RANDOM TOPICS

Transformer, GAN for NLP, paper reading

Attention is all you need

Abstract

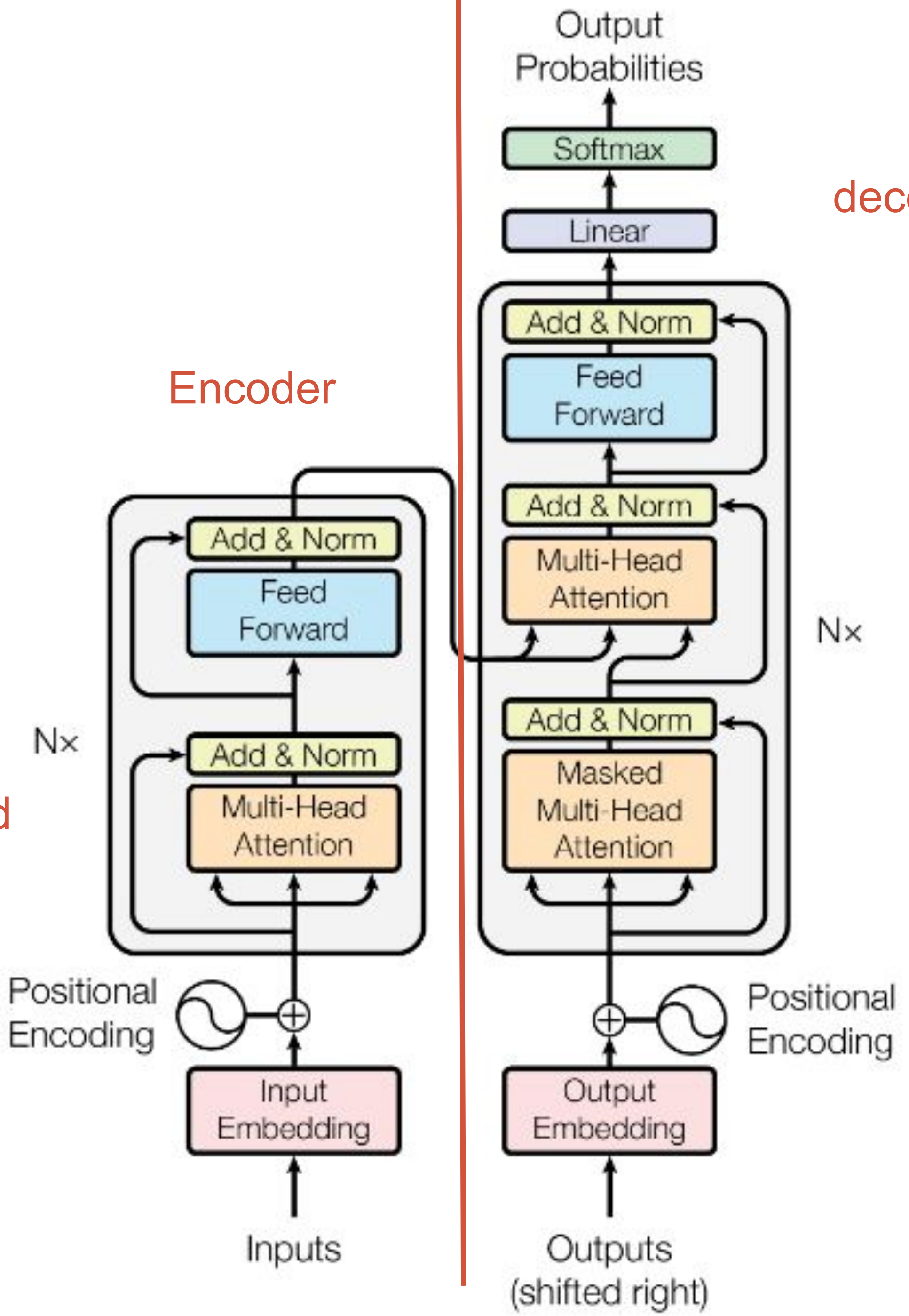
The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, 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

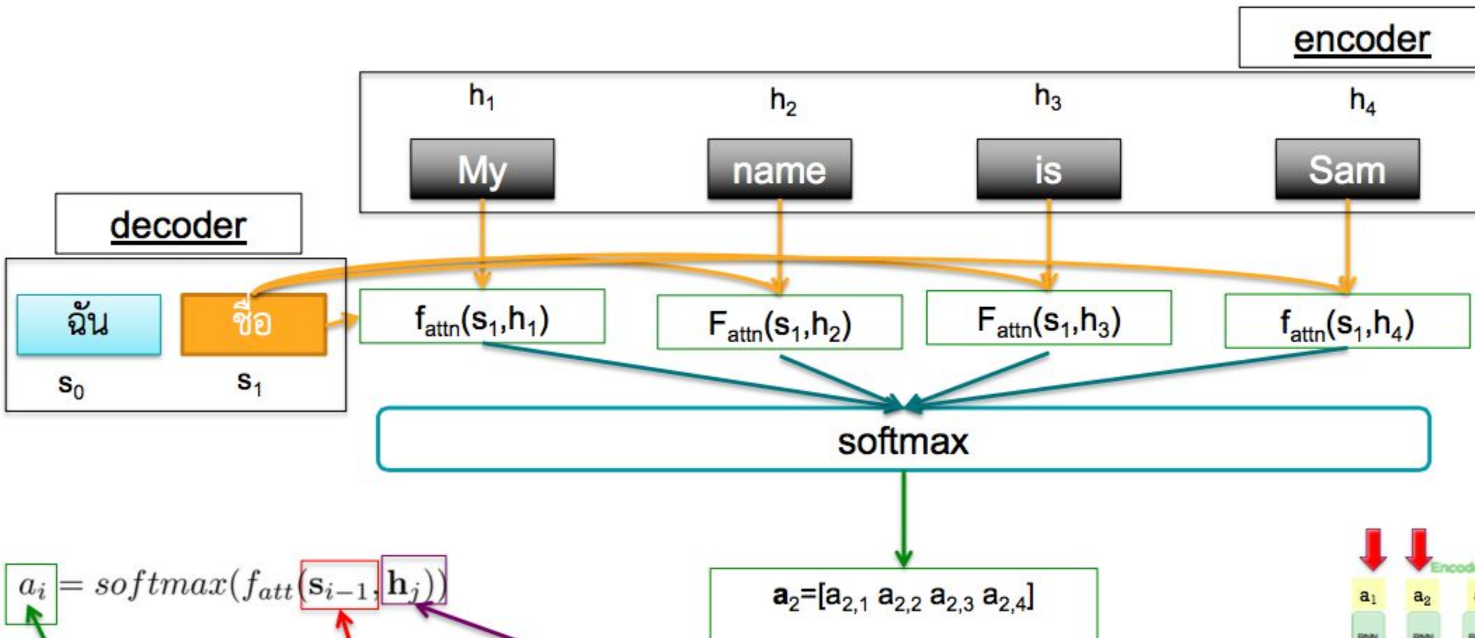
To eliminate GRU which remembers time, encode position instead

Encoder

decoder



Attention Calculation Example (1): Attention Scores

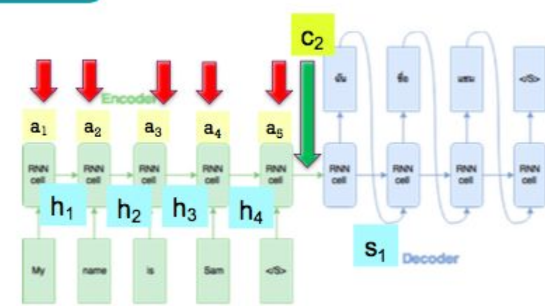


$$\boxed{a_i} = softmax(f_{att}(\boxed{\mathbf{s}_{i-1}}, \boxed{\mathbf{h}_j}))$$

attention scores

current hidden state/ decoder state

previous hidden state/ encoder state



Assume now try to translate “ชื่อ”
to give high weight at “name”

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

	My	name	is	Sam	enco
ฉัน					
ชื่อ					
แซม					
decoder					

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:

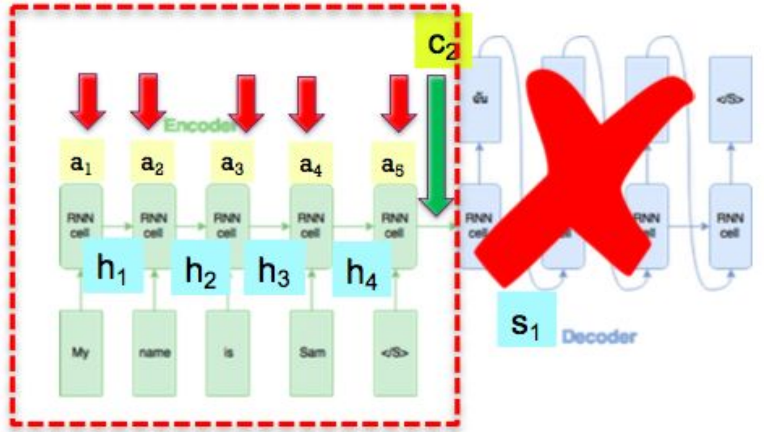
$$f_{attn}(\mathbf{s}_{i-1}, \mathbf{h}_j) = \mathbf{s}_{i-1}^{\top} \mathbf{W}_a \mathbf{h}_j$$

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)

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

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

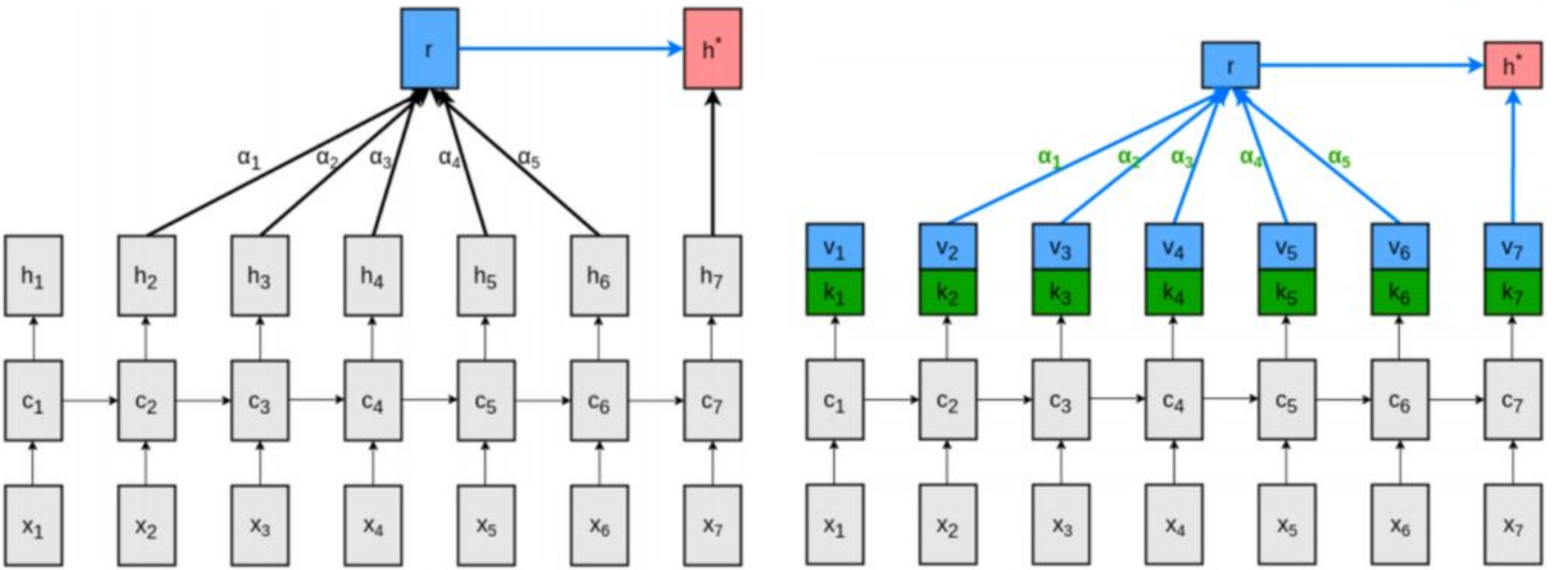
Similar to CNN in flow

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

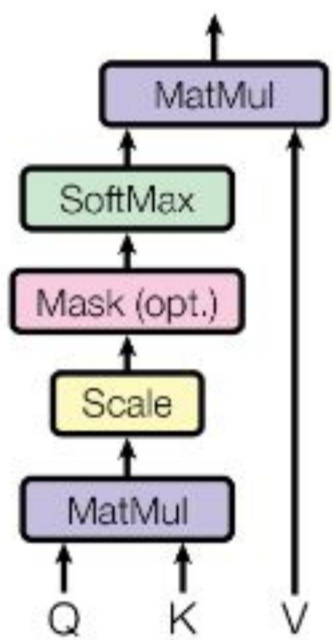


(a) Neural language model with attention.

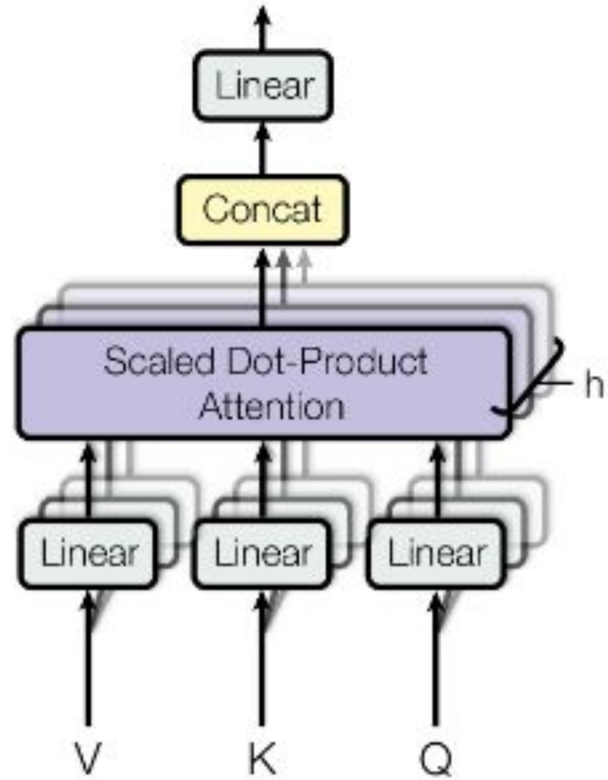
(b) Key-value separation.

Multi-head attention

Scaled Dot-Product Attention



Multi-Head Attention



What's this????

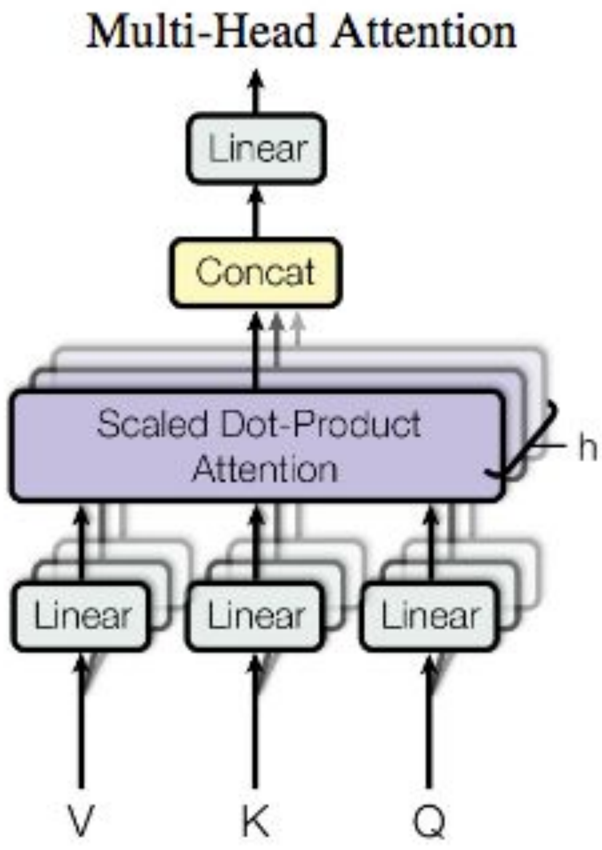
Query – used with Key to determine the position
Value – used as the information after determining the position

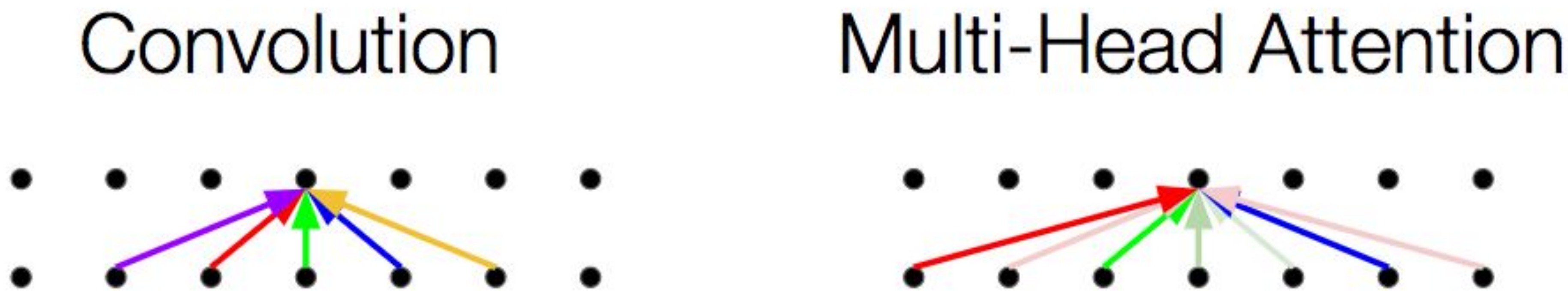
Attention drawback

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

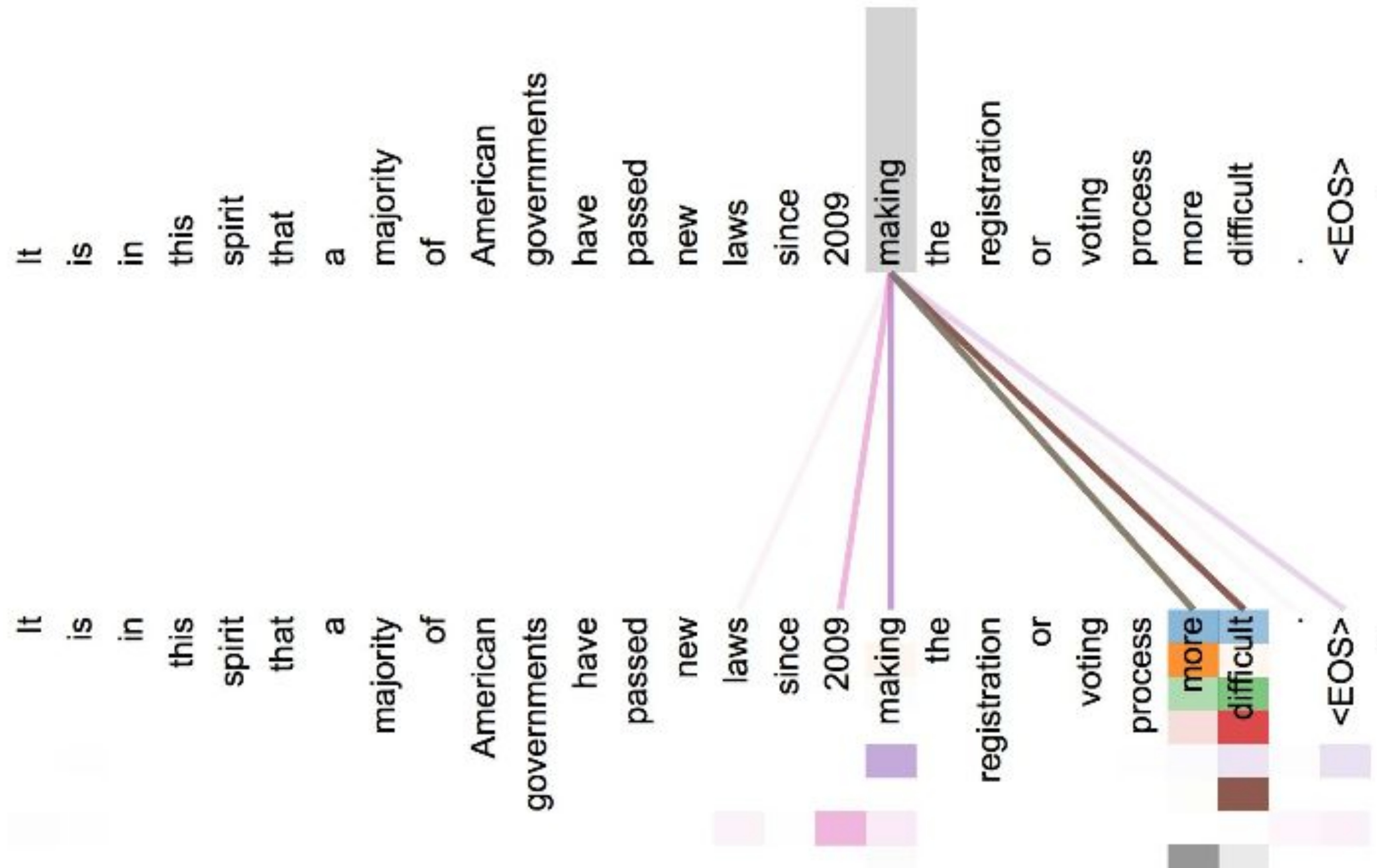
Multi-head attention

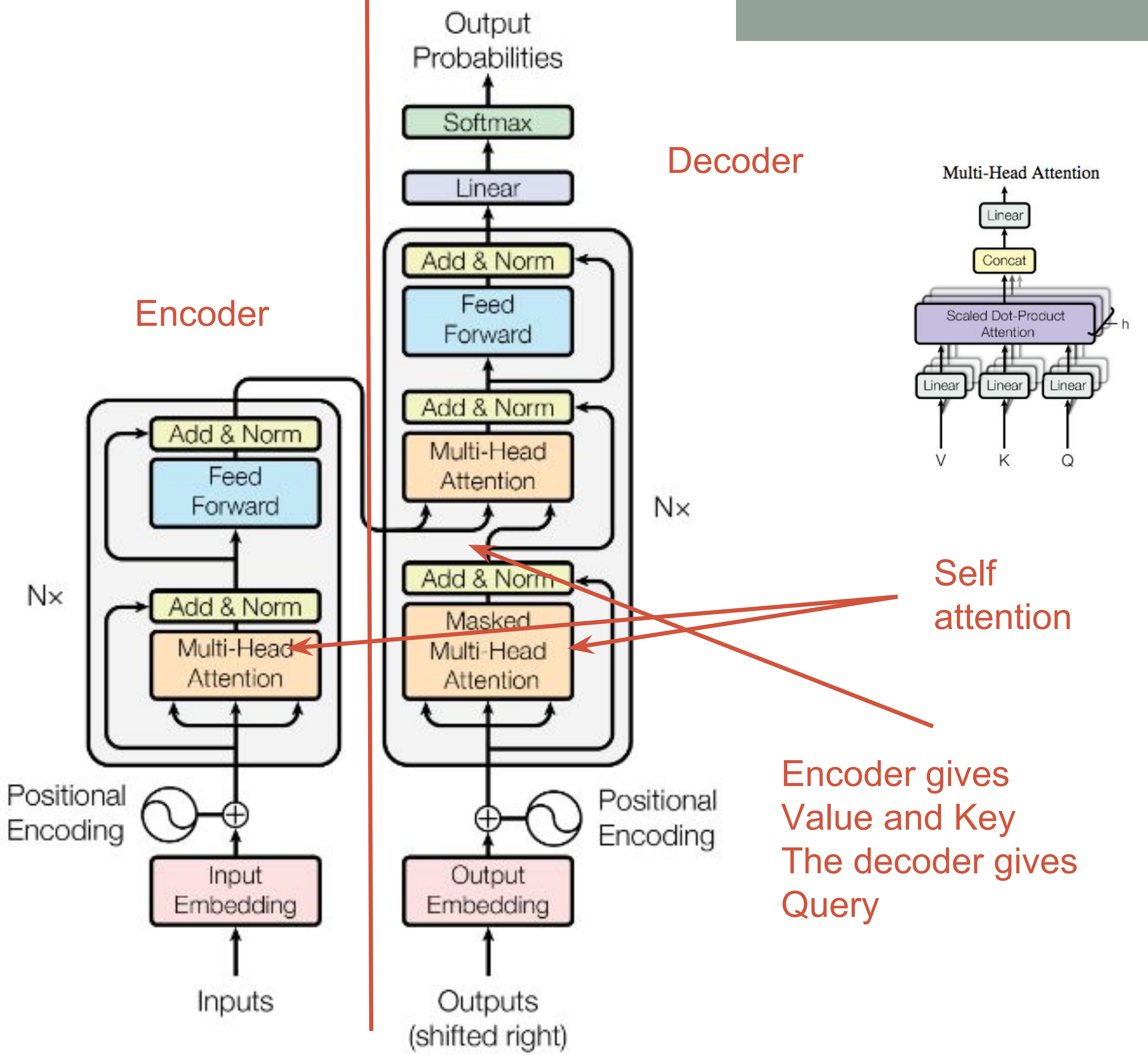
- Multiple attention layers (heads) that run in parallel
- Each head use different weights
- Each head can learn different relationship

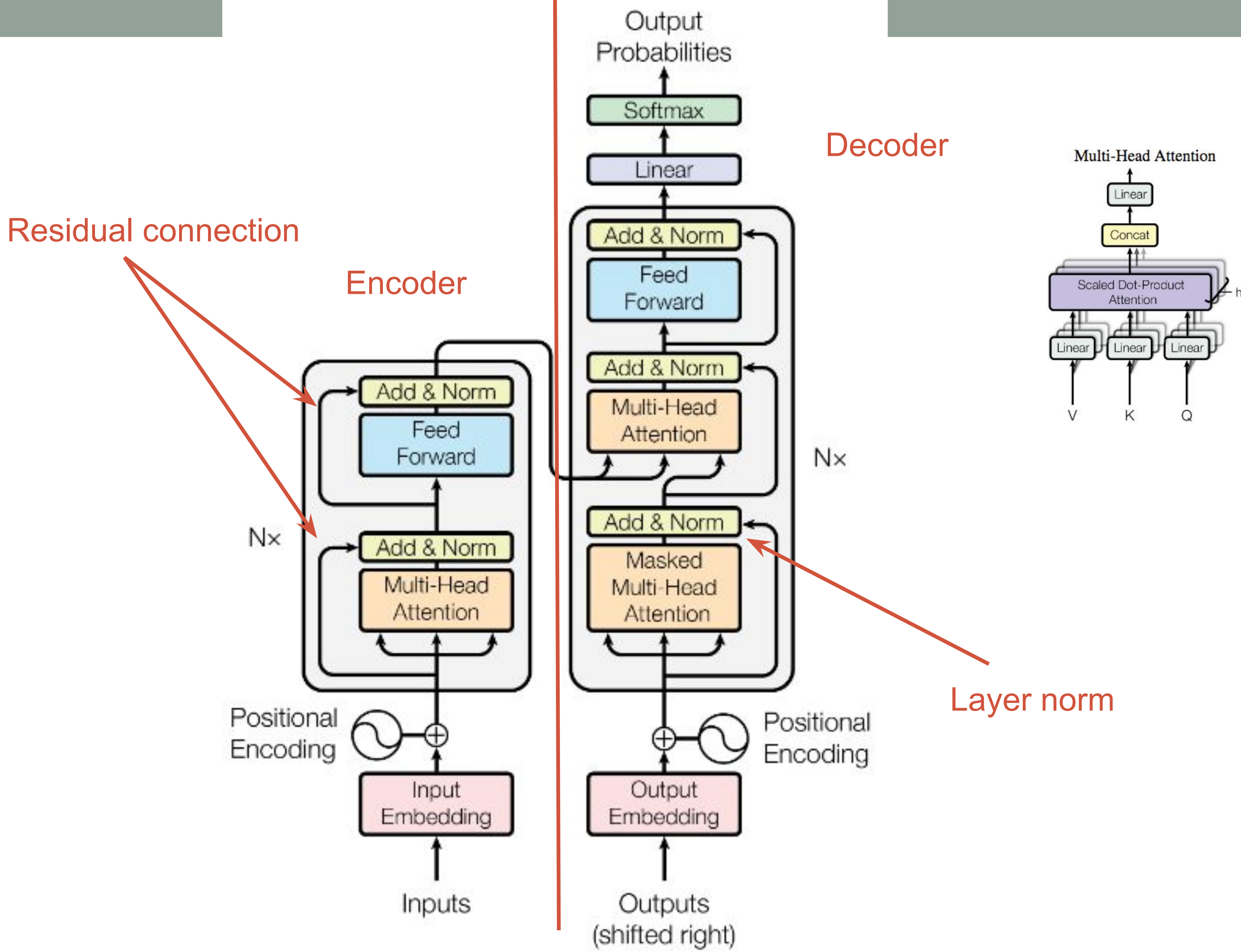




Multi-head visualization



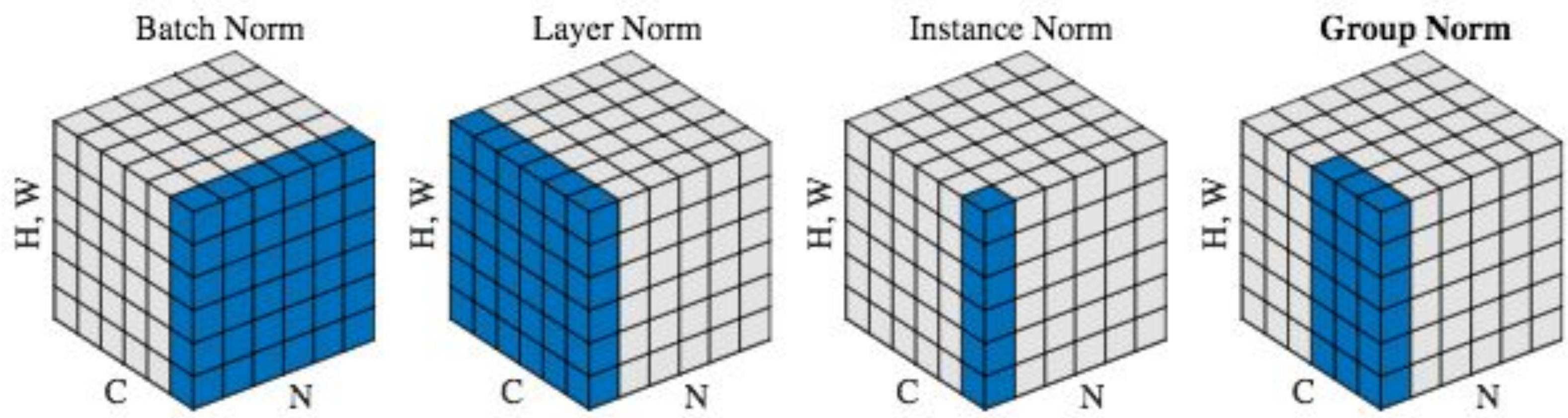

Output Probabilities
Softmax
Linear
Decoder
Encoder
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Add & Norm
Masked Multi-Head Attention
Nx
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Nx
Positional Encoding
Input Embedding
Inputs
Positional Encoding
Output Embedding
Outputs (shifted right)
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
Layer norm
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)
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

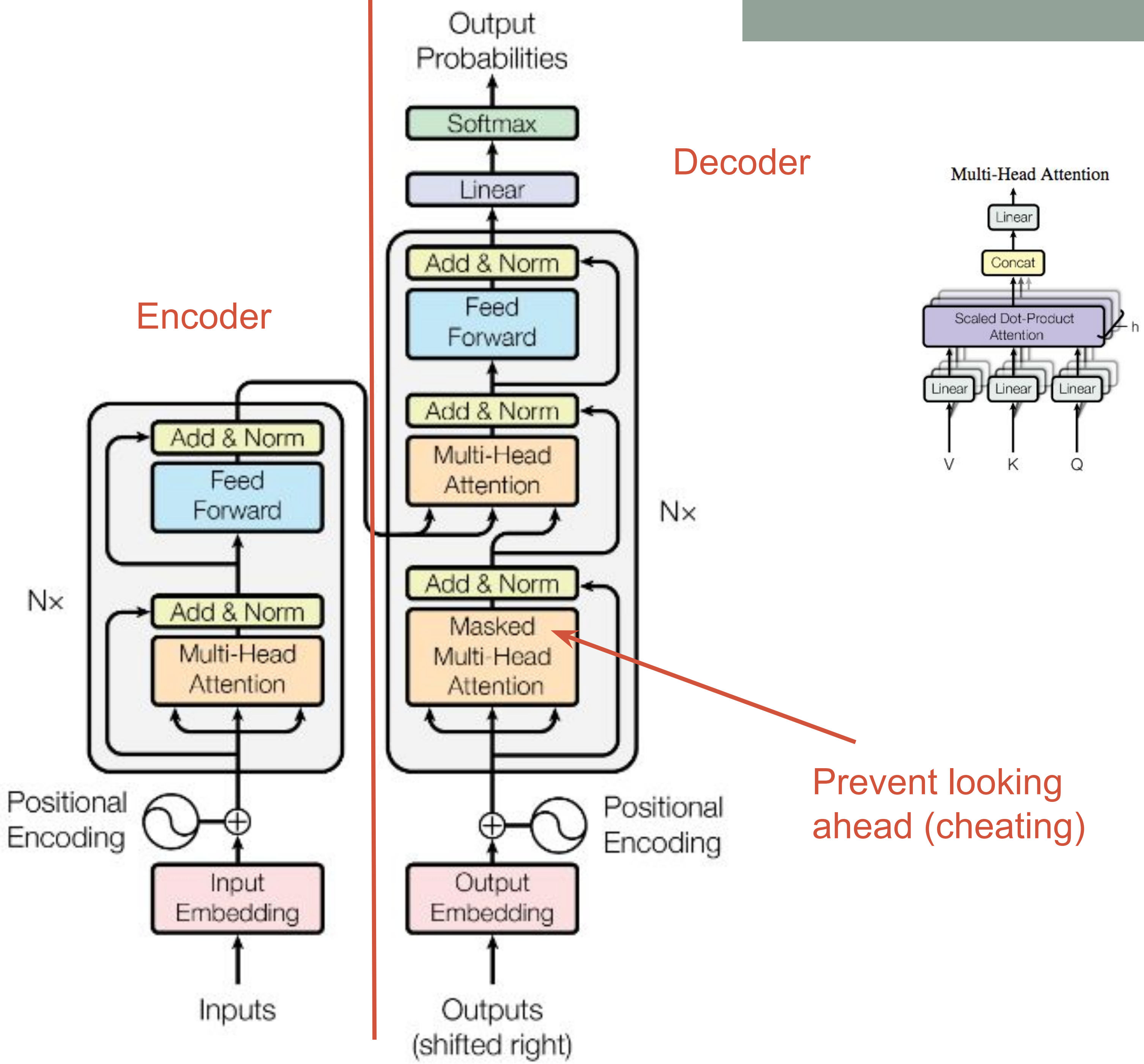


N – example in mini batch
C – Channel output
H,W – spatial coordinates (x,y)

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


Encoder
Decoder
Output Probabilities
Softmax
Linear
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Add & Norm
Masked Multi-Head Attention
Nx
Positional Encoding
Output Embedding
Outputs (shifted right)
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Nx
Positional Encoding
Input Embedding
Inputs
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	$\mathbf{3.3 \cdot 10^{18}}$	
Transformer (big)	**28.4**	**41.8**	$2.3 \cdot 10^{19}$	

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

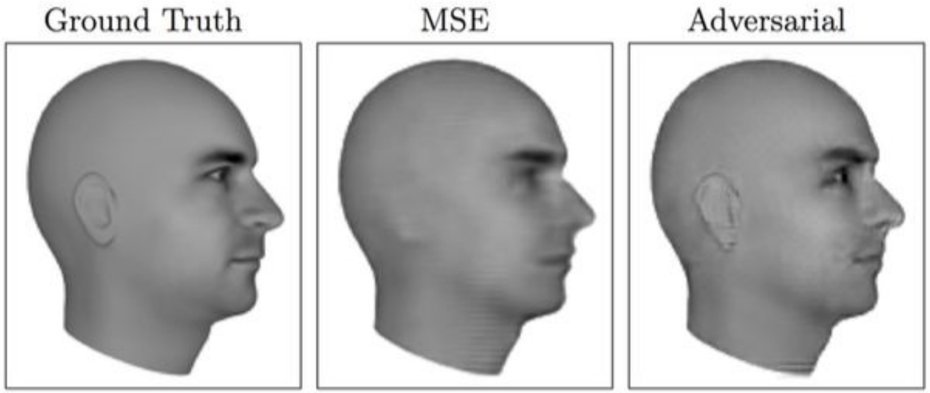
Generative Adversarial Networks (GANs)

Learning distributions

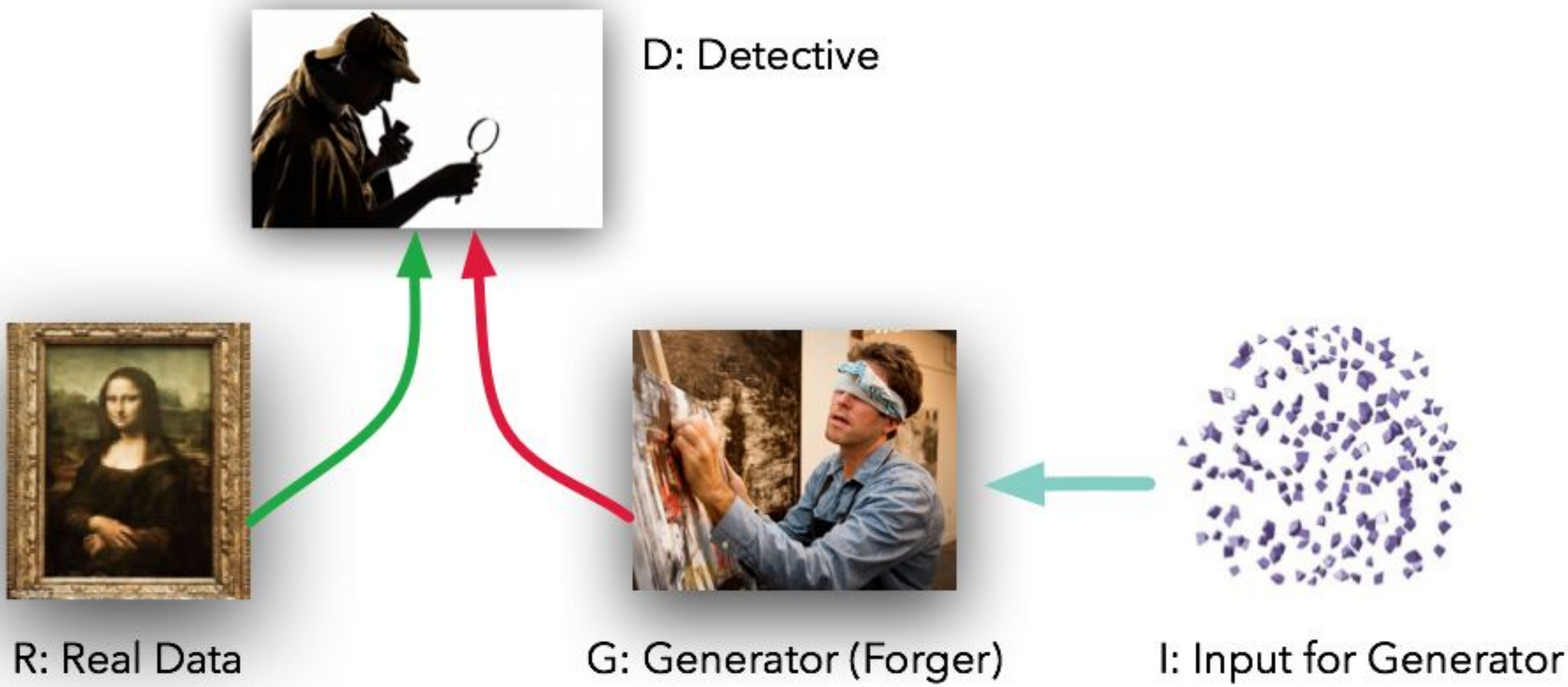
- Supervised learning tasks usually have one correct answer

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?
  - What word is missing from the blank?
    - I eat ____



https://arxiv.org/abs/1701.00160



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

Generative Adversarial Networks (GAN)

0.1, -0.3, .. → Generator →  → Discriminator → Real or Fake

z   x   Y=f(x)

- 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

Generative Adversarial Networks (GAN)

0.1, -0.3, .. → Generator →  → Discriminator → Real or Fake

z   x   Y=f(x)

- Generator (Money Faker):
  - Maximize Y

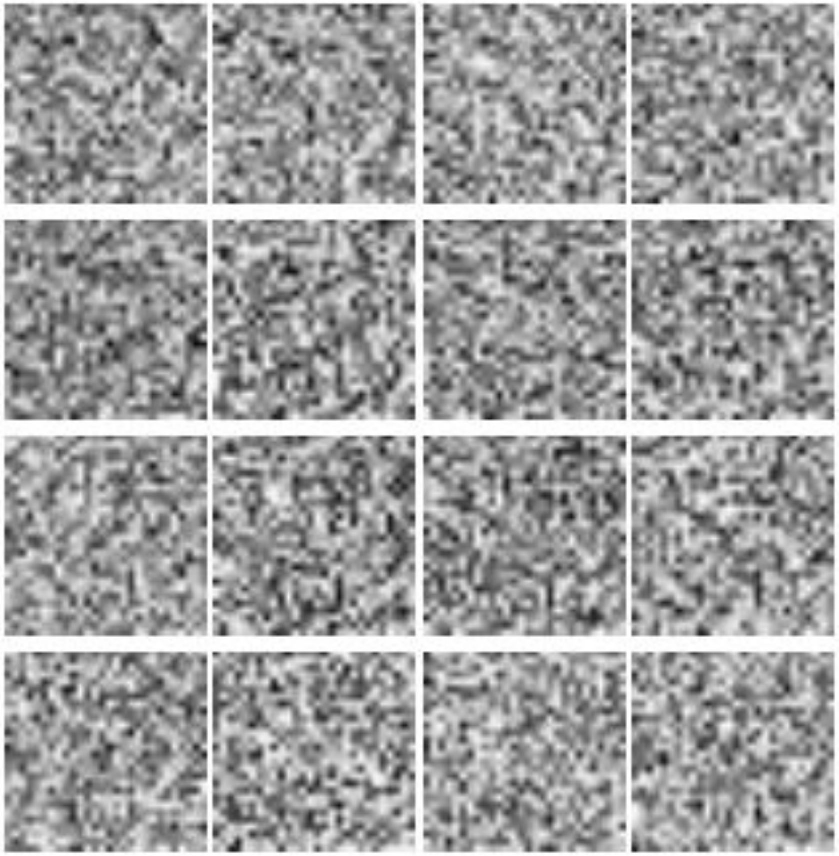
$$\boxed{\min_G \mathbb{E}_{\mathbf{z}}[\log(1 - D(G(\mathbf{z}))]}$$

- Discriminator (Police):
  - For real images => Maximize Y
  - For generated images from the faker => Minimize Y

$$\boxed{\max_D \mathbb{E}_{\mathbf{z}}[\log(1 - D(G(\mathbf{z}))] + \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}}[\log(D(\mathbf{x}))]}$$

GAN example

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



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

GANs Loss Formulations

$$\max_D \mathbb{E}_{\mathbf{z}}[\log(1 - D(G(\mathbf{z}))] + \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}}[\log(D(\mathbf{x}))]$$

$$\min_G \mathbb{E}_{\mathbf{z}}[\log(1 - D(G(\mathbf{z}))]$$

Discriminator

Generator

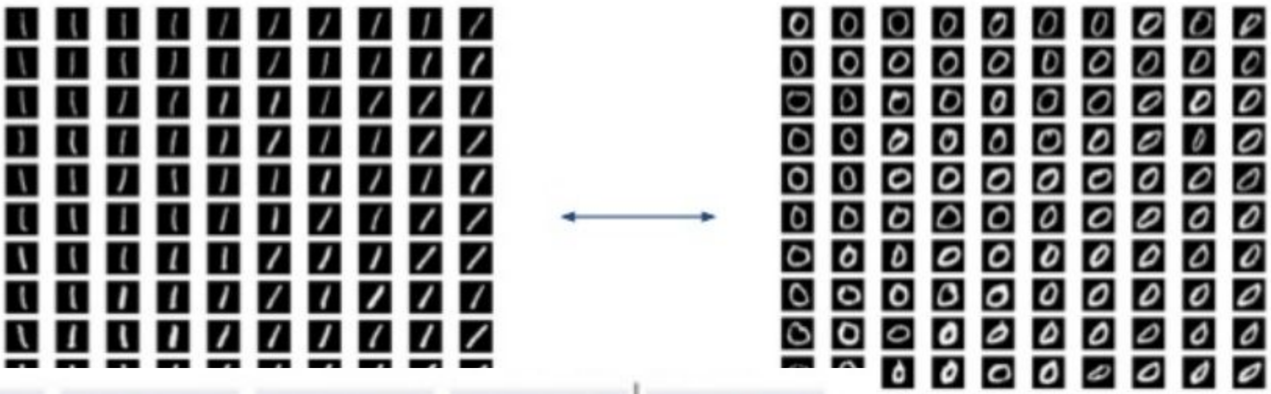
GAN	Discriminator Loss	Generator Loss
MM GAN	$\mathcal{L}_{\text{D}}^{\text{GAN}} = -\mathbb{E}_{x \sim p_d}[\log(D(x))] - \mathbb{E}_{\hat{x} \sim p_g}[\log(1 - D(\hat{x}))]$	$\mathcal{L}_{\text{G}}^{\text{GAN}} = \mathbb{E}_{\hat{x} \sim p_g}[\log(1 - D(\hat{x}))]$
NS GAN	$\mathcal{L}_{\text{D}}^{\text{NSGAN}} = -\mathbb{E}_{x \sim p_d}[\log(D(x))] - \mathbb{E}_{\hat{x} \sim p_g}[\log(1 - D(\hat{x}))]$	$\mathcal{L}_{\text{G}}^{\text{NSGAN}} = -\mathbb{E}_{\hat{x} \sim p_g}[\log(D(\hat{x}))]$
WGAN	$\mathcal{L}_{\text{D}}^{\text{WGAN}} = -\mathbb{E}_{x \sim p_d}[D(x)] + \mathbb{E}_{\hat{x} \sim p_g}[D(\hat{x})]$	$\mathcal{L}_{\text{G}}^{\text{WGAN}} = -\mathbb{E}_{\hat{x} \sim p_g}[D(\hat{x})]$
WGAN GP	$\mathcal{L}_{\text{D}}^{\text{WGANGP}} = \mathcal{L}_{\text{D}}^{\text{WGAN}} + \lambda \mathbb{E}_{\hat{x} \sim p_g}[(\|\|\nabla D(\alpha x + (1 - \alpha \hat{x})\|\|_2 - 1)^2]$	$\mathcal{L}_{\text{G}}^{\text{WGANGP}} = -\mathbb{E}_{\hat{x} \sim p_g}[D(\hat{x})]$
LS GAN	$\mathcal{L}_{\text{D}}^{\text{LSGAN}} = -\mathbb{E}_{x \sim p_d}[(D(x) - 1)^2] + \mathbb{E}_{\hat{x} \sim p_g}[D(\hat{x})^2]$	$\mathcal{L}_{\text{G}}^{\text{LSGAN}} = -\mathbb{E}_{\hat{x} \sim p_g}[(D(\hat{x} - 1))^2]$
DRAGAN	$\mathcal{L}_{\text{D}}^{\text{DRAGAN}} = \mathcal{L}_{\text{D}}^{\text{GAN}} + \lambda \mathbb{E}_{\hat{x} \sim p_d + \mathcal{N}(0,c)}[(\|\|\nabla D(\hat{x})\|\|_2 - 1)^2]$	$\mathcal{L}_{\text{G}}^{\text{DRAGAN}} = \mathbb{E}_{\hat{x} \sim p_g}[\log(1 - D(\hat{x}))]$
BEGAN	$\mathcal{L}_{\text{D}}^{\text{BEGAN}} = \mathbb{E}_{x \sim p_d}[\|\|x - \text{AE}(x)\|\|_1] - k_t \mathbb{E}_{\hat{x} \sim p_g}[\|\|\hat{x} - \text{AE}(\hat{x})\|\|_1]$	$\mathcal{L}_{\text{G}}^{\text{BEGAN}} = \mathbb{E}_{\hat{x} \sim p_g}[\|\|\hat{x} - \text{AE}(\hat{x})\|\|_1]$

Another problem: Mode collapsing

Are GANs Created Equal? A Large-Scale Study [Lucic et al. 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

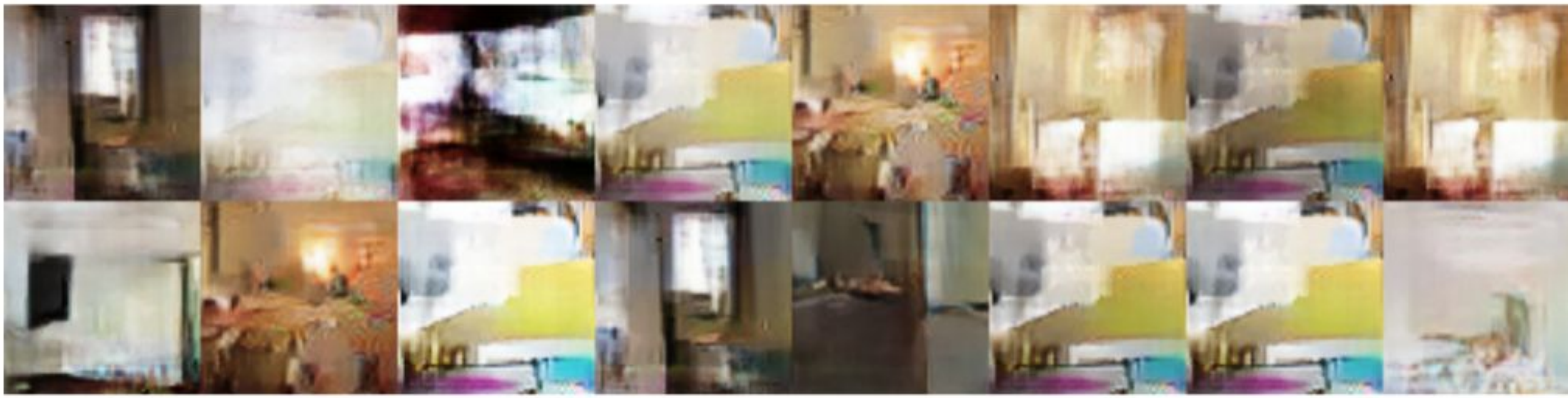


mode collapsing

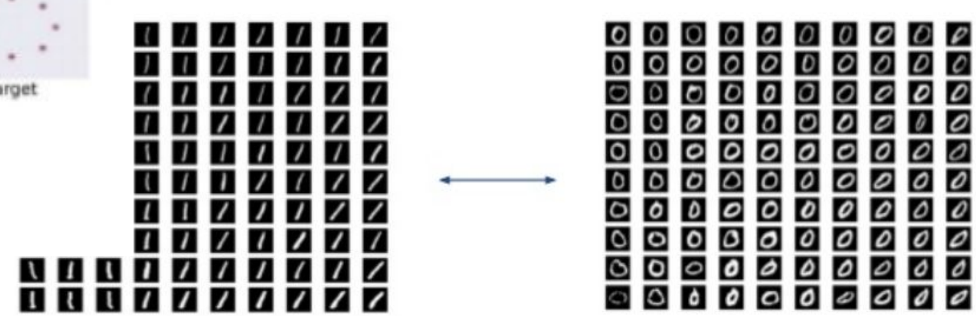
https://www.slideshare.net/ssuser77ee21/generative-adversarial-networks-70896091

Mode collapse

Model only learn a couple types (modes) of inputs



mode collapsing

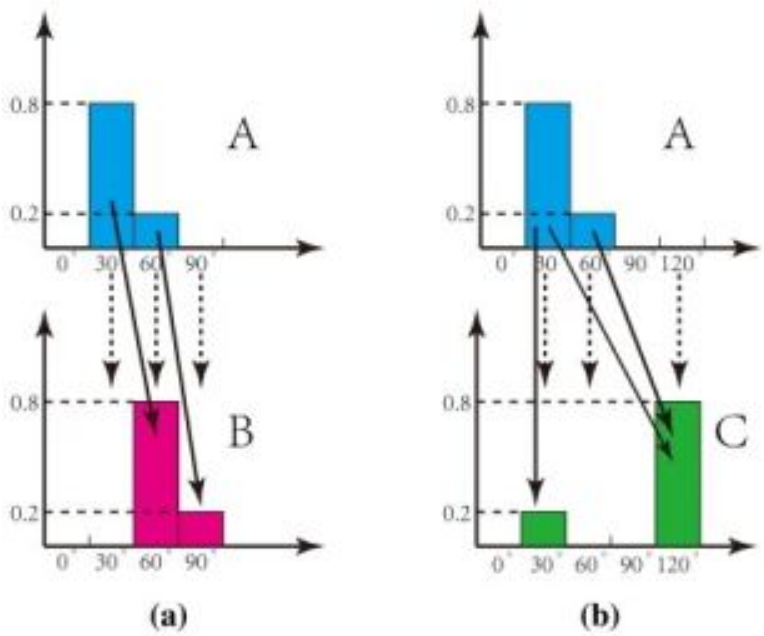


https://arxiv.org/pdf/1701.07875.pdf

Wasserstein GAN (WGAN)

Wasserstein distance? (Earth mover distance)

Energy required to move mass to make two distributions look the same



WGAN

Discriminator to a critic (no fake/real sigmoid) but output a score

WD has better gradient and convergence

	Discriminator/Critic	Generator
GAN	$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right)\right]$	$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right)$
WGAN	$\nabla_w \frac{1}{m} \sum_{i=1}^{m} \left[f\left(x^{(i)}\right) - f\left(G\left(z^{(i)}\right)\right)\right]$	$\nabla_\theta \frac{1}{m} \sum_{i=1}^{m} f\left(G\left(z^{(i)}\right)\right)$

WGAN

WGAN requires the critic model to be a k-Lipschitz function

k-Lipschitz?

- Bounded in slope of k
- $|f(a) - f(b)| < k| a - b |$

Example

- $f(x) = 5x$ is 5-Lipschitz

WGAN to WGAN-GP

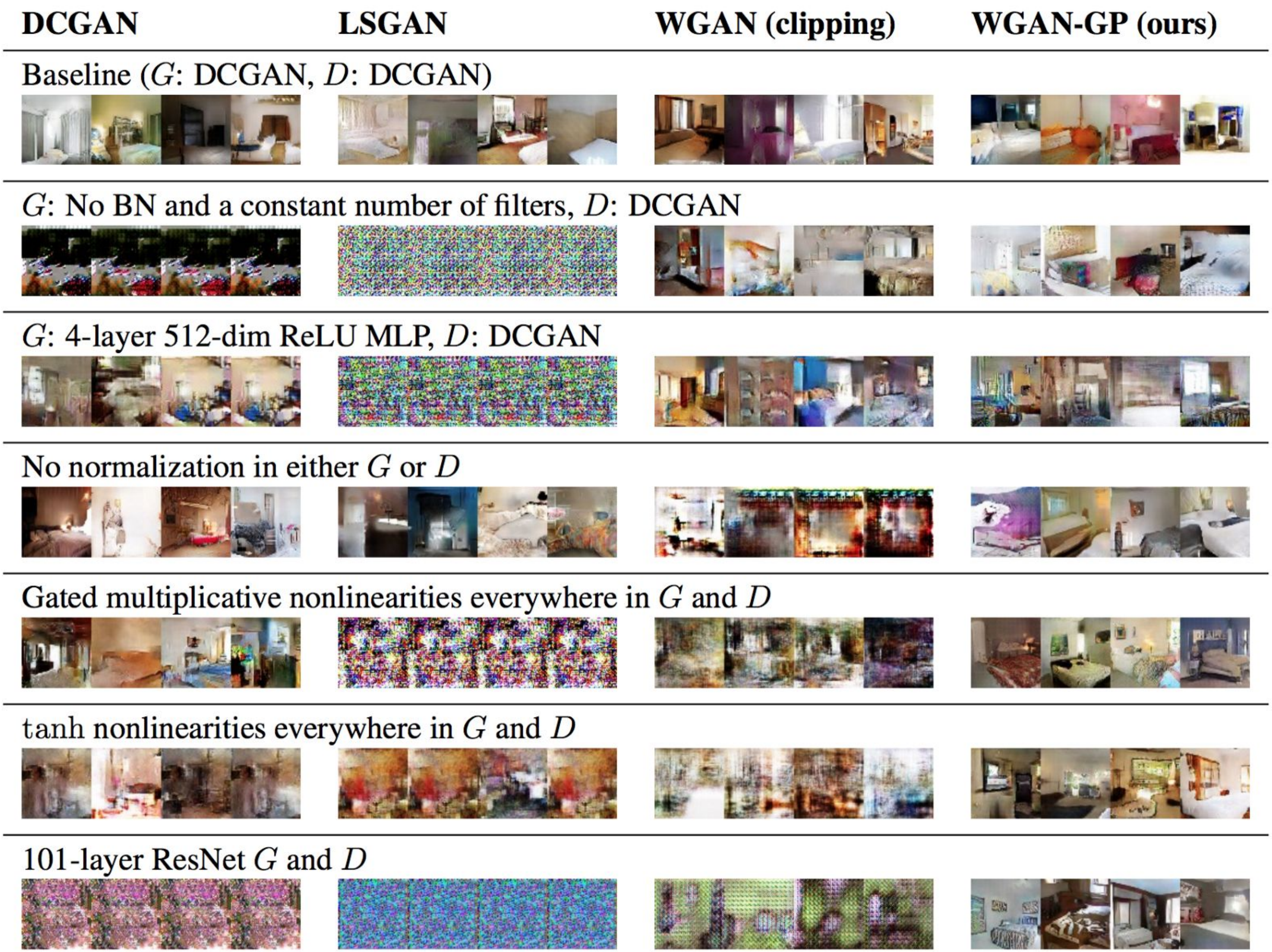
To make k-Lipschitz

WGAN caps the weights of all layers to 1

WGAN-GP improves and add Gradient Penalty to reduce the weights instead

A differentiable function f is 1-Lipschtiz if and only if it has gradients with norm at most 1 everywhere.

$$L = \underbrace{\underset{\tilde{\boldsymbol{x}} \sim \mathbb{P}_g}{\mathbb{E}} \left[D(\tilde{\boldsymbol{x}}) \right] - \underset{\boldsymbol{x} \sim \mathbb{P}_r}{\mathbb{E}} \left[D(\boldsymbol{x}) \right]}_{\text{Original critic loss}} + \underbrace{\lambda \underset{\hat{\boldsymbol{x}} \sim \mathbb{P}_{\hat{\boldsymbol{x}}}}{\mathbb{E}} \left[\left(\| \nabla_{\hat{\boldsymbol{x}}} D(\hat{\boldsymbol{x}}) \|_2 - 1 \right)^2 \right]}_{\text{Our gradient penalty}} .$$


DCGAN
LSGAN
WGAN (clipping)
WGAN-GP (ours)
Baseline (G: DCGAN, D: DCGAN)
G: No BN and a constant number of filters, D: DCGAN
G: 4-layer 512-dim ReLU MLP, D: DCGAN
No normalization in either G or D
Gated multiplicative nonlinearities everywhere in G and D
tanh nonlinearities everywhere in G and D
101-layer ResNet G and D

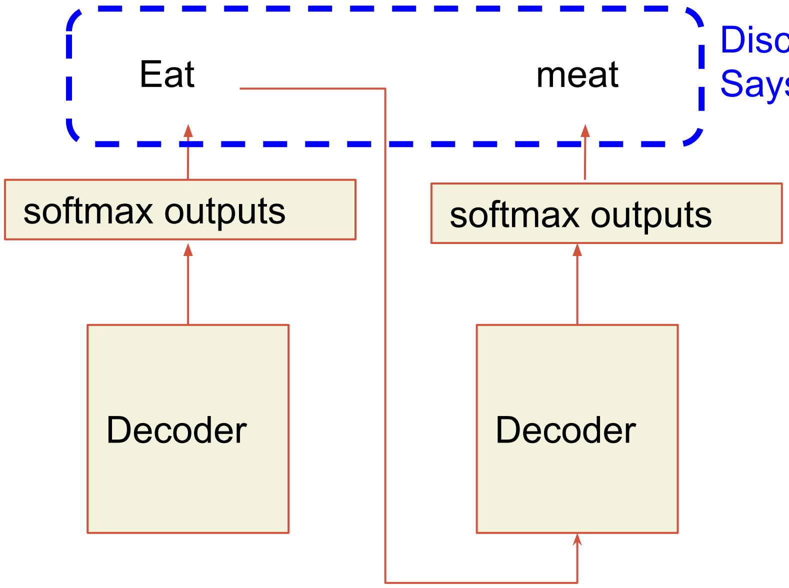
WGAN makes things easier

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

GAN for text generation

Autoregressive decoding includes a sampling process

Cannot gradient descent



Discriminator sees the sampled sentence.
Says it's good or bad

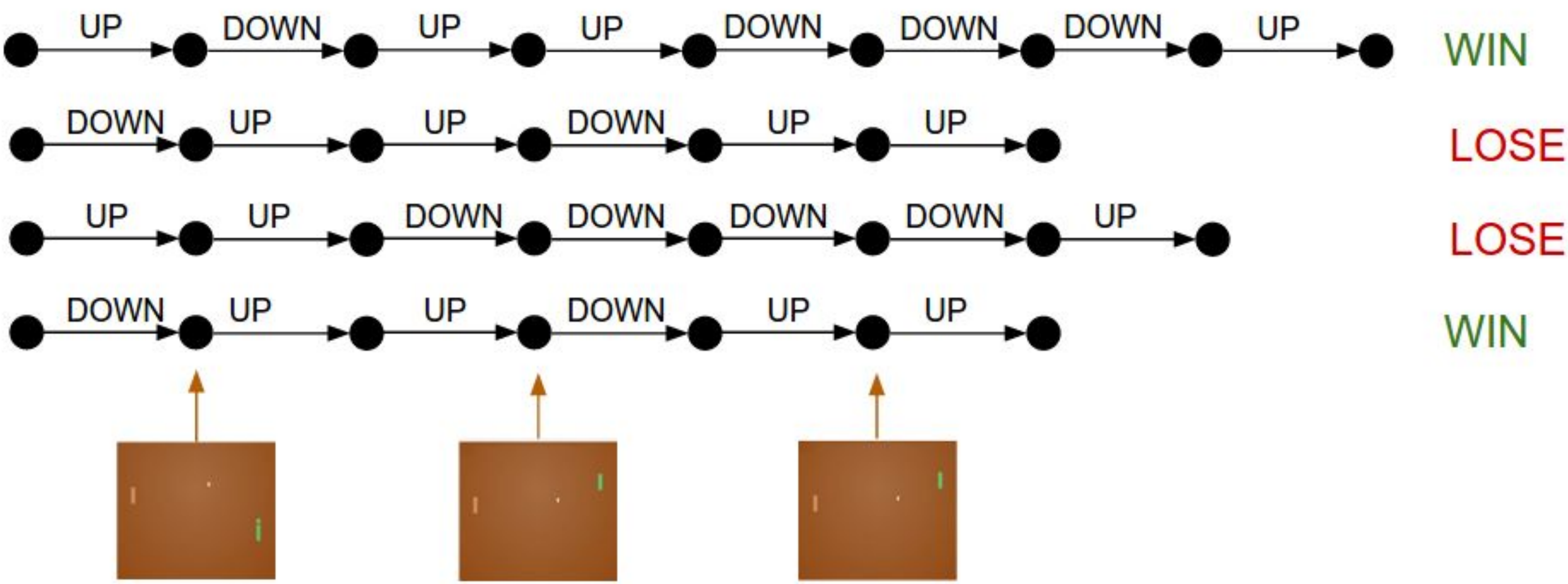
softmax prob [0.3 0.4 0.1 0.2] -> word index z
Pick the max
Pick randomly based on prob

What is the gradient of z?

Two popular methods: REINFORCE, Gumbel-Softmax approximation (https://arxiv.org/abs/1611.01144)

RL and policy gradients

- Credit assignment problem in reinforcement learning

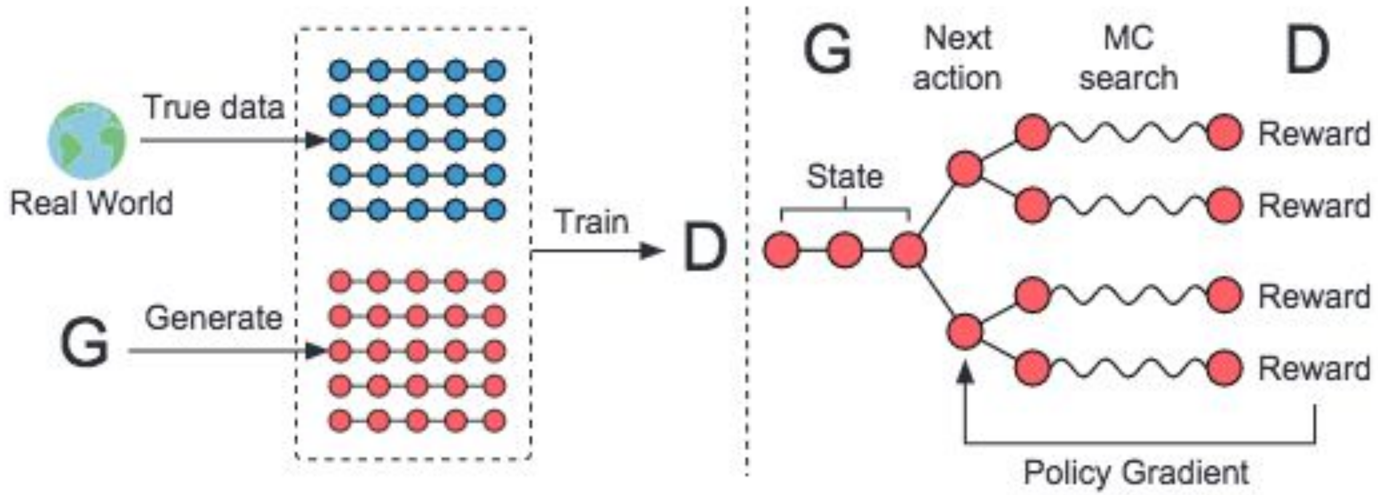


which move makes you win?
For RL with policy gradients, we increase the probability of every move that results in a win (REINFORCE algorithm)

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.

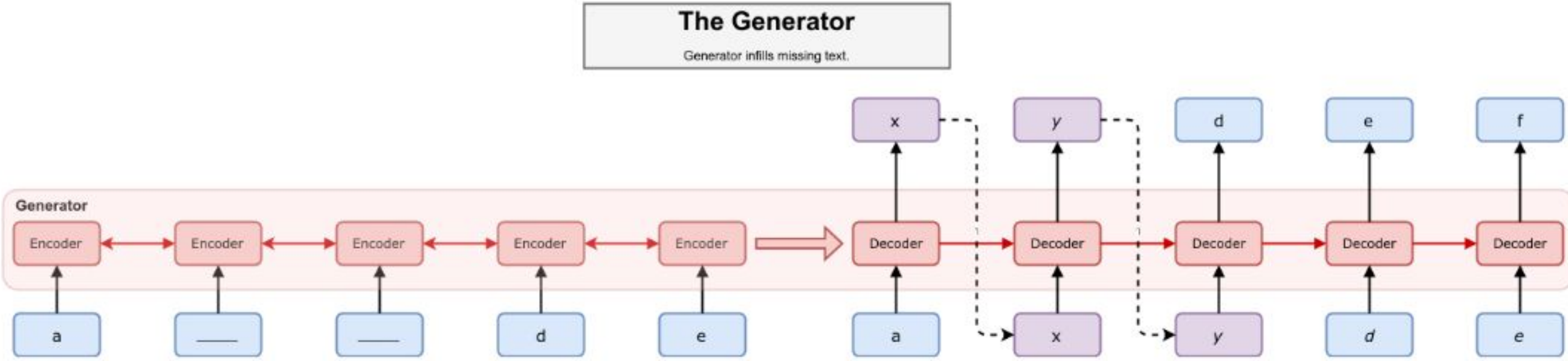
Algorithm	BLEU-3	p-value	BLEU-4	p-value
MLE	0.519	$< 10^{-6}$	0.416	0.00014
SeqGAN	**0.556**		**0.427**	

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

Gumbel Softmax

From softmax prob [0.3 0.4 0.1 0.2] -> word index z

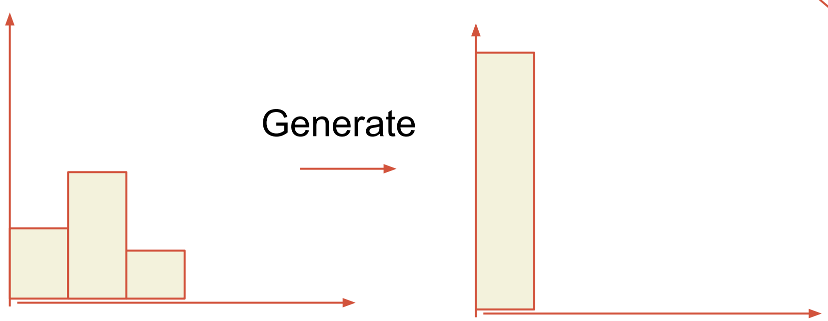
Pick randomly based on prob

random value generated from Gumbel dist.

$$z = \text{one\_hot}\left(\arg\max_{i}\left[g_i + \log \pi_i\right]\right)$$

prob values from softmax

index for each word



https://arxiv.org/pdf/1611.01144.pdf

Gumbel Softmax

From softmax prob [0.3 0.4 0.1 0.2] -> word index z
Pick randomly based on prob

random value generated from Gumbel dist.

$$z = \texttt{one\_hot}\left(\arg\max_i \boxed{[g_i + \log \pi_i]}\right)$$

prob values from softmax

index for each word

$$\boxed{y_i} = \frac{\exp((\log(\pi_i) + g_i)/\tau)}{\sum_{j=1}^{k} \exp((\log(\pi_j) + g_j)/\tau)} \qquad \text{for } i = 1, \ldots, k.$$

Gumbel Softmax

From softmax prob [0.3 0.4 0.1 0.2] -> word index z
Pick randomly based on prob

random value generated from Gumbel dist.

$$z = \texttt{one\_hot}\left(\arg\max_i \boxed{[g_i + \log \pi_i]}\right)$$

prob values from softmax

index for each word

Not a one hot

Temperature parameter

$$\boxed{y_i} = \boxed{\frac{\exp((\log(\pi_i) + g_i)/\tau)}{\sum_{j=1}^{k} \exp((\log(\pi_j) + g_j)/\tau)}} \qquad \text{for } i = 1, \ldots, k.$$

This rescales the distribution

Gumbel Softmax

From softmax prob [0.3 0.4 0.1 0.2] -> word index z
Pick randomly based on prob

Gumbel softmax

Regular argmax

Small T

large T

Not a one hot

Temperature parameter

$$\boxed{y_i} = \boxed{\frac{\exp((\log(\pi_i) + g_i)/\tau)}{\sum_{j=1}^{k} \exp((\log(\pi_j) + g_j)/\tau)}} \quad \text{for } i = 1, \ldots, k.$$

This rescales the distribution

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-GP: https://arxiv.org/abs/1704.00028
- 4) On how GANs are hard to train stil: https://arxiv.org/abs/1711.10337

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