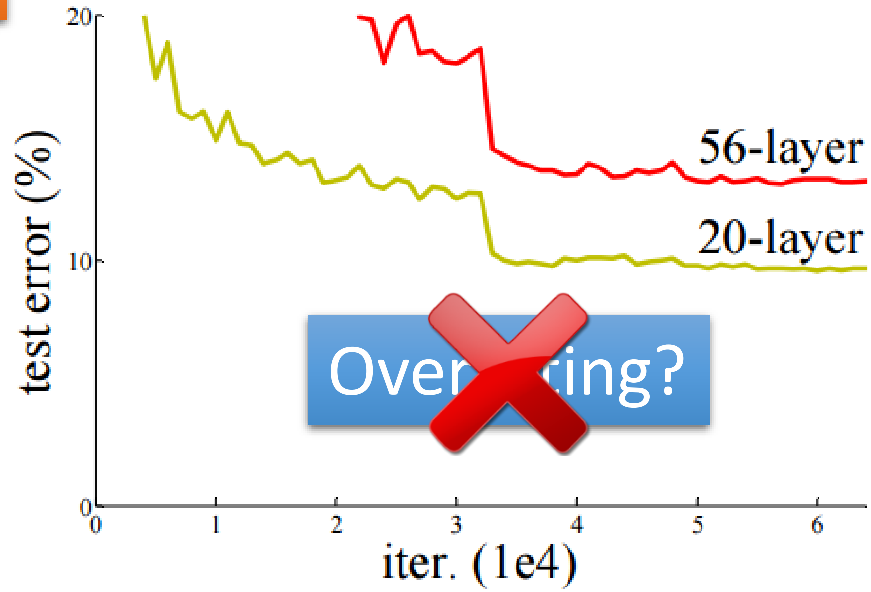
Recipe of Deep Learning



Do not always blame Overfitting



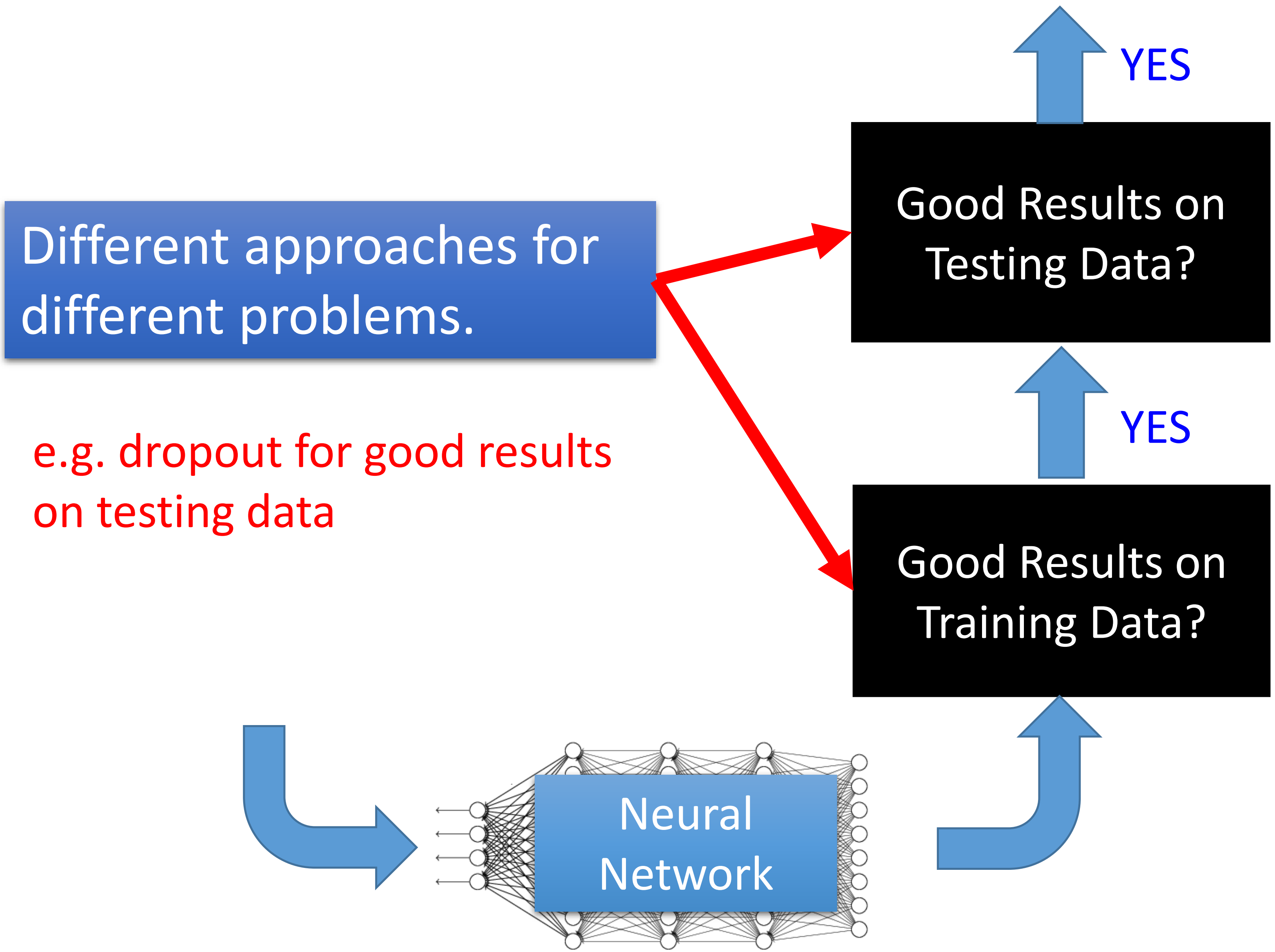
Training Data



Testing Data

Deep Residual Learning for Image Recognition
http://arxiv.org/abs/1512.03385

Recipe of Deep Learning



Outline

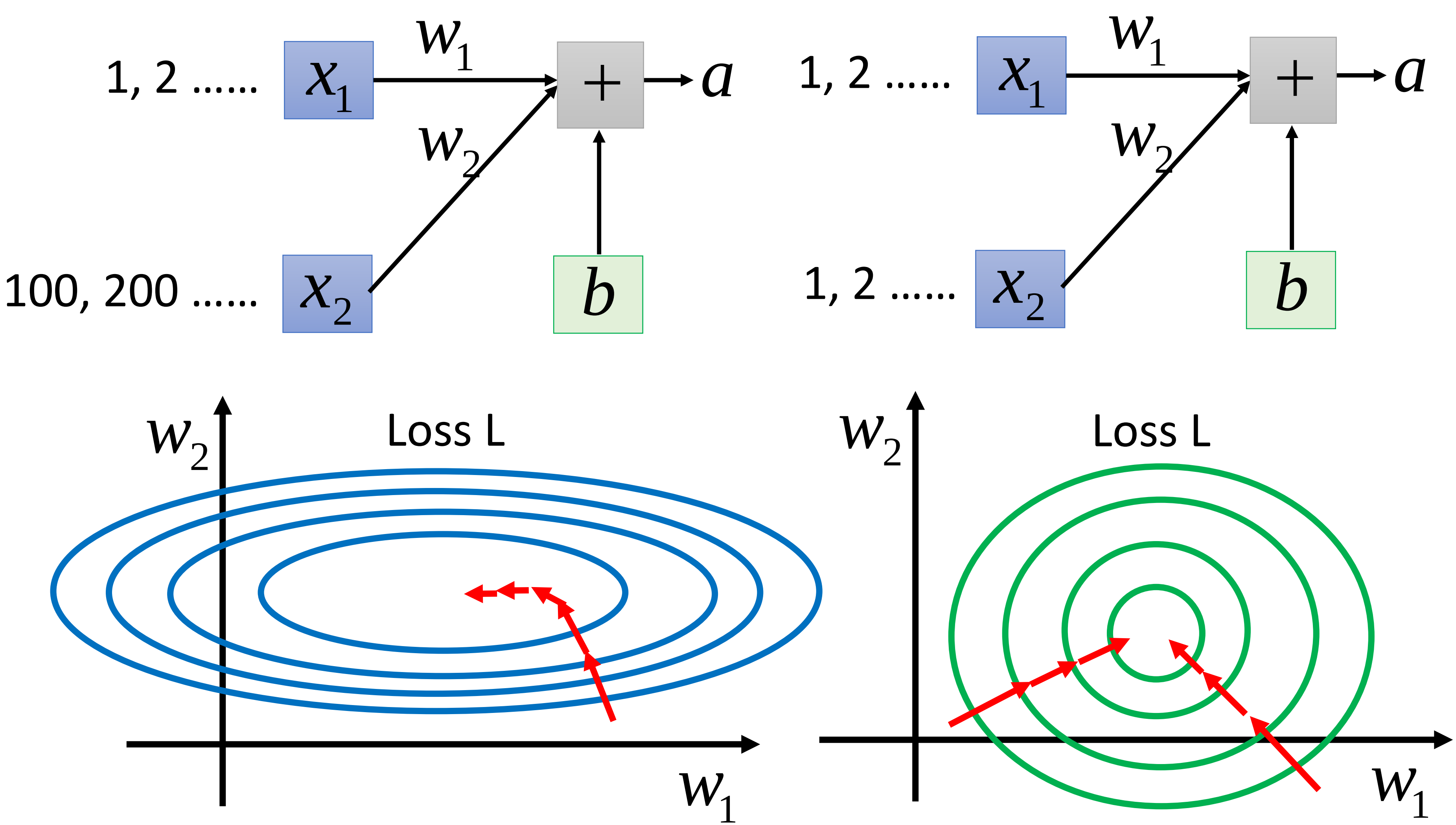
- Batch Normalization
- New Activation Function
- Tuning Hyperparameters
- Interesting facts (?) about deep learning
- Capsule
- New models for QA

Batch Normalization

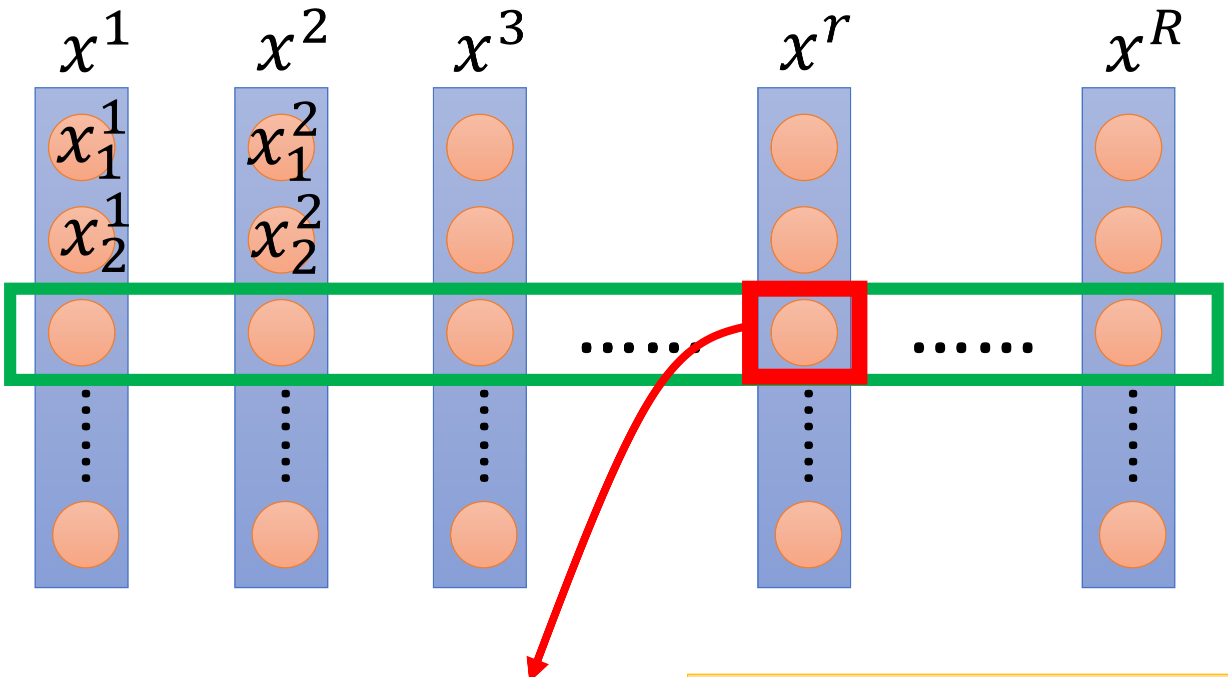
Sergey Ioffe, Christian Szegedy, “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift”, 2015

Feature Scaling

Make different features have the same scaling



Feature Scaling



For each dimension i:

mean: m_i

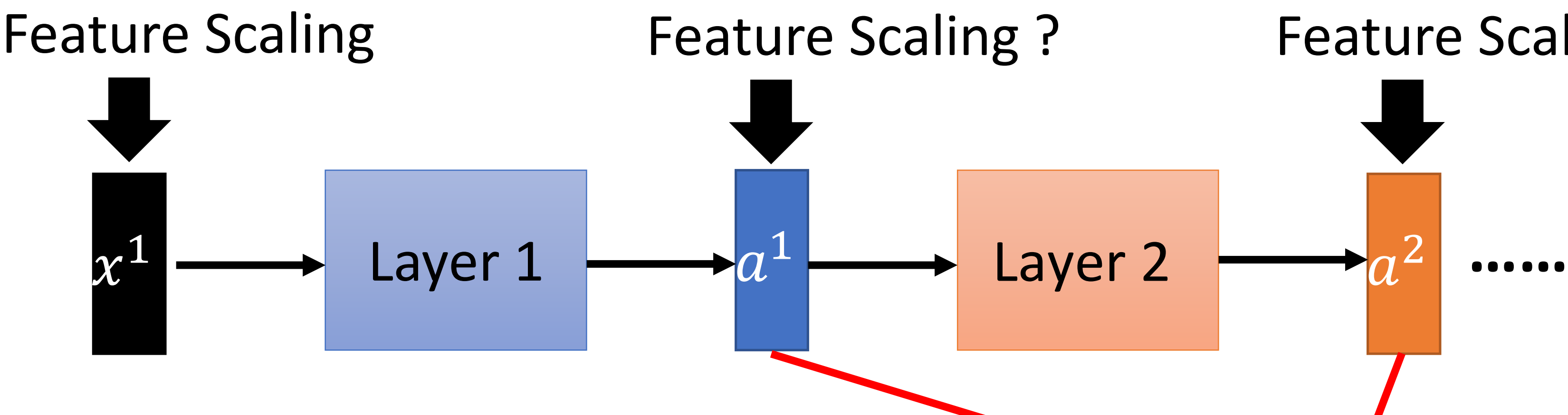
standard deviation: σ_i

$$x_i^r \leftarrow \frac{x_i^r - m_i}{\sigma_i}$$

The means of all dimensions are 0, and the variances are all 1

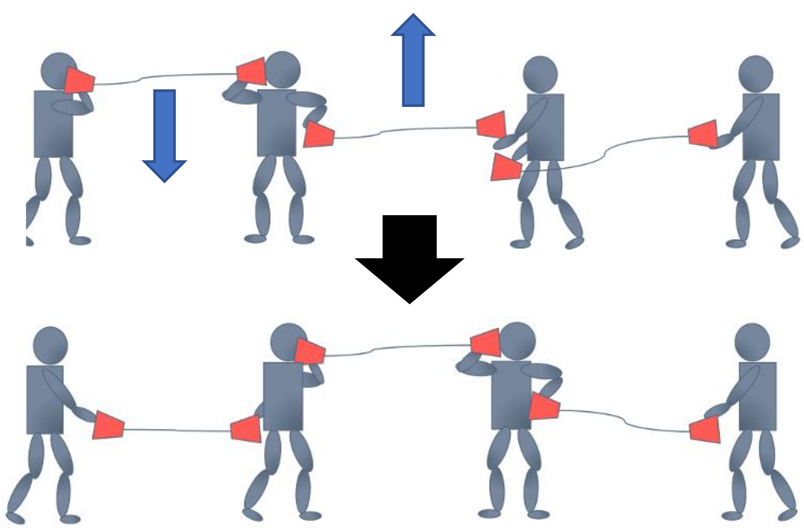
In general, gradient descent converges much faster with feature scaling than without it.

How about Hidden Layer?



Difficulty: their statistics change during the training …

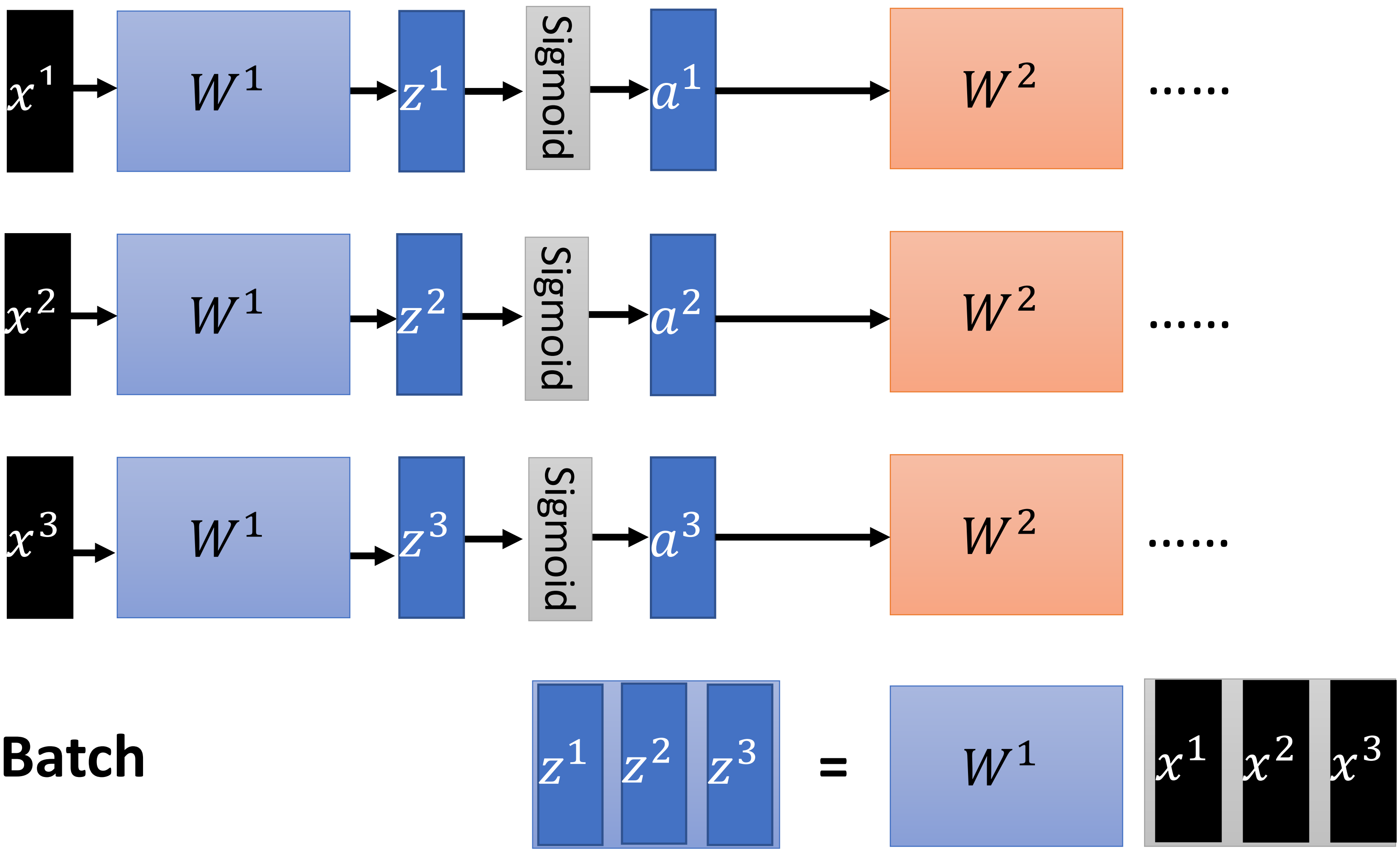
➡ Batch normalization



Internal Covariate Shift

Smaller learning rate can be helpful, but the training would be slower.

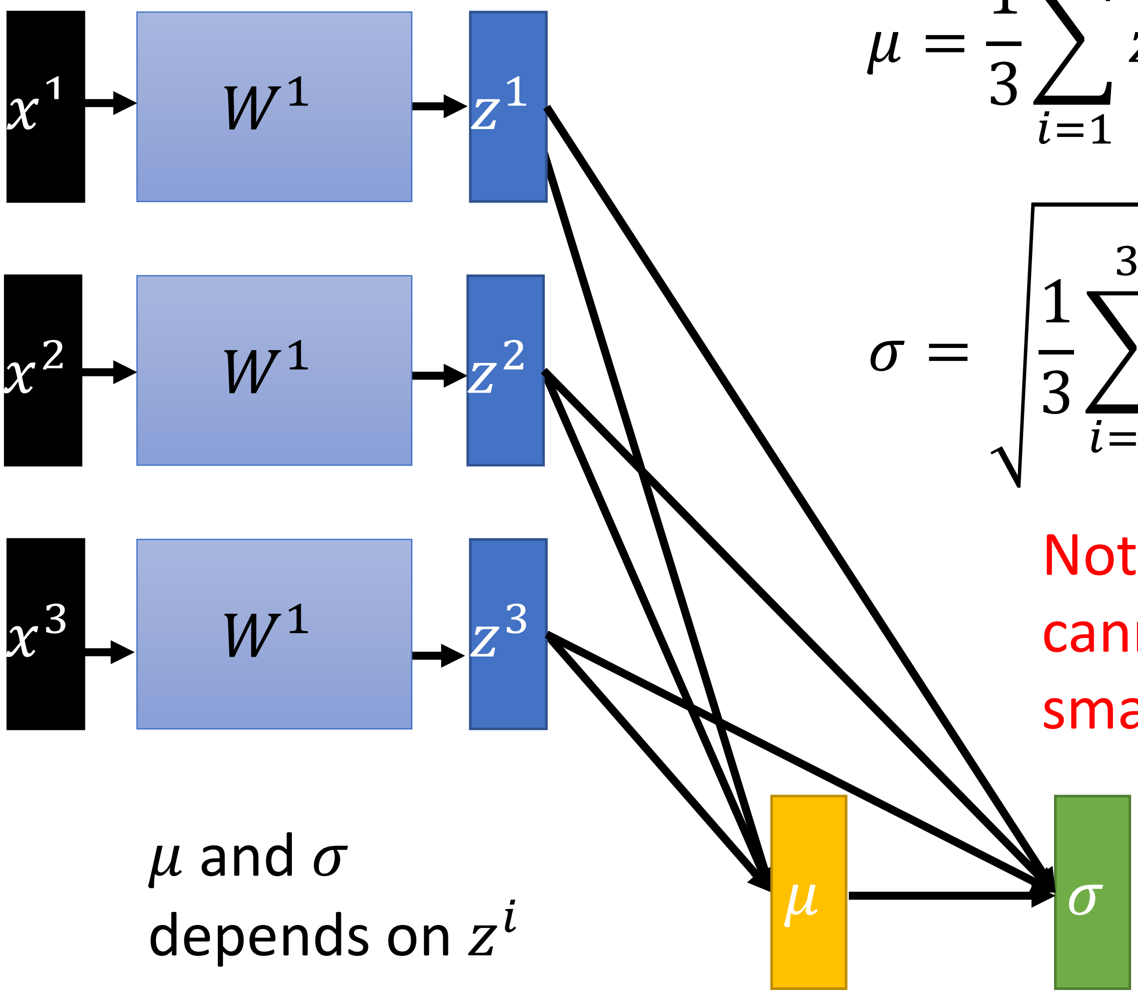
Batch



x^1
W^1
z^1
Sigmoid
a^1
W^2
......
x^2
W^1
z^2
Sigmoid
a^2
W^2
......
x^3
W^1
z^3
Sigmoid
a^3
W^2
......
z^1 z^2 z^3
=
W^1
x^1 x^2 x^3

Batch

Batch normalization



$$\mu = \frac{1}{3}\sum_{i=1}^{3} z^i$$

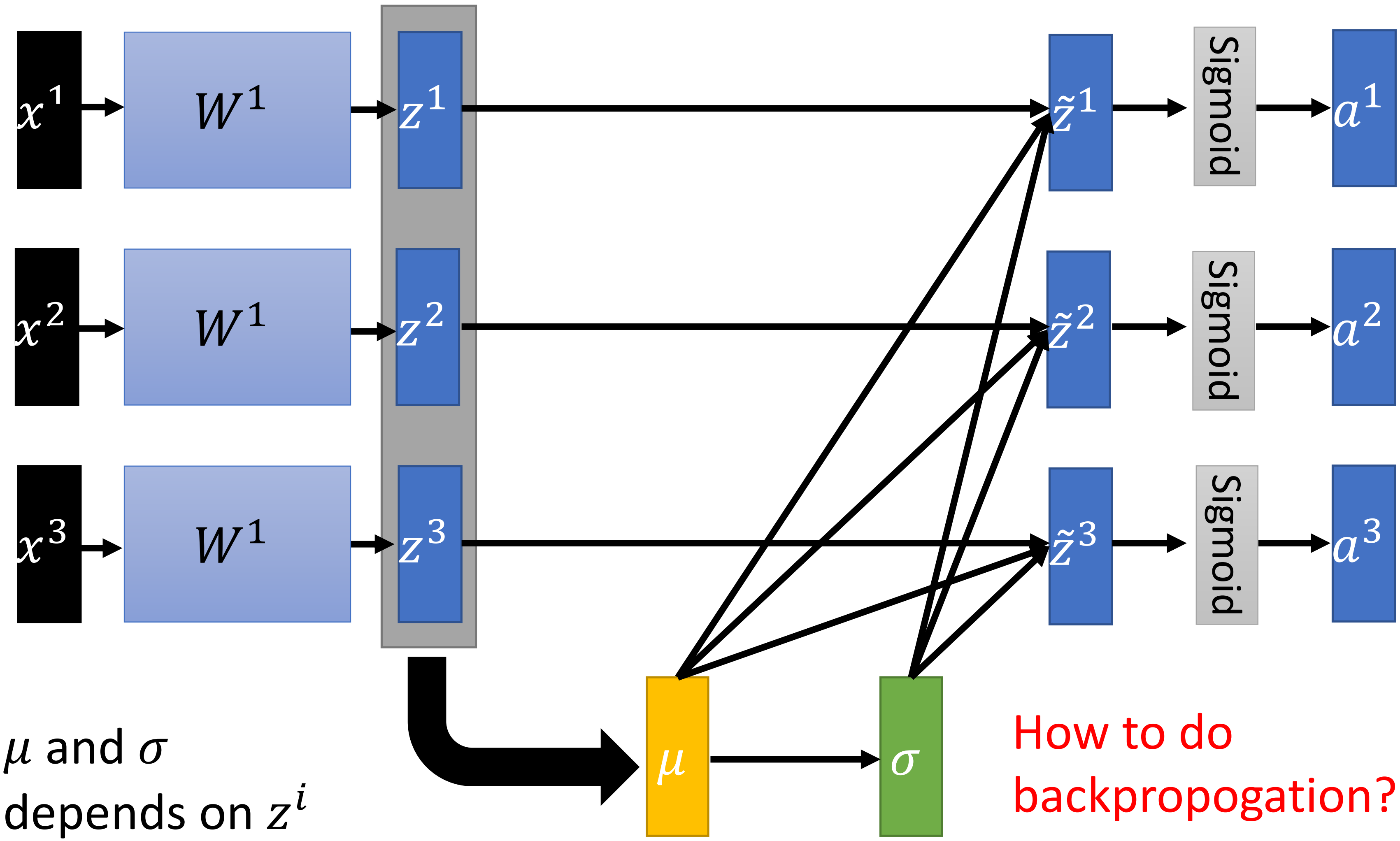
$$\sigma = \sqrt{\frac{1}{3}\sum_{i=1}^{3} (z^i - \mu)^2}$$

Note: Batch normalization cannot be applied on small batch.

μ and σ depends on z^i

Batch normalization

$$\tilde{z}^i = \frac{z^i - \mu}{\sigma}$$



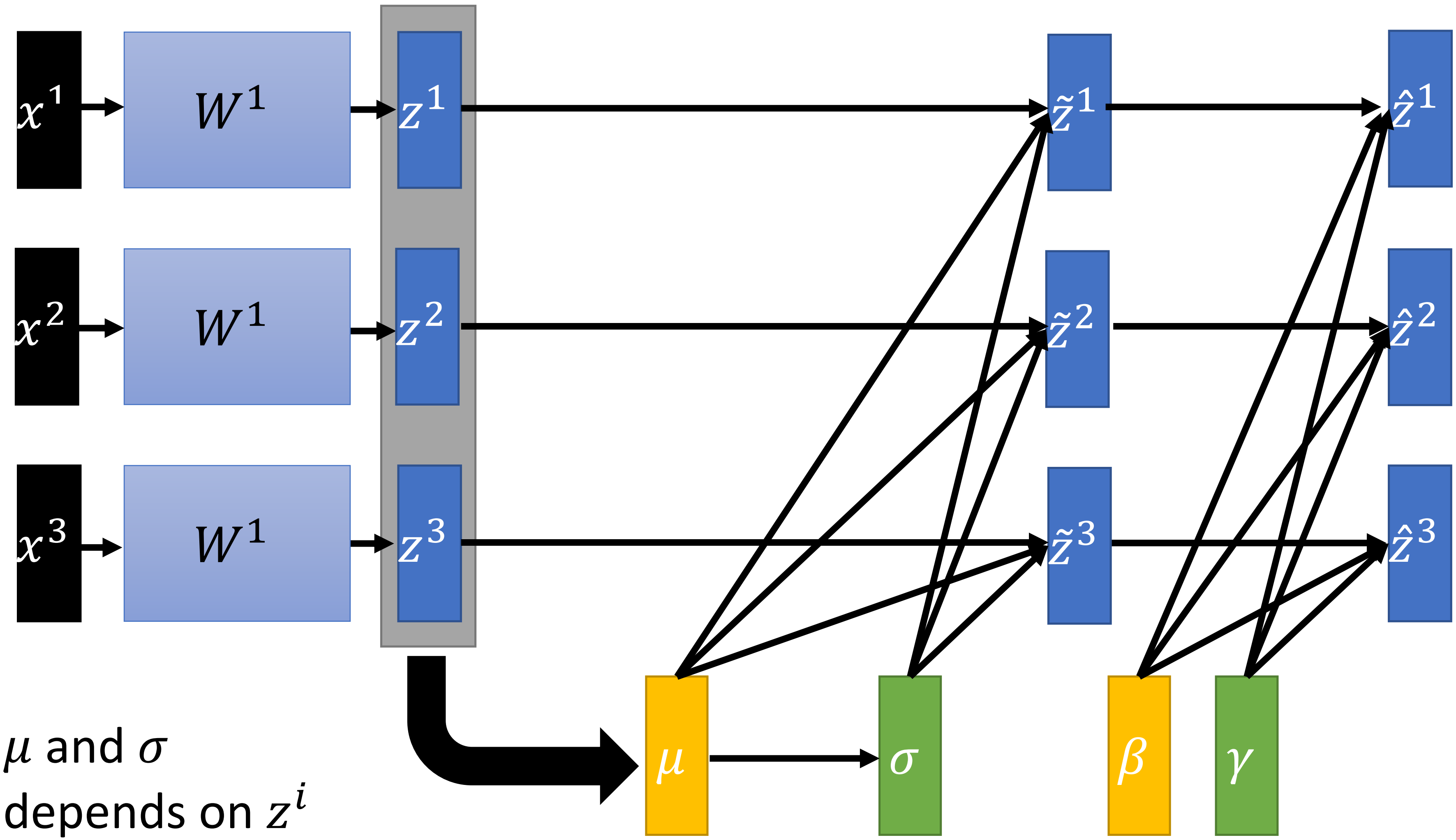
μ and σ depends on z^i

How to do backpropogation?

Batch normalization

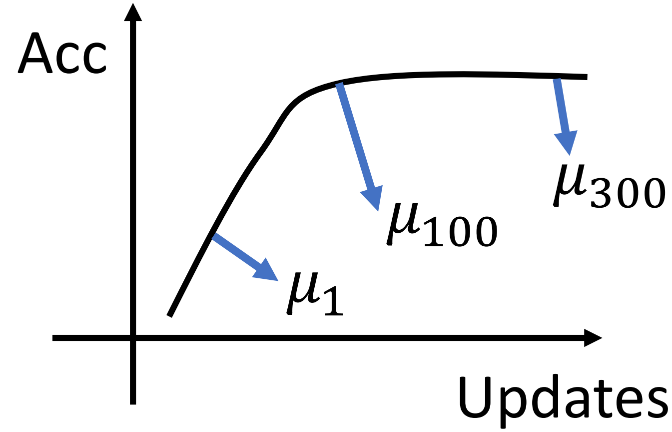
$$\tilde{z}^i = \frac{z^i - \mu}{\sigma}$$

$$\hat{z}^i = \gamma \odot \tilde{z}^i + \beta$$

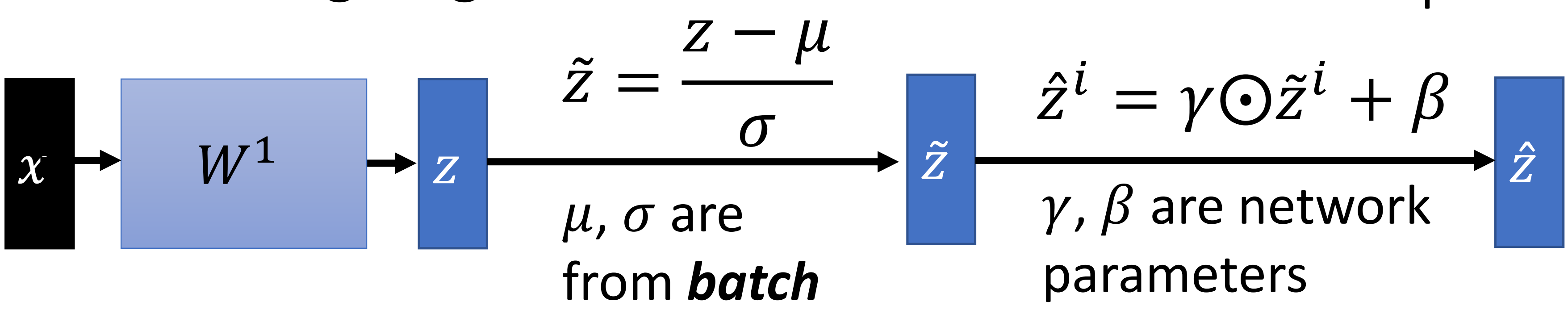


μ and σ depends on z^i

Batch normalization



- At testing stage:



We do not have ***batch*** at testing stage.

Ideal solution:

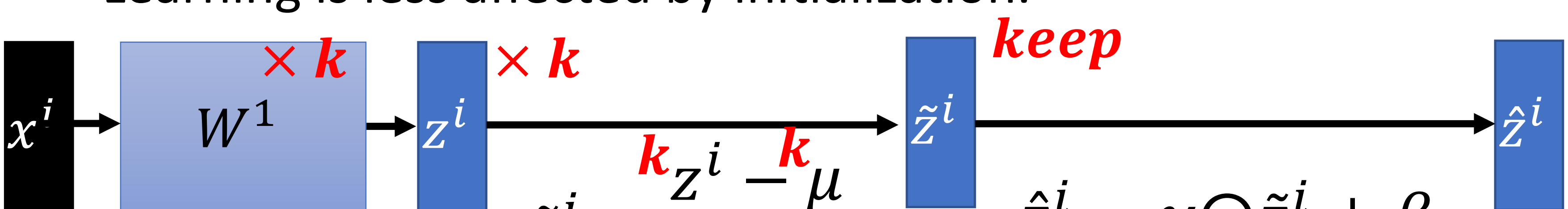
Computing μ and σ using the whole training dataset.

Practical solution:

Computing the moving average of μ and σ of the batches during training.

Batch normalization - Benefit

- BN reduces training times, and make very deep net trainable.
  - Because of less Covariate Shift, we can use larger learning rates.
  - Less exploding/vanishing gradients
    - Especially effective for sigmoid, tanh, etc.
- Learning is less affected by initialization.



$$\tilde{z}^i = \frac{\boldsymbol{k}z^i - \boldsymbol{k}\mu}{\boldsymbol{k}\sigma}$$

$$\hat{z}^i = \gamma \odot \tilde{z}^i + \beta$$

- BN reduces the demand for regularization.

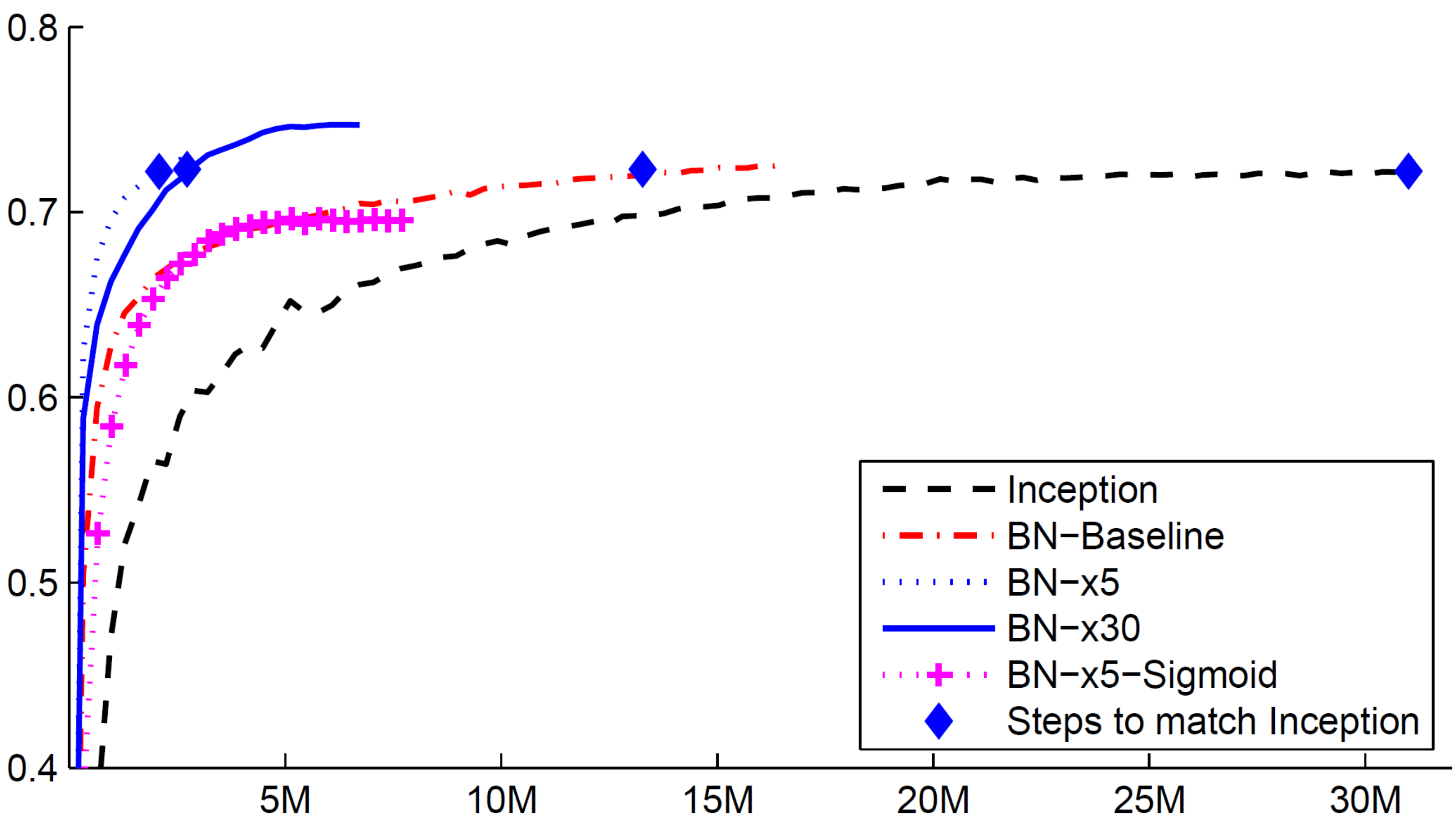


Figure 2: *Single crop validation accuracy of Inception and its batch-normalized variants, vs. the number of training steps.*

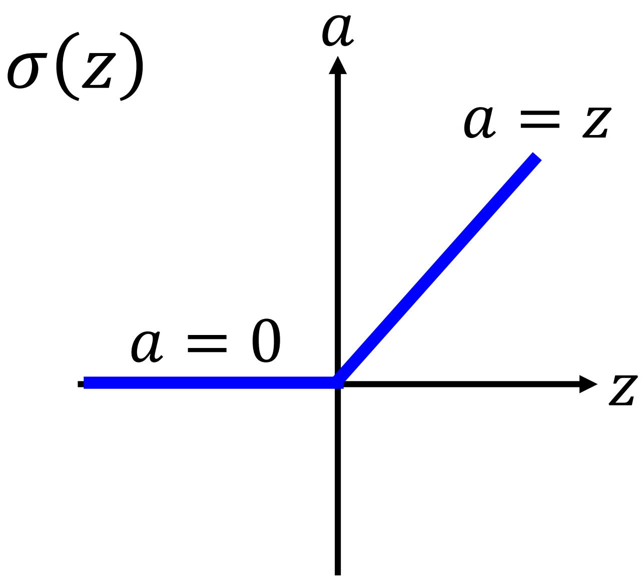
Demo

Activation Function

Günter Klambauer, Thomas Unterthiner, Andreas Mayr, Andreas Mayr, "Self-Normalizing Neural Networks", NIPS, 2017

ReLU

- Rectified Linear Unit (ReLU)



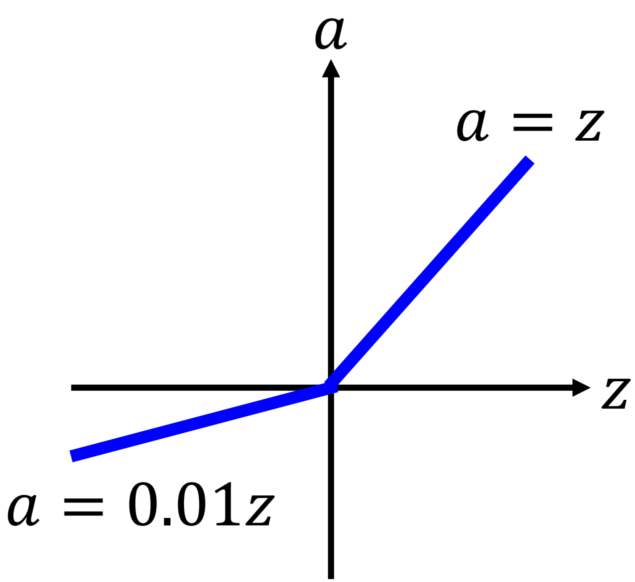
[Xavier Glorot, AISTATS'11]
[Andrew L. Maas, ICML'13]
[Kaiming He, arXiv'15]

Reason:

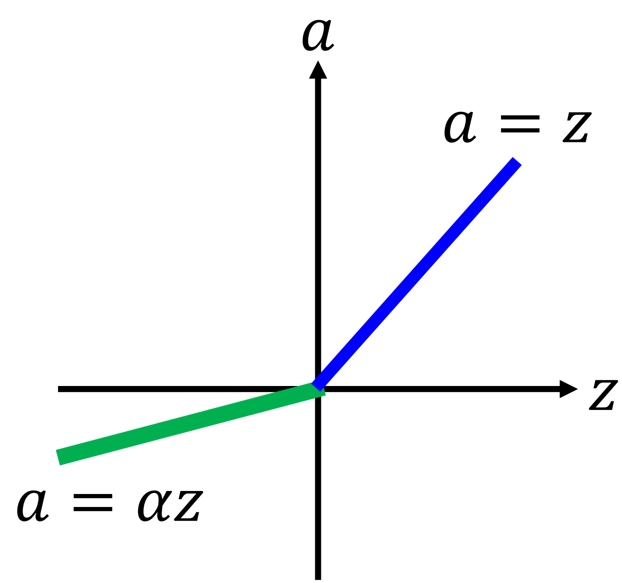
1. Fast to compute
2. Biological reason
3. Infinite sigmoid with different biases
4. Vanishing gradient problem

ReLU - variant

Leaky ReLU



Parametric ReLU

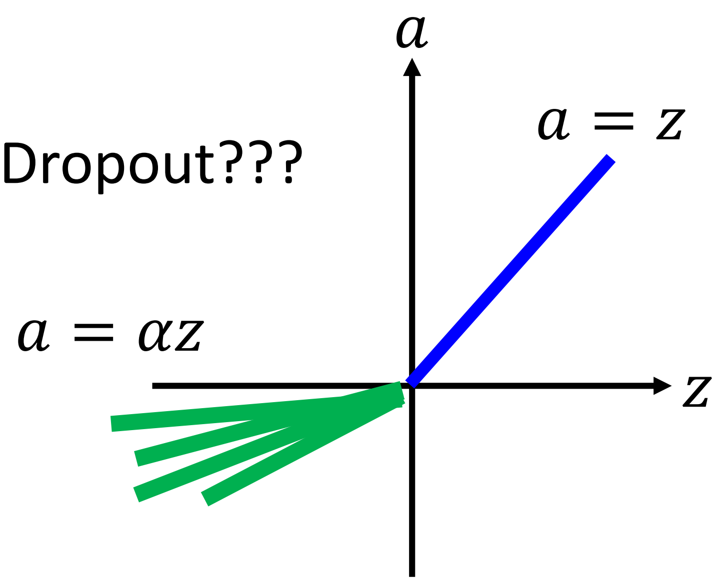


α also learned by gradient descent

ReLU - variant

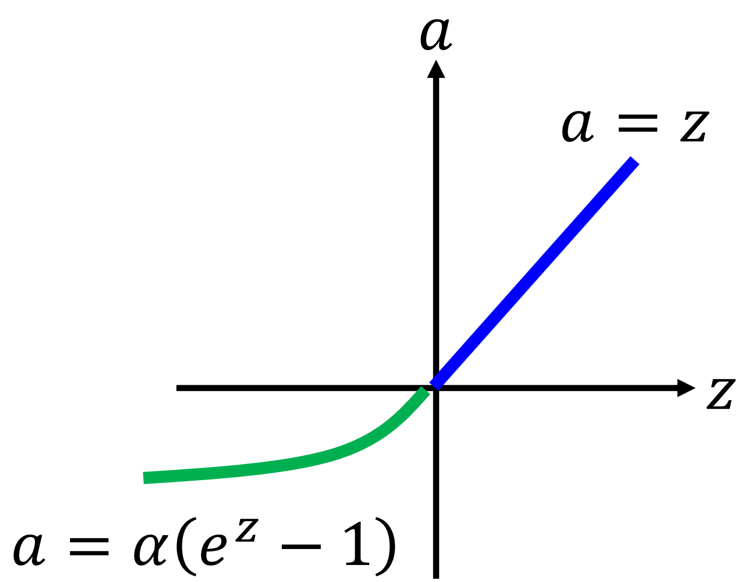
Randomized ReLU

Dropout???



α is sampled from a distribution during training. Fixed during testing.

Exponential Linear Unit (ELU)

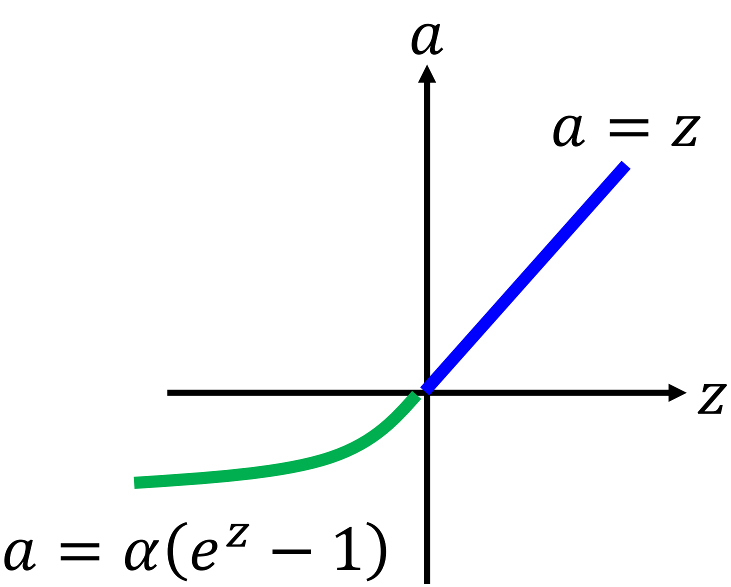


(1) Definition of scaled exponential linear units (SELUs)

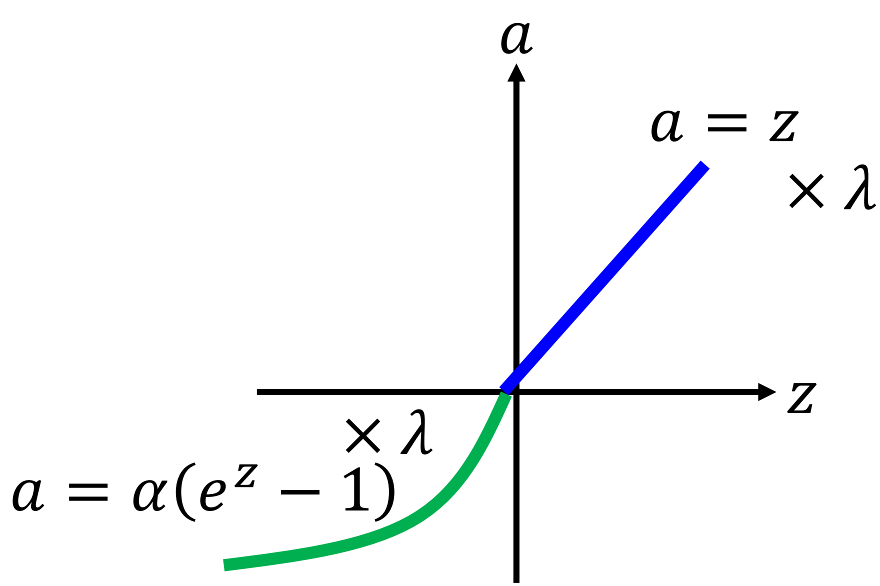
In [3]:
```
def selu(x):
    with ops.name_scope('elu') as scope:
        alpha = 1.6732632423543772848170429916717
        scale = 1.0507009873554804934193349852946
        return scale*tf.where(x>=0.0, x, alpha*tf.nn.elu(x))
```

https://github.com/bioinf-jku/SNNs

Exponential Linear Unit (ELU)



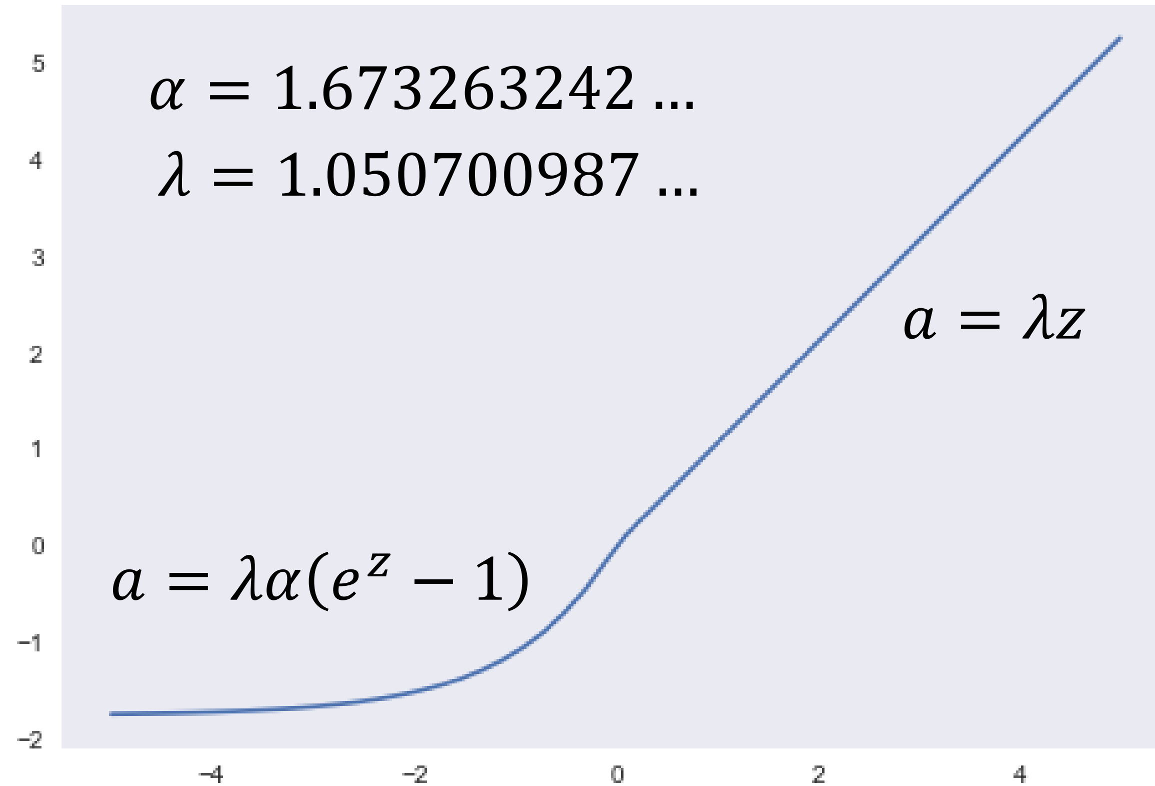
Scaled ELU (SELU)



$$\alpha = 1.6732632423543772848170429916717$$

$$\lambda = 1.0507009873554804934193349852946$$

SELU



Positive and negative values

➡ The whole ReLU family has this property except the original ReLU.

Saturation region ➡ ELU also has this property

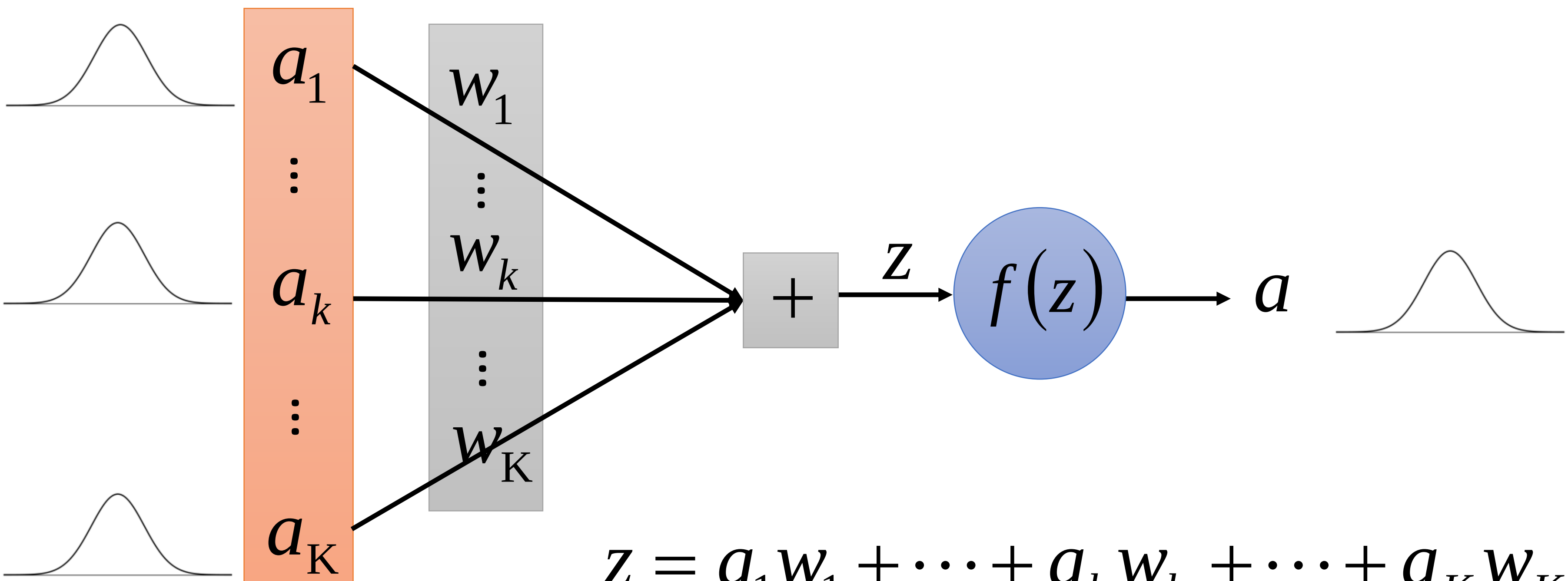
Slope larger than 1 ➡ Only SELU also has this property

SELU

The inputs are i.i.d random variables with mean $\underbrace{\mu}_{=0}$ and variance $\underbrace{\sigma^2}_{=1}$.

$$\mu_z = E[z]$$

$$= \sum_{k=1}^{K} \underbrace{E[a_k]}_{\mu} w_k = \mu \underbrace{\sum_{k=1}^{K} w_k}_{=0} = \mu \cdot K \underbrace{\mu_w}_{=0}$$



$$z = a_1 w_1 + \cdots + a_k w_k + \cdots + a_K w_K$$

Do not have to be Gaussian

SELU

$\mu_z = 0 \qquad \mu_w = 0$

$$\sigma_z^2 = E[(z-\mu_z)^2] = E[z^2]$$

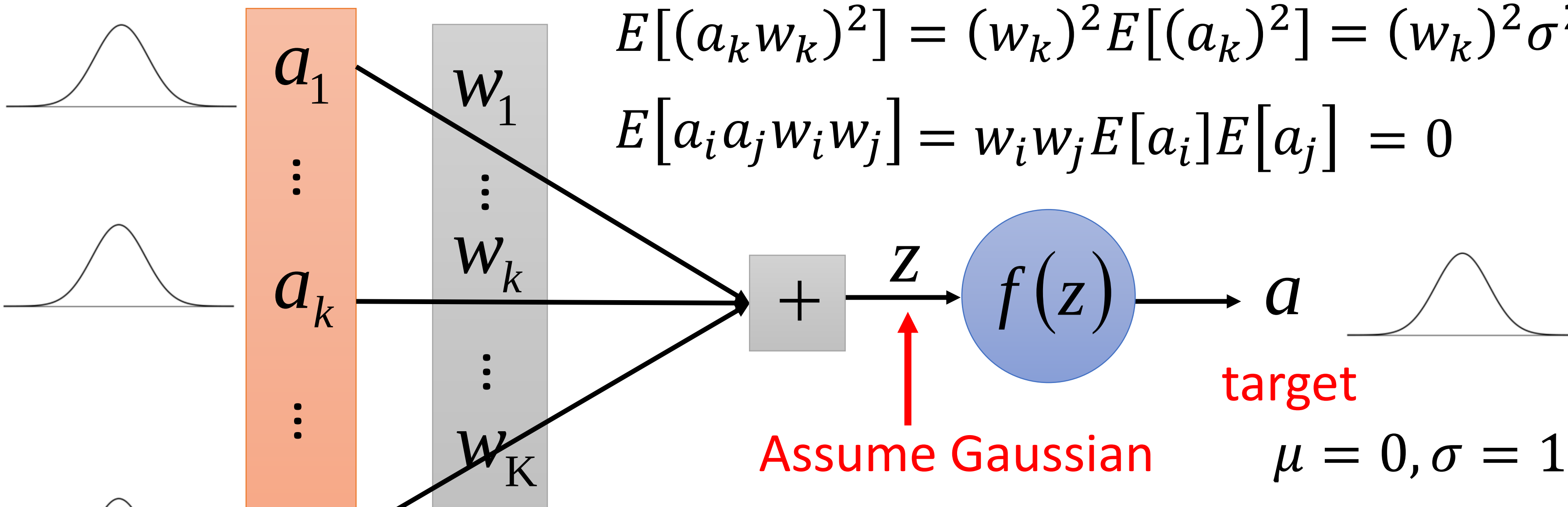
$$= E[(a_1 w_1 + a_2 w_2 + \cdots)^2]$$

$$= \sum_{k=1}^{K} (w_k)^2 \sigma^2 = \underbrace{\sigma^2}_{=1} \cdot \underbrace{K\sigma_w^2}_{=1} = 1$$

The inputs are i.i.d random variables with mean $\underbrace{\mu}_{=0}$ and variance $\underbrace{\sigma^2}_{=1}$.

$$E[(a_k w_k)^2] = (w_k)^2 E[(a_k)^2] = (w_k)^2 \sigma^2$$

$$E\left[a_i a_j w_i w_j\right] = w_i w_j E[a_i] E\left[a_j\right] = 0$$



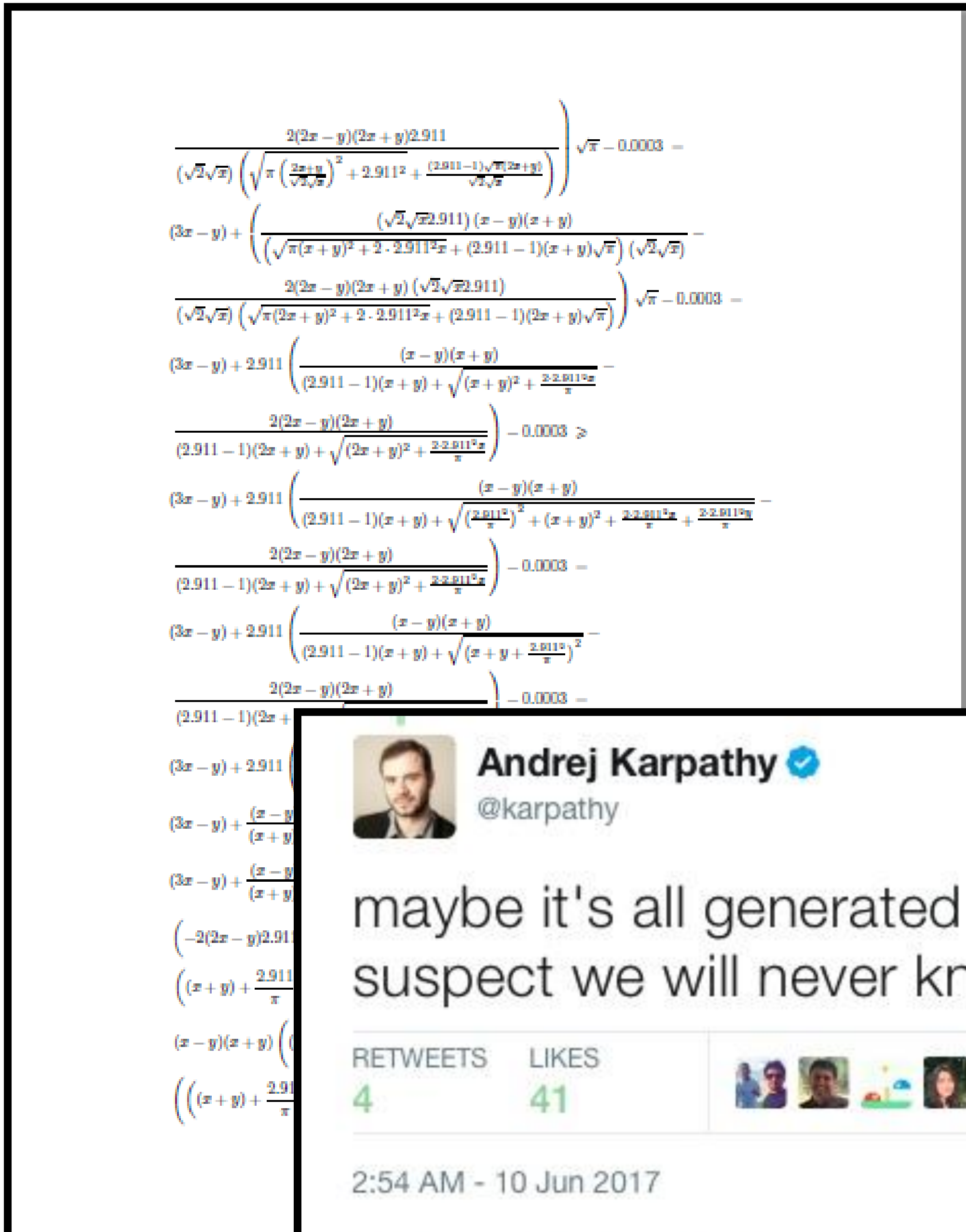
$$z = a_1 w_1 + \cdots + a_k w_k + \cdots + a_K w_K$$

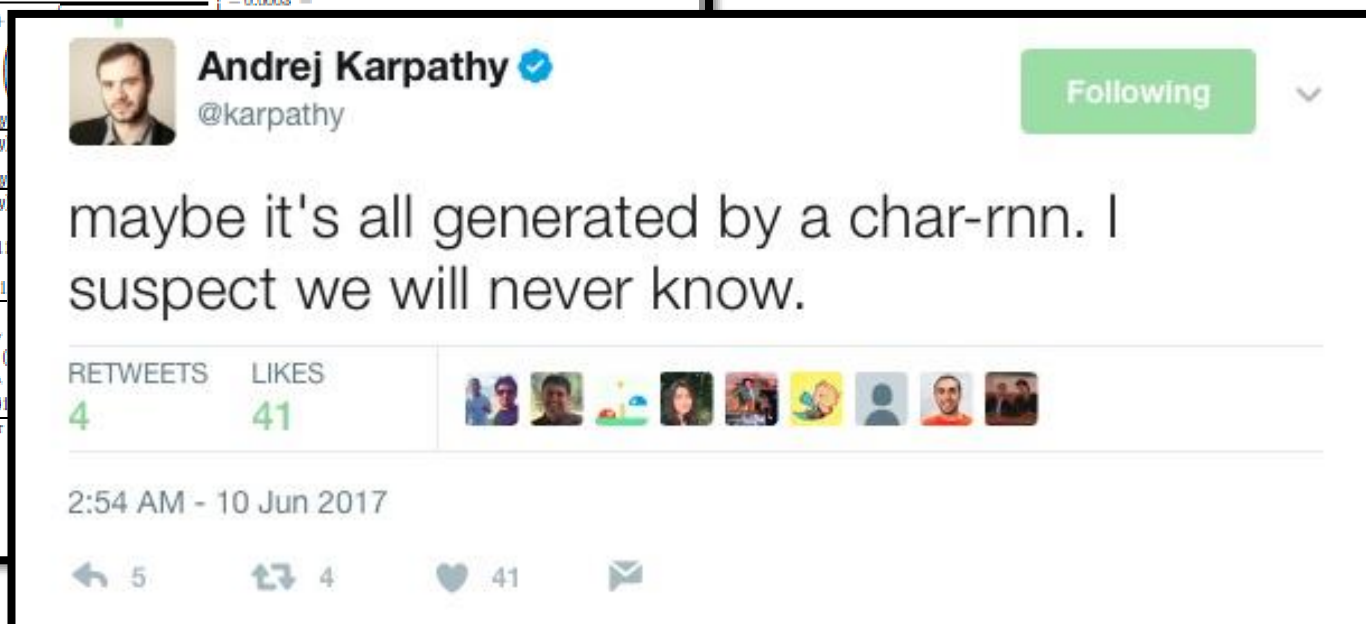
Demo

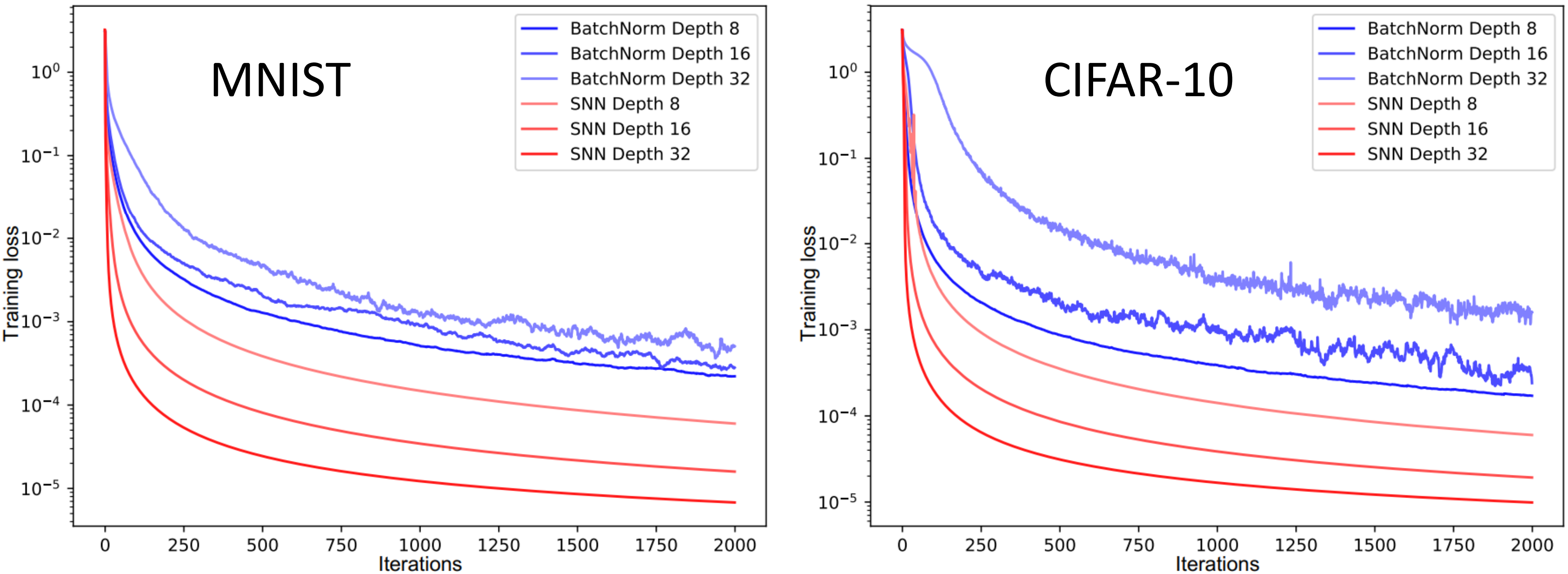
93 頁的證明

Source of joke: https://zhuanlan.zhihu.com/p/27336839

SELU is actually more general.

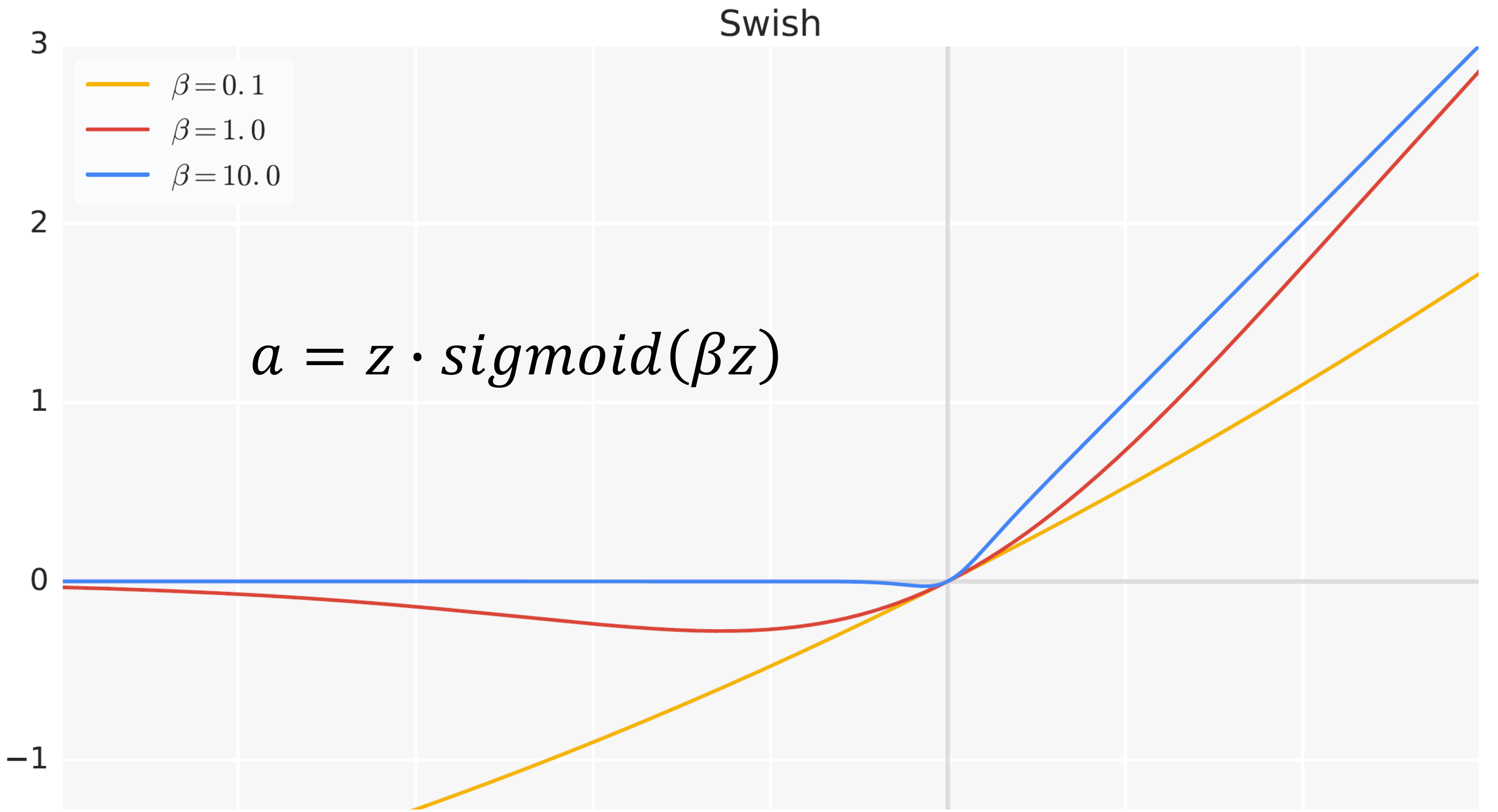






FNN method comparison			ML method comparison		
Method	avg. rank diff.	p-value	Method	avg. rank diff.	p-value
SNN	-0.756		SNN	-6.7	
MSRAinit	-0.240*	2.7e-02	SVM	-6.4	5.8e-01
LayerNorm	-0.198*	1.5e-02	RandomForest	-5.9	2.1e-01
Highway	0.021*	1.9e-03	MSRAinit	-5.4*	4.5e-03
ResNet	0.273*	5.4e-04	LayerNorm	-5.3	7.1e-02
WeightNorm	0.397*	7.8e-07	Highway	-4.6*	1.7e-03
BatchNorm	0.504*	3.5e-06	…	…	…

Demo

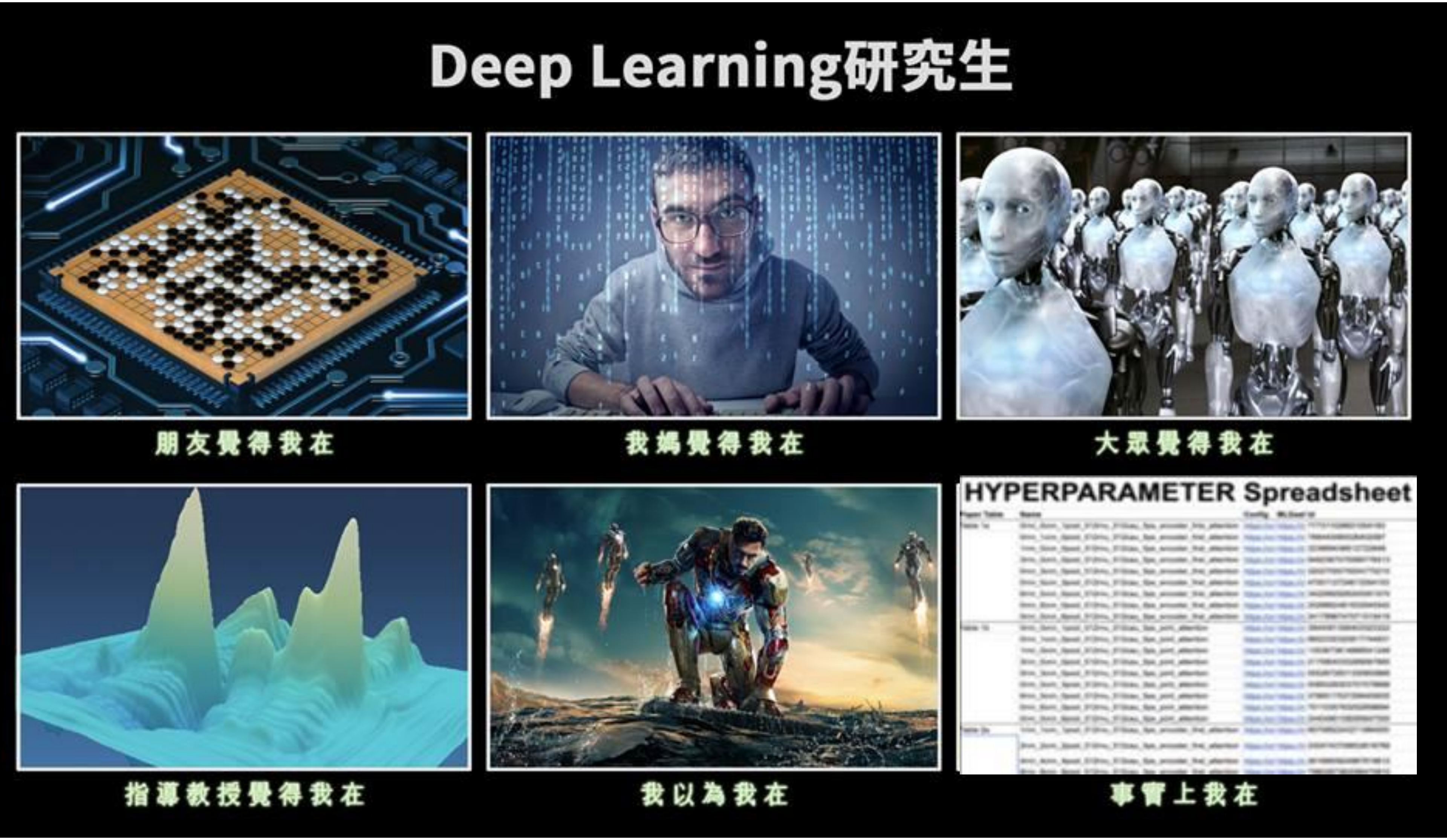


Baselines	**ReLU**	**LReLU**	**PReLU**	**Softplus**	**ELU**	**SELU**	**GELU**
Swish > Baseline	9	7	6	6	8	8	8
Swish = Baseline	0	1	3	2	0	1	1
Swish < Baseline	0	1	0	1	1	0	0

Figure 4: The Swish activation function.

Hyperparameters

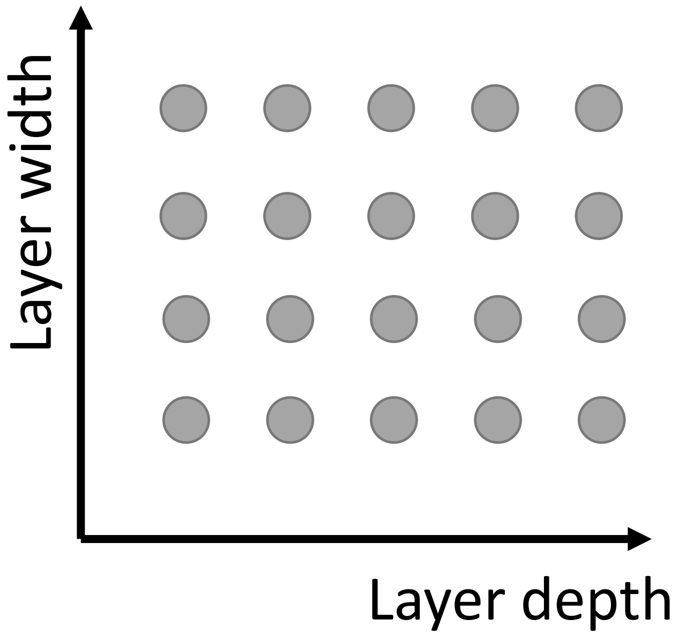
Source of iamge: https://medium.com/intuitionmachine/the-brute-force-method-of-deep-learning-innovation-58b497323ae5 (Denny Britz's graphic)

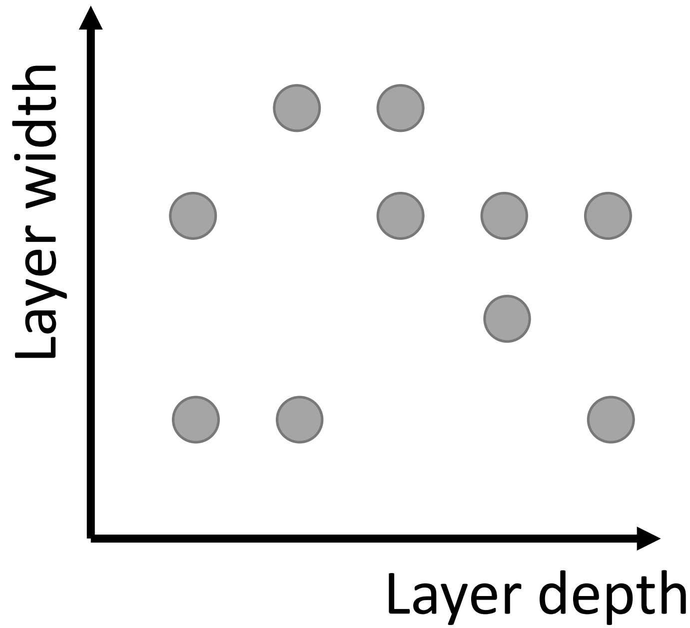


感謝 沈昇勳 同學提供圖檔

http://www.deeplearningbook.org/contents/guidelines.html

Grid Search v.s. Random Search





Assumption: top K results are good enough

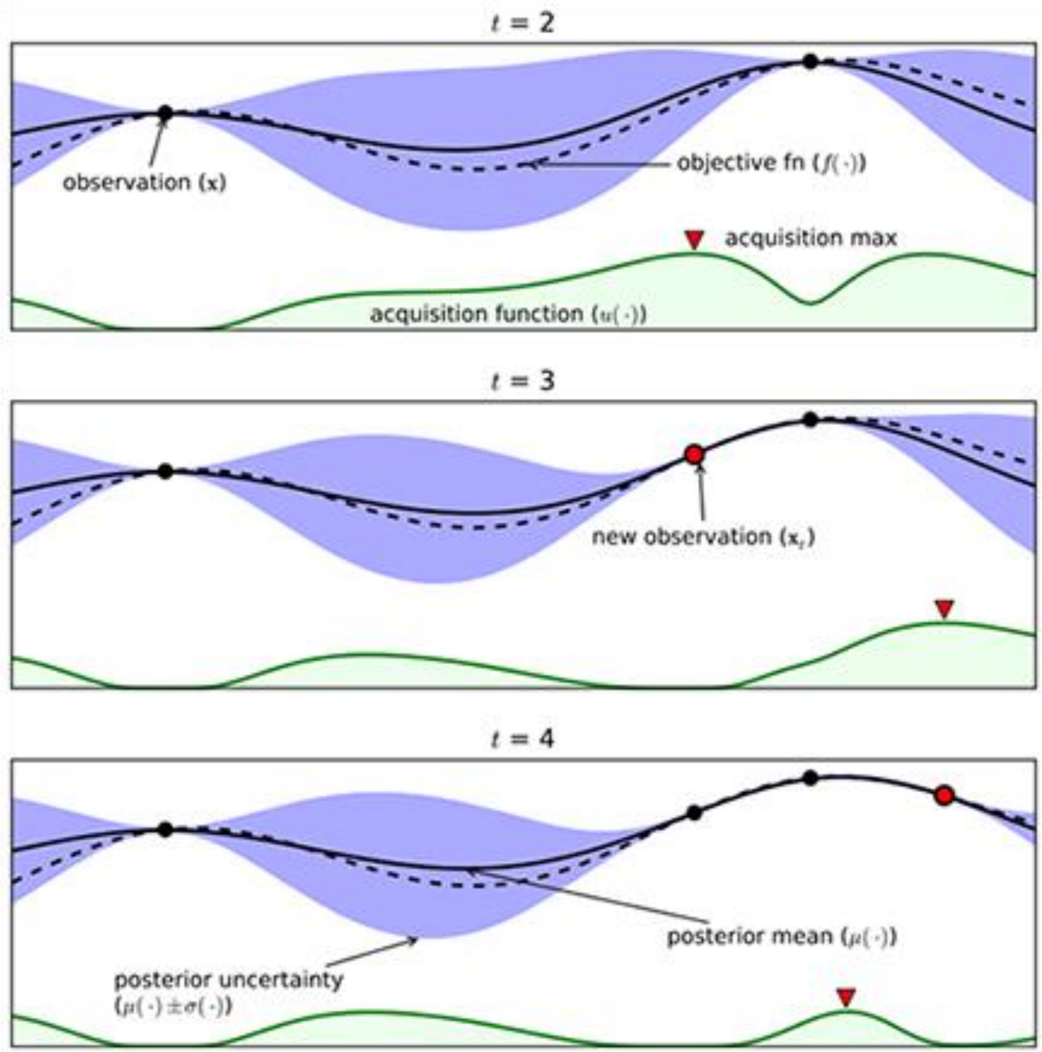
If there are N points, probability K/N that your sample is in top K

Sample x times: $1-(1-K/N)^x$ > 90%

If N = 1000, K = 10 → x = 230

K = 100 → x = 22

Model-based Hyperparameter Optimization

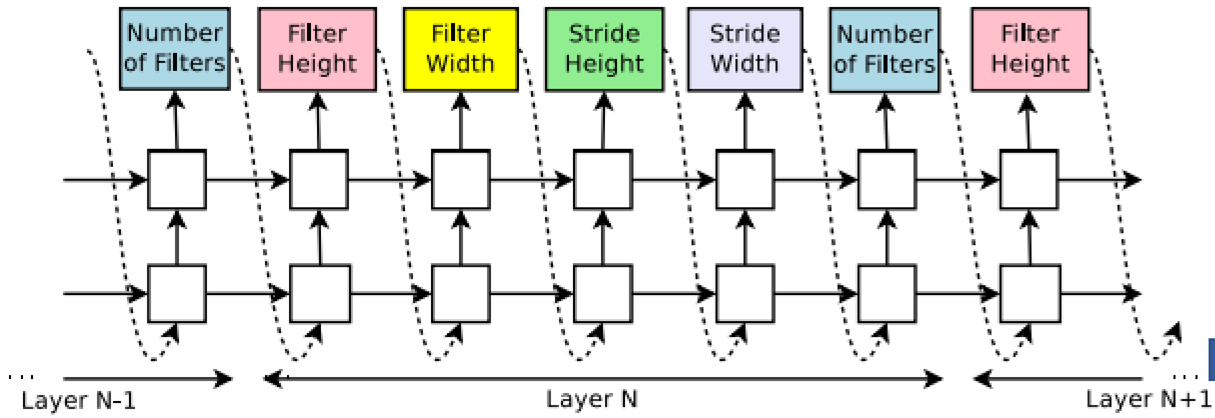


https://cloud.google.com/blog/big-data/2017/08/hyperparameter-tuning-in-cloud-machine-learning-engine-using-bayesian-optimization

Reinforcement Learning

It can design LSTM as shown in the previous lecture.

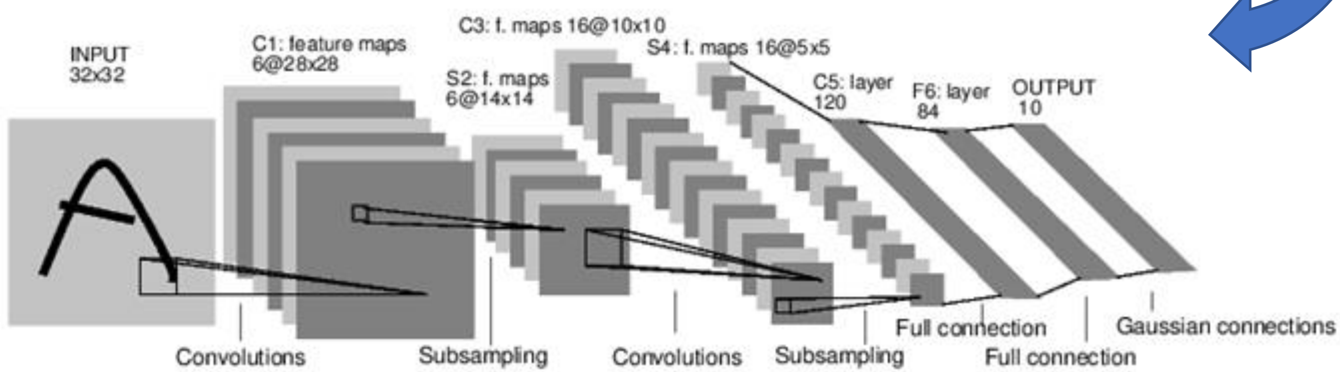
One kind of meta learning (or learn to learn)



Design a network

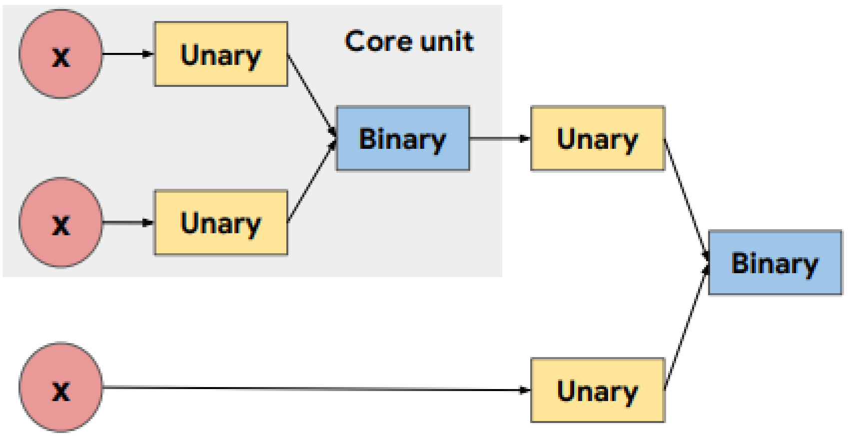
Accuracy as reward

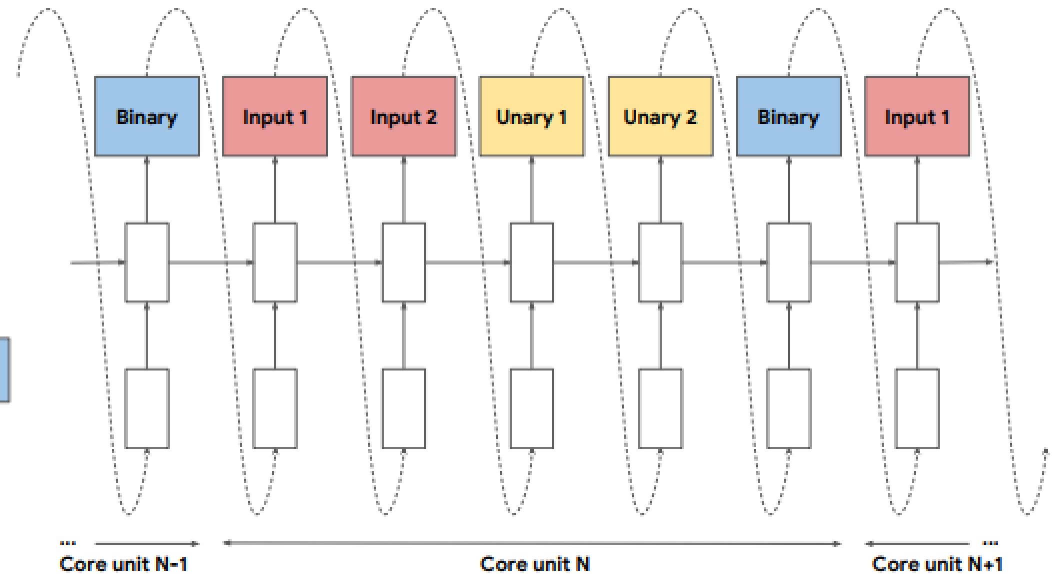
Train the network



A Full Convolutional Neural Network (LeNet)

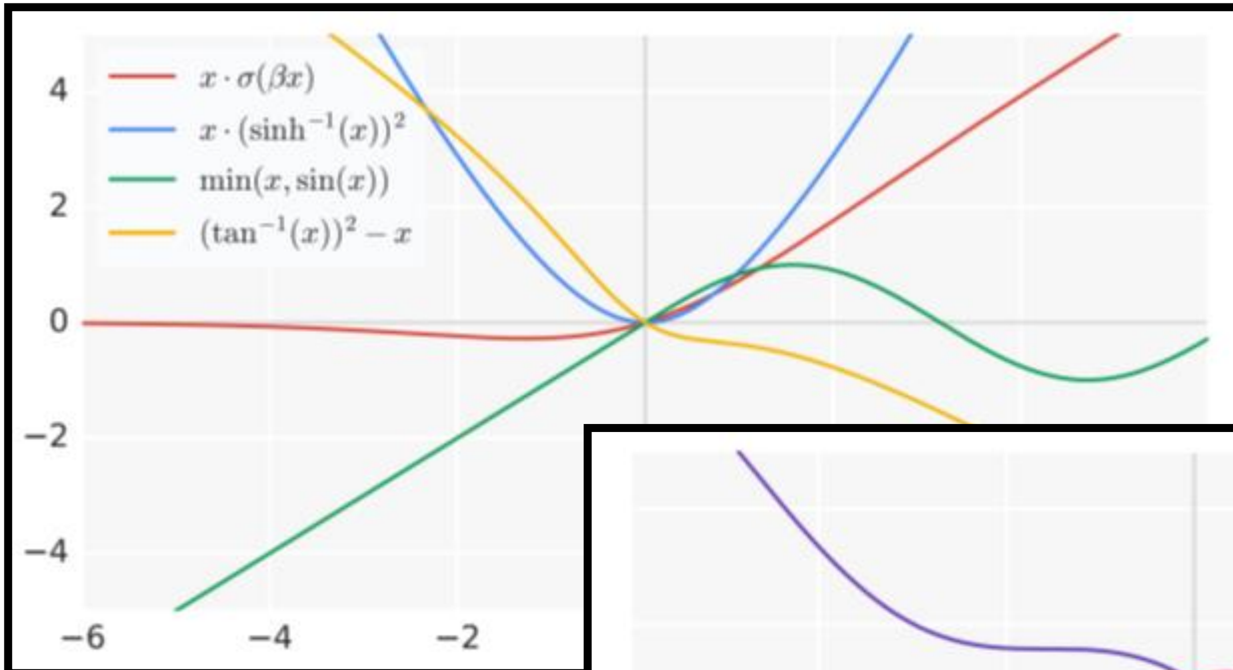
SWISH ......

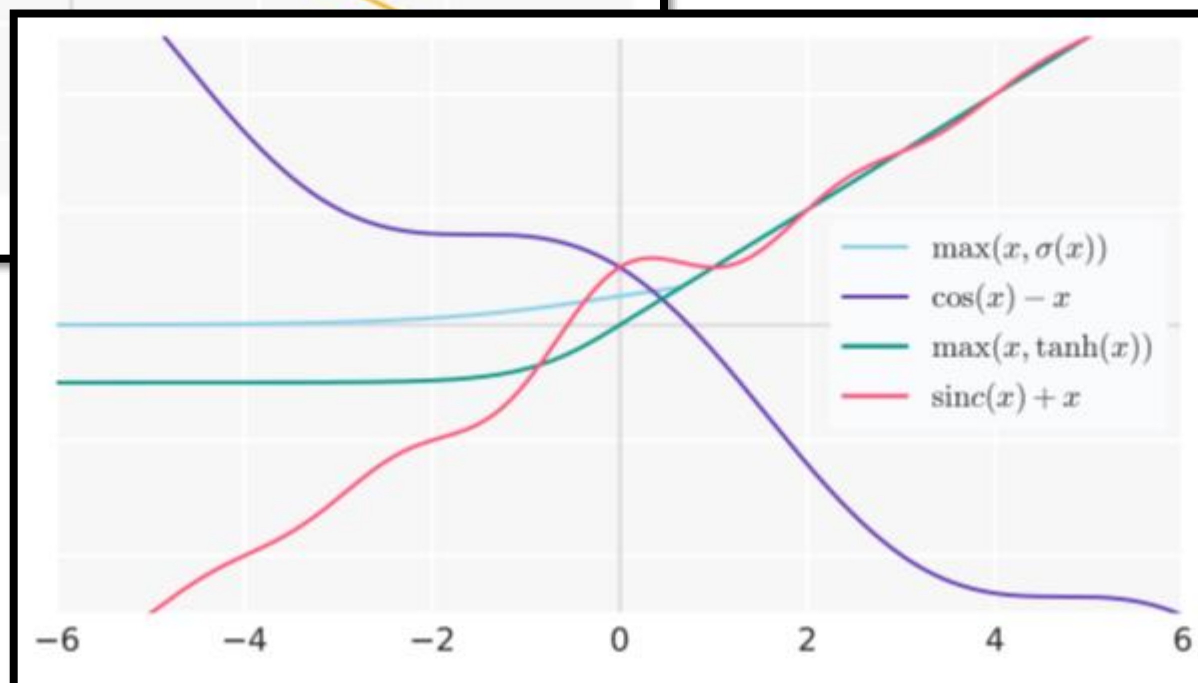




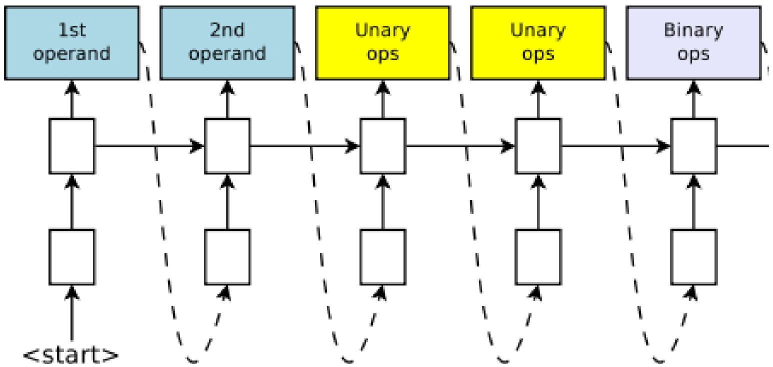
- **Unary functions**: x, $-x$, $|x|$, x^2, x^3, $\sqrt{x}$, βx, $x+\beta$, $\log(|x|+\epsilon)$, $\exp(x)$ $\sin(x)$, $\cos(x)$, $\sinh(x)$, $\cosh(x)$, $\tanh(x)$, $\sinh^{-1}(x)$, $\tan^{-1}(x)$, $\text{sinc}(x)$, $\max(x,0)$, $\min(x,0)$, $\sigma(x)$, $\log(1+\exp(x))$, $\exp(-x^2)$, $\text{erf}(x)$, β
- **Binary functions**: x_1+x_2, $x_1 \cdot x_2$, $x_1 - x_2$, $\frac{x_1}{x_2+\epsilon}$, $\max(x_1,x_2)$, $\min(x_1,x_2)$, $\sigma(x_1)\cdot x_2$, $\exp(-\beta(x_1-x_2)^2)$, $\exp(-\beta|x_1-x_2|)$, $\beta x_1 + (1-\beta)x_2$

SWISH ……

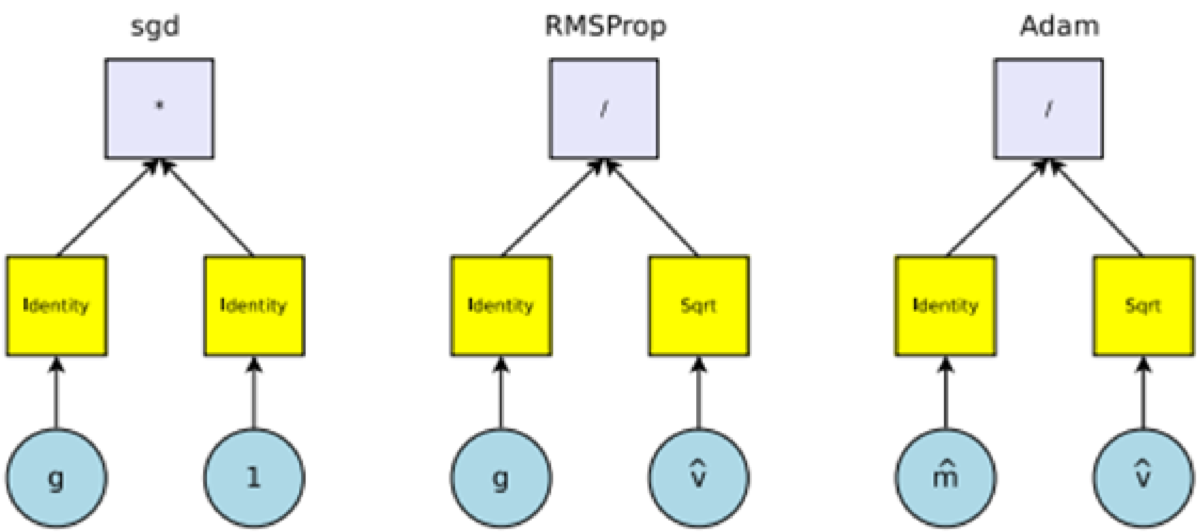

$x \cdot \sigma(\beta x)$
$x \cdot (\sinh^{-1}(x))^2$
$\min(x, \sin(x))$
$(\tan^{-1}(x))^2 - x$
4
2
0
−2
−4
−6
−4
−2

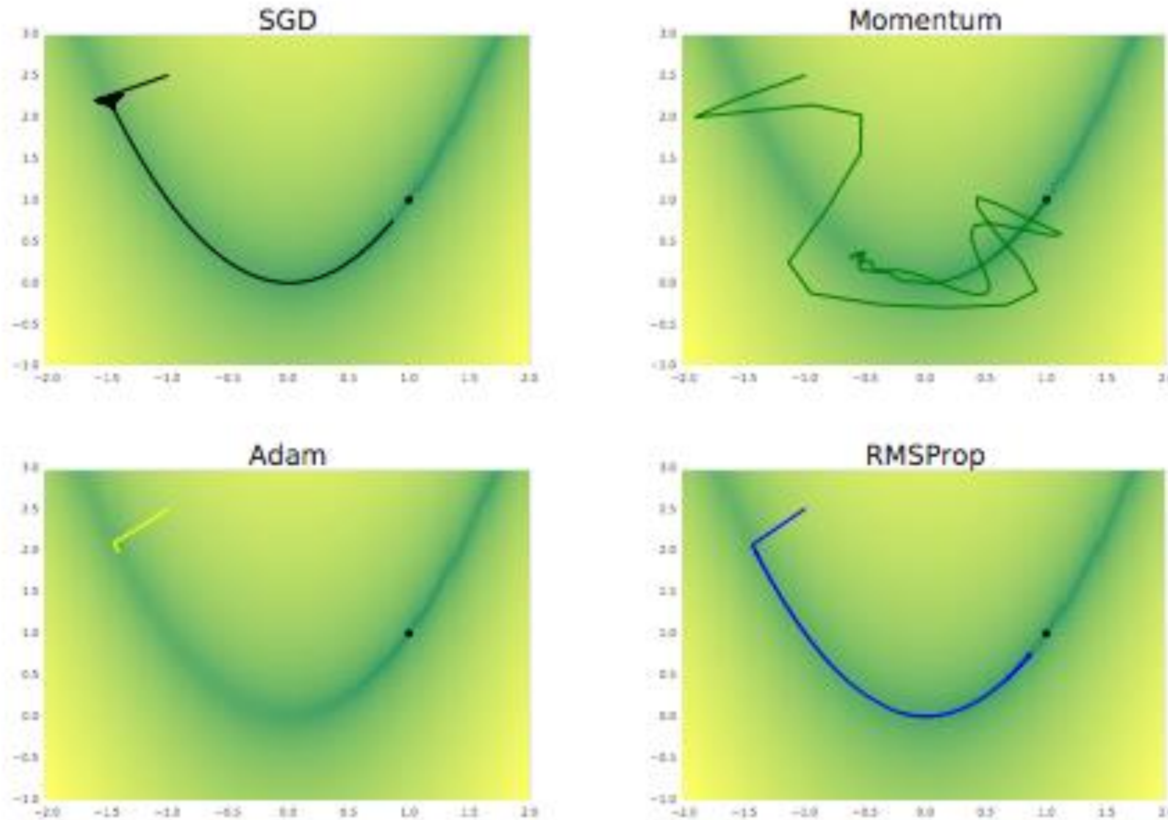

$\max(x, \sigma(x))$
$\cos(x) - x$
$\max(x, \tanh(x))$
$\mathrm{sinc}(x) + x$
−6
−4
−2
0
2
4
6

Learning Rate

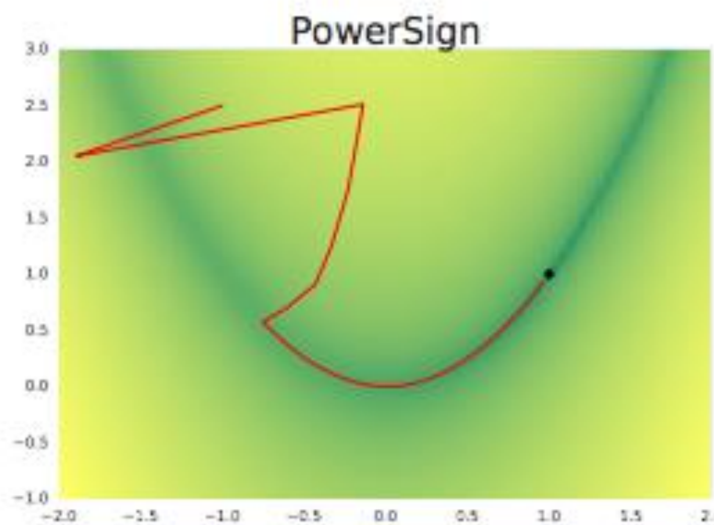


- **Operands:** g, g^2, g^3, $\hat{m}$, $\hat{v}$, $\hat{\gamma}$, $\text{sign}(g)$, $\text{sign}(\hat{m})$, 1, 2, $\epsilon \sim N(0, 0.01)$, $10^{-4}w$, $10^{-3}w$, $10^{-2}w$, $10^{-1}w$, Adam and RMSProp.
- **Unary functions** which map input x to: x, $-x$, e^x, $\log|x|$, $\sqrt{|x|}$, $clip(x, 10^{-5})$, $clip(x, 10^{-4})$, $clip(x, 10^{-3})$, $drop(x, 0.1)$, $drop(x, 0.3)$, $drop(x, 0.5)$ and $\text{sign}(x)$.
- **Binary functions** which map (x, y) to $x + y$ (addition), $x - y$ (subtraction), $x * y$ (multiplication), $\frac{x}{y+\delta}$ (division), x^y (exponentiation) or x (keep left).





$$e^{\mathrm{sign}(g)*\mathrm{sign}(m)} * g$$



Can transfer to new tasks

Capsule

Sara Sabour, Nicholas Frosst, Geoffrey E. Hinton, “Dynamic Routing Between Capsules”, NIPS, 2017

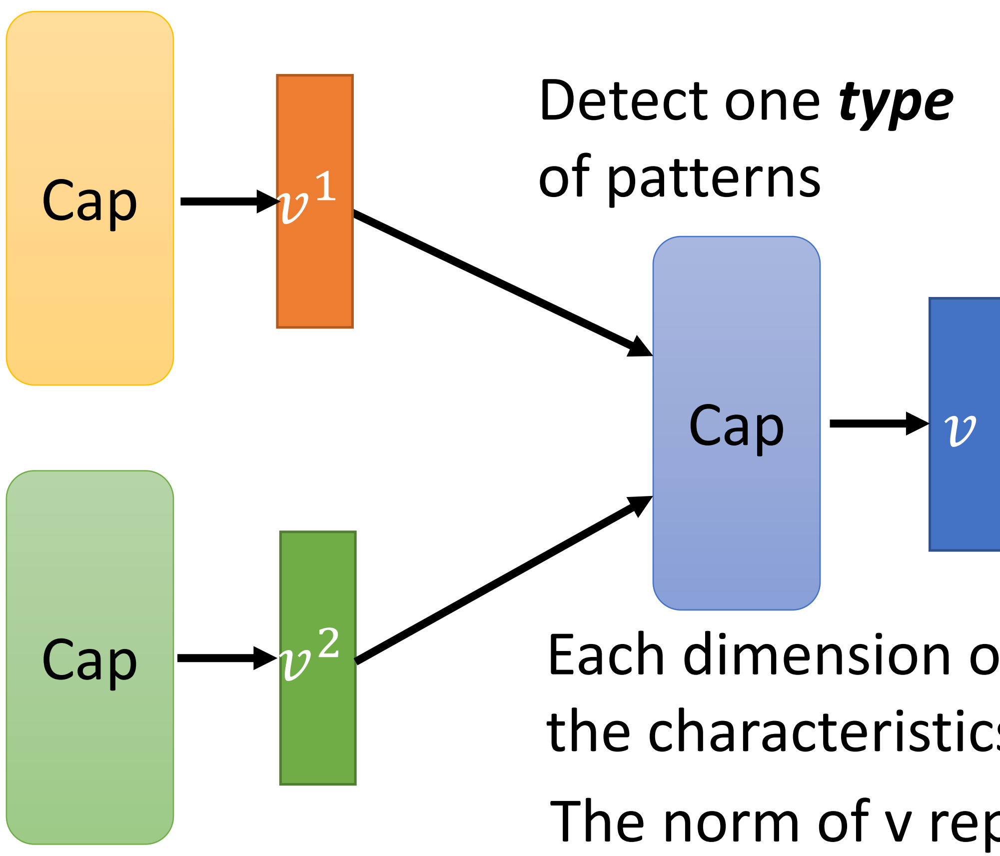
Capsule

A neuron detects a specific pattern.

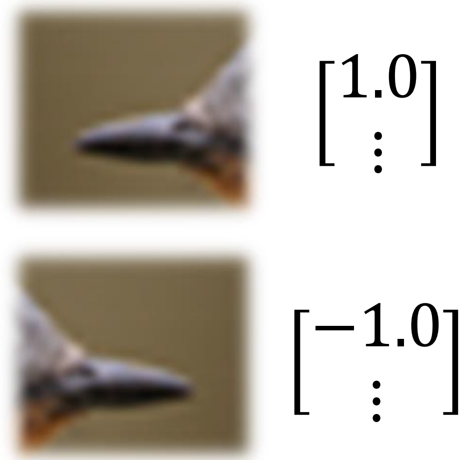
Neuron A

Neuron B

- Neuron: output a value, Capsule: output a vector



Detect one ***type*** of patterns



Each dimension of v represents the characteristics of patterns.

The norm of v represents the existence.

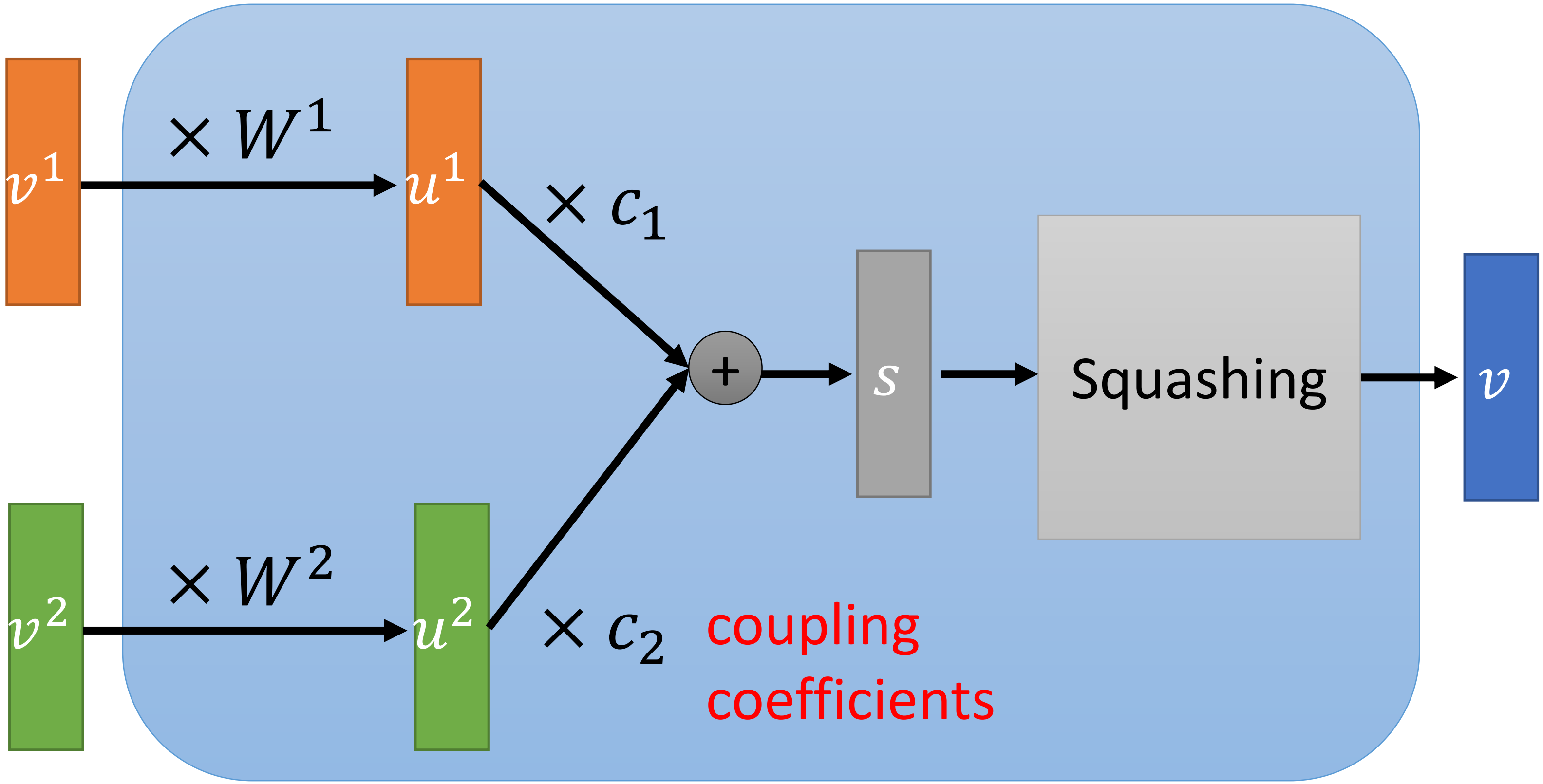
Capsule

$$u^1 = W^1 v^1 \qquad u^2 = W^2 v^2$$

$$s = c_1 u^1 + c_2 u^2$$

$$v = Squash(s)$$

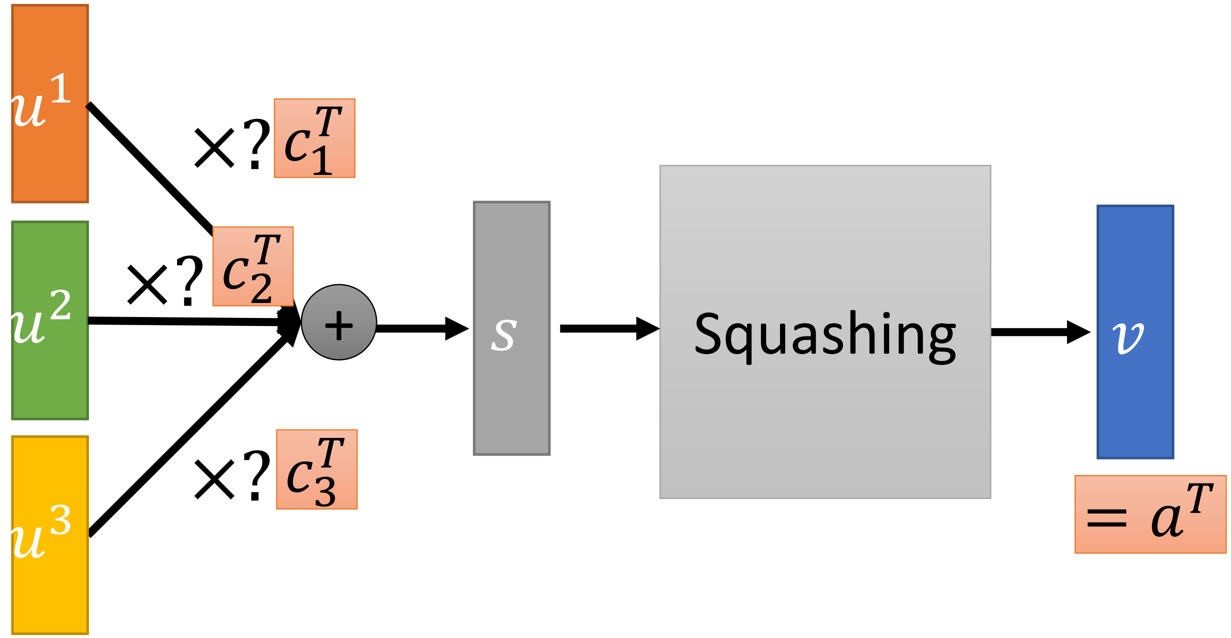
$$v = \frac{\|s\|^2}{1 + \|s\|^2} \frac{s}{\|s\|}$$



c are determined by ***dynamic routing*** during the testing stage.

c.f. pooling

Dynamic Routing



$b_1^0 = 0, b_2^0 = 0, b_3^0 = 0$

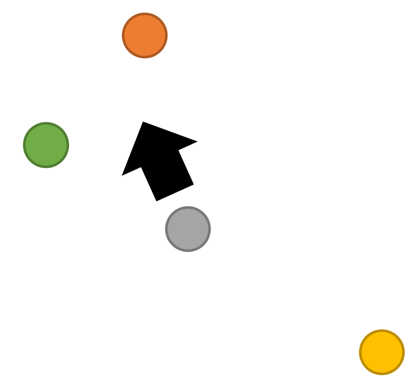
For r = 1 to T do

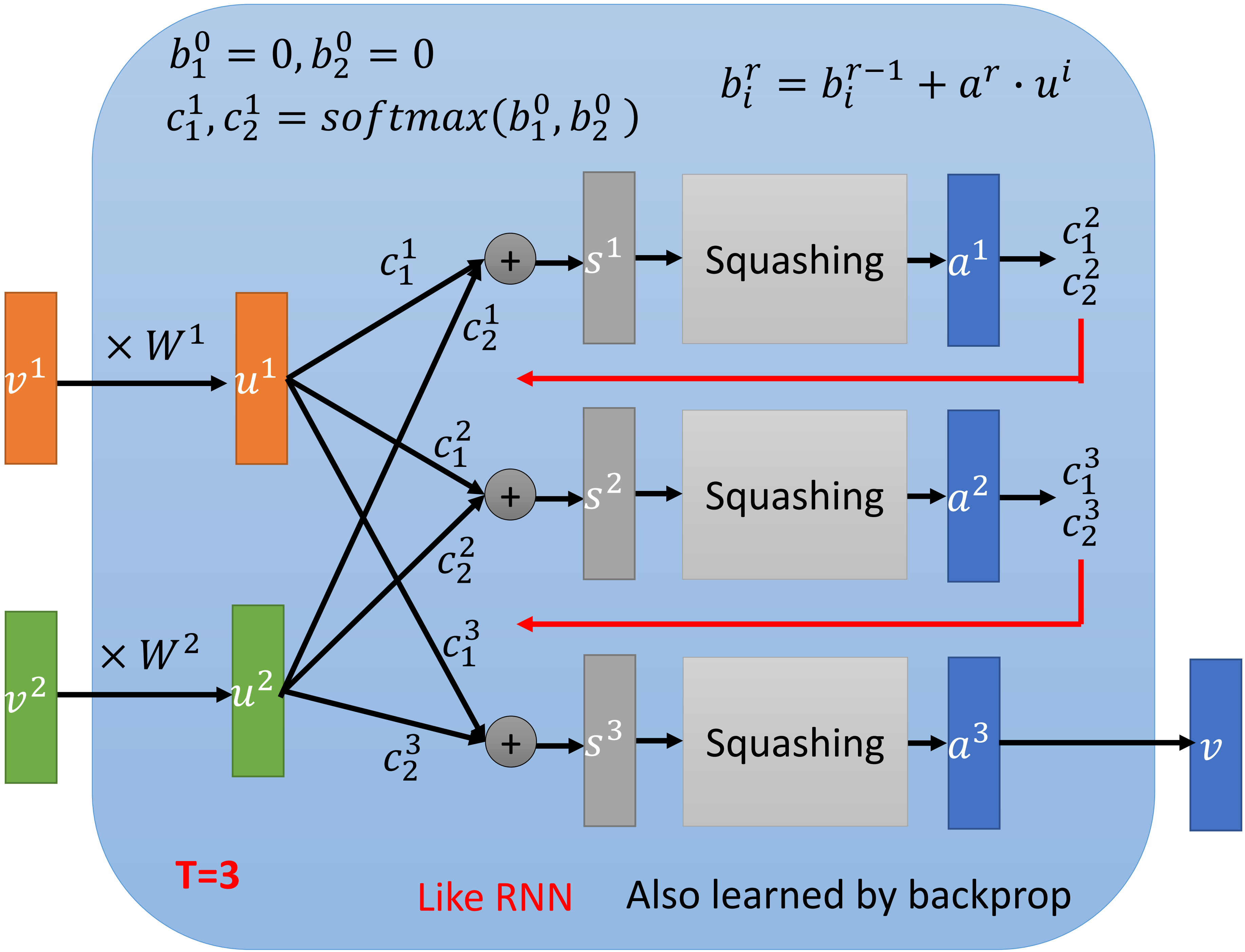
$c_1^r, c_2^r, c_3^r = softmax(b_1^{r-1}, b_2^{r-1}, b_3^{r-1})$

$s^r = c_1 u^1 + c_2 u^2 + c_3 u^3$

$a^r = Squash(s^r)$

$b_i^r = b_i^{r-1} + a^r \cdot u^i$

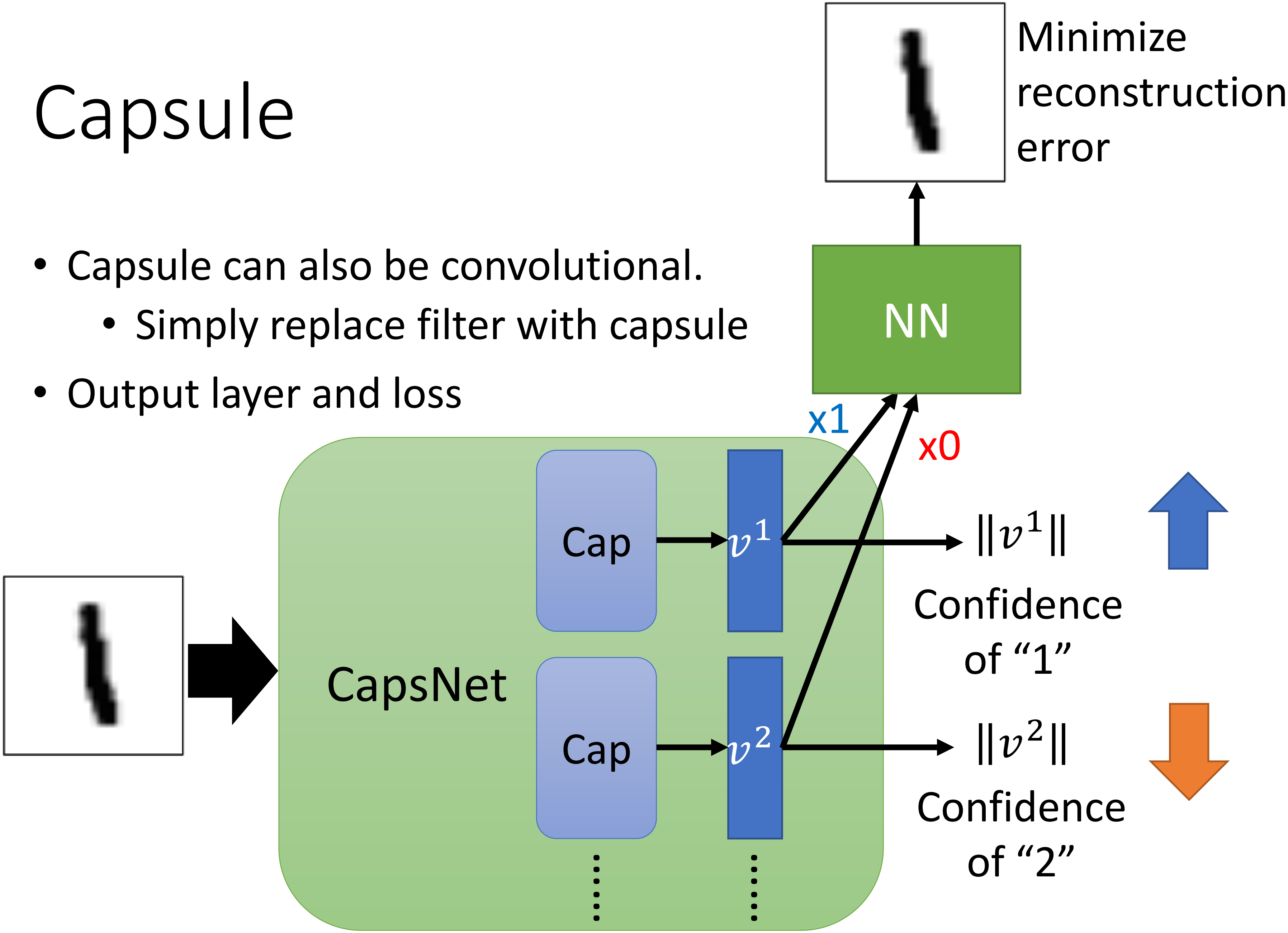




$b_1^0 = 0, b_2^0 = 0$
$c_1^1, c_2^1 = softmax(b_1^0, b_2^0)$
$b_i^r = b_i^{r-1} + a^r \cdot u^i$
v^1
$\times W^1$
u^1
v^2
$\times W^2$
u^2
c_1^1
c_2^1
c_1^2
c_2^2
c_1^3
c_2^3
+
s^1
Squashing
a^1
c_1^2
c_2^2
s^2
Squashing
a^2
c_1^3
c_2^3
s^3
Squashing
a^3
v
T=3
Like RNN
Also learned by backprop

Capsule

- Capsule can also be convolutional.
  - Simply replace filter with capsule
- Output layer and loss



Experimental Results

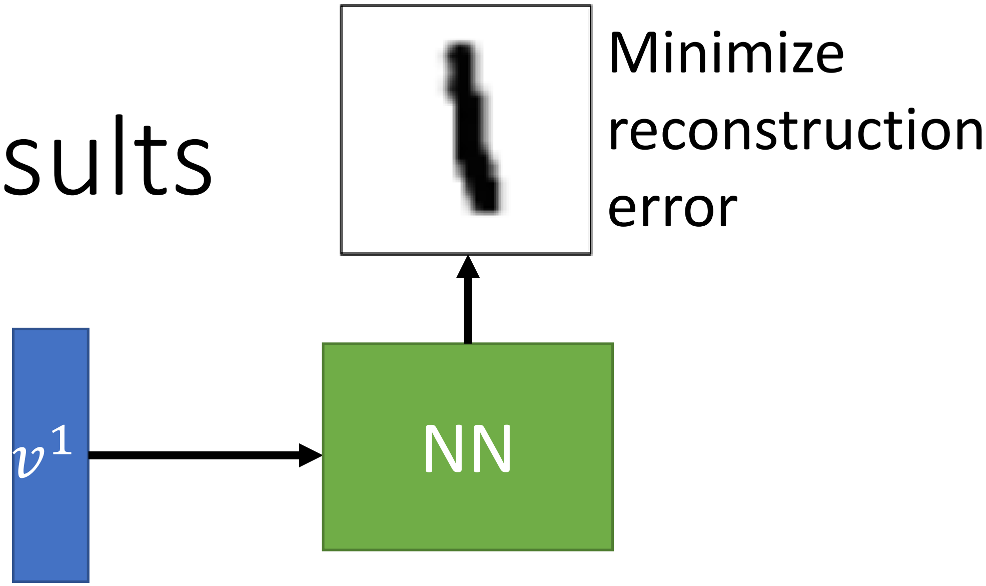
- MNIST

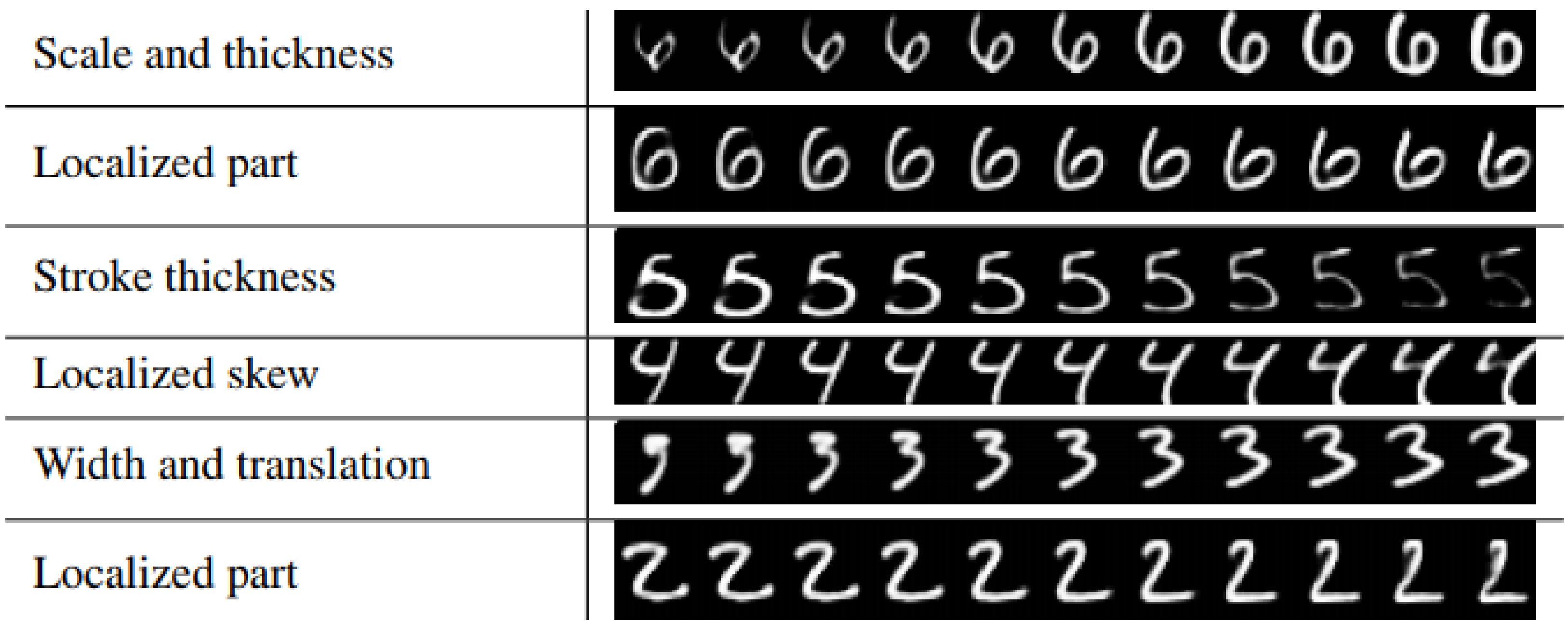
Method	Routing	Reconstruction	MNIST (%)
Baseline	-	-	0.39
CapsNet	1	no	$0.34_{\pm 0.032}$
CapsNet	1	yes	$0.29_{\pm 0.011}$
CapsNet	3	no	$0.35_{\pm 0.036}$
CapsNet	3	yes	$\mathbf{0.25}_{\pm 0.005}$

- Each example is an MNIST digit with a random small affine transformation.
- However, models were never trained with affine transformations
- CapsNet achieved 79% accuracy on the affnist test set.
- A traditional convolutional model with a similar number of parameters which achieved 66%.

Experimental Results

- Each dimension contains specific information.

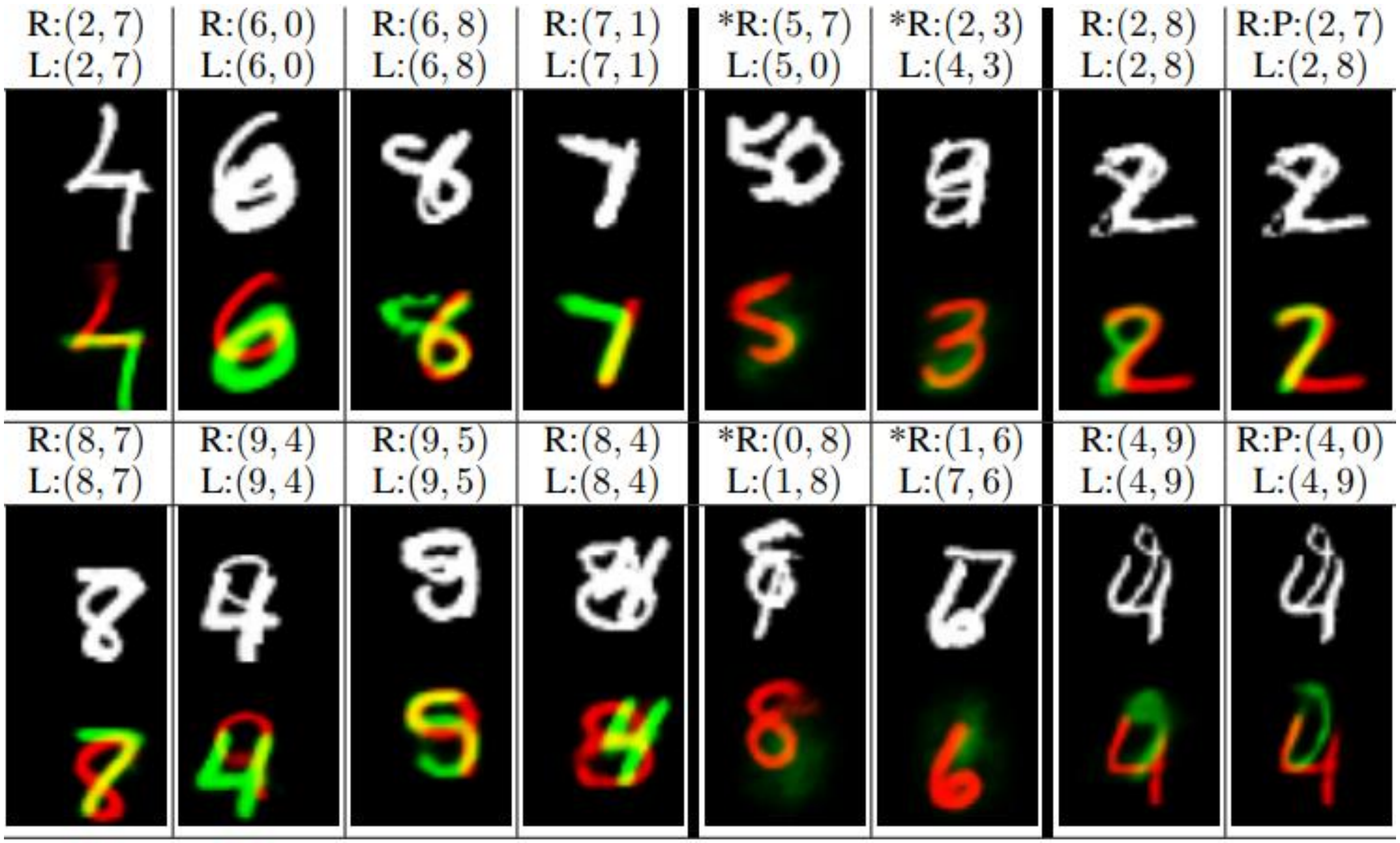




Experimental Results

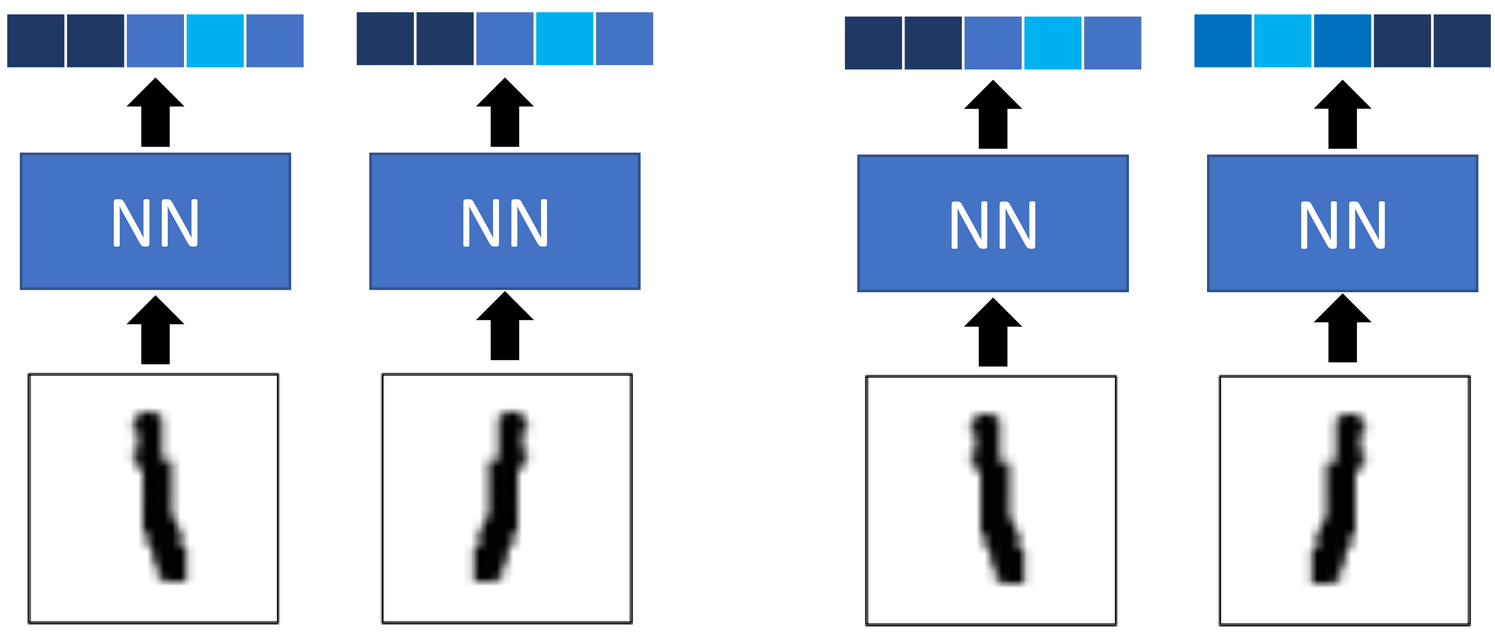
Top: input
Bottom: reconstructed
R: reconstructed digits
L: true labels

- MultiMNIST



Discussion

- Invariance v.s. Equivariance



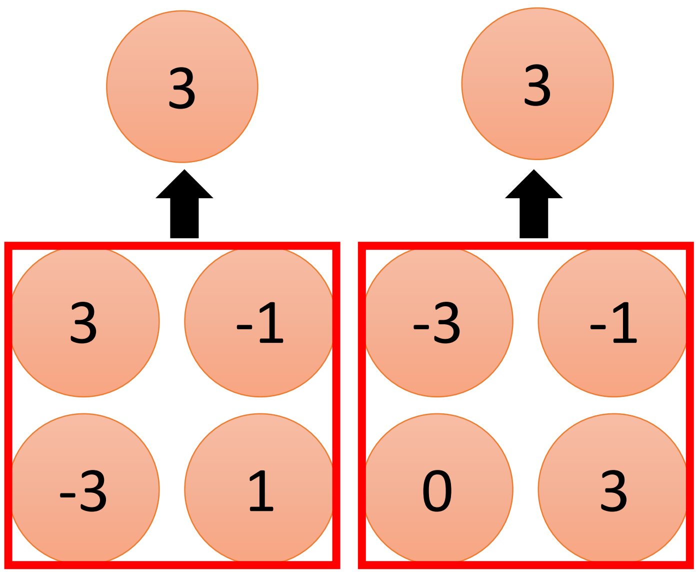
Invariance ***Equivariance***

Discussion

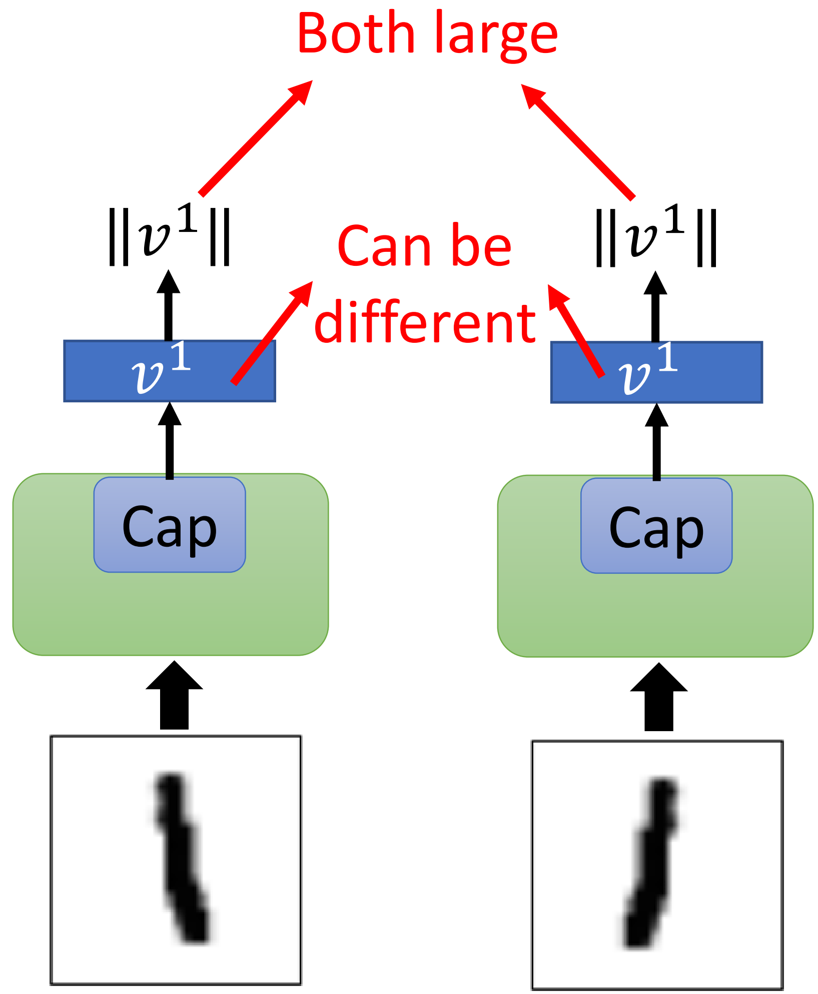
- Invariance v.s. Equivariance

I know the difference, but I do not react to it.

I don't know the difference.

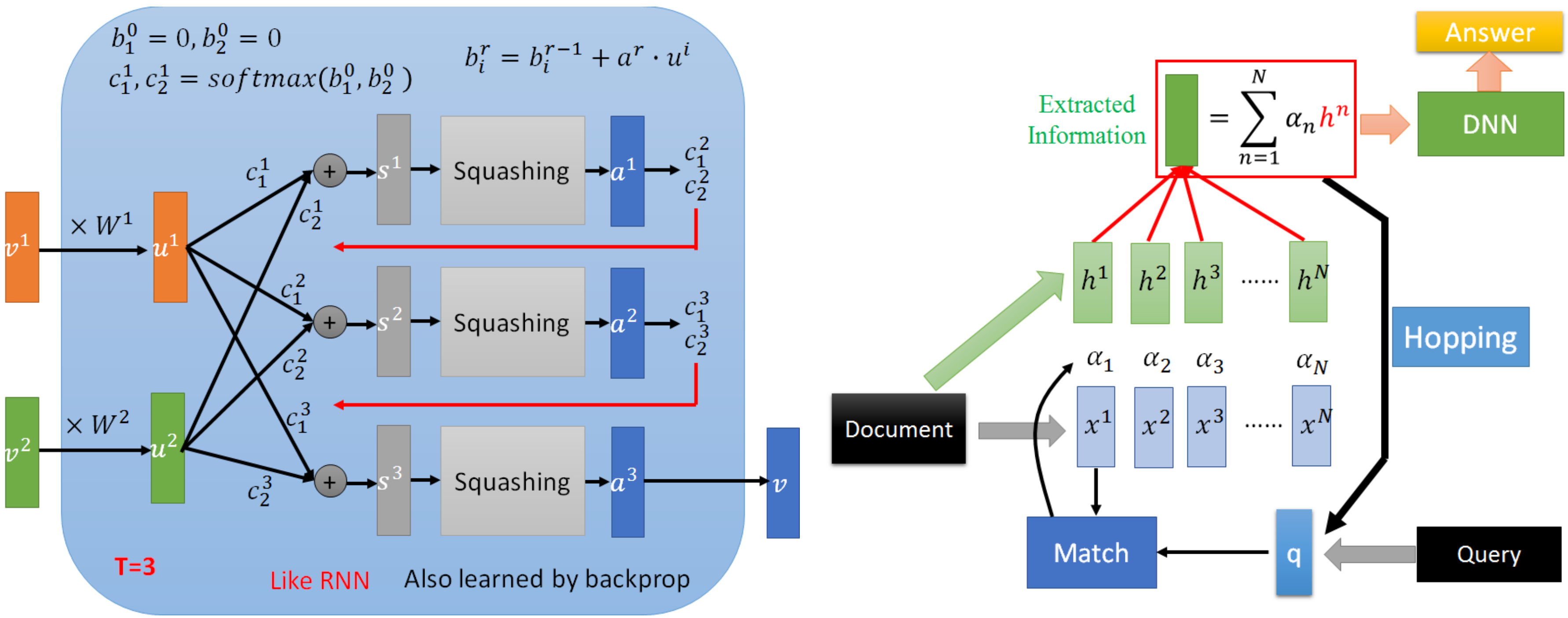


Max pooling has invariance, but not equivariance.



Capsule has both invariance and equivariance.

Dynamic Routing



To Learn More ……

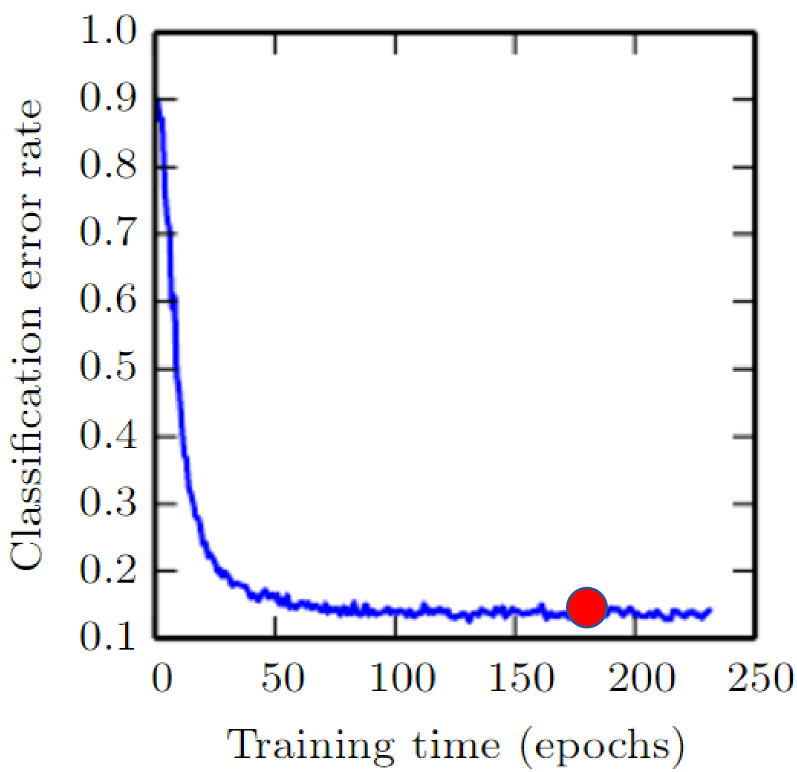
- Hinton's talk: https://www.youtube.com/watch?v=rTawFwUvnLE
- Keras:
  - https://github.com/XifengGuo/CapsNet-Keras
- Tensorflow:
  - https://github.com/naturomics/CapsNet-Tensorflow
- PyTorch
  - https://github.com/gram-ai/capsule-networks
  - https://github.com/timomernick/pytorch-capsule
  - https://github.com/nishnik/CapsNet-PyTorch

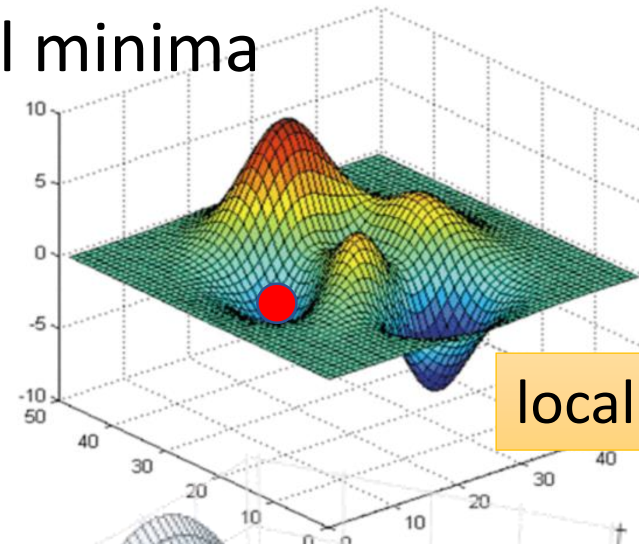
Interesting Facts (?) about Deep Learning

http://www.deeplearningbook.org/contents/optimization.html

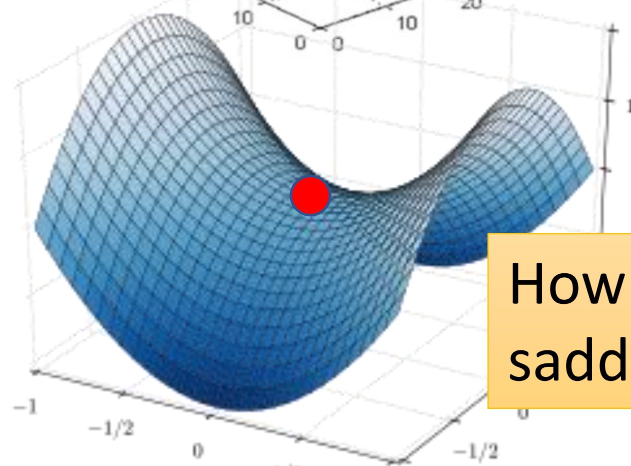
Training stuck because .... ?

- People believe training stuck because the parameters are near a local minima





local minima

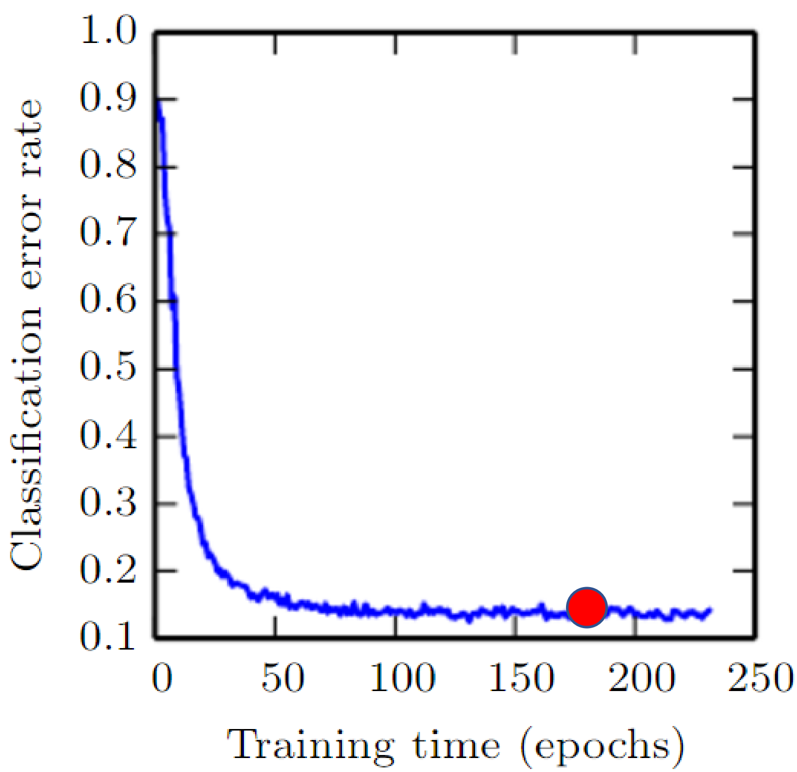


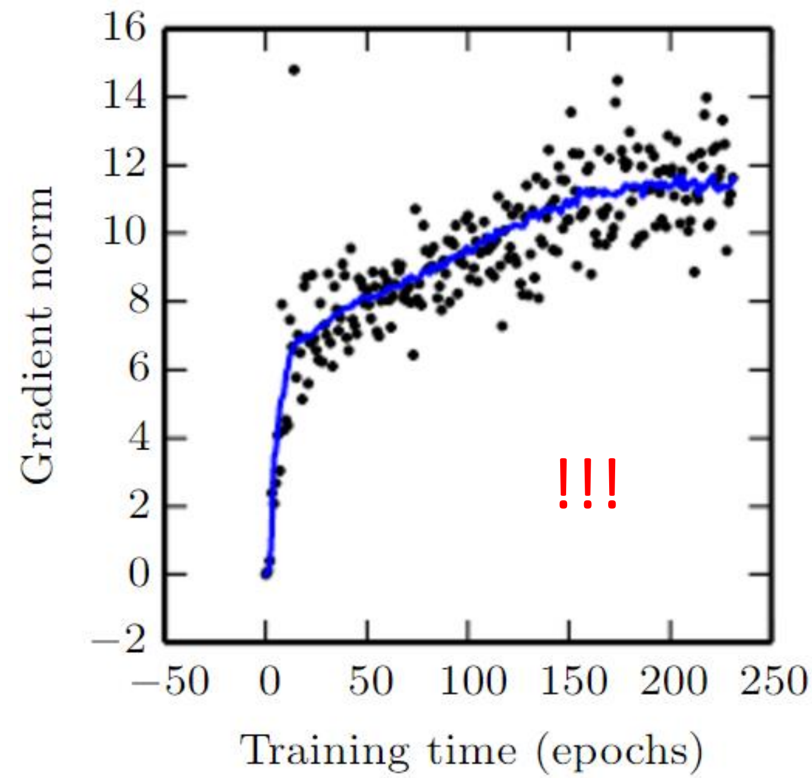
How about saddle point?

http://www.deeplearningbook.org/contents/optimization.html

Training stuck because …. ?

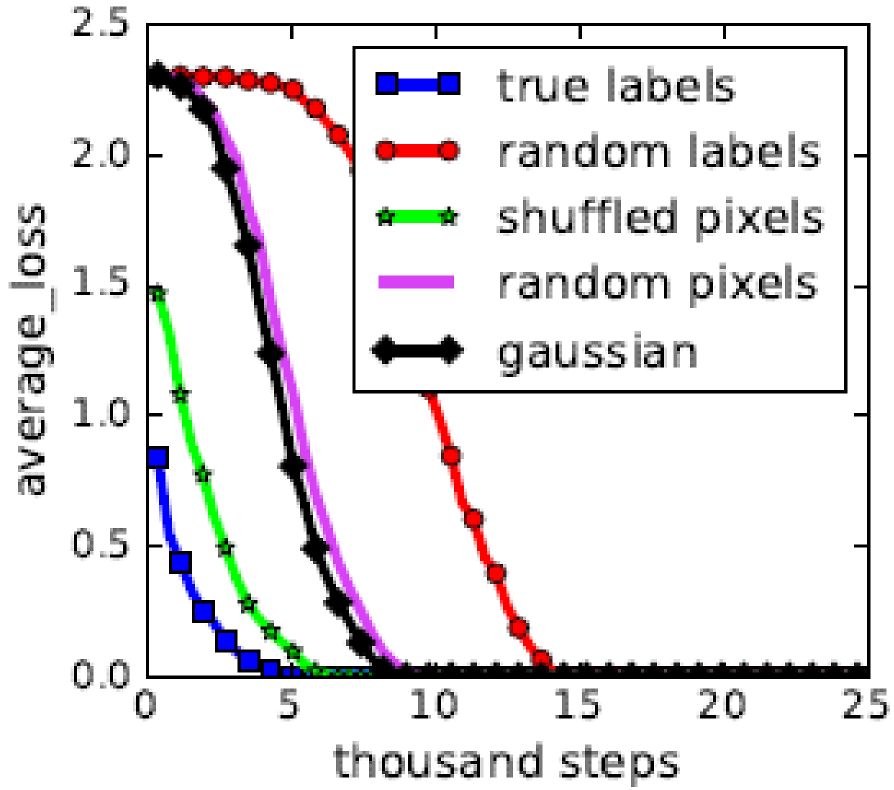
- People believe training stuck because the parameters are around a critical point





https://arxiv.org/pdf/1611.03530.pdf

Brute-force Memorization ?



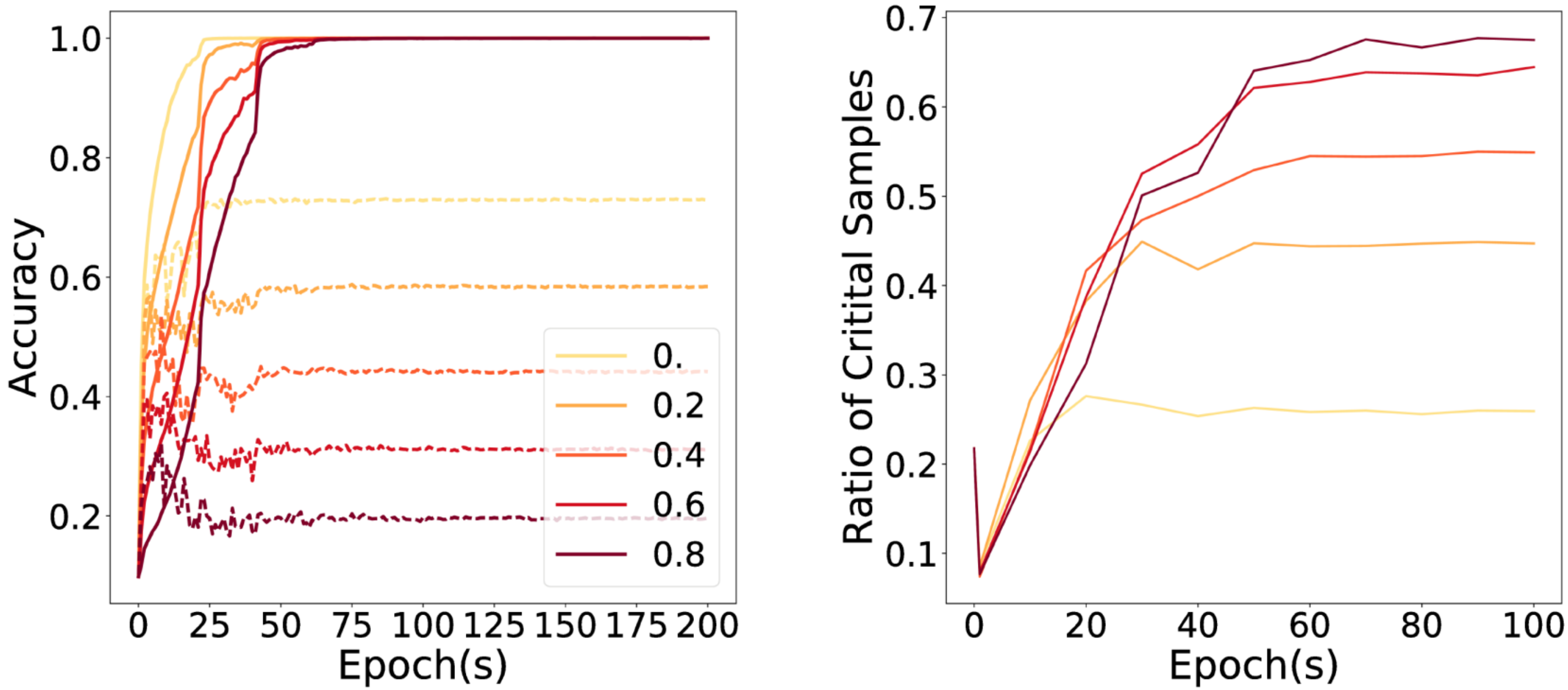
Final of 2017 Spring:
https://ntumlds.wordpress.com/2017/03/27/r05922018_drliao/

Demo

https://arxiv.org/pdf/1706.05394.pdf

Brute-force Memorization ?

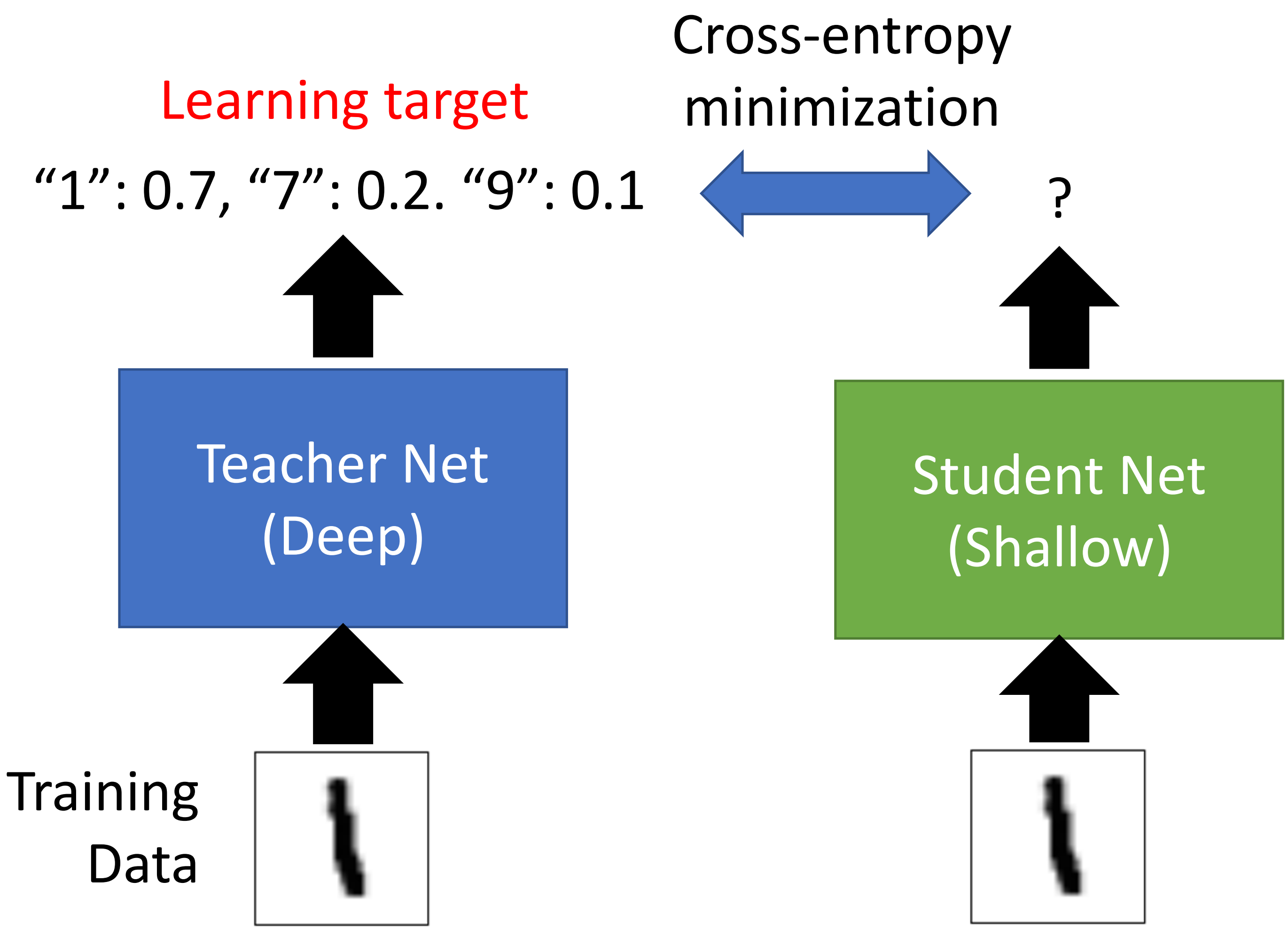
- Simple pattern first, then memorize exception

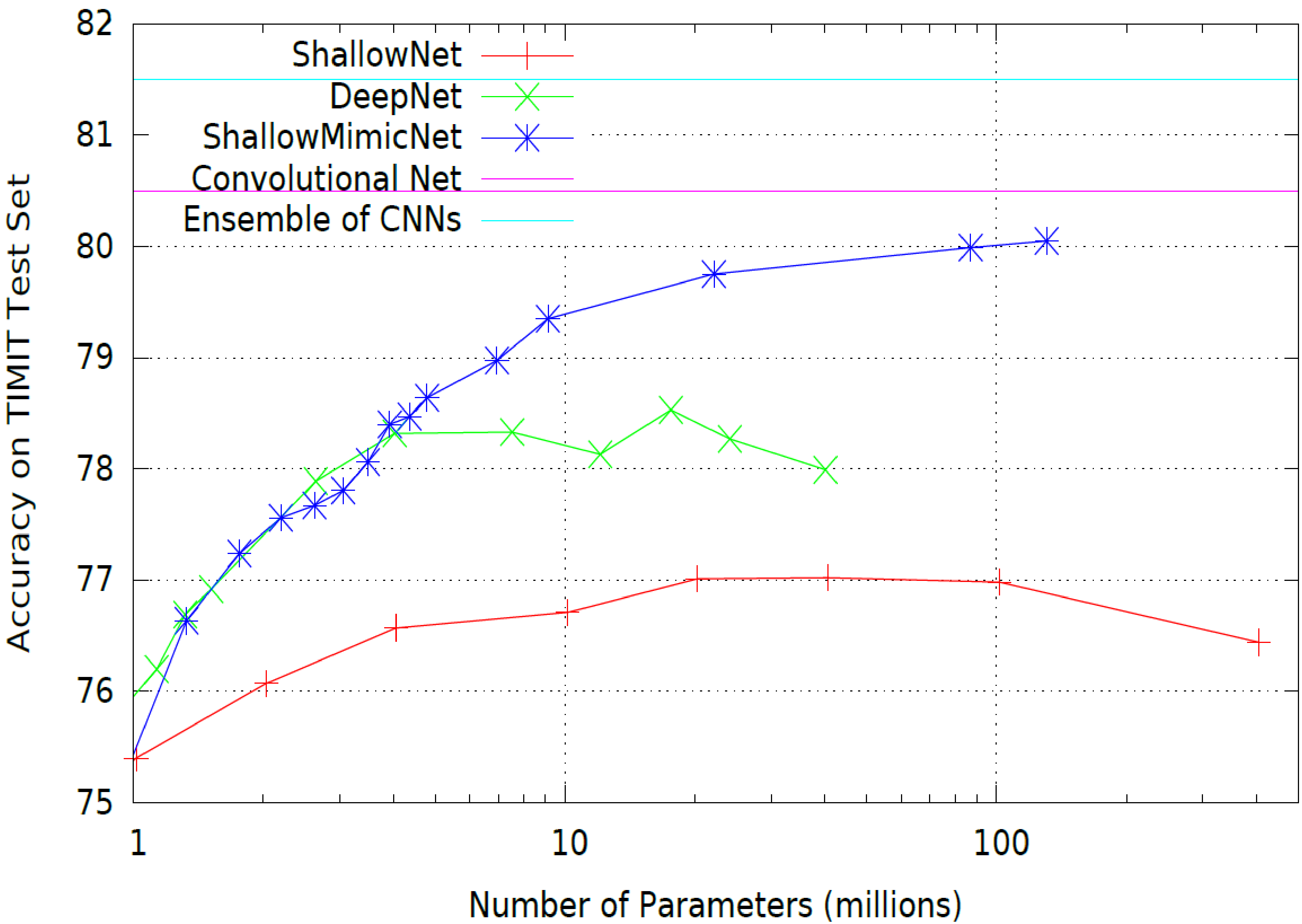


(b) Noise added on classification labels.

Knowledge Distillation

Knowledge Distillation
https://arxiv.org/pdf/1503.02531.pdf
Do Deep Nets Really Need to be Deep?
https://arxiv.org/pdf/1312.6184.pdf





https://arxiv.org/pdf/1312.6184.pdf

Deep Learning for Question Answering

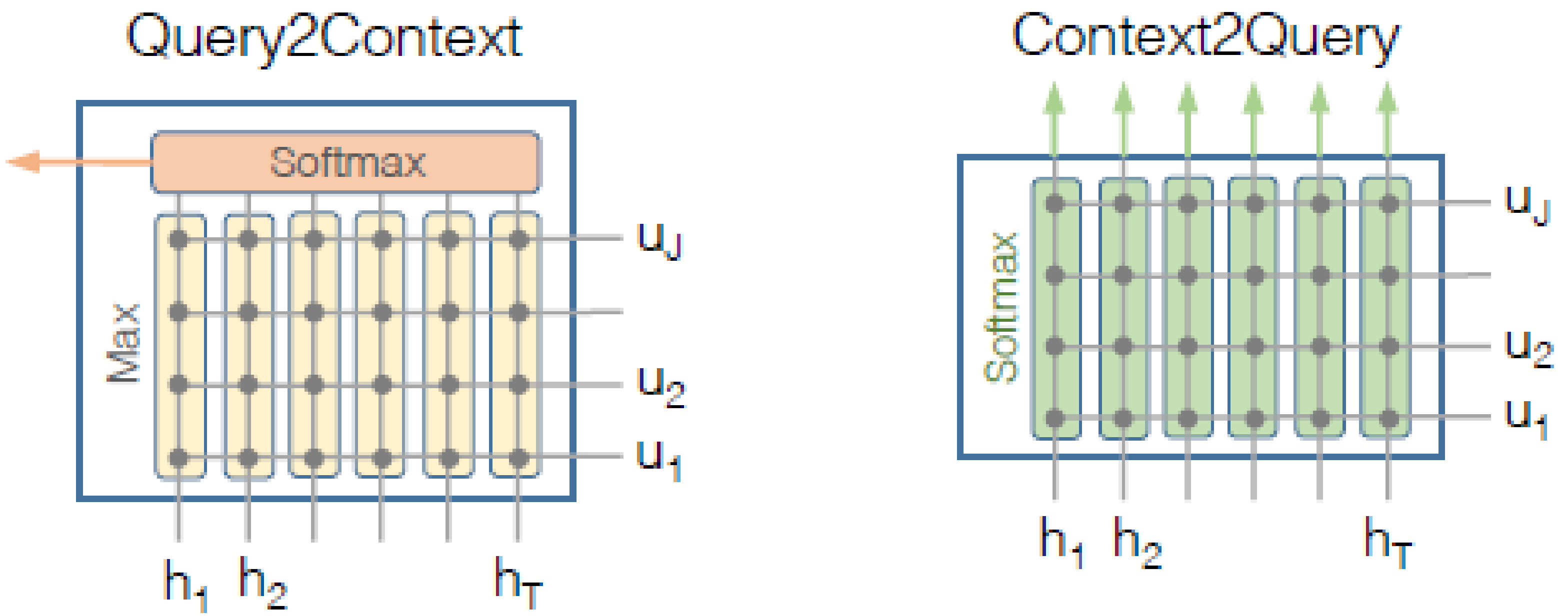
Question Answering

- Given a document and a query, output an answer
- bAbI: the answer is a word
  - https://research.fb.com/downloads/babi/
- SQuAD: the answer is a sequence of words (in the input document)
  - https://rajpurkar.github.io/SQuAD-explorer/
- MS MARCO: the answer is a sequence of words
  - http://www.msmarco.org
- MovieQA: Multiple choice question (output a number)
  - http://movieqa.cs.toronto.edu/home/
- More: https://github.com/dapurv5/awesome-question-answering

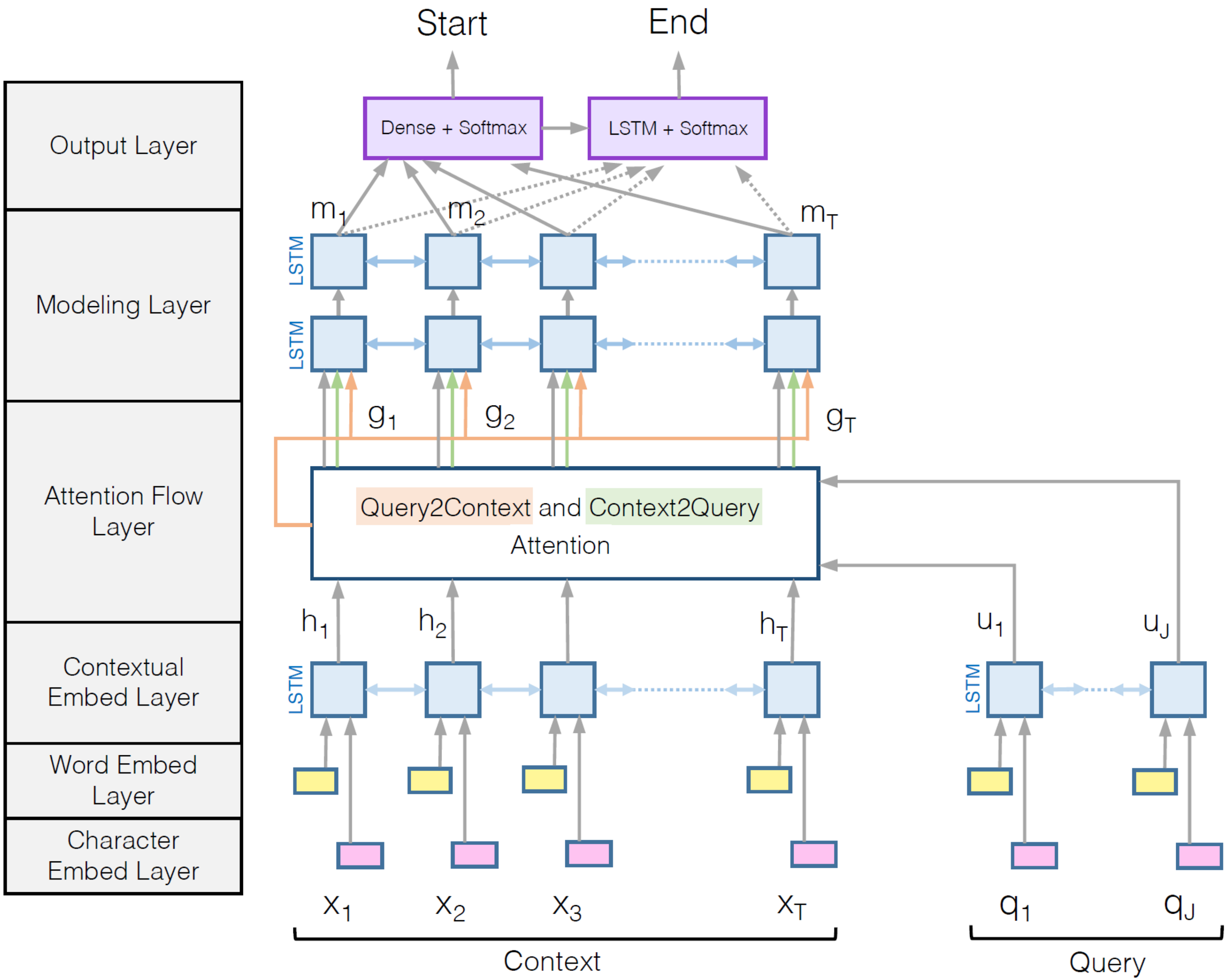
Rank	Model	EM	F1
1 Oct 17, 2017	Interactive AoA Reader+ (ensemble) *Joint Laboratory of HIT and iFLYTEK*	79.083	86.450
2 Oct 24, 2017	FusionNet (ensemble) *Microsoft Business AI Solutions Team*	78.978	86.016
3 Nov 03, 2017	BiDAF + Self Attention + ELMo (single model) *Allen Institute for Artificial Intelligence*	78.580	85.833
3 Oct 12, 2017	r-net (ensemble) *Microsoft Research Asia* http://aka.ms/rnet	78.926	85.722
3 Oct 22, 2017	DCN+ (ensemble) *Salesforce Research*	78.852	85.996
4 Oct 22, 2017	BiDAF + Self Attention + ELMo (single model) *Allen Institute for Artificial Intelligence*	77.856	85.344
5 Jul 25, 2017	Interactive AoA Reader (ensemble) *Joint Laboratory of HIT and iFLYTEK Research*	77.845	85.297
6 Aug 21, 2017	Reinforced Mnemonic Reader (ensemble) *NUDT and Fudan University* https://arxiv.org/abs/1705.02798	77.678	84.888

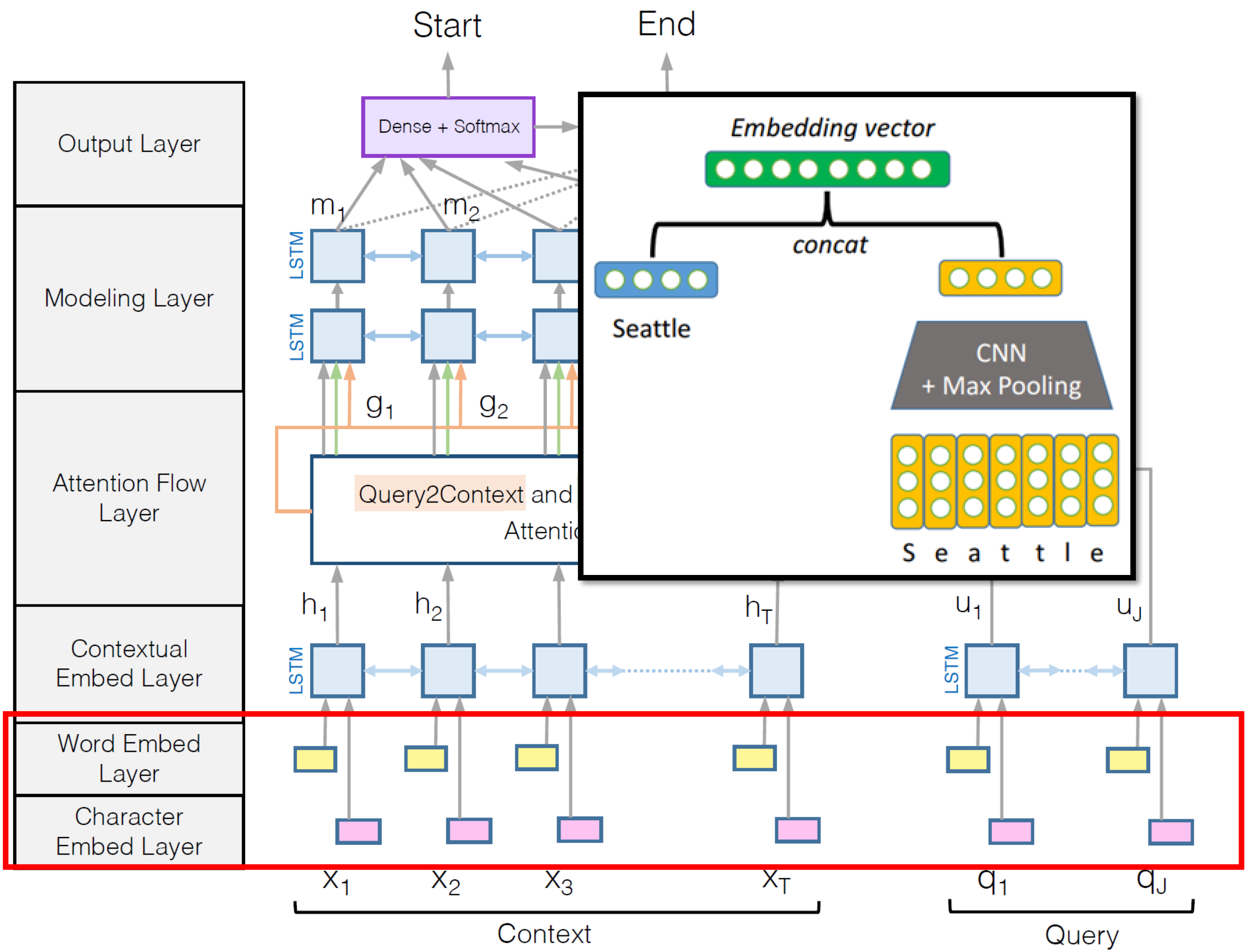
Demo: http://35.165.153.16:1995

Bi-directional Attention Flow

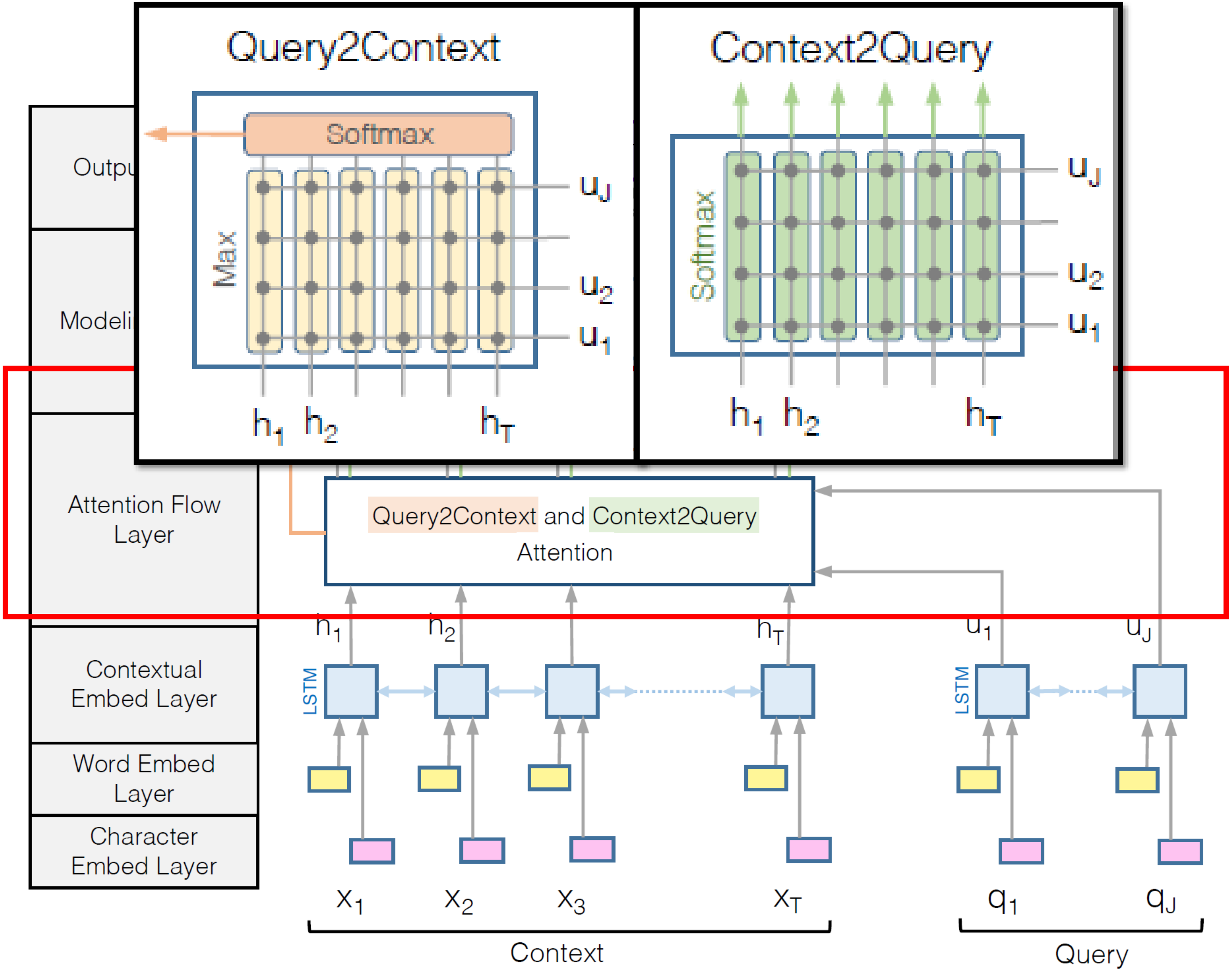


3 Nov 03, 2017	BiDAF + Self Attention + ELMo (single model) *Allen Institute for Artificial Intelligence*	78.580	85.833

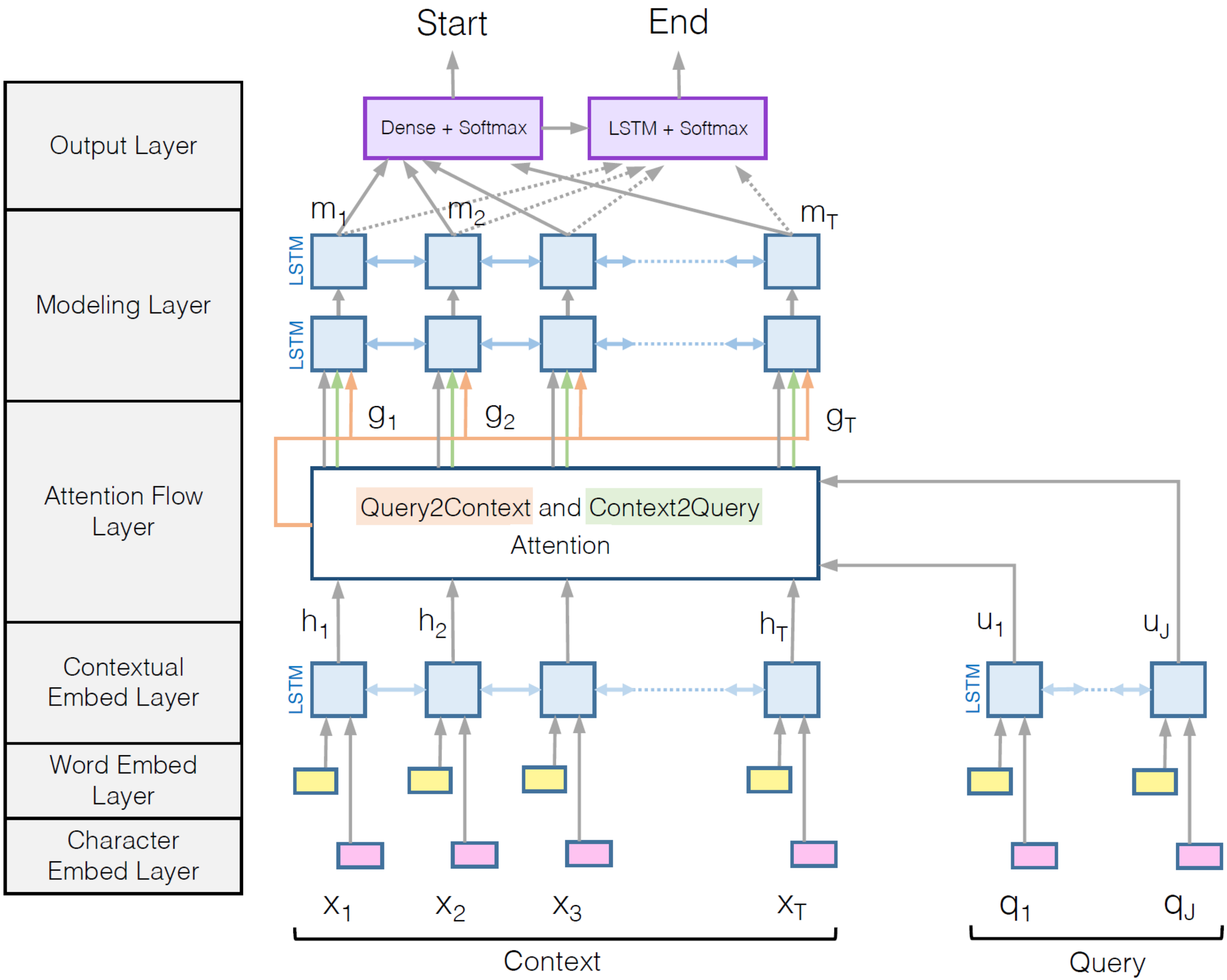

Start
End
Dense + Softmax
LSTM + Softmax
Output Layer
Modeling Layer
Attention Flow Layer
Contextual Embed Layer
Word Embed Layer
Character Embed Layer
m1
m2
mT
LSTM
LSTM
g1
g2
gT
Query2Context and Context2Query Attention
h1
h2
hT
u1
uJ
LSTM
LSTM
x1
x2
x3
xT
q1
qJ
Context
Query



Start
End
Output Layer
Modeling Layer
Attention Flow Layer
Contextual Embed Layer
Word Embed Layer
Character Embed Layer
Dense + Softmax
m1
m2
LSTM
LSTM
g1
g2
Query2Context and
Attenti
h1
h2
hT
u1
uJ
LSTM
LSTM
x1
x2
x3
xT
q1
qJ
Context
Query
Embedding vector
concat
Seattle
CNN
+ Max Pooling
S e a t t l e



Query2Context
Softmax
Max
u_J
u_2
u_1
h_1 h_2 h_T
Context2Query
Softmax
u_J
u_2
u_1
h_1 h_2 h_T
Outpu
Modeli
Attention Flow Layer
Contextual Embed Layer
Word Embed Layer
Character Embed Layer
Query2Context and Context2Query Attention
h_1 h_2 h_T u_1 u_J
LSTM
LSTM
x_1 x_2 x_3 x_T q_1 q_J
Context
Query



Start
End
Dense + Softmax
LSTM + Softmax
Output Layer
Modeling Layer
Attention Flow Layer
Contextual Embed Layer
Word Embed Layer
Character Embed Layer
m1
m2
mT
LSTM
LSTM
g1
g2
gT
Query2Context and Context2Query Attention
h1
h2
hT
u1
uJ
LSTM
LSTM
x1
x2
x3
xT
q1
qJ
Context
Query

Dynamic Coattention Networks

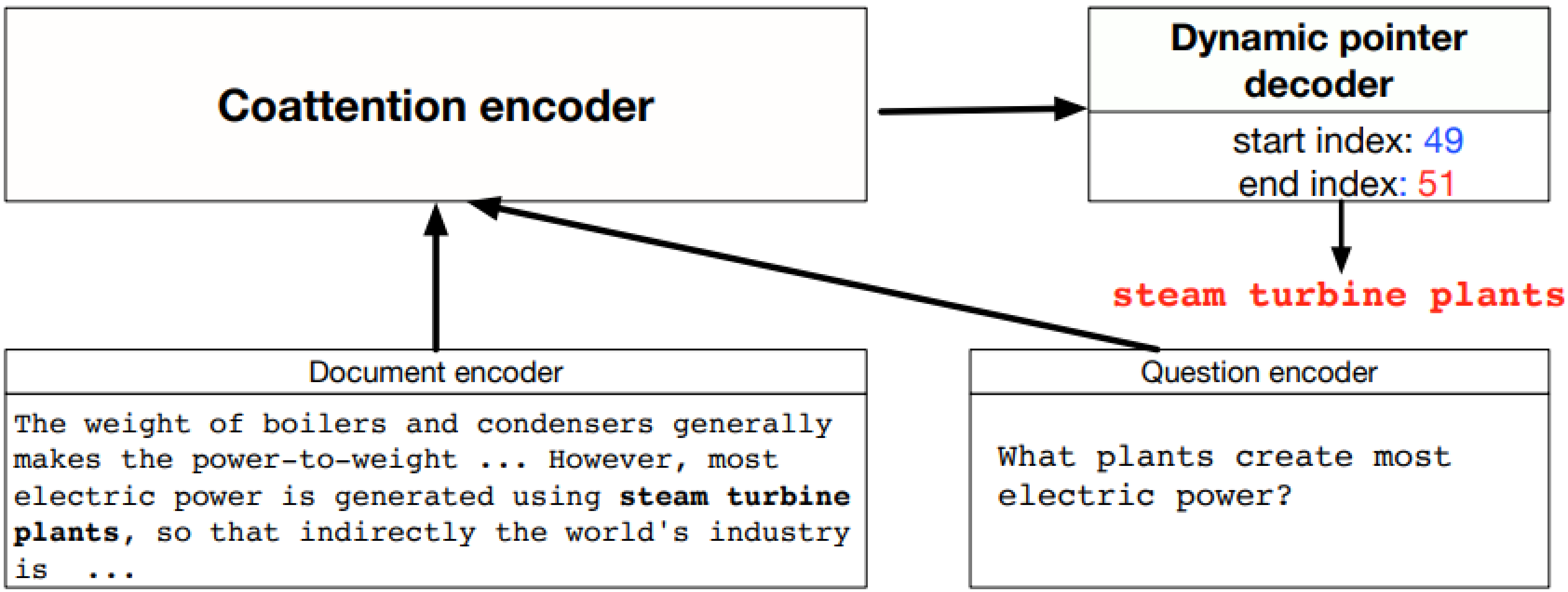
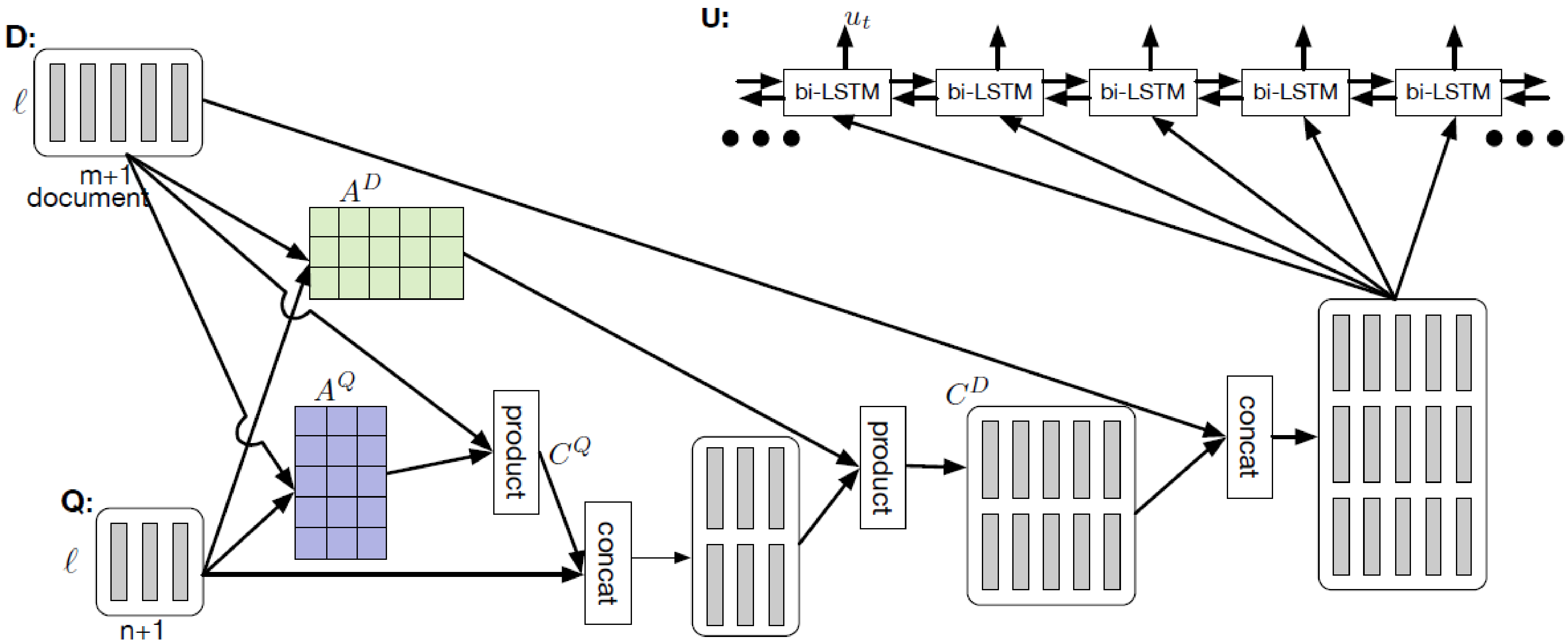


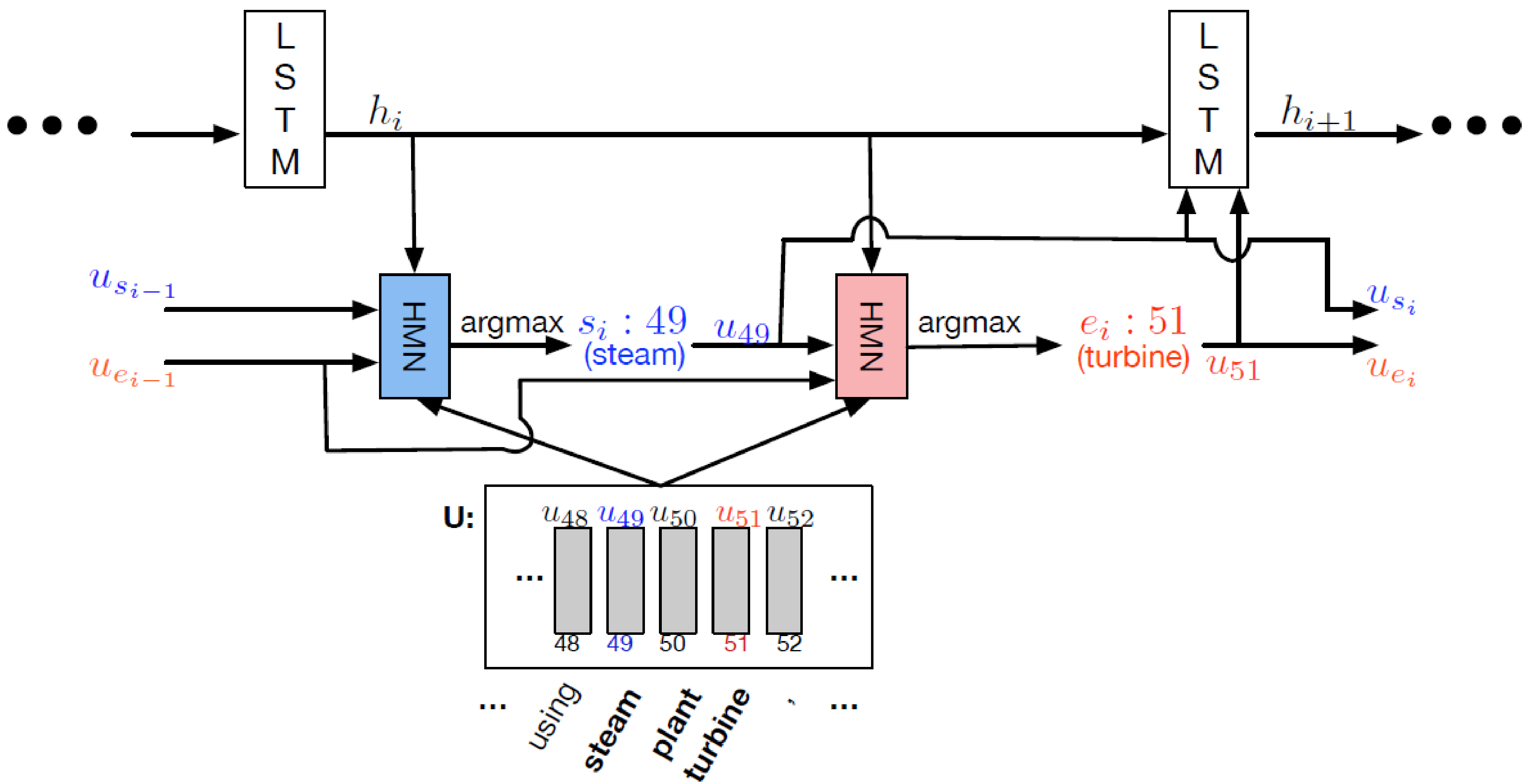
Figure 1: Overview of the Dynamic Coattention Network.

3 Oct 22, 2017	DCN+ (ensemble) *Salesforce Research*	78.852	85.996

Dynamic Coattention Networks

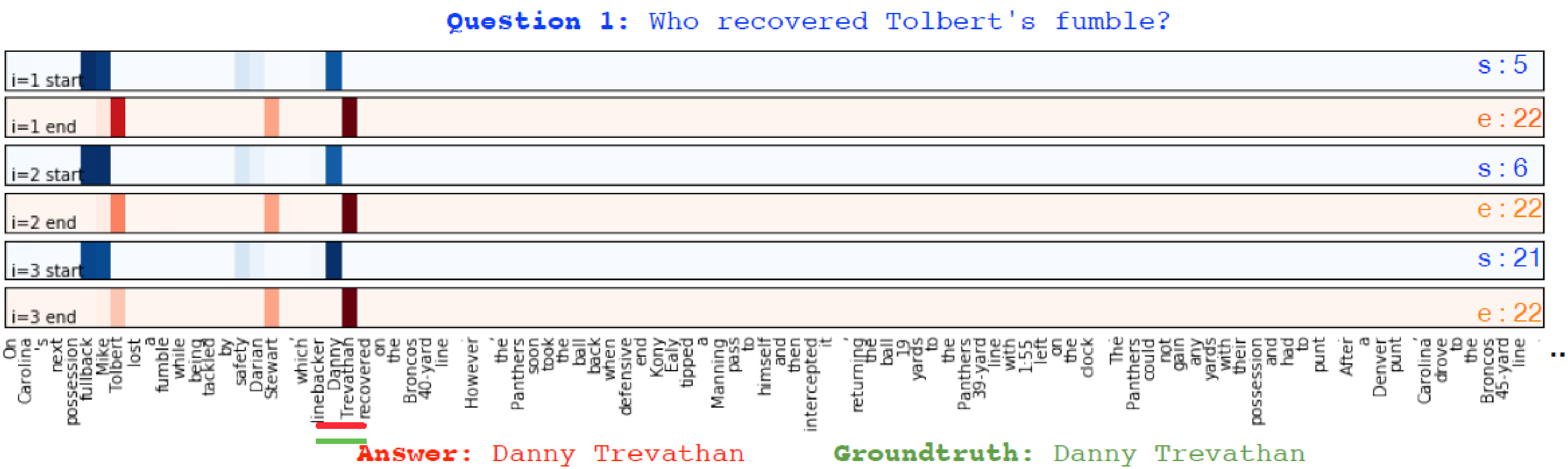


Dynamic Coattention Networks

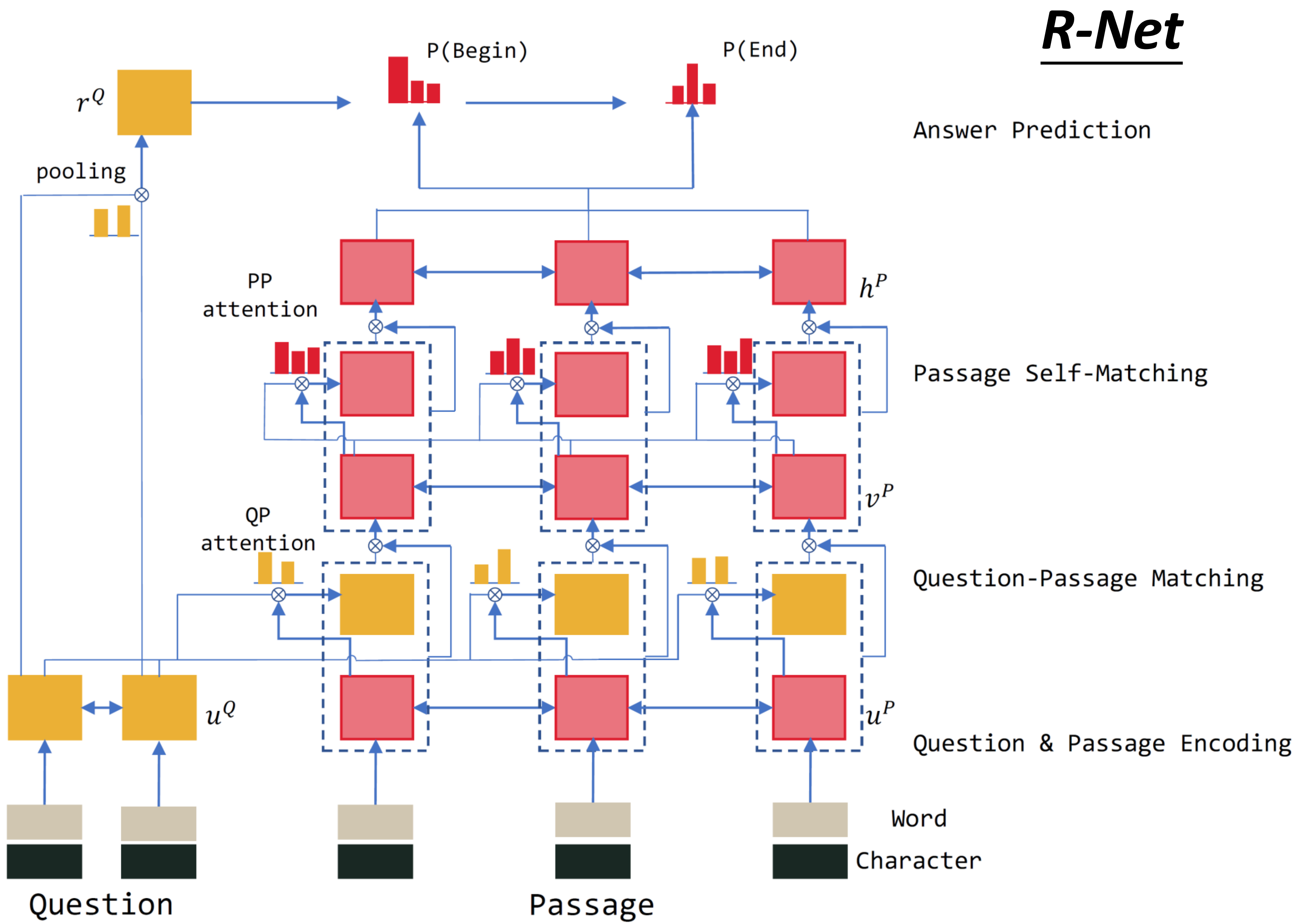


Dynamic Coattention Networks

- Experimental Results

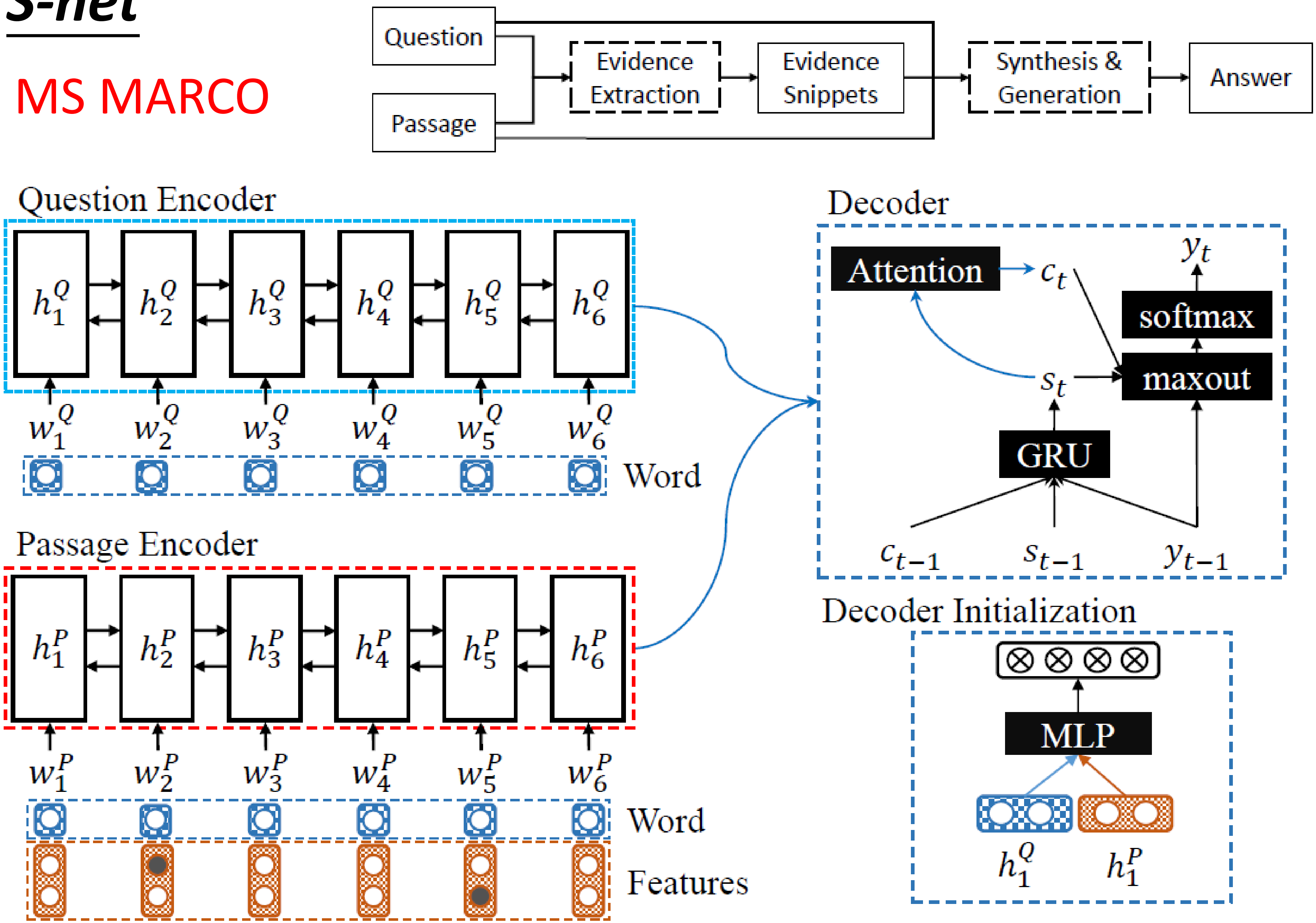


DCN+: https://arxiv.org/pdf/1711.00106.pdf

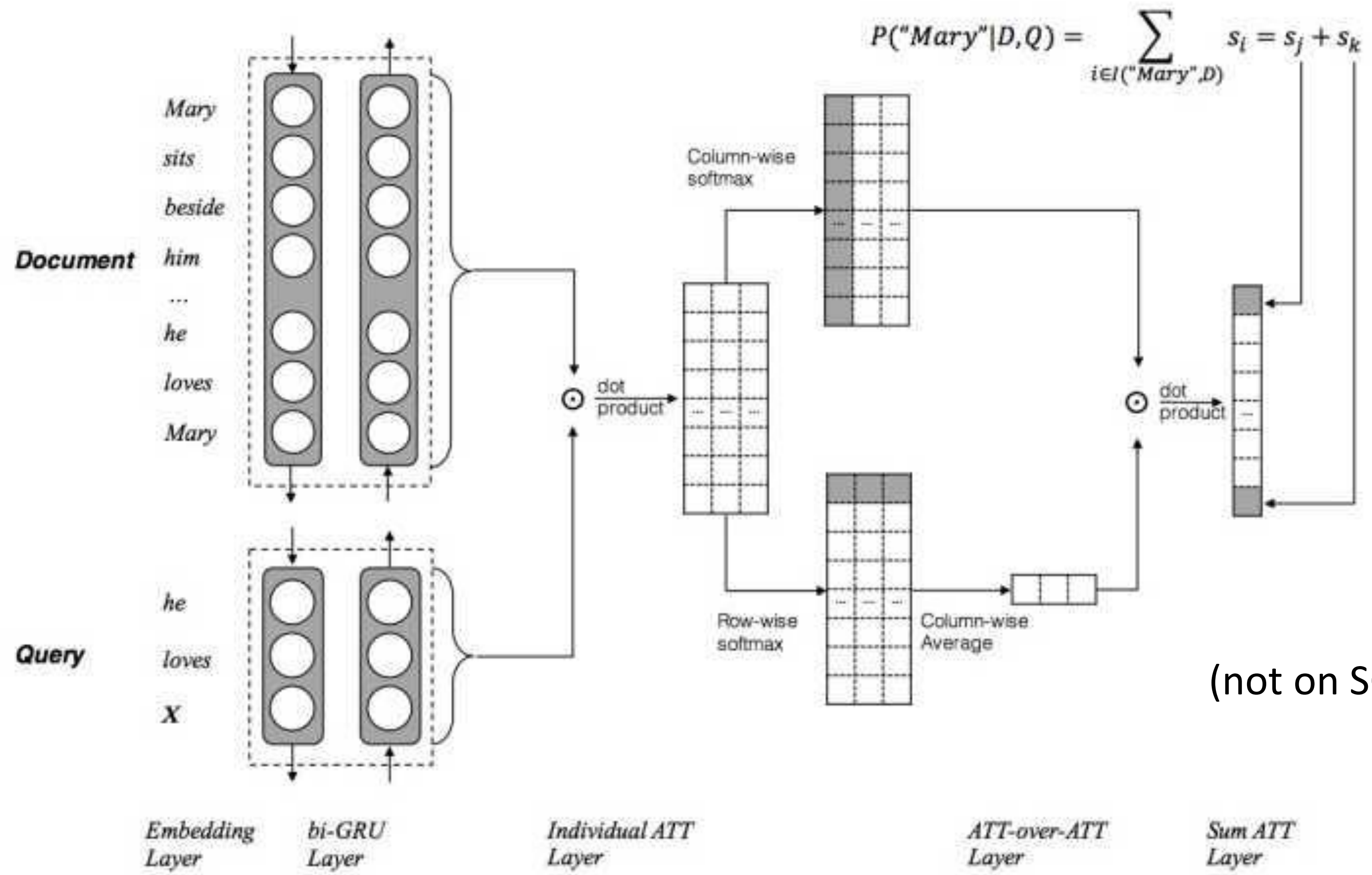

R-Net
P(Begin)
P(End)
Answer Prediction
r^Q
pooling
PP
attention
h^P
Passage Self-Matching
v^P
QP
attention
Question-Passage Matching
u^Q
u^P
Question & Passage Encoding
Word
Character
Question
Passage

S-net

MS MARCO



Attention-over-Attention (AoA)



(not on SQUAD)

Figure 1: Neural network architecture of the proposed Attention-over-Attention Reader (AoA Reader).

1 Oct 17, 2017	Interactive AoA Reader+ (ensemble) *Joint Laboratory of HIT and iFLYTEK*	79.083	86.450

Reinforced Mnemonic Reader

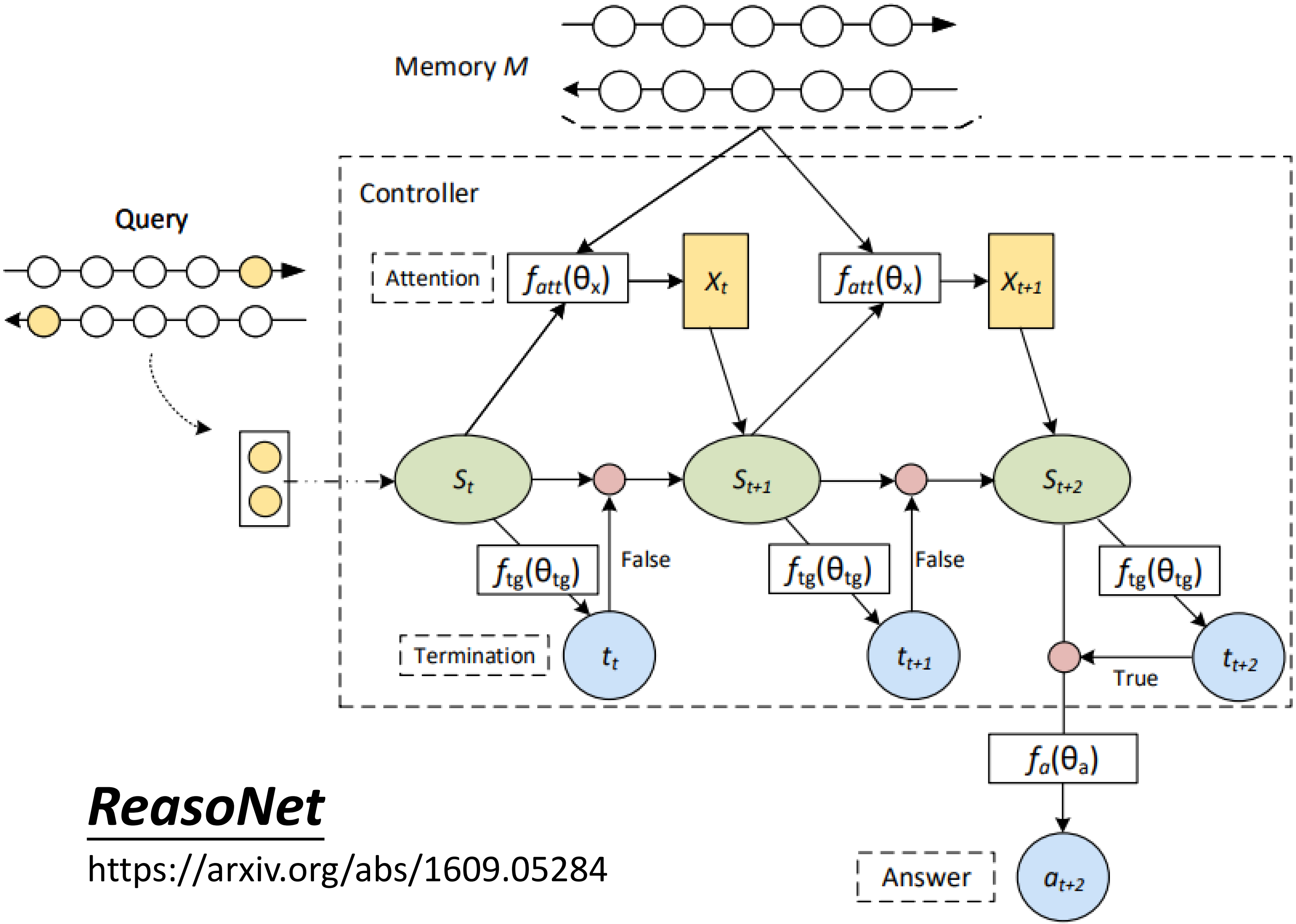
Reinforcement Learning for Machine Comprehension One way to tackle this problem is to directly optimizing the F1 score with reinforcement learning. The F1 score measures the overlap between the predicted answer and the ground-truth answer, serving as a "soft" metric compared to the "hard" EM. Taking the F1 score as reward, we use the REINFORCE algorithm (Williams 1992) to maximize the model's expected reward. For each sampled answer $\hat{A}$, we define the loss as:

$$J_{RL}(\theta) = -\mathbb{E}_{\hat{A}\sim p_\theta(A|C,Q)}[R(\hat{A}, A^*)] \tag{10}$$

where p_θ is the policy to be learned, and $R(\hat{A}, A^*)$ is the reward function for a sampled answer, computed as the F1 score with the ground-truth answer A^*. $\hat{A}$ is obtained by sampling from the predicted probability distribution $p_\theta(A|C,Q)$.

6 Aug 21, 2017	Reinforced Mnemonic Reader (ensemble) *NUDT and Fudan University* https://arxiv.org/abs/1705.02798	77.678	84.888

Multiple-hop



ReasoNet

https://arxiv.org/abs/1609.05284

FusionNet

Architectures	(1)	(2)	(2')	(3)	(3')
Match-LSTM (Wang & Jiang, 2016)		✓			
DCN (Xiong et al., 2017)		✓			✓
FastQA (Weissenborn et al., 2017)	✓				
FastQAExt (Weissenborn et al., 2017)	✓	✓		✓	
BiDAF (Seo et al., 2017)		✓			✓
RaSoR (Lee et al., 2016)	✓		✓		
DrQA (Chen et al., 2017)	✓				
MPCM (Wang et al., 2016)	✓	✓			
Mnemonic Reader (Hu et al., 2017)	✓	✓		✓	
R-net (Wang et al., 2017)		✓		✓	

Table 1: A summarized view on the fusion processes used in several state-of-the-art architectures.

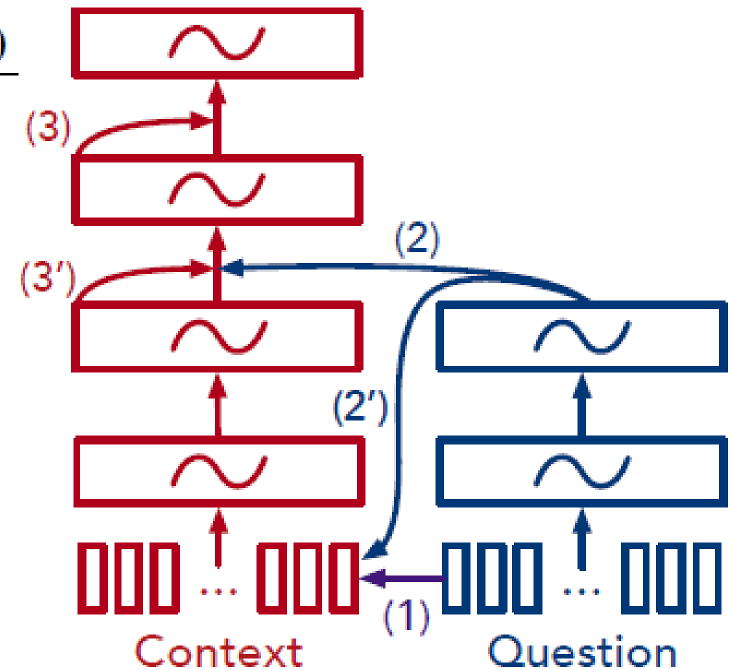
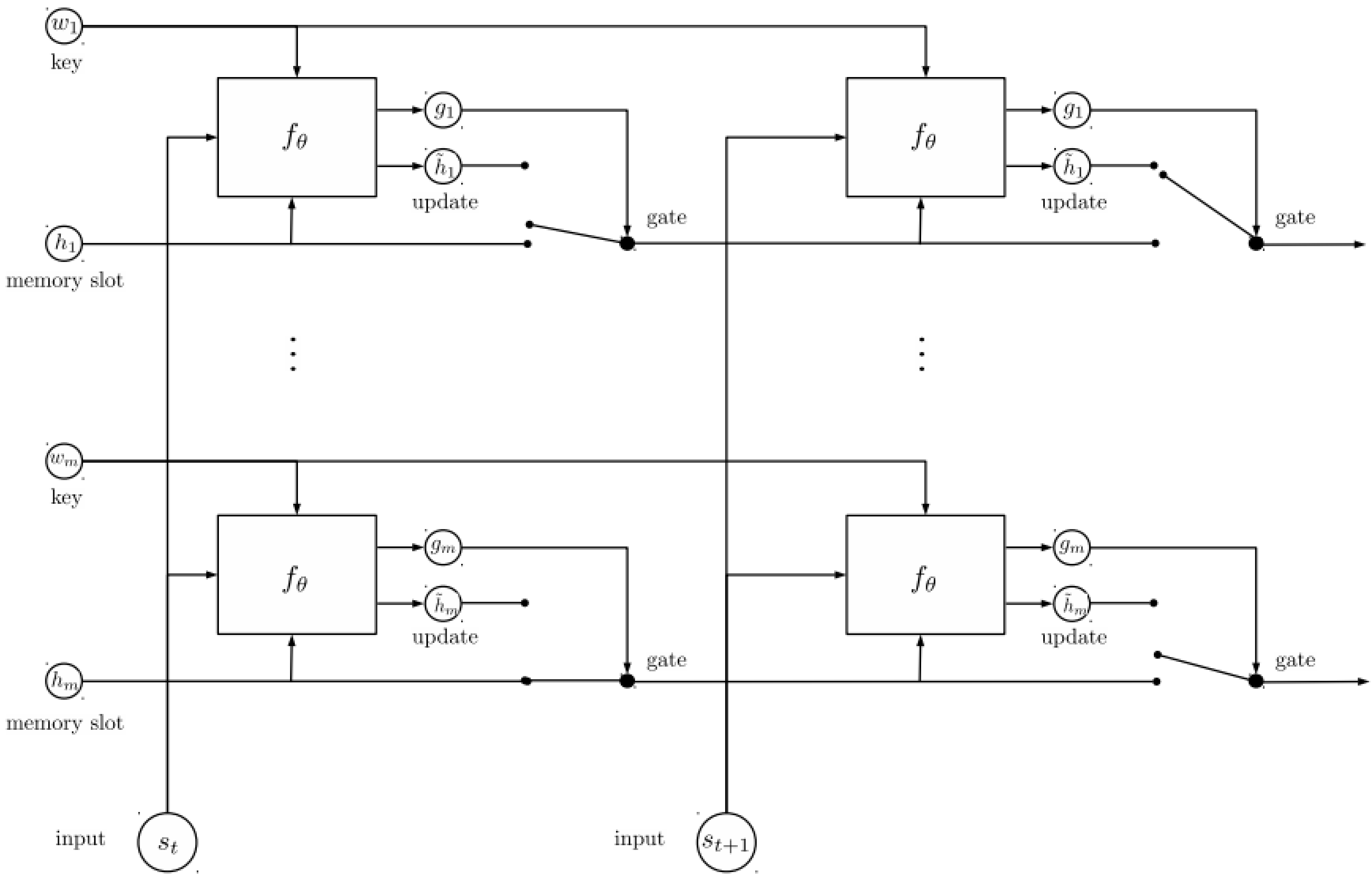


Figure 2: A conceptual architecture illustrating recent advances in MRC.

1 Oct 24, 2017	FusionNet (ensemble) *Microsoft Business AI Solutions Team*	78.978	86.016

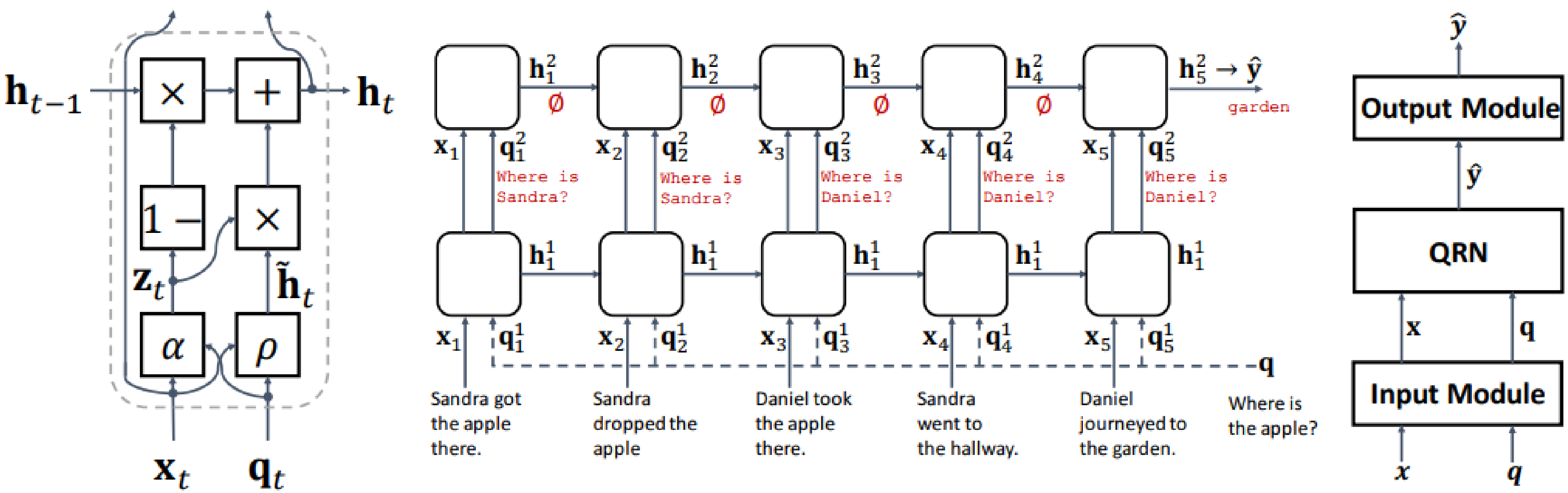
Recurrent Entity Networks



Task	NTM	D-NTM	MemN2N	DNC	DMN+	EntNet
1: 1 supporting fact	31.5	4.4	0	0	0	0
2: 2 supporting facts	54.5	27.5	0.3	0.4	0.3	0.1
3: 3 supporting facts	43.9	71.3	2.1	1.8	1.1	4.1
4: 2 argument relations	0	0	0	0	0	0
5: 3 argument relations	0.8	1.7	0.8	0.8	0.5	0.3
6: yes/no questions	17.1	1.5	0.1	0	0	0.2
7: counting	17.8	6.0	2.0	0.6	2.4	0
8: lists/sets	13.8	1.7	0.9	0.3	0.0	0.5
9: simple negation	16.4	0.6	0.3	0.2	0.0	0.1
10: indefinite knowledge	16.6	19.8	0	0.2	0	0.6
11: basic coreference	15.2	0	0.0	0	0.0	0.3
12: conjunction	8.9	6.2	0	0	0.2	0
13: compound coreference	7.4	7.5	0	0	0	1.3
14: time reasoning	24.2	17.5	0.2	0.4	0.2	0
15: basic deduction	47.0	0	0	0	0	0
16: basic induction	53.6	49.6	51.8	55.1	45.3	0.2
17: positional reasoning	25.5	1.2	18.6	12.0	4.2	0.5
18: size reasoning	2.2	0.2	5.3	0.8	2.1	0.3
19: path finding	4.3	39.5	2.3	3.9	0.0	2.3
20: agent's motivation	1.5	0	0	0	0	0
Failed Tasks ($> 5\%$ error):	16	9	3	2	1	**0**
Mean Error:	20.1	12.8	4.2	3.8	2.8	**0.5**

Query-Reduction Networks for Question Answering

- https://arxiv.org/pdf/1606.04582.pdf



(a) QRN unit (b) 2-layer QRN (c) Overview

Query-Reduction Networks for Question Answering

- https://arxiv.org/pdf/1606.04582.pdf

Task	1k										10k						
	Previous works				QRN						Previous works			QRN			
	LSTM	N2N	DMN+	GMemN2N	1r	2	2r	3r	6r	6r200*	N2N	DMN+	GMemN2N	2r	2rv	3r	6r200
1: Single supporting fact	50.0	0.1	1.3	0.0	0.0	0.0	0.0	0.0	0.0	13.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2: Two supporting facts	80.0	18.8	72.3	8.1	65.7	1.2	0.7	0.5	1.5	15.3	0.3	0.3	0.0	0.4	0.8	0.4	0.0
3: Three supporting facts	80.0	31.7	73.3	38.7	68.2	17.5	5.7	1.2	15.3	13.8	2.1	1.1	4.5	0.4	1.4	0.0	0.0
4: Two arg relations	39.0	17.5	26.9	0.4	0.0	0.0	0.0	0.7	9.0	13.6	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5: Three arg relations	30.0	12.9	25.6	1.0	1.0	1.1	1.1	1.2	1.3	12.5	0.8	0.5	0.2	0.5	0.2	0.3	0.0
6: Yes/no questions	52.0	2.0	28.5	8.4	0.1	0.0	0.9	1.2	50.6	15.5	0.1	0.0	0.0	0.0	0.0	0.0	0.0
7: Counting	51.0	10.1	21.9	17.8	10.9	11.1	9.6	9.4	13.1	15.3	2.0	2.4	1.8	1.0	0.7	0.7	0.0
8: Lists/sets	55.0	6.1	21.9	12.5	6.8	5.7	5.6	3.7	7.8	15.1	0.9	0.0	0.3	1.4	0.6	0.8	0.4
9 : Simple negation	36.0	1.5	42.9	10.7	0.0	0.6	0.0	0.0	32.7	13.0	0.3	0.0	0.0	0.0	0.0	0.0	0.0
10: Indefinite knowledge	56.0	2.6	23.1	16.5	0.8	0.6	0.0	0.0	3.5	12.9	0.0	0.0	0.2	0.0	0.0	0.0	0.0
11: Basic coreference	38.0	3.3	4.3	0.0	11.3	0.5	0.0	0.0	0.9	14.7	0.1	0.0	0.0	0.0	0.0	0.0	0.0
12: Conjunction	26.0	0.0	3.5	0.0	0.0	0.0	0.0	0.0	0.0	15.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0
13: Compound coreference	6.0	0.5	7.8	0.0	5.3	5.5	0.0	0.3	8.9	13.7	0.0	0.0	0.0	0.0	0.0	0.0	0.0
14: Time reasoning	73.0	2.0	61.9	1.2	20.2	1.3	0.8	3.8	18.2	14.5	0.1	0.0	0.0	0.2	0.0	0.0	0.1
15: Basic deduction	79.0	1.8	47.6	0.0	39.4	0.0	0.0	0.0	0.1	14.7	0.0	0.0	0.0	0.0	0.0	0.0	0.0
16: Basic induction	77.0	51.0	54.4	0.1	50.6	54.8	53.0	53.4	53.5	15.5	51.8	45.3	0.0	49.4	50.4	49.1	0.0
17: Positional reasoning	49.0	42.6	44.1	41.7	40.6	36.5	34.4	51.8	52.0	13.0	18.6	4.2	27.8	0.9	0.0	5.8	4.1
18: Size reasoning	48.0	9.2	9.1	9.2	8.2	8.6	7.9	8.8	47.5	14.9	5.3	2.1	8.5	1.6	8.4	1.8	0.7
19: Path finding	92.0	90.6	90.8	88.5	88.8	89.8	78.7	90.7	88.6	13.6	2.3	0.0	31.0	36.1	1.0	27.9	0.1
20: Agents motivations	9.0	0.2	2.2	0.0	0.0	0.0	0.2	0.3	5.5	14.6	0.0	0.0	0.0	0.0	0.0	0.0	0.0
# Failed	20	10	16	10	12	8	7	**5**	13	20	3	1	3	2	2	3	**0**
Average error rates (%)	51.3	15.2	33.2	12.7	20.1	11.7	**9.9**	11.3	20.5	14.2	4.2	2.8	3.7	4.6	3.2	4.3	**0.3**

Acknowledgement

- 感謝 曹爗文 同學發現投影片 上的錯字