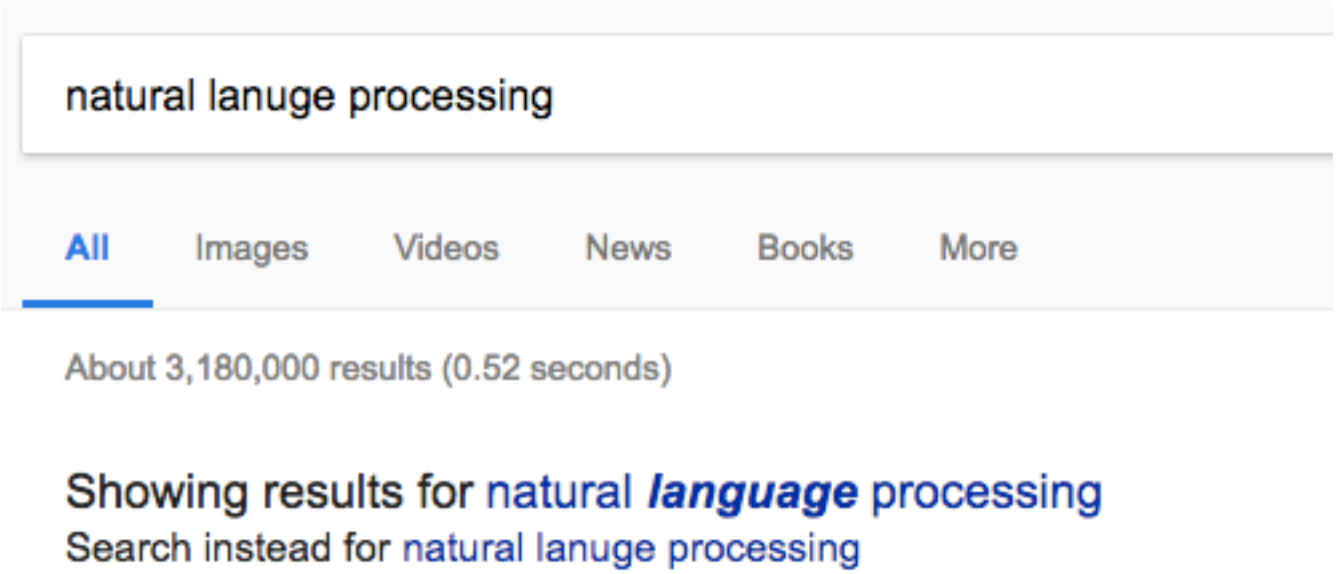
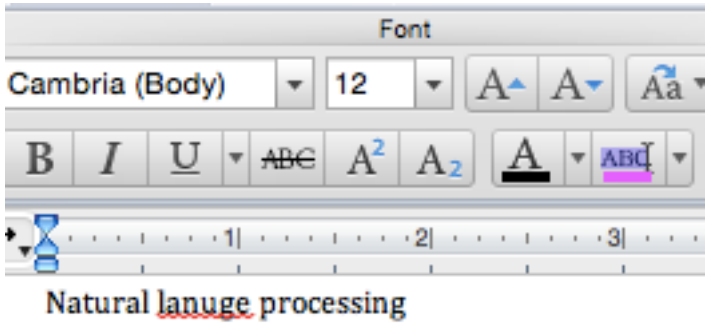
RANDOM TOPICS

Spell correction, Transformer, GAN for NLP,
Unsupervised MT, paper reading

Spell correction



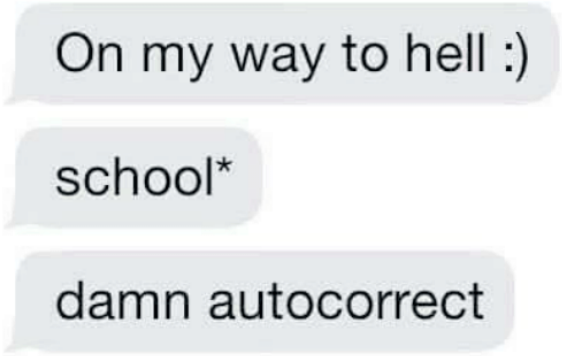


Rates of spelling errors ~1-20% depending on application

Tasks

- Spell error detection
- Spell error correction
  - Suggestion list
  - Suggest a correction
  - Autocorrect

Confidence of the system



Type of spelling errors

- Non-word errors
  - Languaeg -> Language
- Real-word errors
  - Typographical errors
    - Three -> there
  - Cognitive errors (homophones)
    - piece->peace
    - too->two
    - ค่ะ -> คะ

- Dictionary is required.

A note on dictionary

- Require a large dictionary (millions) to handle real world internet usage
- Automatically generated from unigrams list
  - This list includes misspellings
  - Misspellings occur less frequently than correct words in similar context
- Or let users add new words when not accepting corrections

Whitelaw, C., Hutchinson, B., Chung, G. Y., and Ellis,G. (2009). Using the web for language independent spellchecking and autocorrection.

Non-word spelling error

- Detection:
  - Any word not in dictionary is an error
- Correction:
  1. Generate candidates
  2. Choose the one that has the best
     1. Smallest edit distance
     2. Best noisy channel probability

Real word spelling error

- Detection:
  - ???
- Correction:
  1. Generate candidates
     1. From similar pronunciations
     2. From similar spellings
  2. Choose the one that has the best
     1. Smallest edit distance
     2. Best noisy channel probability

     The criterions has to be context sensitive!

     ไปไหนดีค่ะ -> ไปไหนดีคะ

Candidate generation

- Words with similar spelling
  - Edit distance
  - Typically at most 2 edit distance
  - Can be weighted based on the task, e.g. keyboard
  - Allow merging and splitting of words
    - Ladygaga -> Lady Gaga
    - |การะ|เกด| -> การะเกด
- Words with similar pronunciation
  - Example: Aspell, Soundex – a hash based on sounds
    - Jurafsky, Jarofsky, Jarovsky, and Jarovski all map to J612
  - Modern methods use Letter-2-sound (L2S) rules (also called Grapheme-to-Phoneme G2P)

- How to generate this in an efficient manner?

Candidate generation

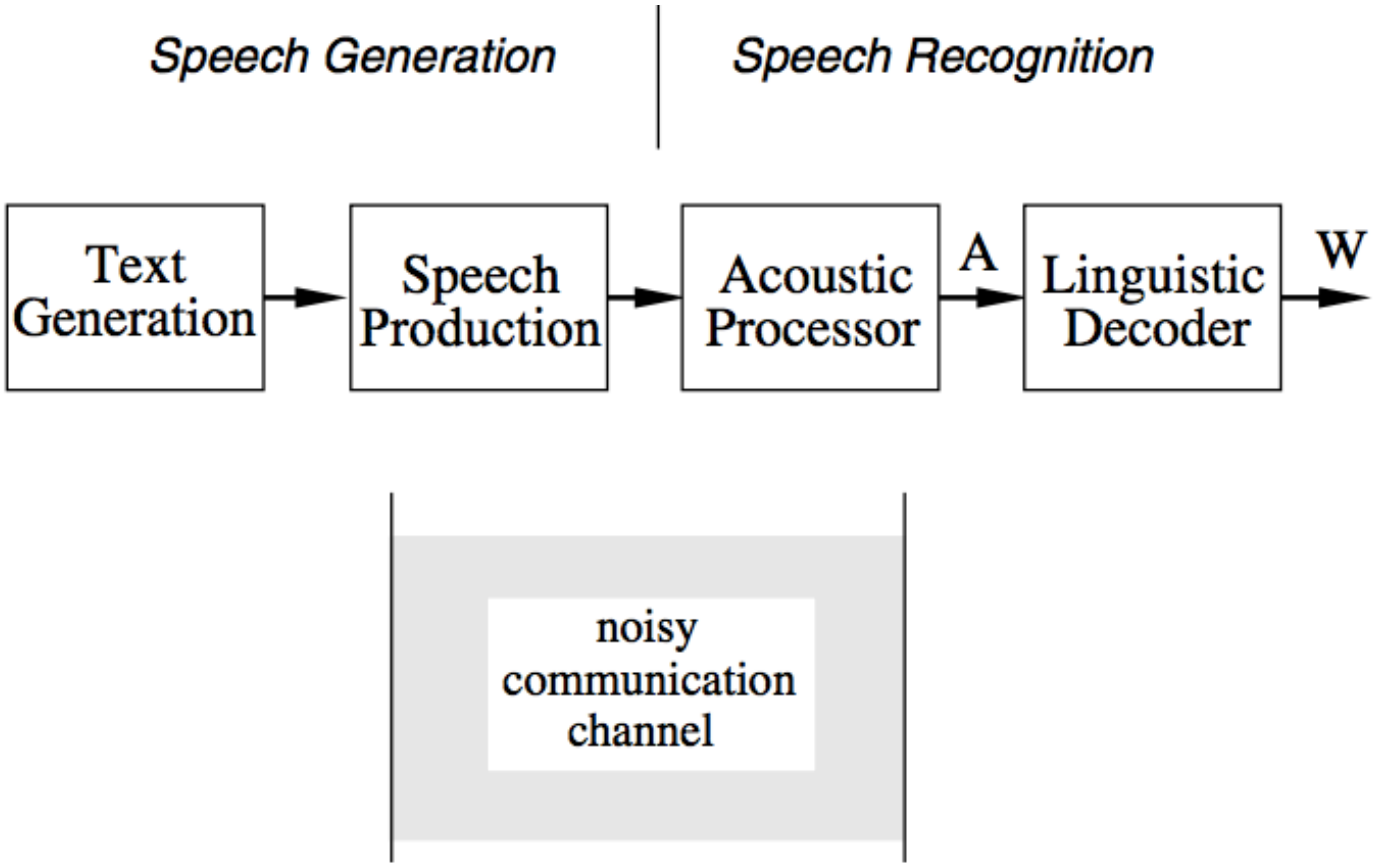
1. Run through dictionary, check edit distance (slow)
2. Generate all words with desired edit distance, then intersect with dictionary
3. Character n-gram inverted index. Find words which shared the most n-gram (fast)
   - Google -> goo oog ogl gle
   - Inverted index: goo -> google good goon good-bye
4. Compute with Finite State Transducer (FST) (fast)
5. Precomputed map of words
6. Mixture (https://github.com/wolfgarbe/SymSpell) (fast)

Tommi A Pirinen. Weighted Finite-State Methods for Spell-Checking and Correction 2014

The noisy channel model

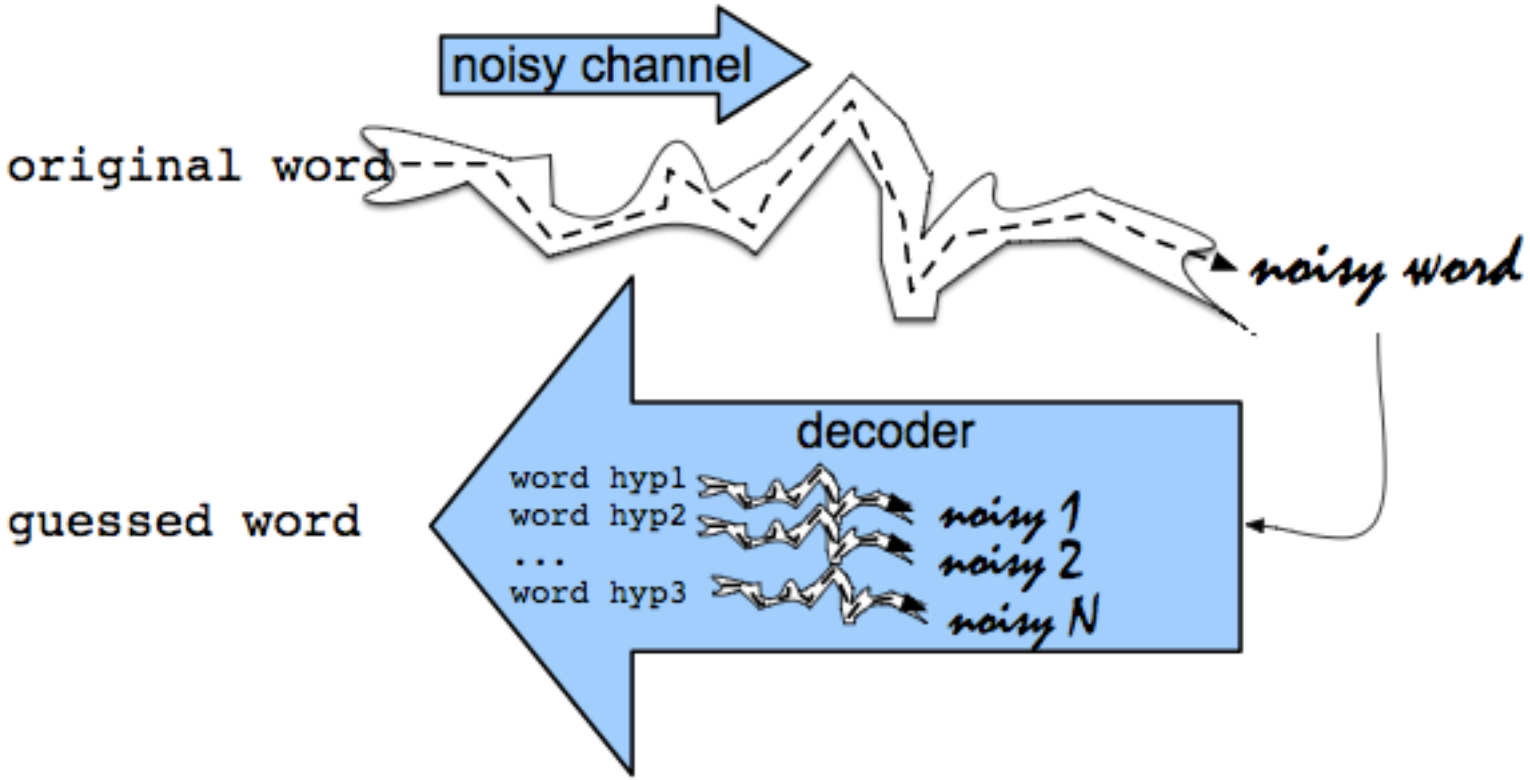
- Used to rank the best candidates

Information theoretic formulation



$$W^* = \arg\max_W P(W \mid A) \qquad P(W \mid A) = \frac{P(A \mid W)P(W)}{P(A)}$$

The noisy channel intuition



Jurafsky, chapter 5

Bayes rule

- An observation x of the misspelled word
- Find the correct word w* from vocab list V

$$= argmax_{w \in V} P(w|x)$$

$$= argmax_{w \in V} \frac{P(x|w)P(w)}{P(x)}$$

Language model

$$= argmax_{w \in V} P(x|w)P(w)$$

Channel/error model

Noisy channel example

- acress
- Let’s generate candidates

Noisy channel example

Edit distance 1 (Damerau-Levenshtein distance)

Levenshtein edit distance – instertion, deletion, substituion
Damerau-Levenshtein edit distance - instertion, deletion, substituion, transposition

		Transformation			
Error	Correction	Correct Letter	Error Letter	Position (Letter #)	Type
acress	actress	t	—	2	deletion
acress	cress	—	a	0	insertion
acress	caress	ca	ac	0	transposition
acress	access	c	r	2	substitution
acress	across	o	e	3	substitution
acress	acres	—	s	5	insertion
acress	acres	—	s	4	insertion

Jurafsky, chapter 5

Language model (unigram)

w	count(w)	p(w)
actress	9,321	.0000231
cress	220	.000000544
caress	686	.00000170
access	37,038	.0000916
across	120,844	.000299
acres	12,874	.0000318

Channel/error model

- Easiest to use a confusion matrix conditioned on the previous character.
- Based on counts

$$P(x|w) = \begin{cases} \dfrac{\text{del}[x_{i-1}, w_i]}{\text{count}[x_{i-1} w_i]}, & \text{if deletion} \\ \dfrac{\text{ins}[x_{i-1}, w_i]}{\text{count}[w_{i-1}]}, & \text{if insertion} \\ \dfrac{\text{sub}[x_i, w_i]}{\text{count}[w_i]}, & \text{if substitution} \\ \dfrac{\text{trans}[w_i, w_{i+1}]}{\text{count}[w_i w_{i+1}]}, & \text{if transposition} \end{cases}$$

w_i is the i-th character

Learning the confusion matrix

1. Have a data of misspellings-corrections pair
2. Have a data of misspellings and use EM
    1. Initialize the confusion matrix to be equally likely
    2. E-step: run the spell checker and correct words
    3. M-step: re-estimate the confusion matrix based on corrections

Or use nearby keys
(Each phone need to have different model)

Channel/error model

Candidate Correction	Correct Letter	Error Letter	x|w	P(x|w)
actress	t	-	c|ct	.000117
cress	-	a	a|#	.00000144
caress	ca	ac	ac|ca	.00000164
access	c	r	r|c	.000000209
across	o	e	e|o	.0000093
acres	-	s	es|e	.0000321
acres	-	s	ss|s	.0000342

Noisy channel model

Candidate Correction	Correct Letter	Error Letter	x|w	P(x|w)	P(w)	10^9*P(x|w)P(w)
actress	t	-	c|ct	.000117	.0000231	2.7
cress	-	a	a|#	.00000144	.000000544	0.00078
caress	ca	ac	ac|ca	.00000164	.00000170	0.0028
access	c	r	r|c	.000000209	.0000916	0.019
across	o	e	e|o	.0000093	.000299	2.8
acres	-	s	es|e	.0000321	.0000318	1.0
acres	-	s	ss|s	.0000342	.0000318	1.0

Note1: It is typical to use more than unigrams for the LM (very important for real word errors)

Note2: typically added a weight parameter tuned on dev set

$$P(x|w)P(w)^{\alpha}$$

Real word case for noisy channel

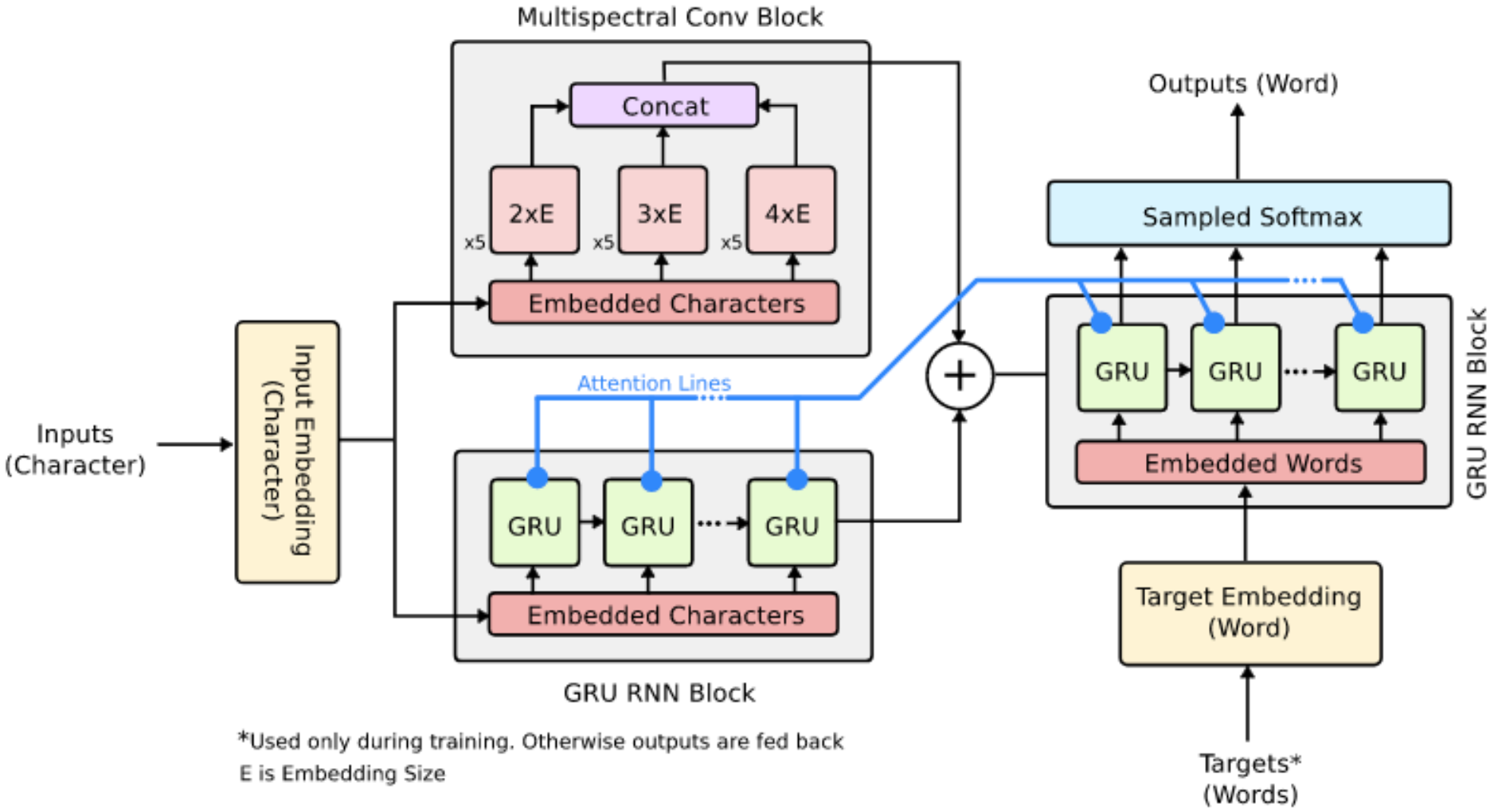
- Need to compare candidates against no correction
  - Error model needs a score for no correction
- Set to some parameter, c, representing how errorful our input is
- $P(w|x) = c$ for $w = x$
- The rest of the probability 1-c is weighted by the confusion matrix

Classifier-based methods

- Have a pair of correction and features
  - whether/weather
  - Features:
    1. cloudy in +- 5 words
    2. Followed by “to VERB”
    3. Followed by “or not”

- End-to-end model using neural networks
  - Seq2Seq model

Neural-based spell correction example



https://arxiv.org/pdf/1709.06429.pdf

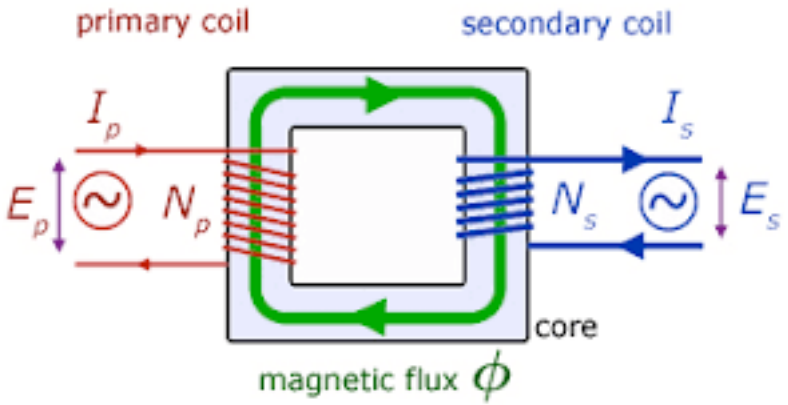
		Model Sizes		
Model	*Corpus Size (words)*	*No of Parameters*	*Training Time (hours)*	*Memory Footprint (Disk)*
CCEAD (Ours)	12M	17M	24	200MB
Velocitap [1]	1.8B	-	days	1.5GB
Transformer [13]	12M	44M	2	541MB
LM + SC	12M	26K	24	320MB
CCED	12M	17M	24	200MB

TABLE 8: Model comparison. Cell with a '-' is not applicable for the given model comparison.

User No.	Input medium	CCED	LM+SC	CCEAD(Ours)	Velocitap [1]
User 1	Mobile touch-screen	1.4	1.0	1.0	1.0
User 2	Mobile touch-screen	1.9	2.0	1.9	1.9
User 3	Mobile touch-screen	0.1	0.1	0.1	0.0
User 4	Mobile touch-screen	0.8	1.3	1.0	1.0
User 5	Mobile touch-screen	1.1	1.3	0.8	1.0
User 6	Mobile touch-screen	5.8	6.3	1.1	1.0
User 7	Mobile touch-screen	0.8	0.9	0.8	1.0
User 8	Mobile touch-screen	5.1	5.1	5.3	3.0
User 2	Physical keyboard	1.1	1.4	1.3	-
User 3	Physical keyboard	0.5	0.9	0.5	-
User 4	Physical keyboard	1.6	2.8	1.4	-
User 5	Physical keyboard	1.6	1.9	1.6	-
User 6	Physical keyboard	3.9	3.8	4.1	-
User 7	Physical keyboard	3.3	3.3	3.2	-
User 8	Physical keyboard	4.9	5.1	4.9	-

Character Error rate

Transformer?



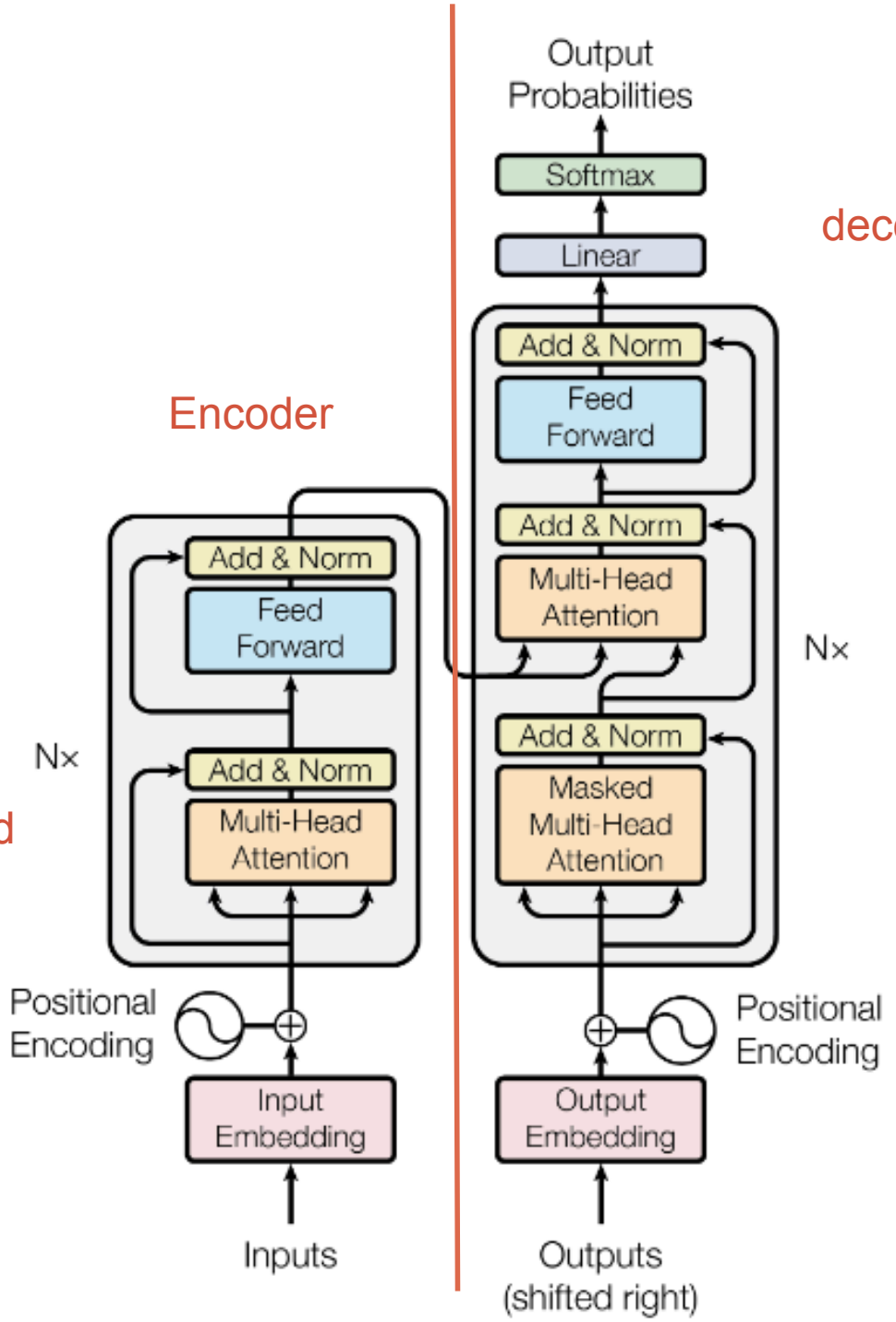
Attention is all you need

Abstract

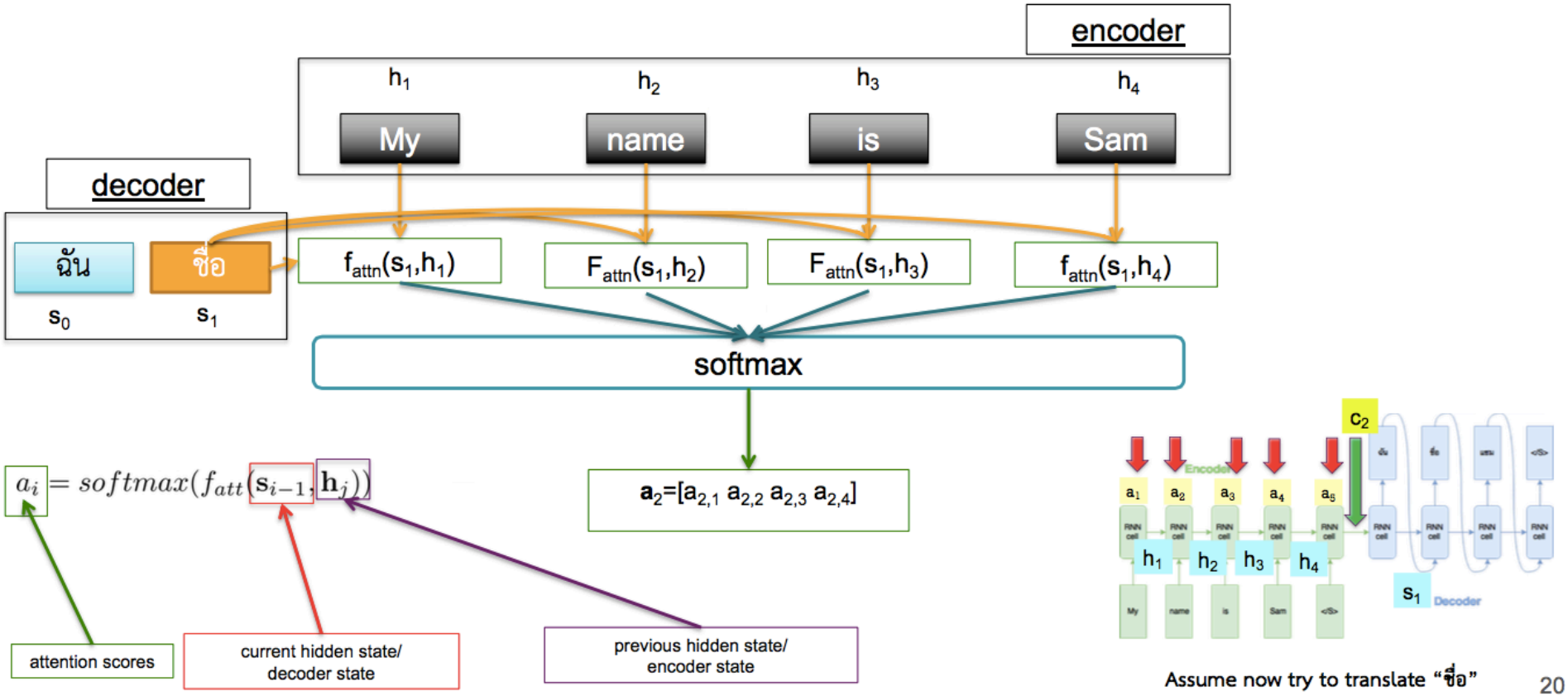
The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 English-to-German translation task, improving over the existing best results, including ensembles, by over 2 BLEU. On the WMT 2014 English-to-French translation task, our model establishes a new single-model state-of-the-art BLEU score of 41.8 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature. We show that the Transformer generalizes well to other tasks by applying it successfully to English constituency parsing both with large and limited training data.

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

To eliminate GRU which remembers time, encode position instead



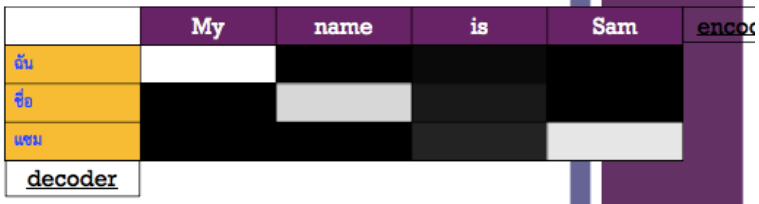
Attention Calculation Example (1): Attention Scores



Assume now try to translate "ชื่อ"
to give high weight at "name"

$$a_i = softmax(f_{att}(\mathbf{s}_{i-1}, \mathbf{h}_j))$$

Type of Attention mechanisms



(Remember that there are many variants of attention function f_{attn})

Additive attention: The original attention mechanism (Bahdanau et al., 2015) uses a one-hidden layer feed-forward network to calculate the attention alignment:

$$f_{attn}(\mathbf{s}_{i-1}, \mathbf{h}_j) = tanh(\mathbf{W}_a[\mathbf{s}_{i-1}; \mathbf{h}_j])$$

Multiplicative attention: Multiplicative attention (Luong et al., 2015) simplifies the attention operation by calculating the following function:

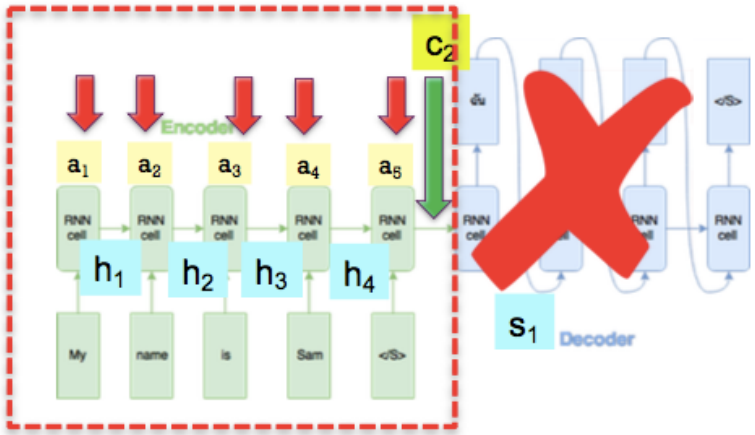
$$f_{attn}(\mathbf{s}_{i-1}, \mathbf{h}_j) = \mathbf{s}_{i-1}^{\top} \mathbf{W}_a \mathbf{h}_j$$

Self-attention: Without any additional information, however, we can still extract relevant aspects from the sentence by allowing it to attend to itself using self-attention (Lin et al., 2017)

$$\mathbf{a} = softmax(\mathbf{w}_{s_2} tanh(\mathbf{W}_{s_1} \mathbf{H}^T))$$

Key-value attention: key-value attention (Daniluk et al., 2017) is a recent attention variant that separates form from function by keeping separate vectors for the attention calculation.

Self attention



Assume now try to translate “ชื่อ”

to give high weight at “name”

No need for additional information in order to select where to attend

Convolution

Similar to CNN in flow

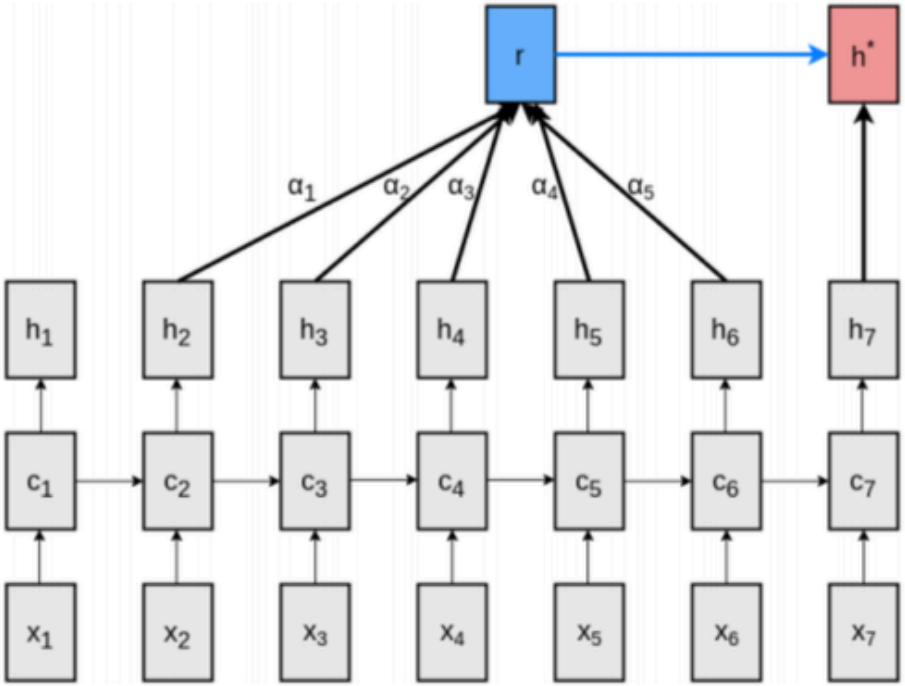
Self-Attention

https://nlp.stanford.edu/seminar/details/lkaiser.pdf

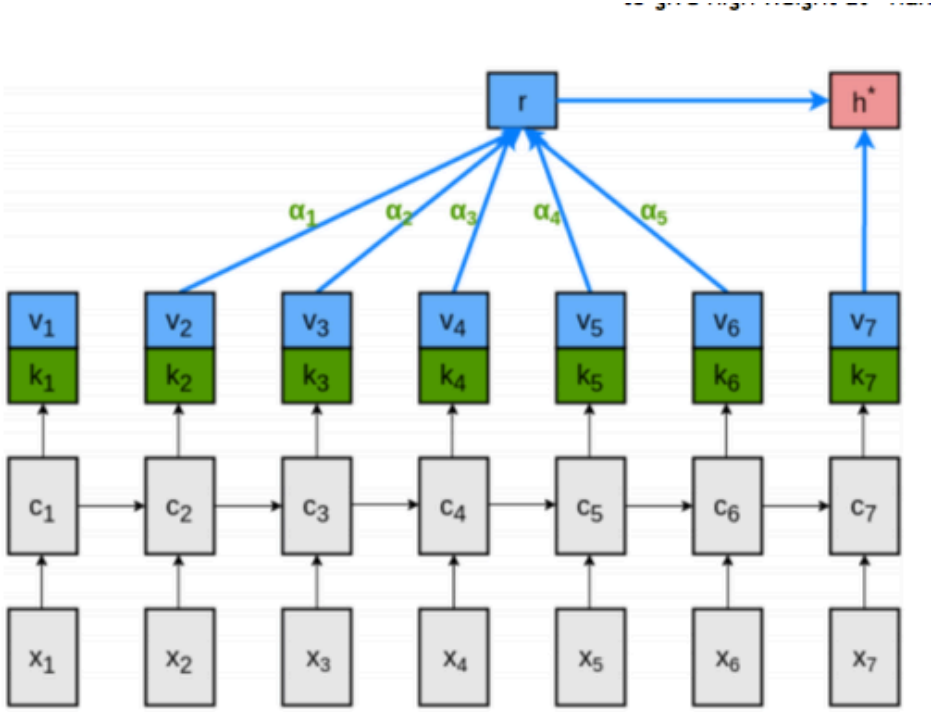
Key-value attention

Normal attention use the same vector to find the position and use as values

Use key to find the position, and use the values as information

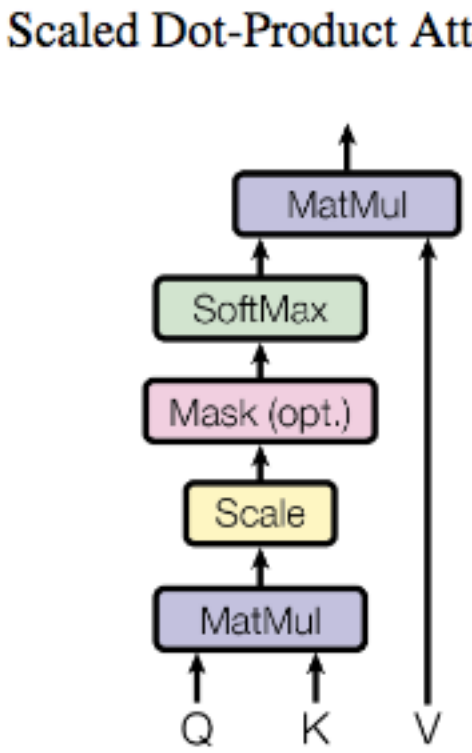


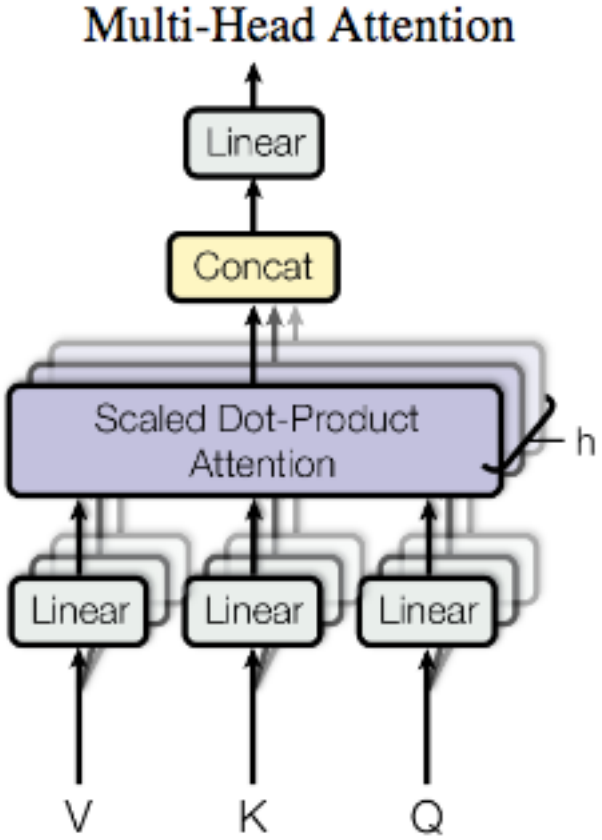
(a) Neural language model with attention.



(b) Key-value separation.

Multi-head attention





What's this????

Query – used with Key to determine the position
Value – used as the information after determining the position

Attention drawback

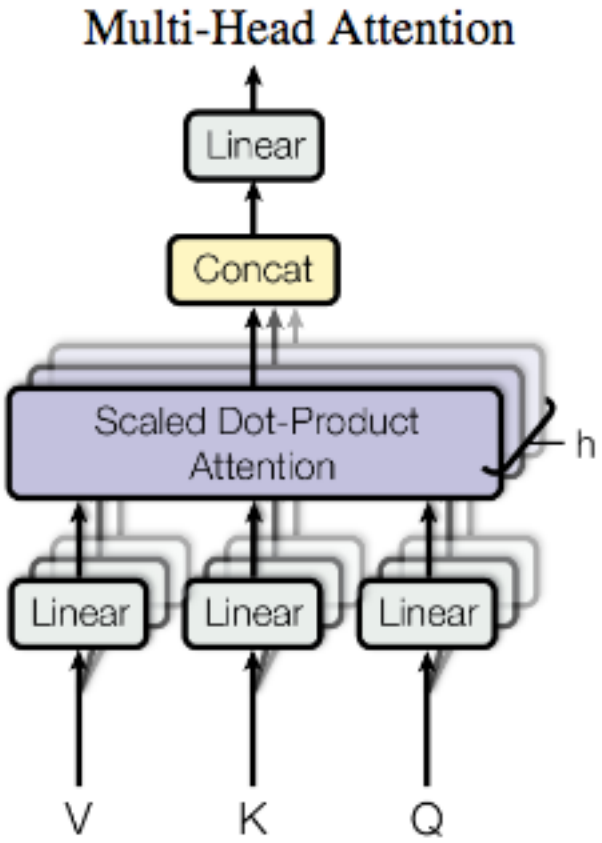
- Convolution: weights * input. Each weights are different. So position is encoded.
- Self-attention: a weighted average. Position information is lost at the output

Convolution

Self-Attention

Multi-head attention

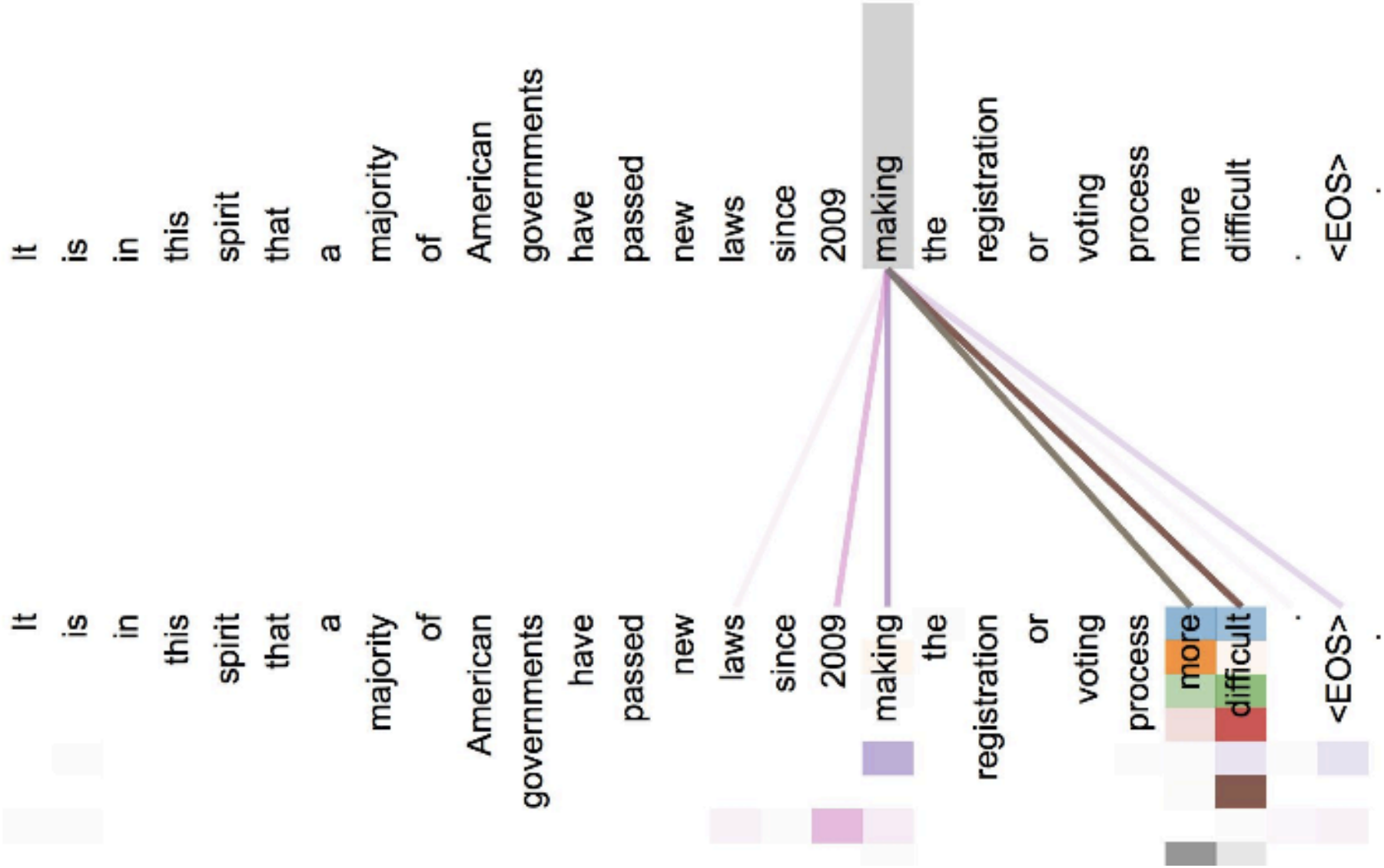
- Multiple attention layers (heads) that run in parallel
- Each head use different weights
- Each head can learn different relationship

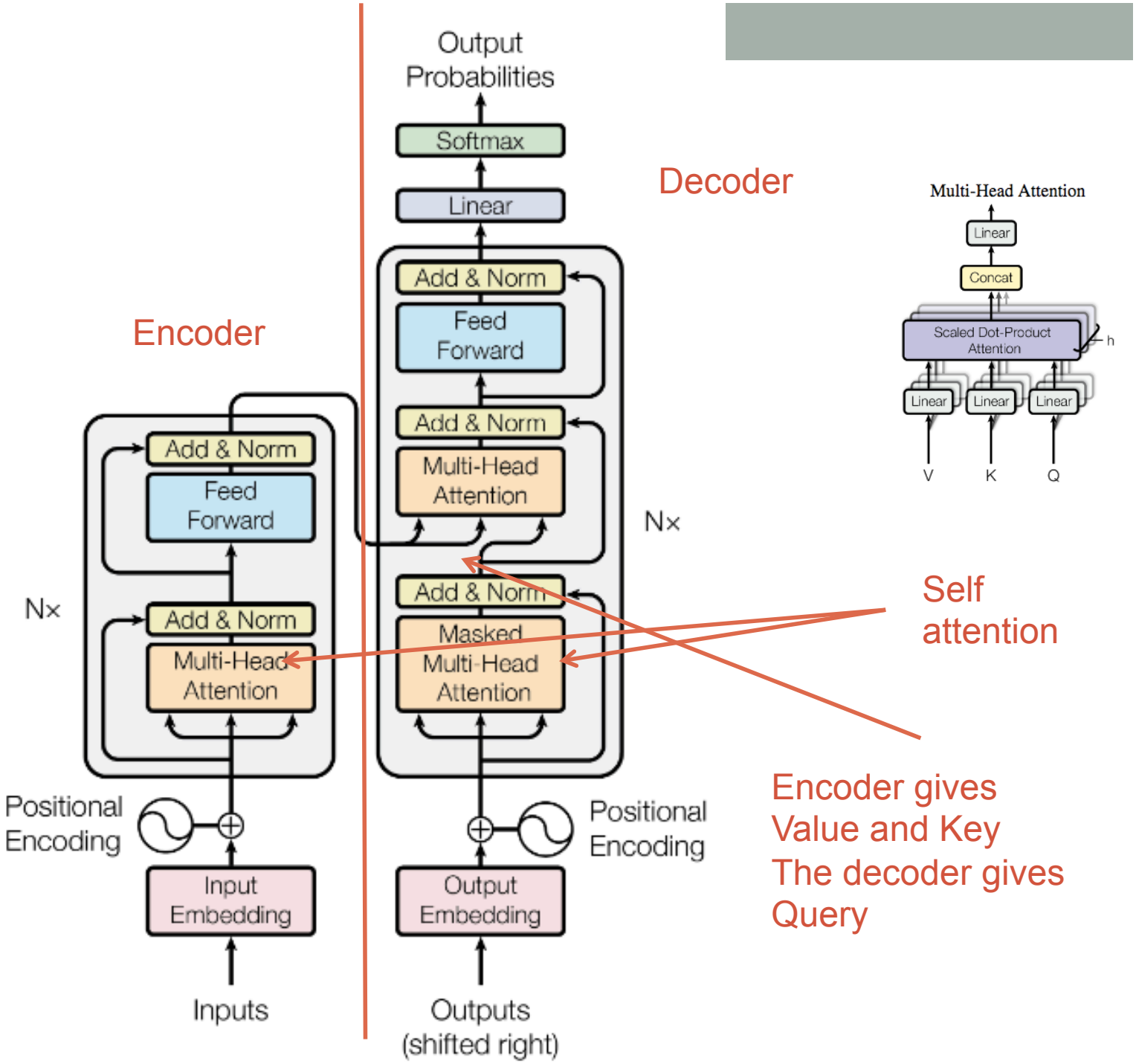


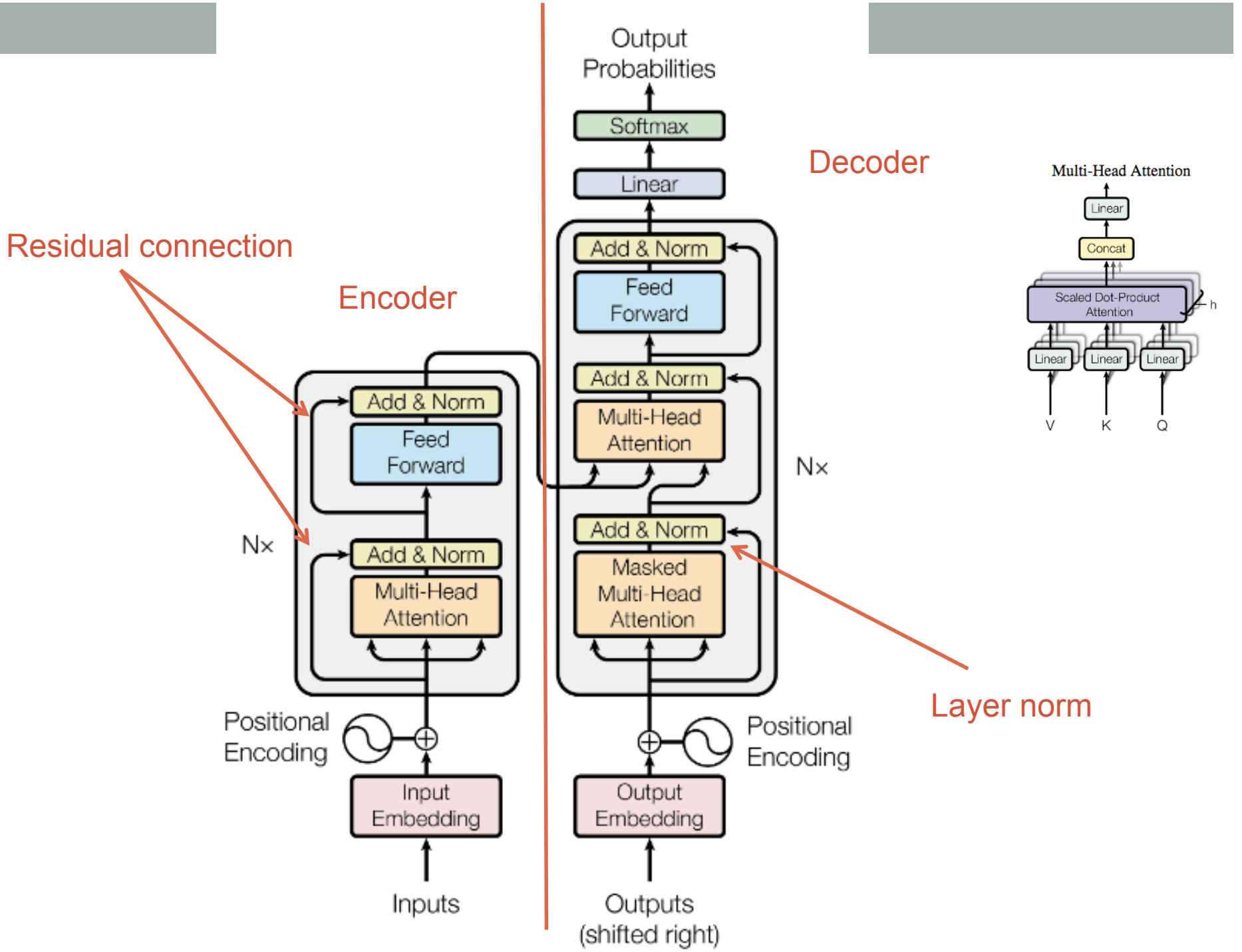
Convolution

Multi-Head Attention

Multi-head visualization



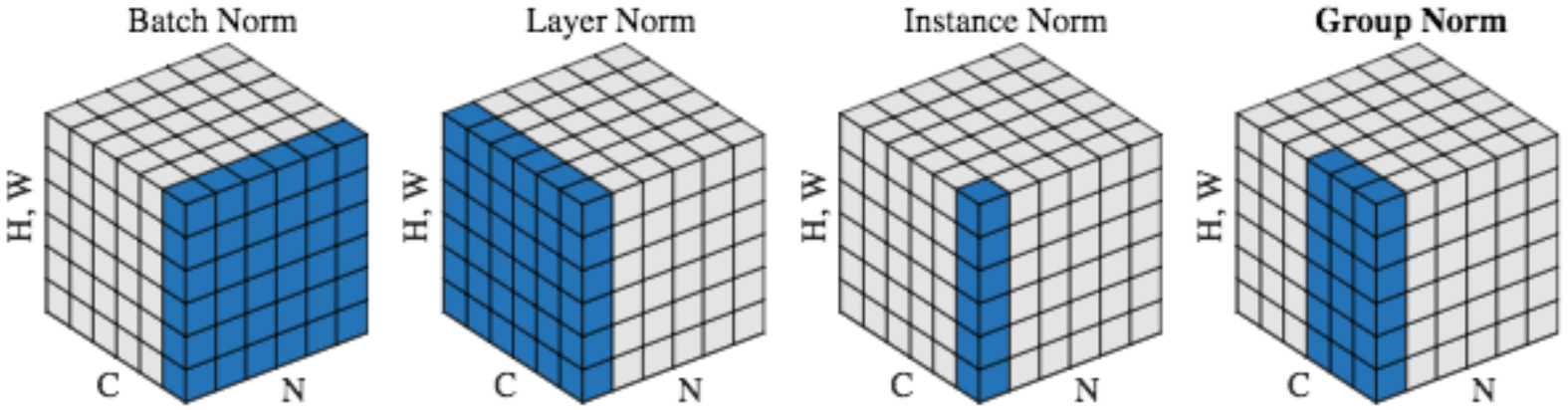

Output Probabilities
Softmax
Linear
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Add & Norm
Masked Multi-Head Attention
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Positional Encoding
Positional Encoding
Input Embedding
Output Embedding
Inputs
Outputs (shifted right)
Nx
Nx
Encoder
Decoder
Multi-Head Attention
Linear
Concat
Scaled Dot-Product Attention
h
Linear
Linear
Linear
V
K
Q
Self attention
Encoder gives Value and Key
The decoder gives Query


Residual connection
Encoder
Decoder
Output Probabilities
Softmax
Linear
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Add & Norm
Masked Multi-Head Attention
Nx
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Nx
Positional Encoding
Positional Encoding
Input Embedding
Output Embedding
Inputs
Outputs (shifted right)
Layer norm
Multi-Head Attention
Linear
Concat
Scaled Dot-Product Attention
h
Linear
Linear
Linear
V
K
Q

Layer norm

- Normalize the mean and SD
- Batch norm vs layer norm vs Instance norm vs group norm

Group is used to distributed models into multiple GPUs

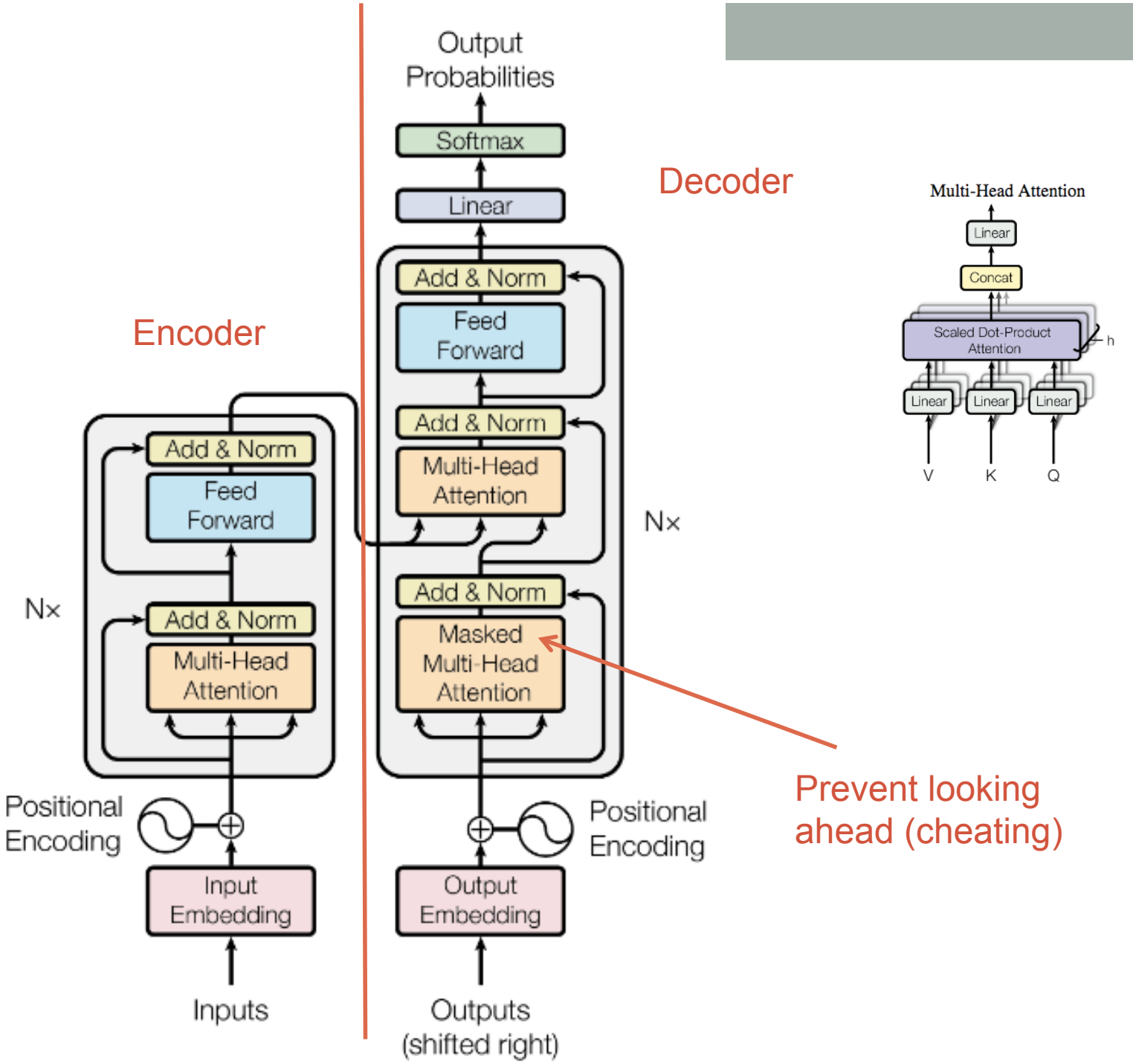


N – example in mini batch
C – Channel output
H,W – spatial coordinates (x,y)

Box is output tensor from CNN

BN and GN are usually best, GN is better when batch size is small (Vision task)

https://arxiv.org/abs/1803.08494


Output Probabilities
Softmax
Linear
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Add & Norm
Masked Multi-Head Attention
Nx
Decoder
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Nx
Encoder
Positional Encoding
Input Embedding
Inputs
Positional Encoding
Output Embedding
Outputs (shifted right)
Prevent looking ahead (cheating)
Multi-Head Attention
Linear
Concat
Scaled Dot-Product Attention
h
Linear
Linear
Linear
V
K
Q

MT results

Table 2: The Transformer achieves better BLEU scores than previous state-of-the-art models on the English-to-German and English-to-French newstest2014 tests at a fraction of the training cost.

Model	BLEU		Training Cost (FLOPs)	
	EN-DE	EN-FR	EN-DE	EN-FR
ByteNet [18]	23.75			
Deep-Att + PosUnk [39]		39.2		$1.0 \cdot 10^{20}$
GNMT + RL [38]	24.6	39.92	$2.3 \cdot 10^{19}$	$1.4 \cdot 10^{20}$
ConvS2S [9]	25.16	40.46	$9.6 \cdot 10^{18}$	$1.5 \cdot 10^{20}$
MoE [32]	26.03	40.56	$2.0 \cdot 10^{19}$	$1.2 \cdot 10^{20}$
Deep-Att + PosUnk Ensemble [39]		40.4		$8.0 \cdot 10^{20}$
GNMT + RL Ensemble [38]	26.30	41.16	$1.8 \cdot 10^{20}$	$1.1 \cdot 10^{21}$
ConvS2S Ensemble [9]	26.36	**41.29**	$7.7 \cdot 10^{19}$	$1.2 \cdot 10^{21}$
Transformer (base model)	27.3	38.1	**$3.3 \cdot 10^{18}$**	
Transformer (big)	**28.4**	**41.8**	$2.3 \cdot 10^{19}$	

Can use for other tasks, like ASR, parsing, etc.

Generative Adversarial Networks (GANs)

Learning distributions

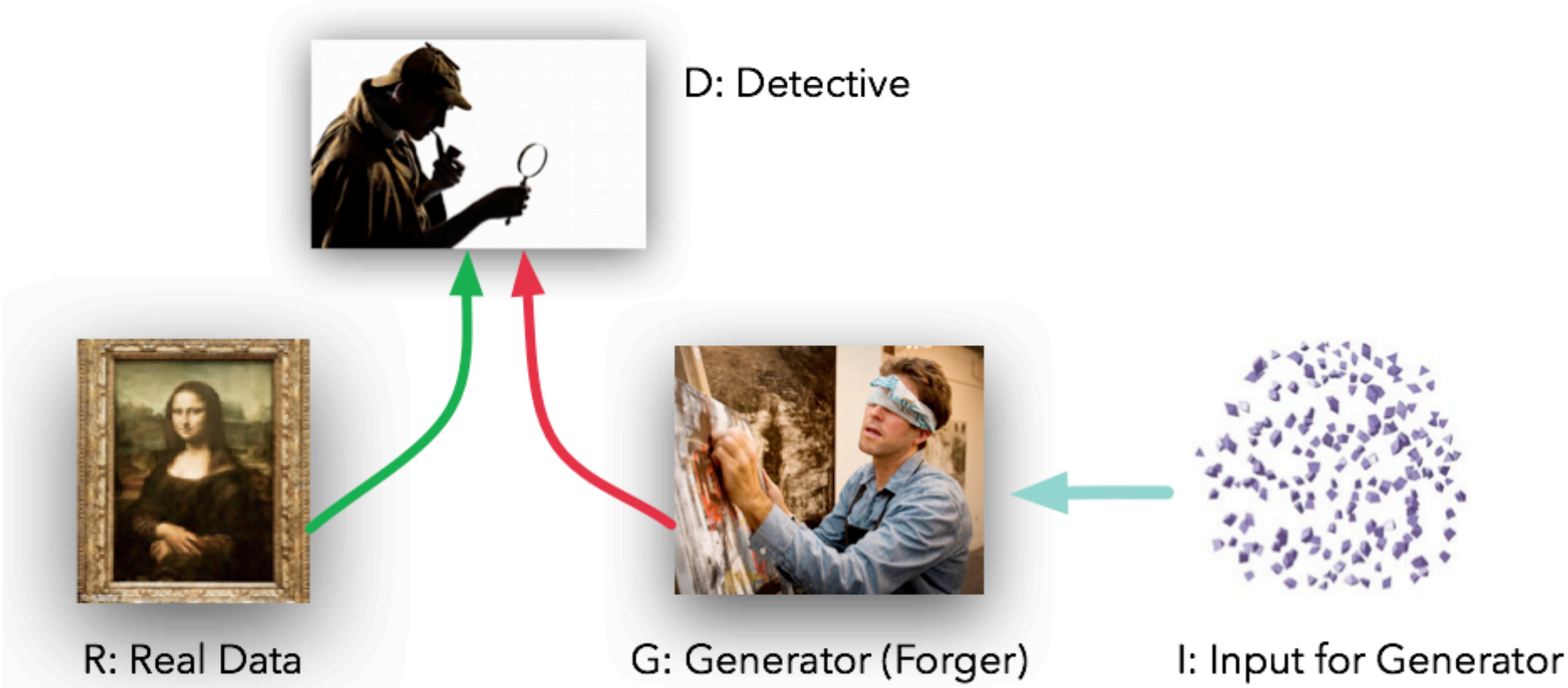
- Supervised learning tasks usually have one correct answer

Learning distributions

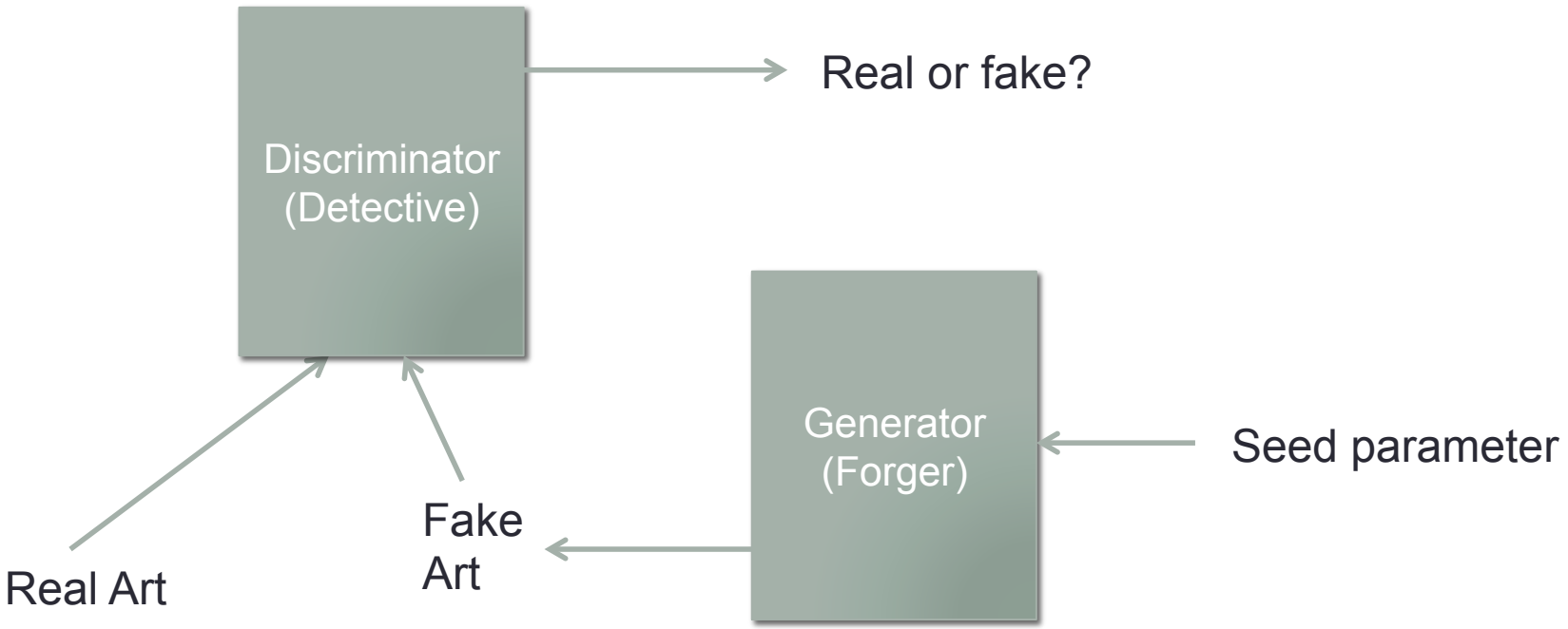
- Supervised learning tasks usually have one correct answer
- Sometimes there are more than one possibility
  - What is the next frame of a video?
  - What is the missing pixels in an image?
  - Which side will a pencil fall over?
  - What word is missing from the blank?
    - I eat ____

Generative Adversarial Networks (GAN)

- Consider a money counterfeiter
  - He wants to make fake money that looks real
  - There's a police that tries to differentiate fake and real money.
- The counterfeiter is the adversary and is generating fake inputs. – Generator network
- The police is try to discriminate between fake and real inputs. – Discriminator network

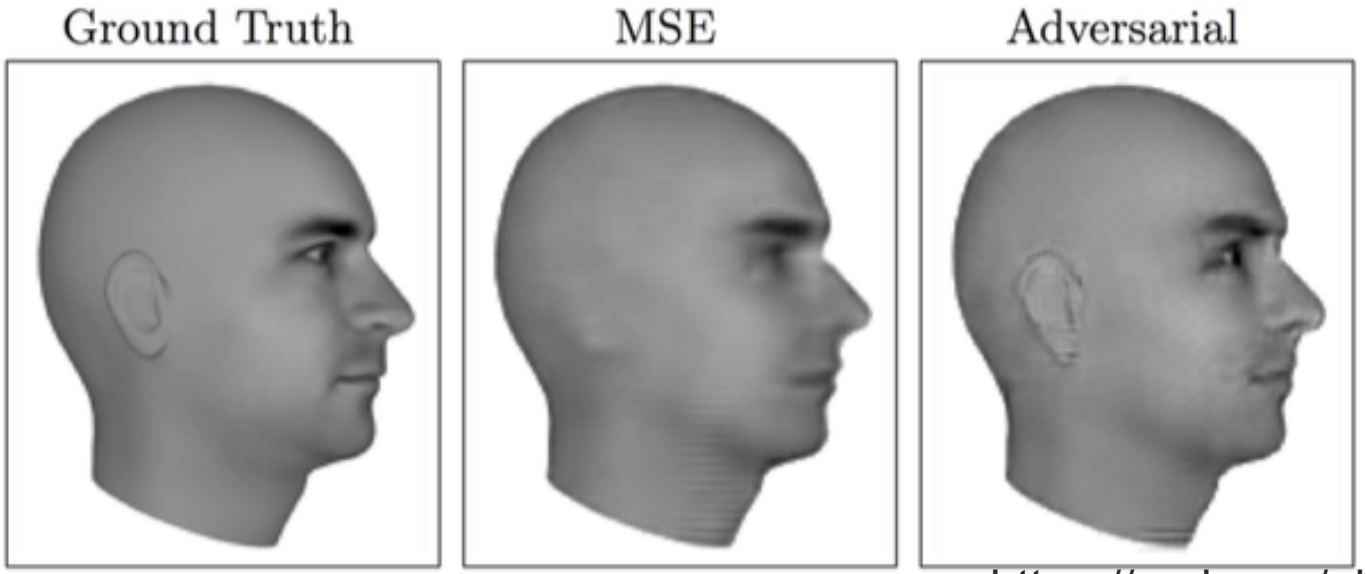


https://medium.com/@devnag/generative-adversarial-networks-gans-in-50-lines-of-code-pytorch-e81b79659e3f


Discriminator
(Detective)
Real or fake?
Generator
(Forger)
Seed parameter
Real Art
Fake
Art

Why GANs

- Learning distribution instead of the average answer



https://arxiv.org/abs/1701.00160

GAN in a nutshell

D( ) – discriminator, outputs probability of real input
G( ) – generator, generates new data from random noise

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))].$$

Samples from z (noise), and generate G(z). The discriminator wants to give a low probability

Samples from data (real), The discriminator wants to give a high probabiliy

- Minmax: minimize the loss in the worst case scenario.
- At equilibrium, the discriminator will output 0.5 for real and fake data.

GAN training

- Create two networks, D and G
  - Discriminator training
    - Generates mini-batch of fake things + real things
    - GSD with reward

$$\frac{1}{m}\sum_{i=1}^{m}\left[\log D\left(\boldsymbol{x}^{(i)}\right)+\log\left(1-D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right)\right]$$

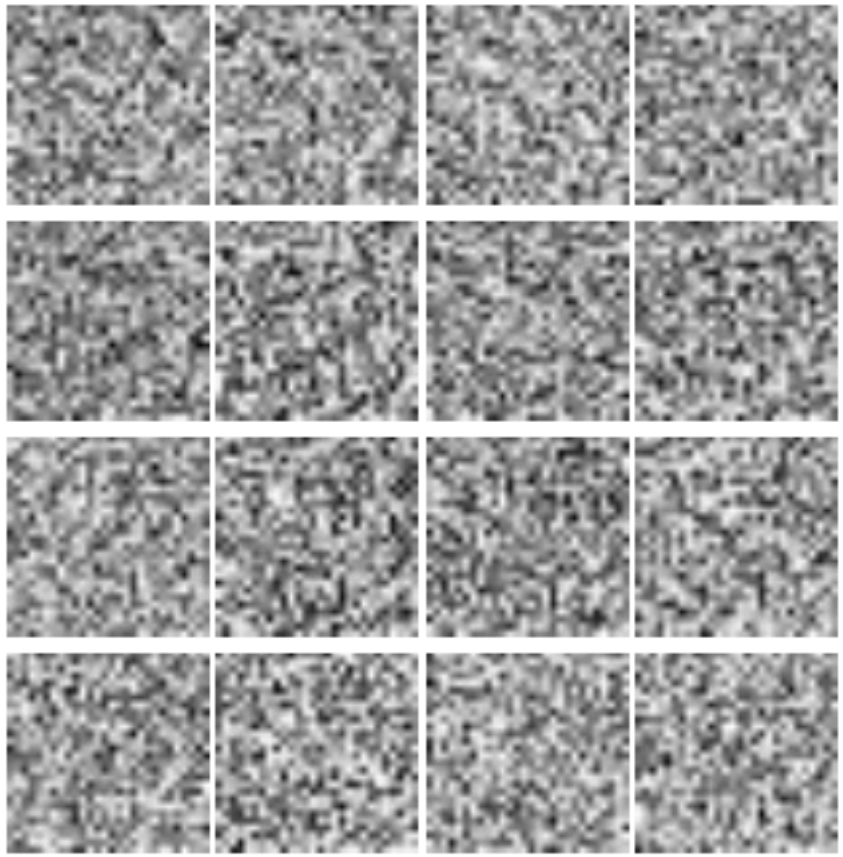
  - Generator training
    - Generate fake things
    - GSD with loss

$$\frac{1}{m}\sum_{i=1}^{m}\log\left(1-D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right)$$

- Alternate between the two

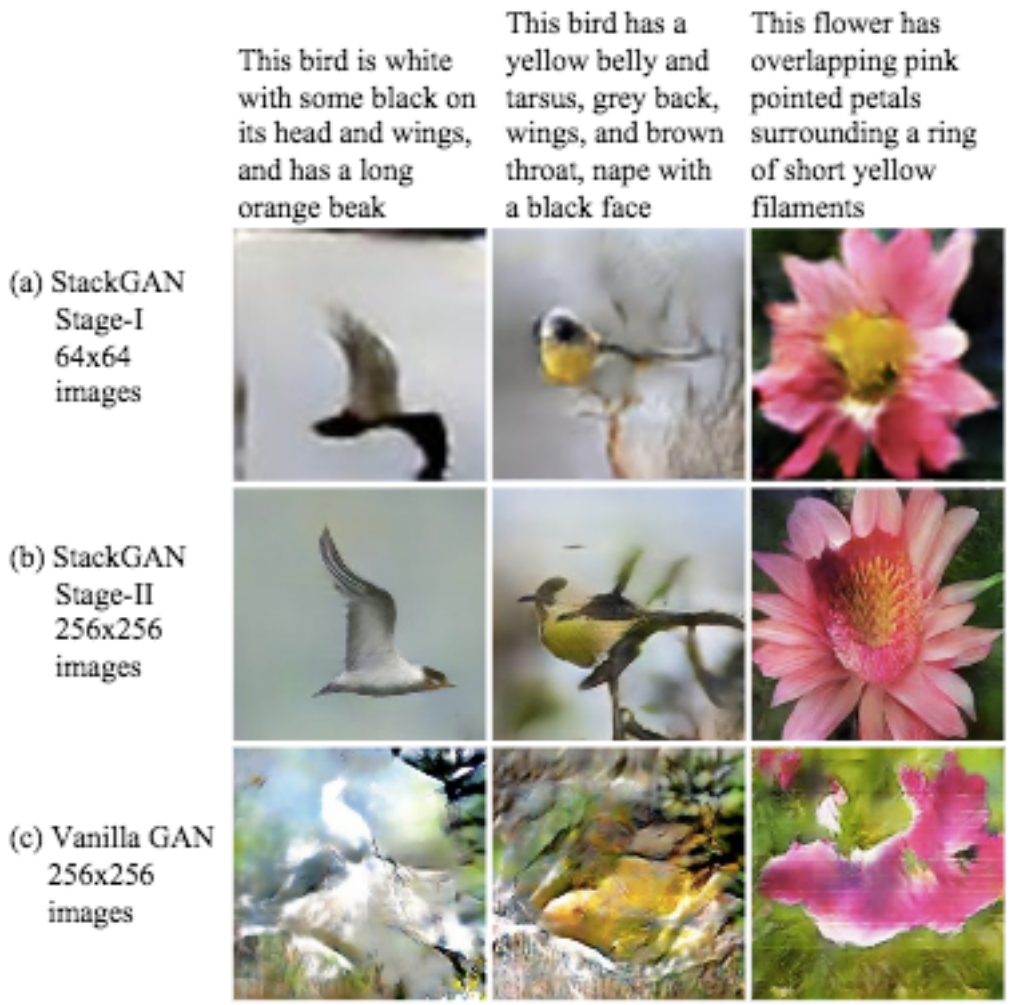
GAN example

Generator output starts from random noise and gets better as we train.



https://wiseodd.github.io/techblog/2016/09/17/gan-tensorflow/

StackGA



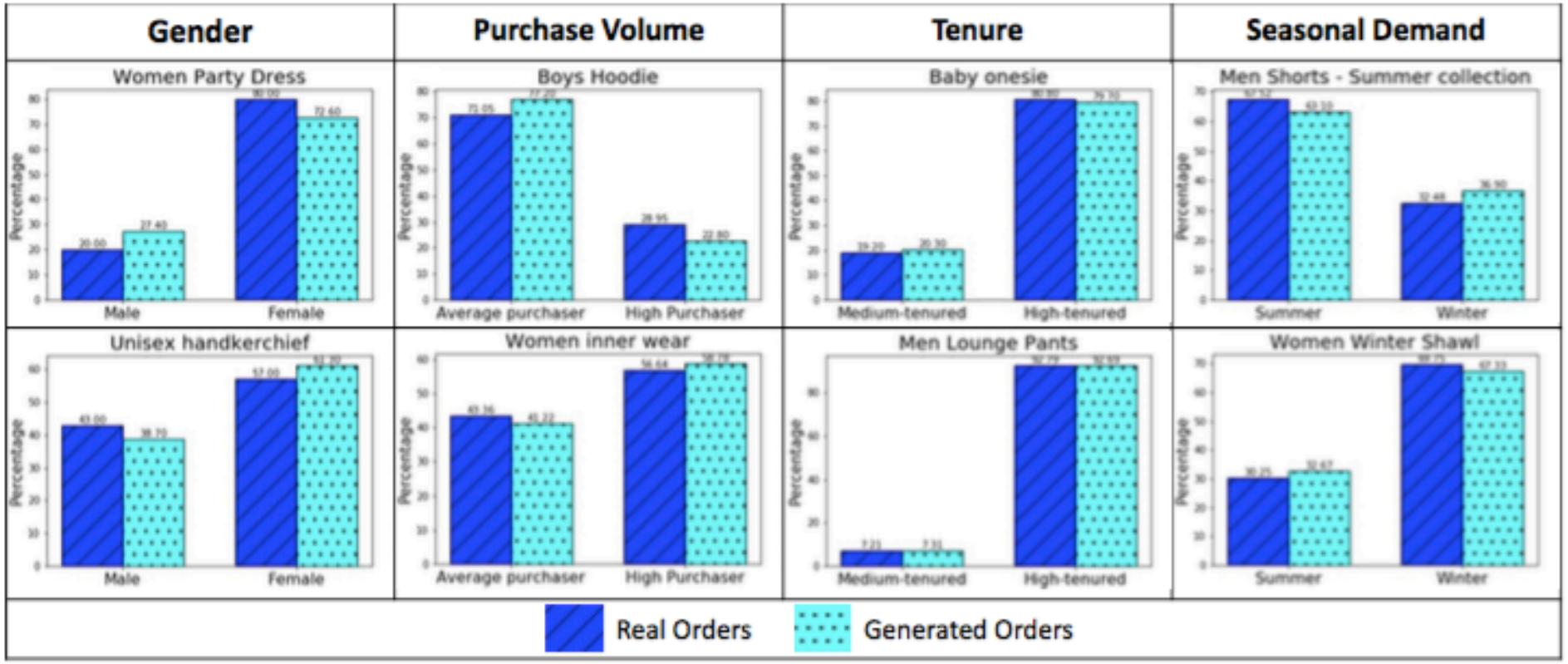
H Zhang, StackGAN: Text to Photo-realistic Image Synthesis with Stacked Generative Adversarial Networks, 2017

TextureGAN

W Xian, TextureGAN: Controlling Deep Image Synthesis with Texture Patches, 2017

eCommerceGAN

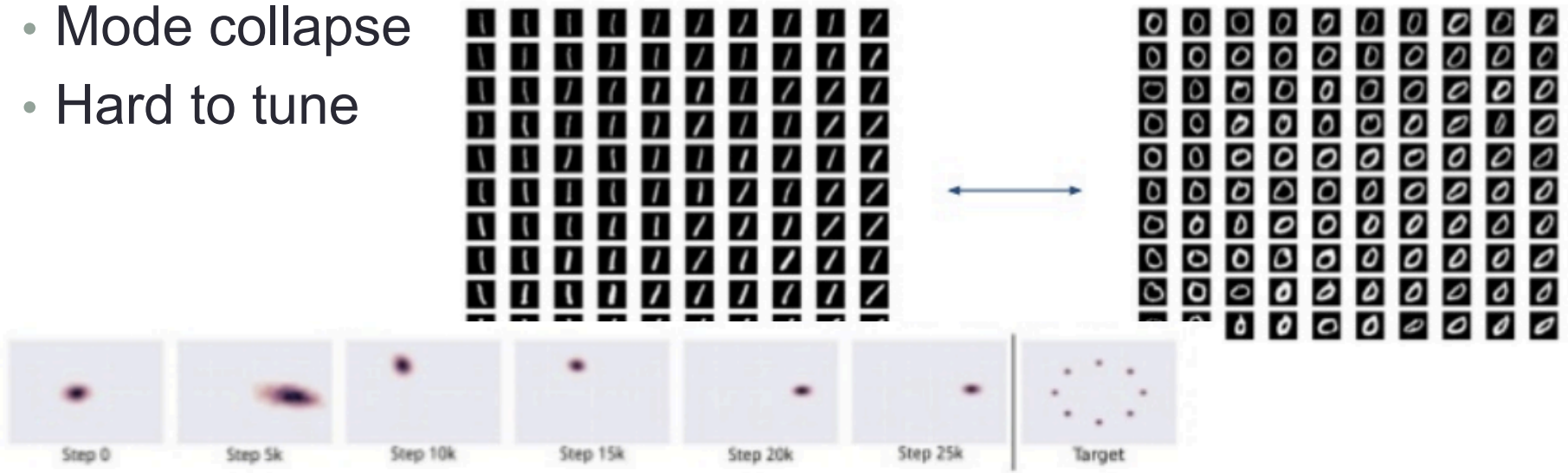
- Predicts sales, prices, and customers using text information for a new product



A Kumar, eCommerceGAN : A Generative Adversarial Network for E-commerce, 2018

GAN problems

- Hard to tune
  - Loss is not meaningful (model evaluation is hard)
  - Prone to initialization
  - “An art” to tune
- Mode collapse
- Hard to tune



mode collapsing

https://www.slideshare.net/ssuser77ee21/generative-adversarial-networks-70896091

Wasserstein GAN (WGAN)

- Original GAN loss has bad properties
  - Hard to learn from a discriminator (adversarial)
- Use a different loss function
  - Switch to actor-critic model
  - Critic gives a score (distance) rather than classifying authenticity
  - Use Wasserstein distance
  - Better gradients overall

WGAN vs GAN

$$\frac{1}{m}\sum_{i=1}^{m}\left[\log D\left(\boldsymbol{x}^{(i)}\right)+\log\left(1-D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right)\right]$$

Discriminator reward

$$\frac{1}{m}\sum_{i=1}^{m}\log\left(1-D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right)$$

Generator loss

F is not constrained to be between 0 and 1. However, the weights of F are clipped to some maximum

$$\frac{1}{m}\sum_{i=1}^{m}[F(x^{(i)})-F(G(z^{(i)}))]$$

Critic reward

$$\frac{1}{m}\sum_{i=1}^{m}[-F(G(z^{(i)}))]$$

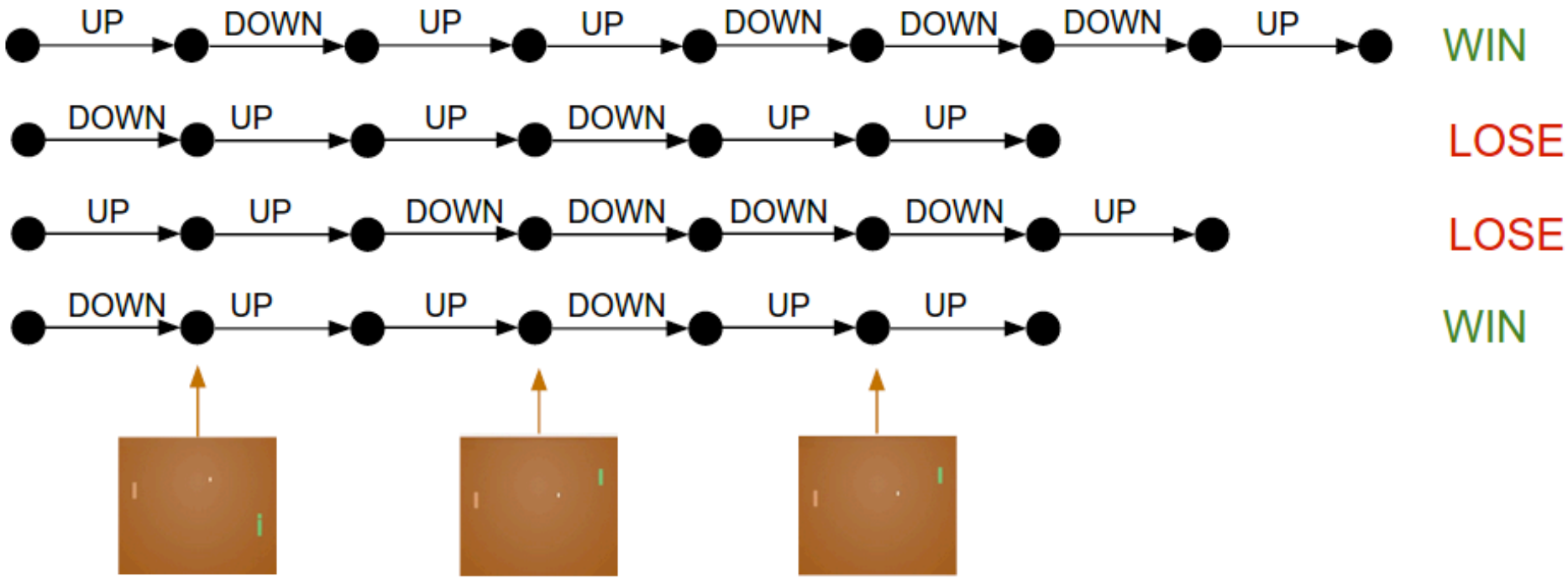
Generator loss

WGAN makes things easier

- With WGAN many people start exploring usage of GANs in more domains

Aside: RL and policy gradients

- Credit assignment problem in reinforcement learning

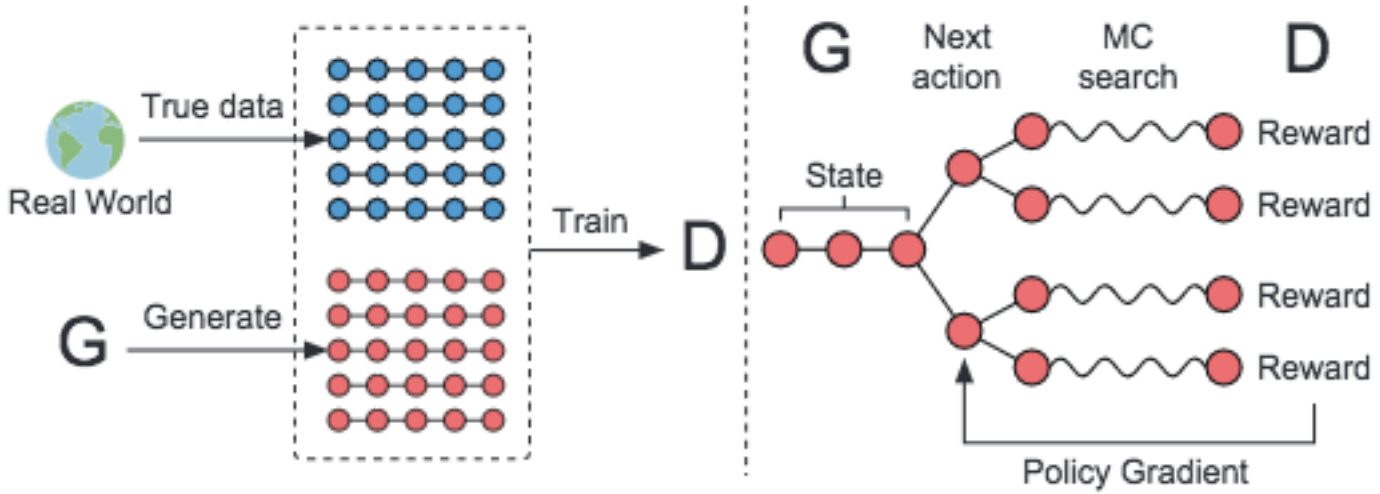


which move makes you win?
For RL with policy gradients, we increase the probability of every move that results in a win

http://karpathy.github.io/2016/05/31/rl/

GAN with text generation (SeqGAN)

- Use policy gradient to update the generator (the agent in RL setting)
- The discriminator (critic) gives the reward



How is this different from our previous text generation? (Maximum likelihood)
Want to generate exact vs Want to generate “real” sentences

https://arxiv.org/pdf/1609.05473.pdf

Table 2: Chinese poem generation performance comparison.

Algorithm	Human score	p-value	BLEU-2	p-value
MLE	0.4165	0.0034	0.6670	$< 10^{-6}$
SeqGAN	**0.5356**		**0.7389**	
Real data	0.6011		0.746	

Table 3: Obama political speech generation performance.

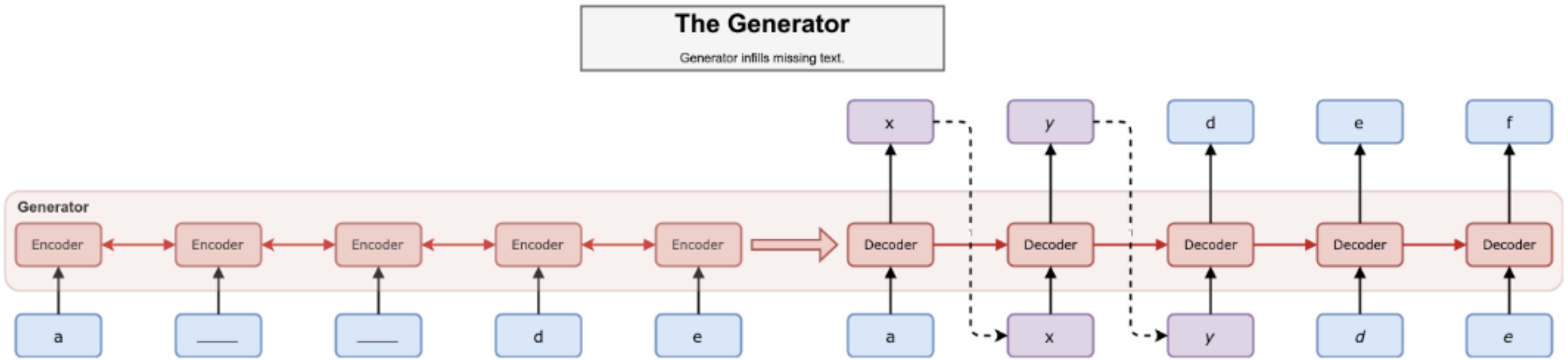
Algorithm	BLEU-3	p-value	BLEU-4	p-value
MLE	0.519	$< 10^{-6}$	0.416	0.00014
SeqGAN	**0.556**		**0.427**	

Table 4: Music generation performance comparison.

Algorithm	BLEU-4	p-value	MSE	p-value
MLE	0.9210	$< 10^{-6}$	22.38	0.00034
SeqGAN	**0.9406**		**20.62**	

MaskGAN

- GAN to fill in the blank. Encoder - Decoder



https://arxiv.org/pdf/1801.07736.pdf

MaskGAN

- GAN to fill in the blank. Encoder - Decoder

Ground Truth	**Pitch Black was a complete shock to me when I first saw it back in 2000 In the previous years I**
MaskGAN	Pitch Black was a complete shock to me when I first saw it back in 1979 I was really looking forward
	Pitch Black was a complete shock to me when I first saw it back in 1976 The promos were very well
	Pitch Black was a complete shock to me when I first saw it back in the days when I was a
MaskMLE	Black was a complete shock to me when I first saw it back in 1969 I live in New Zealand
	Pitch Black was a complete shock to me when I first saw it back in 1951 It was funny All Interiors
	Pitch Black was a complete shock to me when I first saw it back in the day and I was in

https://arxiv.org/pdf/1801.07736.pdf

GAN readings

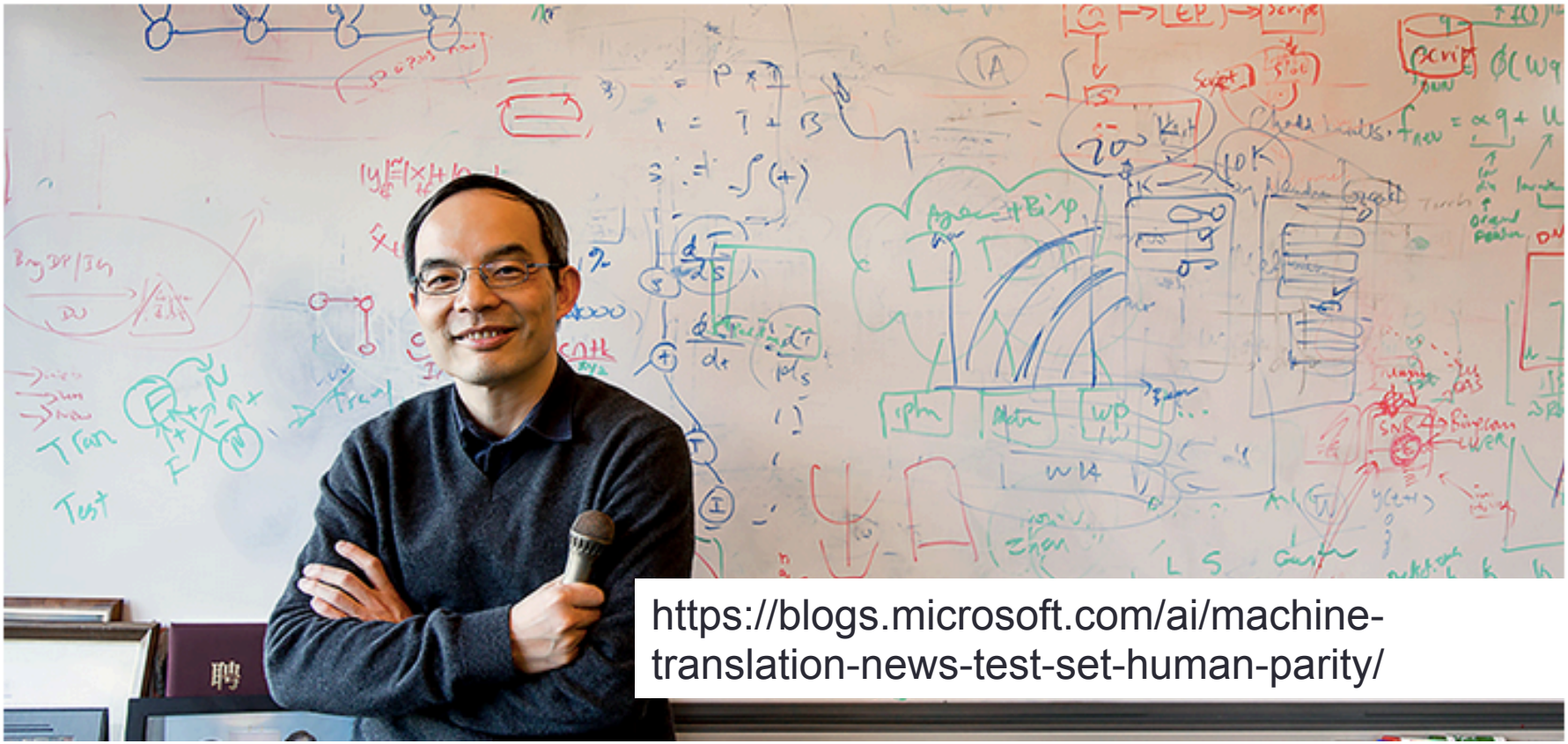
- 1) GAN tutorial: https://arxiv.org/pdf/1701.00160.pdf
- 2) WGAN: https://arxiv.org/abs/1701.07875
  - 2.1) Blog explanation: https://www.alexirpan.com/2017/02/22/wasserstein-gan.html Read 2) and 2.1) together section by section.
- 3) WGAN followup: https://arxiv.org/abs/1704.00028
- 4) On how GANs are hard to train stil: https://arxiv.org/abs/1711.10337

Machine Translation (again)

Microsoft reaches a historic milestone, using AI to match human performance in translating news from Chinese to English

Mar 14, 2018 | Allison Linn

Based on transformer
Dual learning – translate back and forth
Advance decoding (beyond beamsearch)
Lots of system combinations…



https://blogs.microsoft.com/ai/machine-translation-news-test-set-human-parity/

Machine translation with no parallel text

- MT usually requires parallel text

  This is a cat

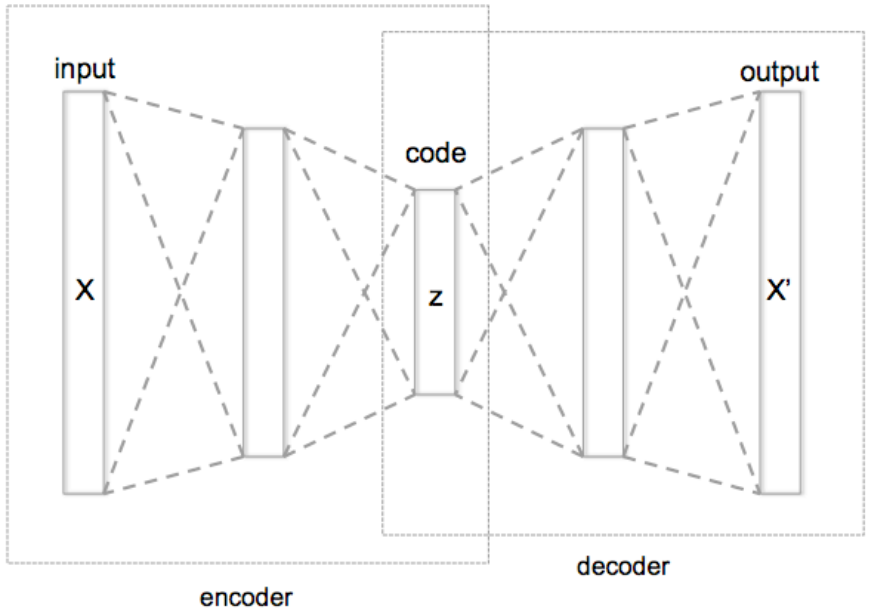
  นี่คือแมว

- Most of the time we don’t have parallel text. Just text.
- Can we still do MT?
  - Use GANs + Autoencoders

https://arxiv.org/abs/1711.00043

Aside: autoencoders

- Use neural network properties to compress data
  - Automatically learn features
  - Loss function = $||x-x'||^2$
  -



Aside: Denoising autoencoder

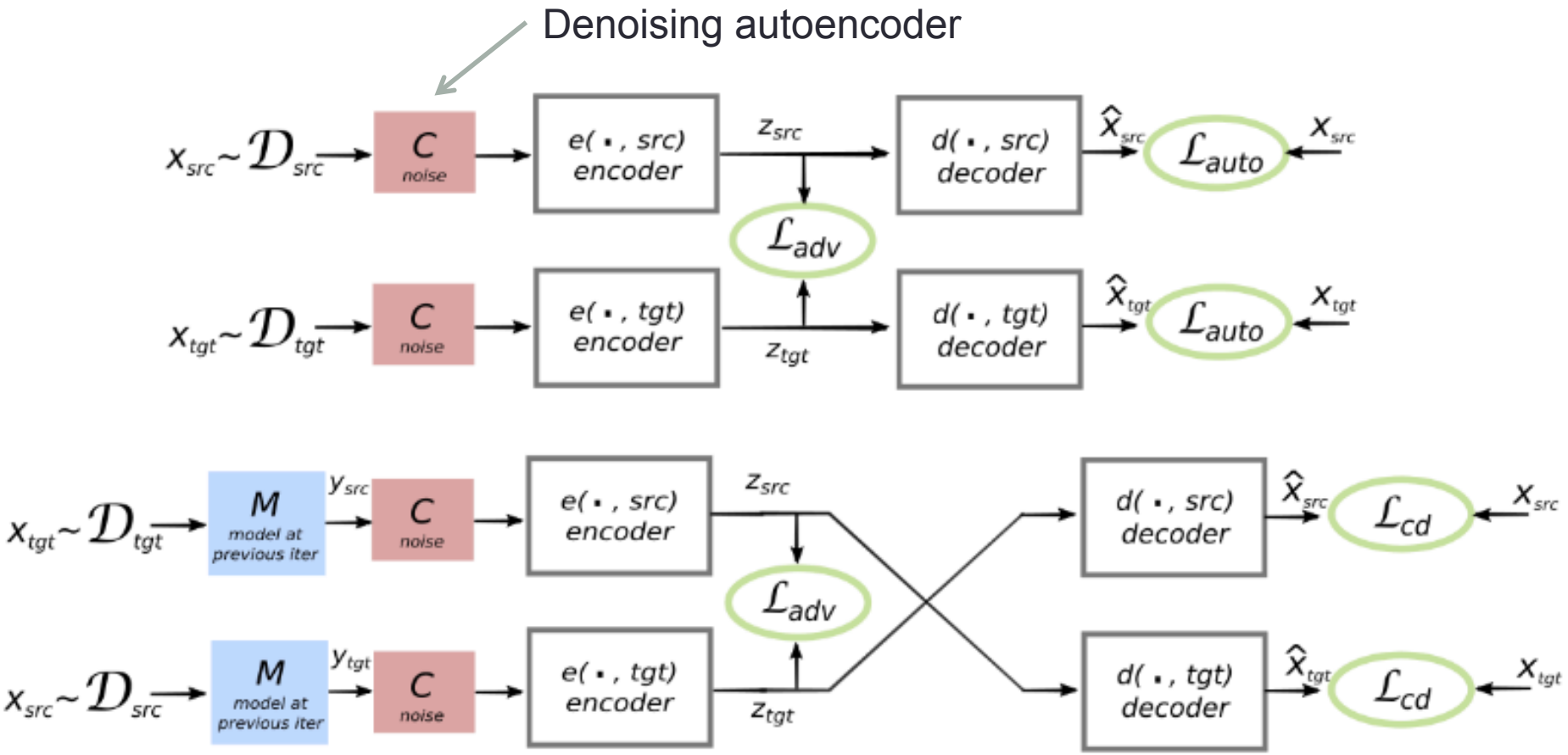
- An autoencoder that denoise corrupted inputs
  - Blur image, sound corrupted by noise
- How?
  - Corrupt your input x -> x''
  - Use x'' as the input, and minimize the same loss
  - Loss = $||x-x'||^2$
- Noising text?
  - Word order swap
  - Randomly dropped words

Aside: denoising example

Original Corrupted Generated

https://www.doc.ic.ac.uk/~js4416/163/website/autoencoders/denoising.html

Use GAN Loss (L_{adv}) to enforce that source and target language pairs share the same distributions.
Then enforce the translation distribution matches.
A form of dual learning



https://arxiv.org/abs/1711.00043

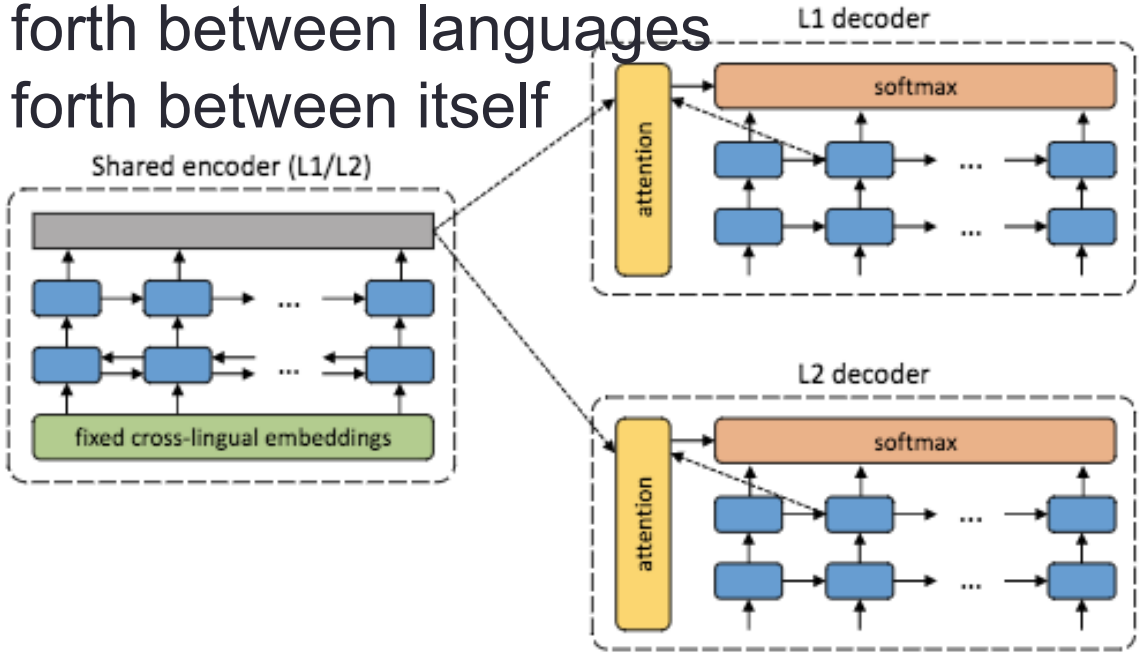
Results

	Multi30k-Task1				WMT			
	en-fr	fr-en	de-en	en-de	en-fr	fr-en	de-en	en-de
Supervised	56.83	50.77	38.38	35.16	27.97	26.13	25.61	21.33
word-by-word	8.54	16.77	15.72	5.39	6.28	10.09	10.77	7.06
word reordering	-	-	-	-	6.68	11.69	10.84	6.70
oracle word reordering	11.62	24.88	18.27	6.79	10.12	20.64	19.42	11.57
Our model: 1st iteration	27.48	28.07	23.69	19.32	12.10	11.79	11.10	8.86
Our model: 2nd iteration	31.72	30.49	24.73	21.16	14.42	13.49	13.25	9.75
Our model: 3rd iteration	32.76	32.07	26.26	22.74	15.05	14.31	13.33	9.64

https://arxiv.org/abs/1711.00043

Another approach

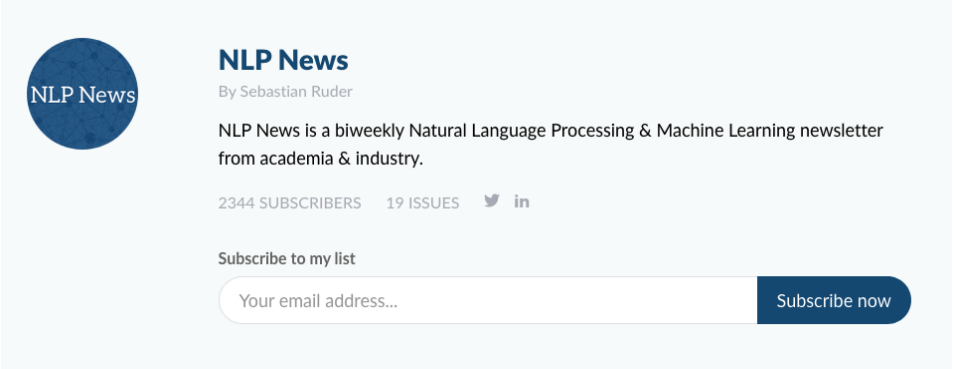
- Train a shared embedding space (utilizing a bilingual dictionary)
- Encoder is shared since embedding is shared
- To train the decoder, use dual learning
  Translate back and forth between languages
  Translate back and forth between itself
  (with noise)



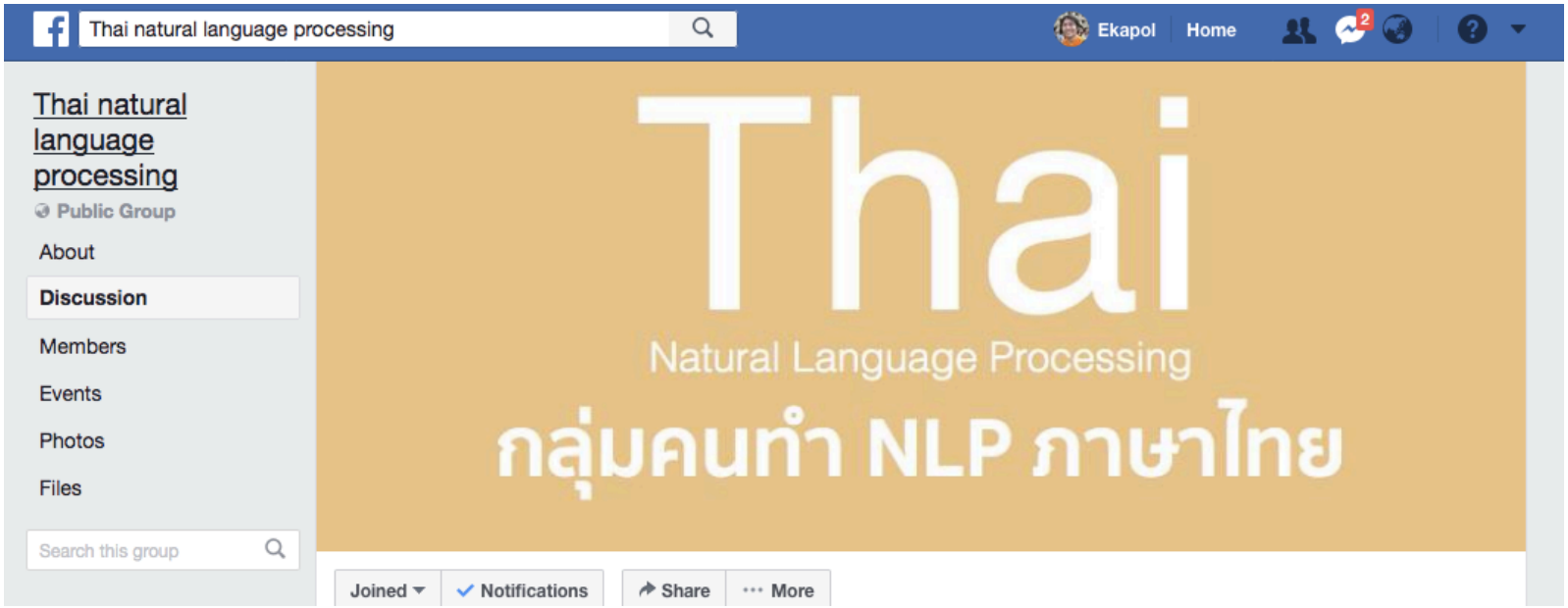
https://arxiv.org/pdf/1710.11041.pdf

Where to learn more

- http://newsletter.ruder.io/



- Thai NLP group



HOW TO READ A SCIENTIFIC ARTICLE

2 Paper types

- Review article/tutorial
  - Give insights about the field
  - Useful for learning about a new field
  - Read multiple to avoid the author's bias
  - Title usually has "review" or "tutorial"
- Primary research article
  - More details on the experiments and results

Parts of an article

- Abstract
- Introduction
- Methods
- Results and discussion
- Conclusion
- Reference

Things to look for before reading an article

- Publication date
- Author names
  - Previous and newer publications
- Keywords
- Acknowledgements and funding sources

Getting the big picture

- Read the abstract
- Read the introduction
  - What is the research question?
  - What is the method?
  - What had been done? How is it different from other work?
- Look at figures and results

Tip: keep track of terms you don’t understand

First reading

- Reread the introduction
- Skim methods
- Read results and discussion
  - Does the figures make sense now?
- Write on the article!

Understanding the article

- Reread the article (until you get what you want)
- Check references for parts you don’t understand
- Reread the abstract
  - Does your understanding match the abstract?
- Note down important points. This might come in handy when you write you paper/thesis!

Evaluating the article

- Does the method make sense?
  - What are the limitations that the authors mention?
  - Are there other limitations?
  - Can it be used in other situations?
- Are the experiments legitimate?
  - The sample size is big enough?
  - What kind of dataset is used? How big?
  - The evaluation criterion is sound?
- Have these results been reproduced?
  - Look for articles that cite this paper

ML paper checklist

- What is being done?
- How is it being done?
  - How is it different from previous work
- What is the dataset?
  - Nature of dataset
  - How many training/testing samples? How many classes/vocab size?
- Evaluation metric
  - What are the baselines?
- Practicality
  - Prone to parameter tuning?
  - Computing resource
  - Runtime (training and testing)

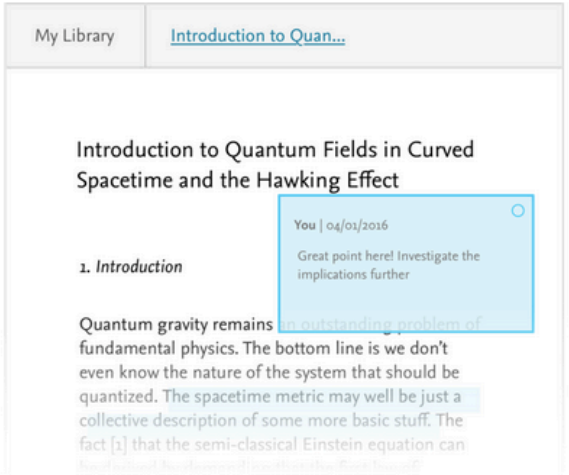
Useful tools

- https://scholar.google.com
  - For finding other articles by the same authors or paper that cites the article

- https://www.mendeley.com/
  - Reference manager

Annotate as you read

Easily add your thoughts on documents in your own library, even from mobile devices. For ease of collaboration, you can also share documents with groups of colleagues and annotate them together.



Project (35%)

- Presentation day (30th April) 15 mins presentation + 5 mins demo + 5 mins QA
  - Integration 10 (GUI, chat services integration)
  - Completeness 10 (No crashes, handling error cases)
  - Techniques 10 (Must include at least 2 type of models for comparison)
  - Evaluation 10 (Come up with a reasonable metric based on your task)
  - Presentation 10
- Presentation paper (24th April) 10 (follow the checklist)
- Report submission day (7th May)
  - Report 10
  - VDO 5
  - Code, model, data 5