A Taste of Deep Generative Models: Transformers and GANs

Melissa Chen

ECE 208/408 - The Art of Machine Learning

3/24/2023

UNIVERSITY of ROCHESTER

Audio Information Research Laboratory

Language Models

Recent Trends

GPT: Generative Pre-trained Transformer

Source:
https://www.nature.com/articles/d41586-023-00816-5
https://investingnews.com/invest-in-openai-chatgpt/
https://tech.hindustantimes.com/tech/news/dalle-2-to-stable-diffusion-generate-photos-freely-with-these-ai-tools-71674561212855.html

Brief History of Transformers

Sequence to sequence model

An example: language translation

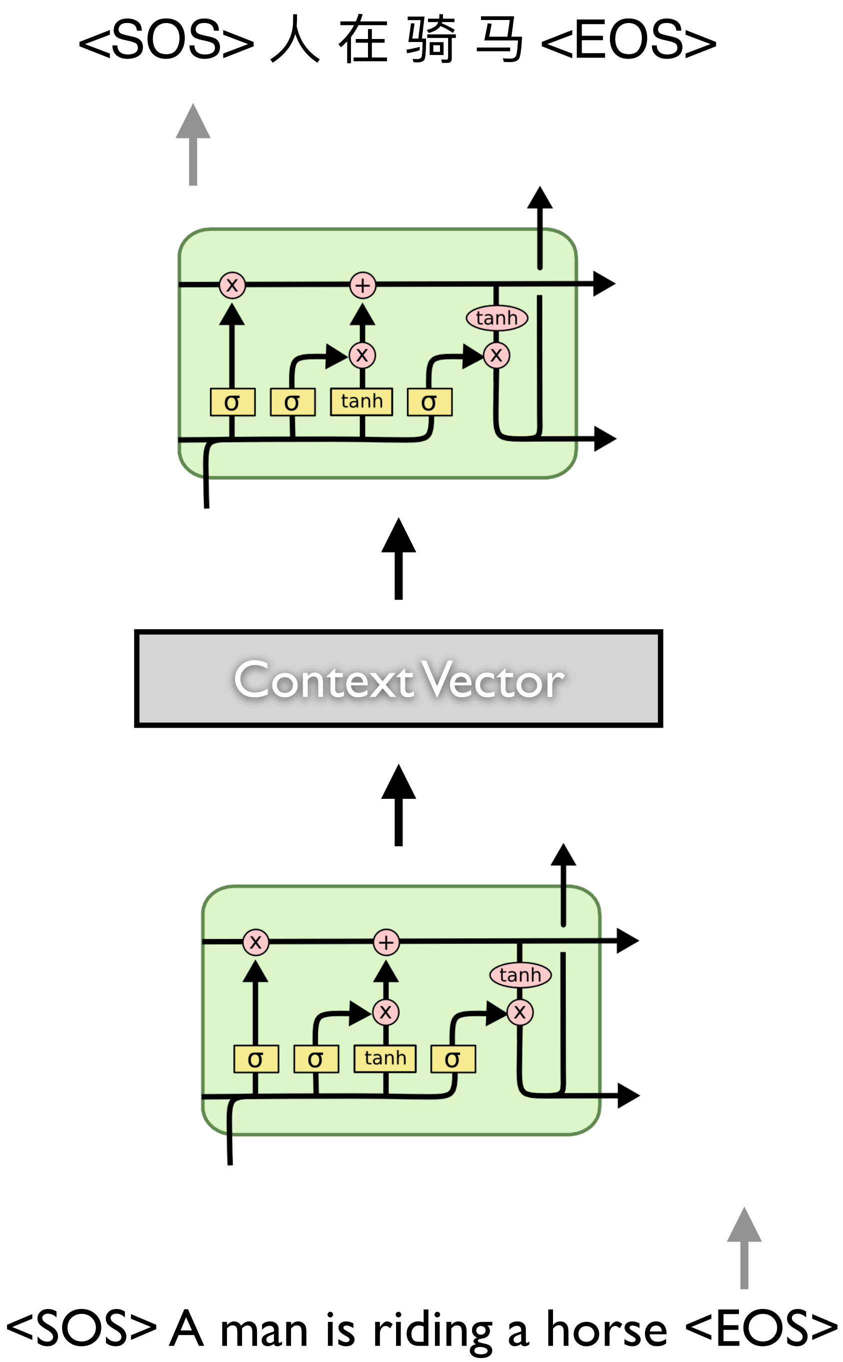
<SOS> 人 在 骑 马 <EOS>

Translator

<SOS> A man is riding a horse <EOS>

Brief History of Transformers

RNN: Encoder-Decoder architecture



1. Input Preparation: tokenize

3. Encoder RNN:
   - one token at a time, generates a hidden state for each input token
   - the final hidden state captures the information of the entire input sentence

4. Decoder RNN:
   - takes the context vector from the encoder

5. Output Generation: highest probability

7. Stopping Criterion: EOS token

9. Training: minimize the difference between its generated and the ground-truth

RNNs suffer from:

- Long sentences
- Slow

"The rapid advancements in artificial intelligence have led to groundbreaking innovations in various fields."

"人工智能的快速发展已经带来了各个领域里的突破性创新。"

Brief History of Transformers

RNN + Attention: longer range memory

“The rapid advancements in artificial intelligence have led to groundbreaking innovations in various fields.”

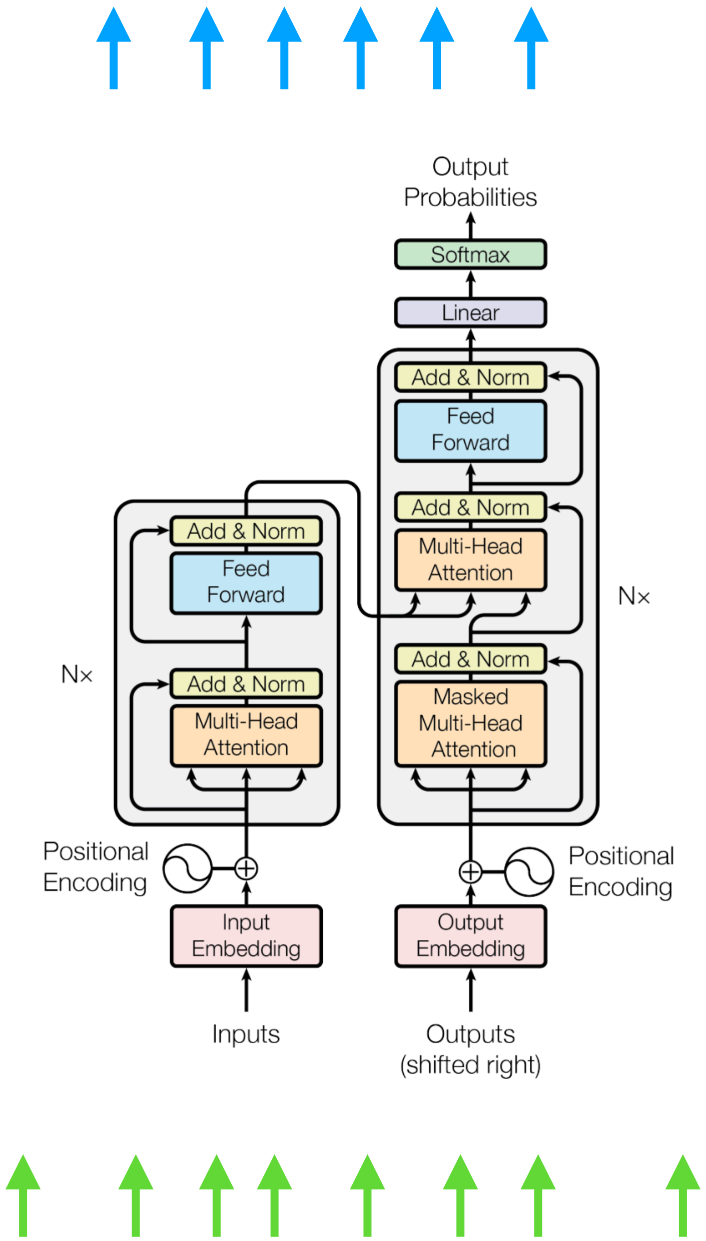
"人工智能的快速发展已经带来了各个领域里的突破性创新。"

Attention mechanisms help the model focus on the most relevant words in a given context.

Brief History of Transformers

Attention is all you need: parallel computing

<SOS> 人 在 骑 马 <EOS>



<SOS> A man is riding a horse <EOS>

Vaswani, Ashish, et al. "Attention is all you need." *Advances in neural information processing systems* 30 (2017).

Brief History of Transformers

- The original transformer is designed for text generation
- The decoder's self-attention is unidirectional

1. She examined the **bark** carefully and noticed the insect damage.
2. She examined the **bark** carefully and recognized it as the sound of her neighbor's dog.

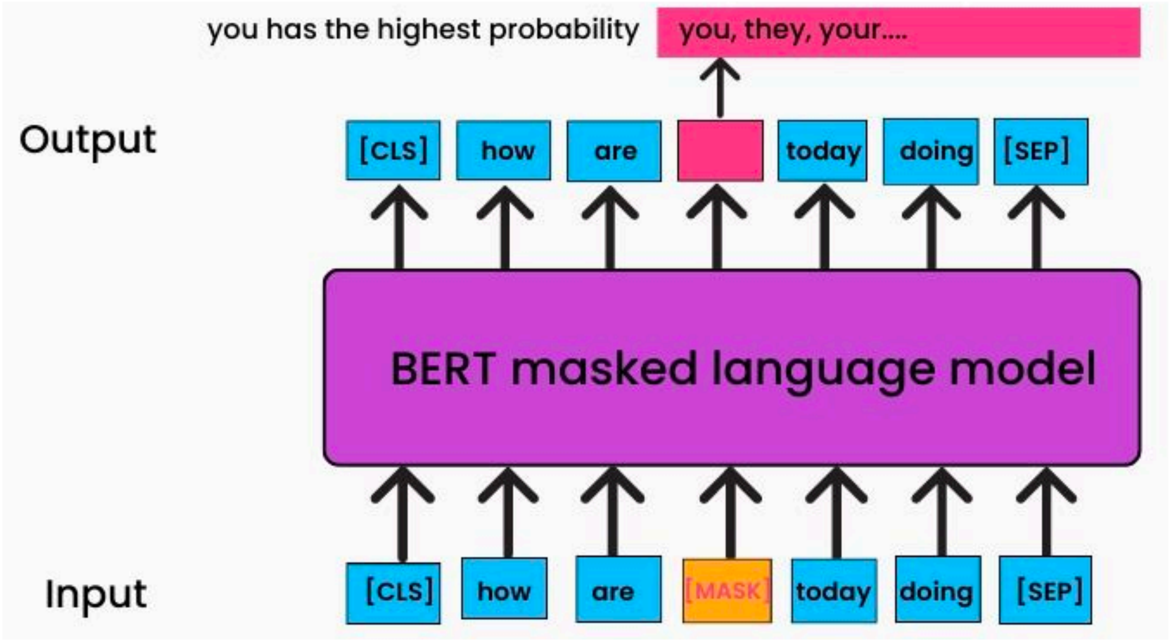
BERT: Add bidirectionally
Bidirectional Encoder Representations from Transformers

Goal: create context-aware word representations for various natural language understanding tasks

Focus on the encoder part

Masked language modeling
- During training, a certain percentage of input words are randomly masked (hidden)
- Predict masked words based on the surrounding context

Exposed to the entire input sequence at once



Devlin, Jacob, et al. "Bert: Pre-training of deep bidirectional transformers for language understanding." *arXiv preprint arXiv:1810.04805* (2018).
Figure source: https://www.turing.com/kb/how-bert-nlp-optimization-model-works

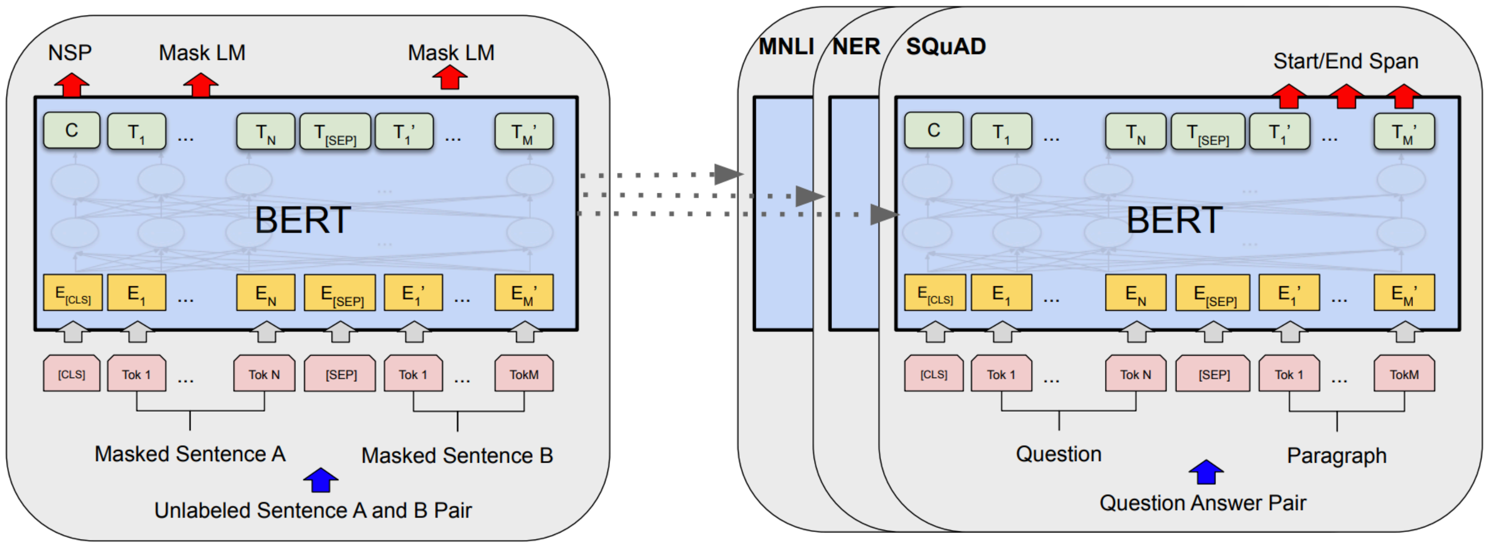
Pertaining and fine-tuning: BERT

During pretraining

- BERT is trained on a large corpus of unannotated text using unsupervised learning
- Learn general language understanding and context-aware word representations

During fine-tuning

- Train on a smaller dataset specific to a target task, e.g. sentiment analysis
- Adding task-specific layers
- Update the weights of the entire model



Pre-training   Fine-Tuning

Masked Language Model (MLM)
Next Sentence Prediction (NSP): To understand the relationship between sentences, predict whether two input sentences follow each other.

Devlin, Jacob, et al. "Bert: Pre-training of deep bidirectional transformers for language understanding." *arXiv preprint arXiv:1810.04805* (2018).

Brief History of Transformers

Vision transformer

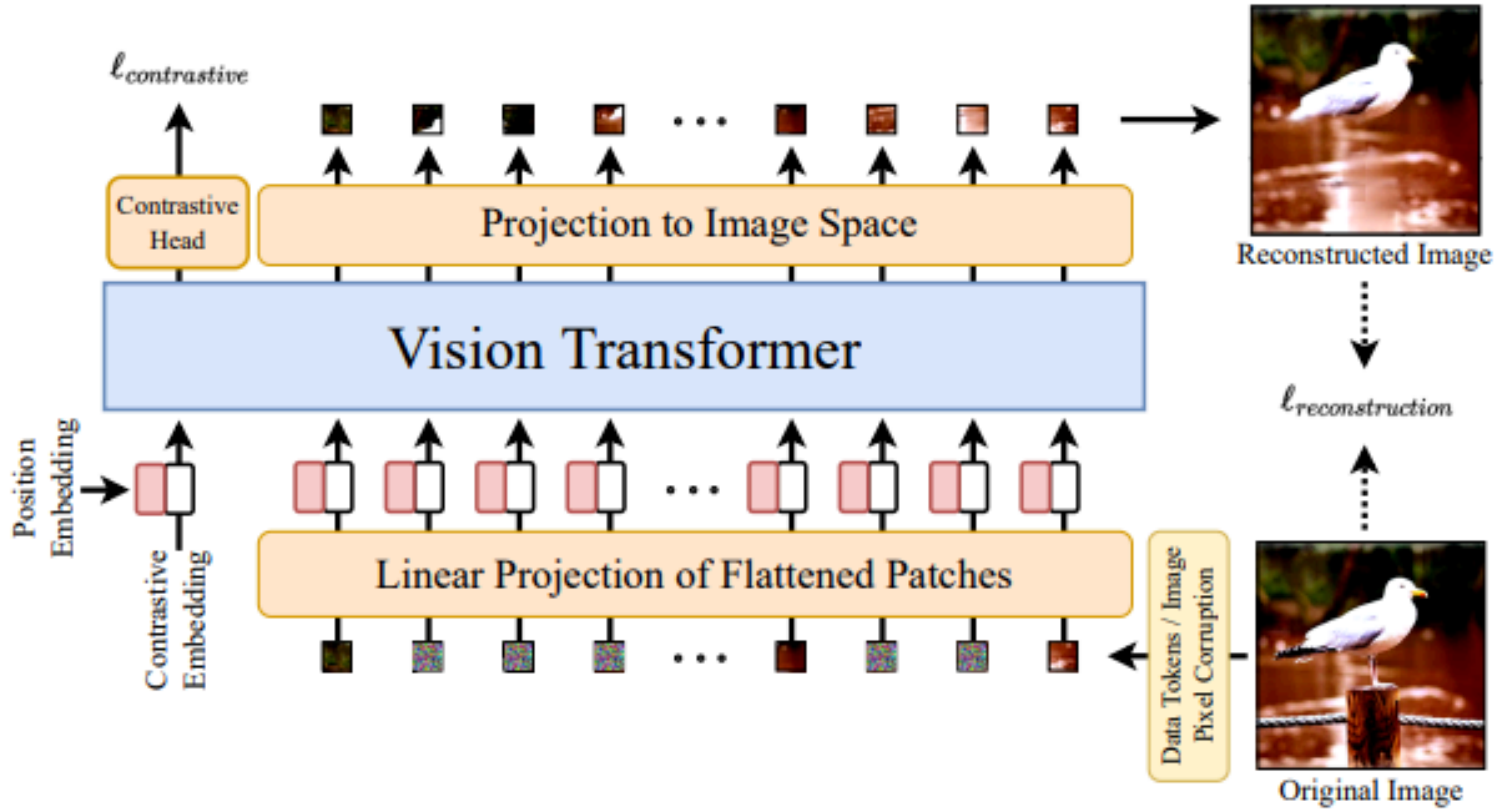


Fig. 1: Self-supervised vIsion Transformer (SiT)

Atito, Sara, Muhammad Awais, and Josef Kittler. "Sit: Self-supervised vision transformer." *arXiv preprint arXiv:2104.03602* (2021).

Brief History of Transformers

Language-vision transformer

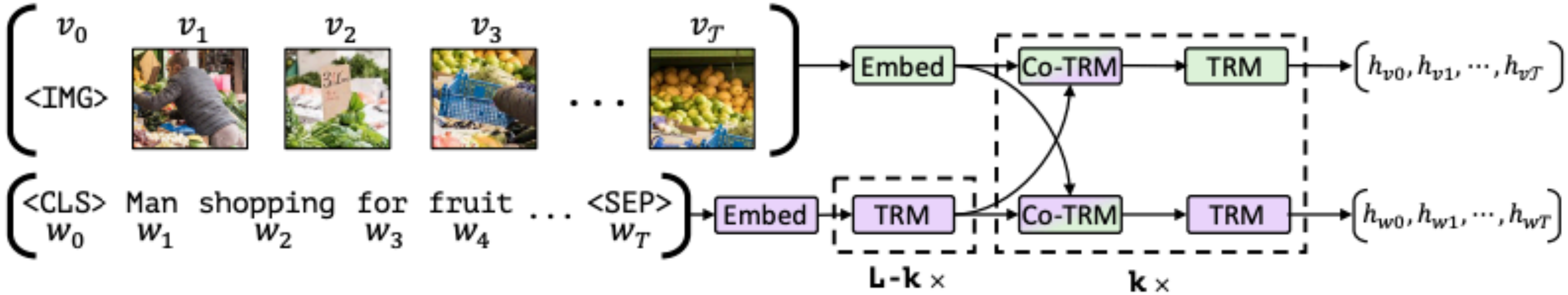
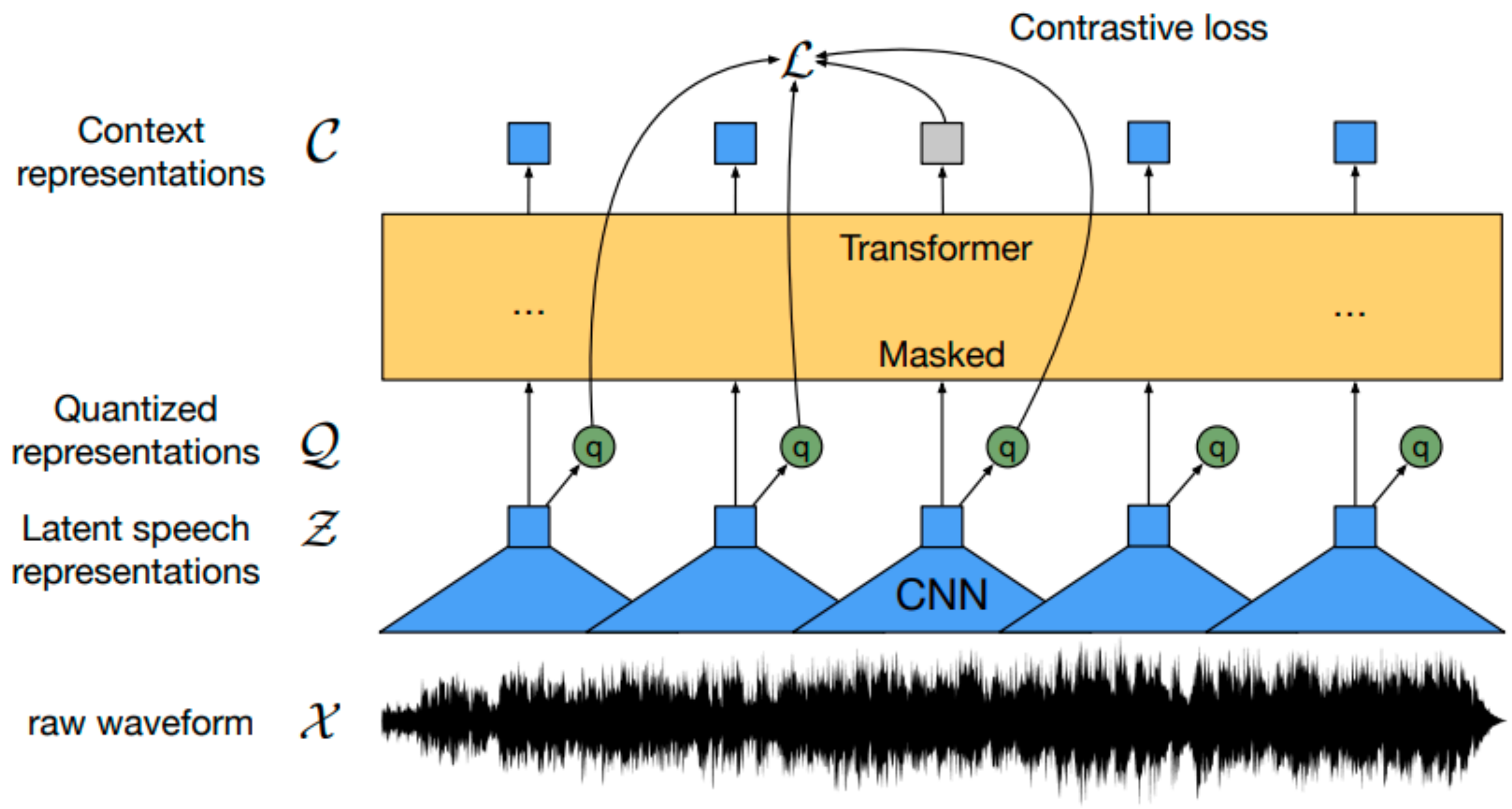


Figure 1: Our ViLBERT model consists of two parallel streams for visual (green) and linguistic (purple) processing that interact through novel co-attentional transformer layers. This structure allows for variable depths for each modality and enables sparse interaction through co-attention. Dashed boxes with multiplier subscripts denote repeated blocks of layers.

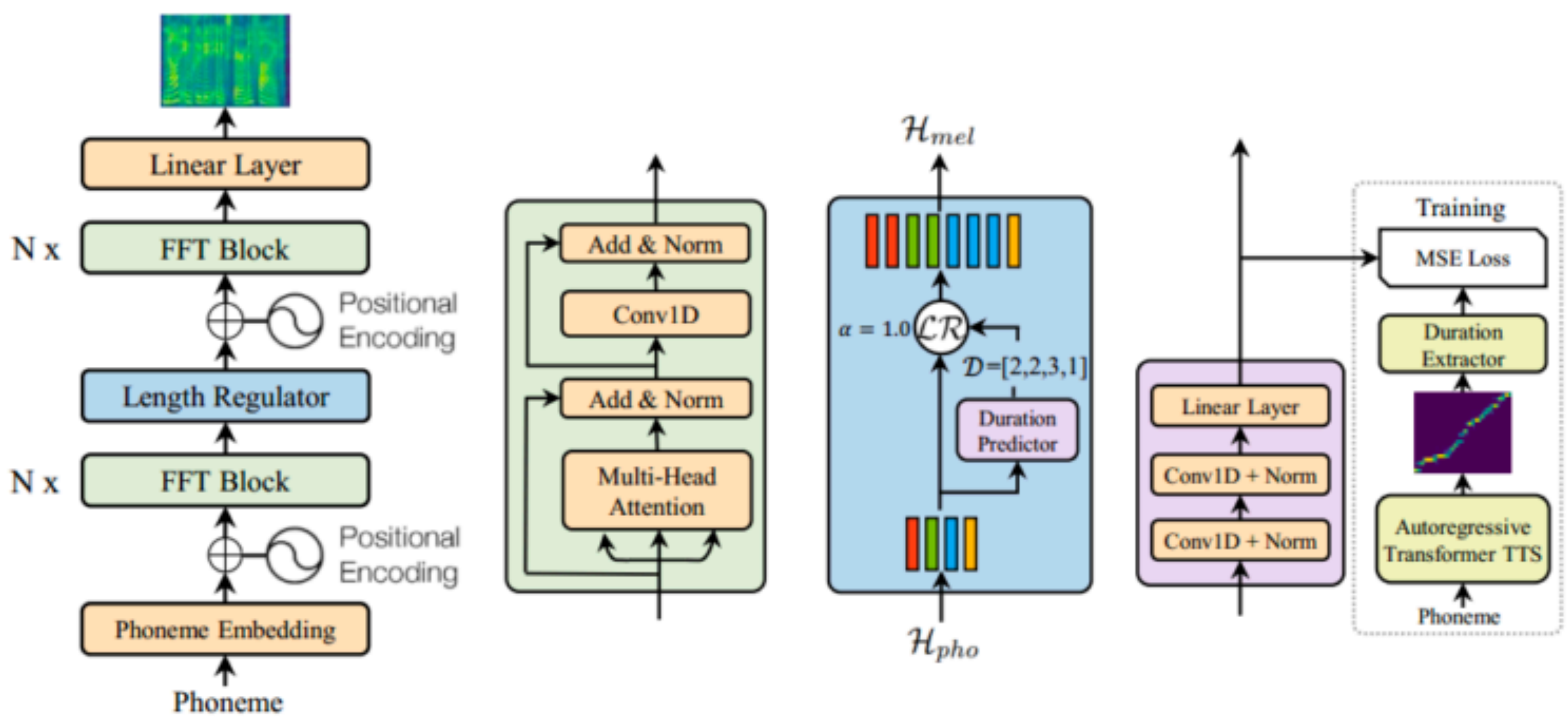
Lu, Jiasen, et al. "Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks." Advances in neural information processing systems 32 (2019).

Speech-language Transformer



Baevski, Alexei, et al. "wav2vec 2.0: A framework for self-supervised learning of speech representations." *Advances in neural information processing systems* 33 (2020): 12449-12460.

Text-to-speech Transformer

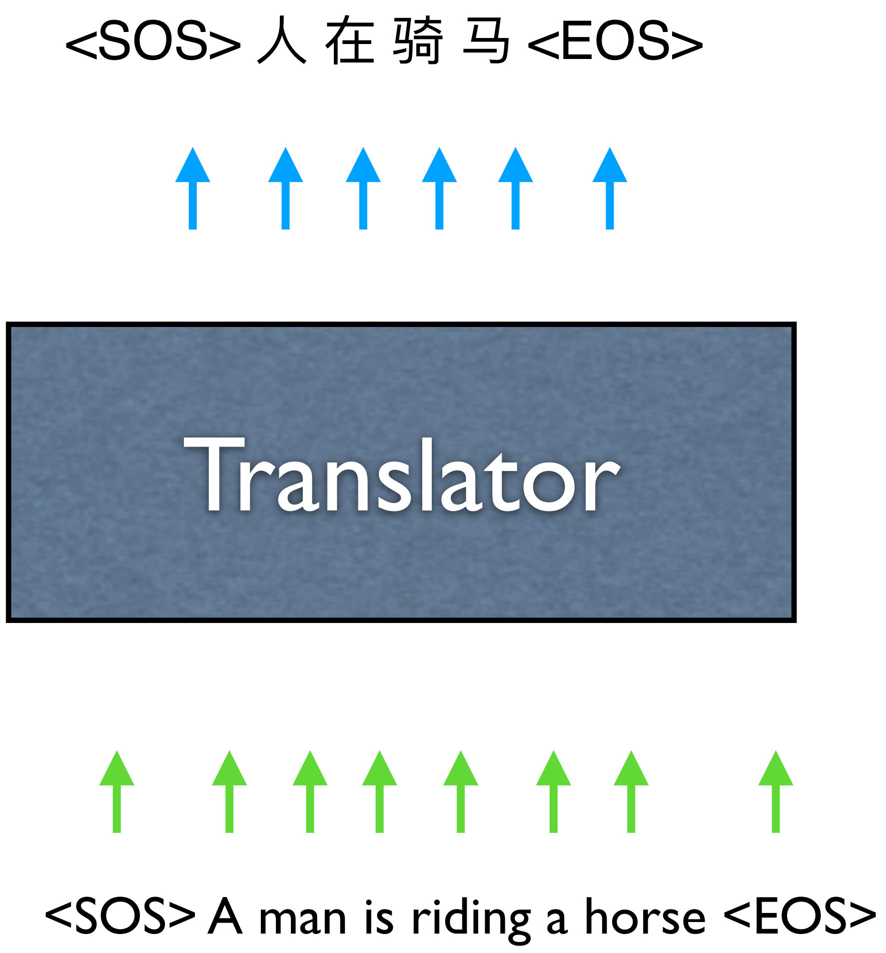


(a) Feed-Forward Transformer (b) FFT Block (c) Length Regulator (d) Duration Predictor

Ren, Yi, et al. "Fastspeech: Fast, robust and controllable text to speech." *Advances in neural information processing systems* 32 (2019).

Transformer Architecture

Back to Machine Translation



Attention is all you need

Encoder

Decoder

Encoder Inputs

Decoder Inputs

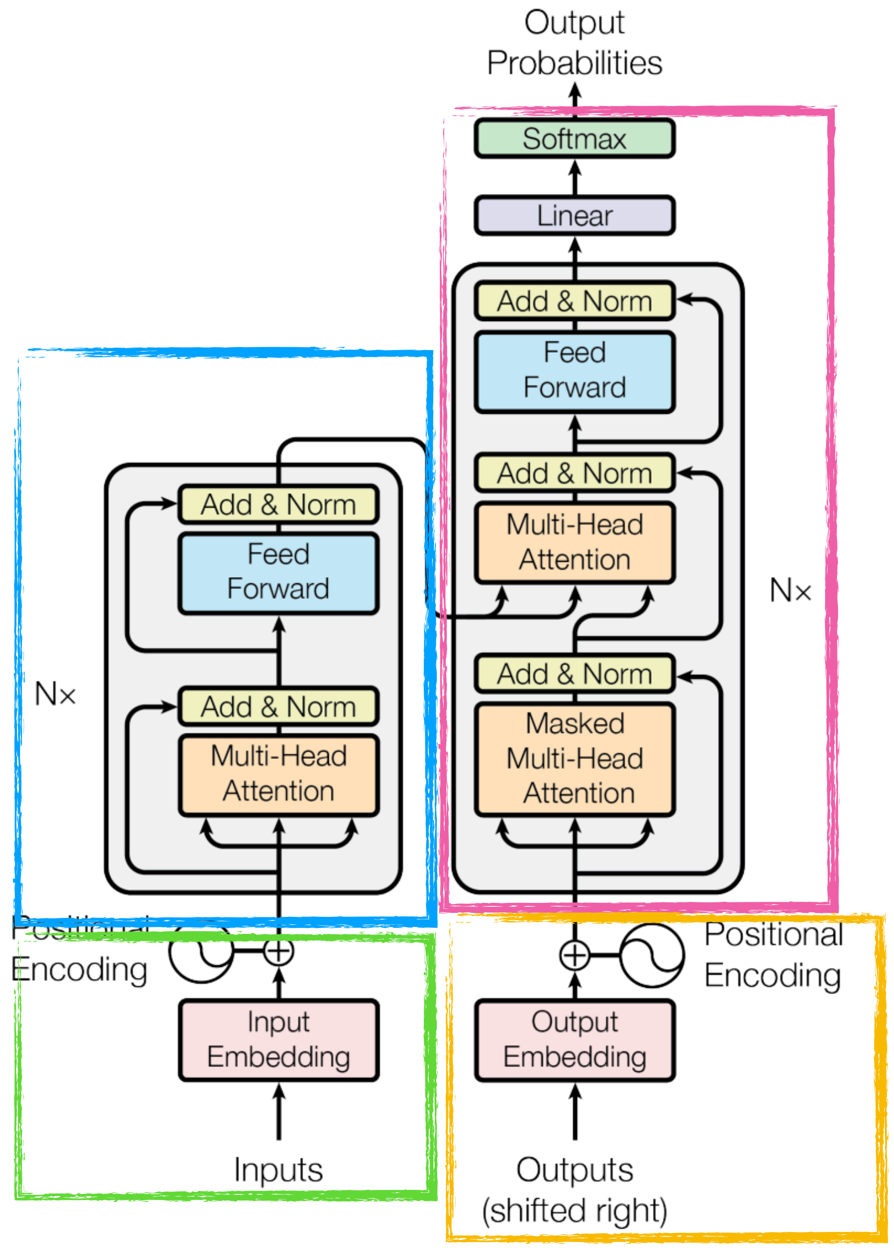


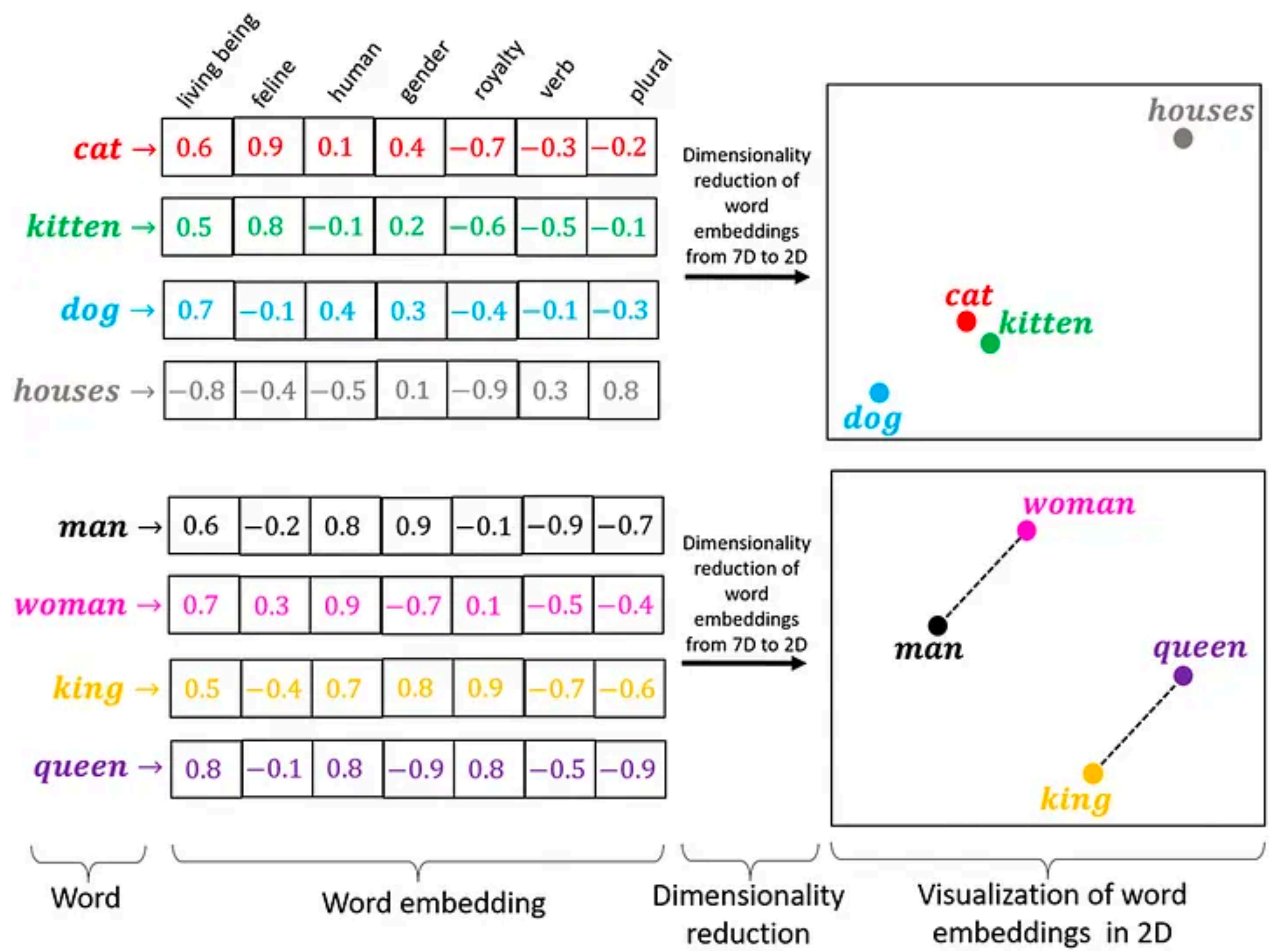
Figure 1: The Transformer - model architecture.

Vaswani, Ashish, et al. "Attention is all you need." *Advances in neural information processing systems* 30 (2017).

Input Embeddings

Word2Vec Embedding

- Convert words into numerical representations called word embeddings
- Words with similar meanings close to each other in the high-dimensional space
- Encodes semantic information



Source: https://swatimeena989.medium.com/training-word2vec-using-gensim-14433890e8e4

Mikolov, Tomas, et al. "Efficient estimation of word representations in vector space." *arXiv preprint arXiv:1301.3781* (2013).

Input Embeddings

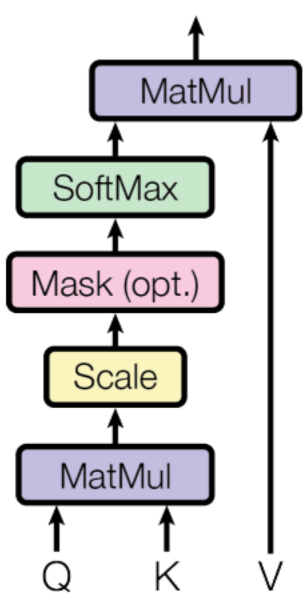
Positional embedding

$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}})$$
$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$

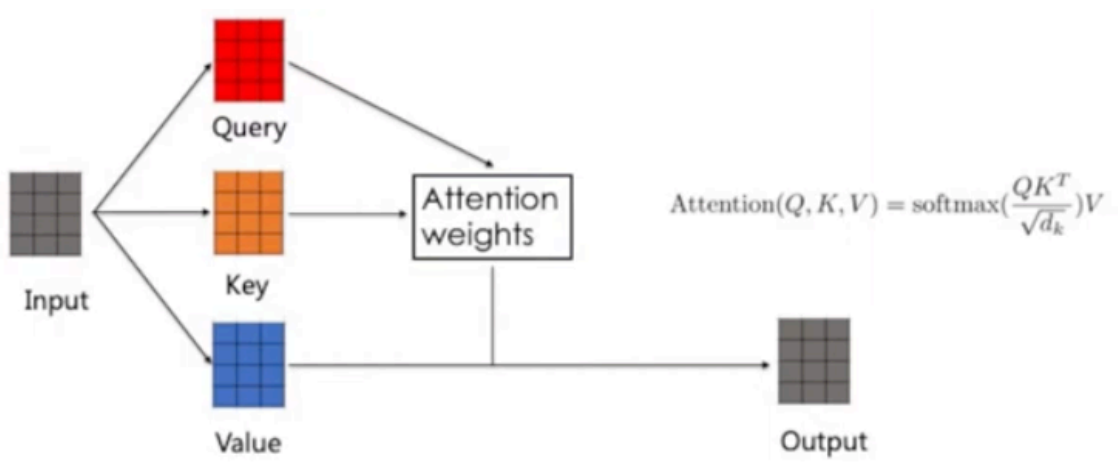
Self-attention

Self-attention helps the models associate each word in the input with other words appropriately

Scaled Dot-Product Attention



Query, Key, and Value

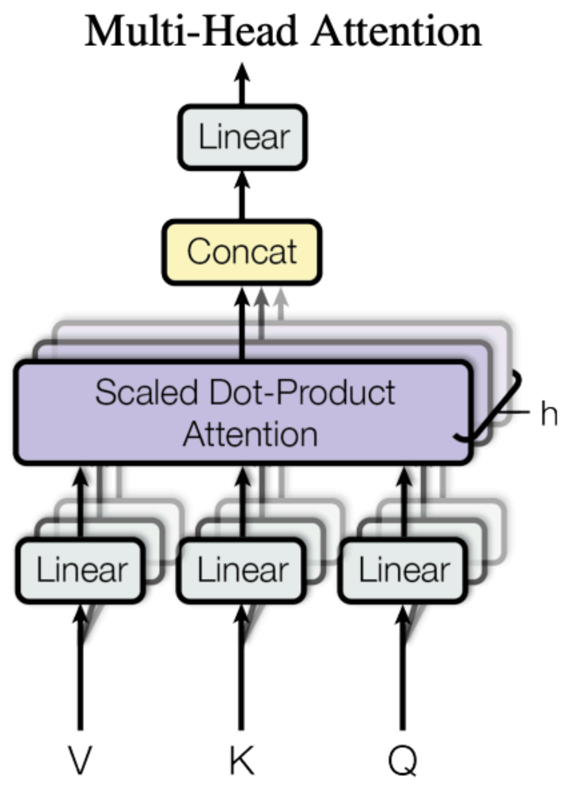


Output is a weighted sum of values

The weights are determined by the dot-product of the query with all the keys

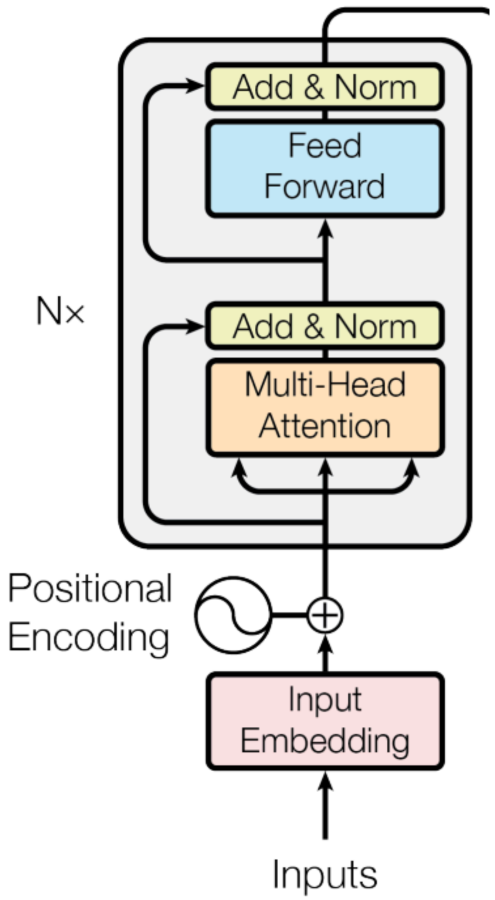
Left figure is from the original paper, right is from https://www.youtube.com/watch?v=5T38-2J5CcY

Multi-Head Attention



- Allows the model to focus on different aspects of the input sequence simultaneously
- Multiple self-attention mechanisms in parallel (i.e. heads)
- Each head computes its own attention weights and generates its own context vectors
- The outputs from all heads are combined to form the final context representation

Encoder



MH Attentions

Help the decoder focus on the appropriate words during decoding

Residual Connections

Allows gradients to flow through the networks directly

Layer Normalization

Stabilize the network, reducing the training time

Pointwise Feed Forward

Project the attention outputs potentially giving it a richer representation

Attention is all you need

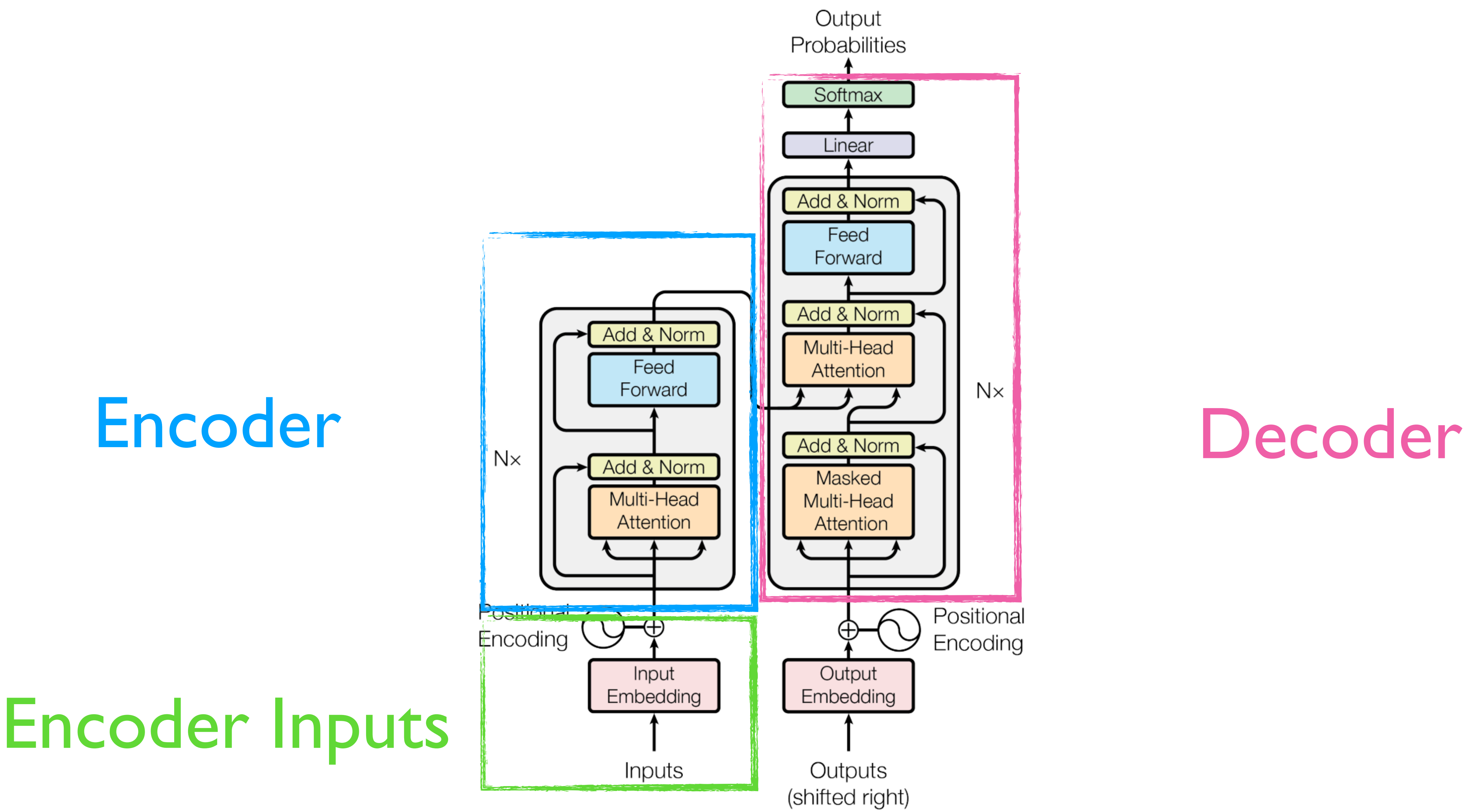
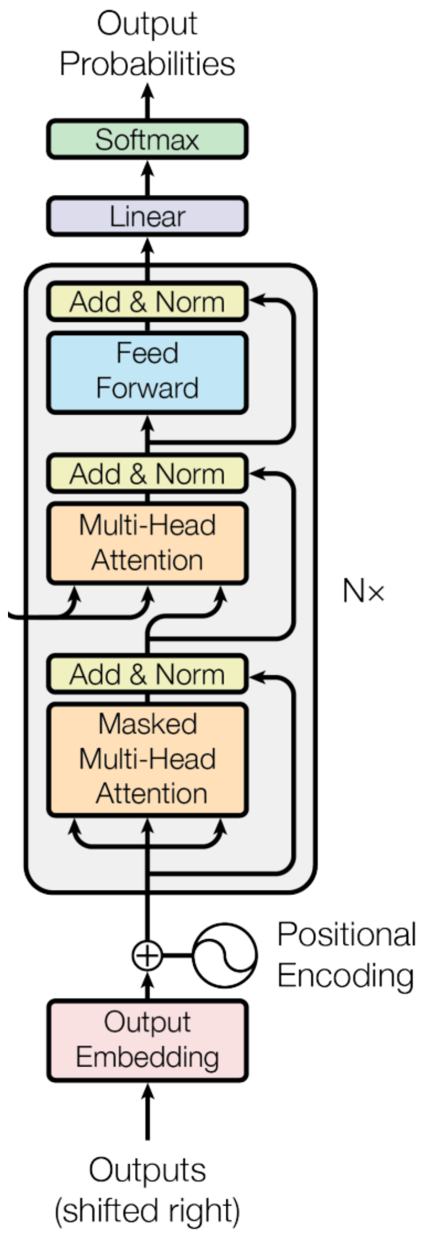


Figure 1: The Transformer - model architecture.

Vaswani, Ashish, et al. "Attention is all you need." *Advances in neural information processing systems* 30 (2017).

Masked Multi-head attention



- Input to decoder is shifted to the right by one position
- No future information
- Simulate inference

I Am Learn' Math

SCALED SCORES

	I	Am	Learning	Math
I	0.7	0.1	0.1	0.1
Am	0.1	0.6	0.2	0.1
Learning	0.1	0.3	0.6	0.1
Math	0.1	0.3	0.3	0.3

*

LOOK-AHEAD MASK

1	-inf	-inf	-inf
1	1	-inf	-inf
1	1	1	-inf
1	1	1	1

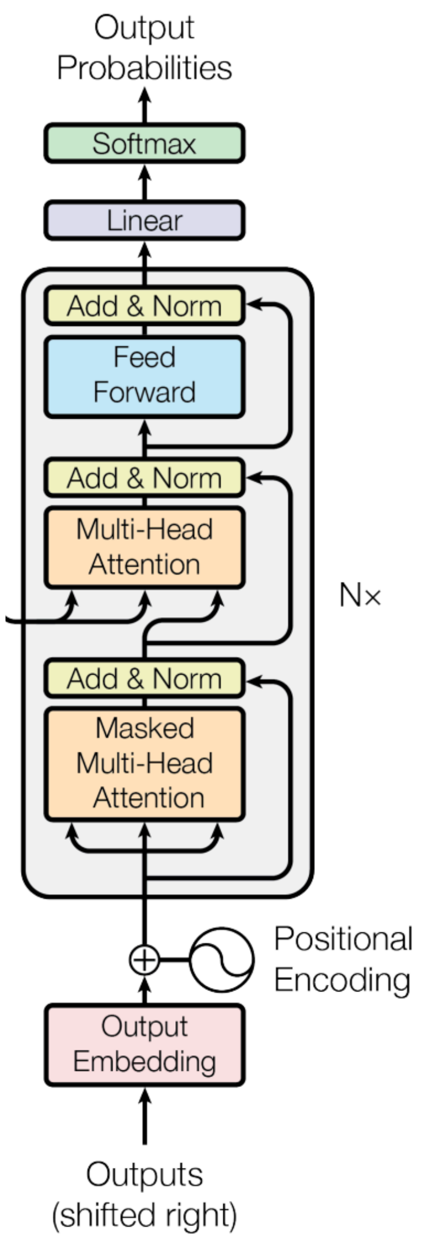
=

MASKED SCORES

0.7	-inf	-inf	-inf
0.1	0.6	-inf	-inf
0.1	0.3	0.6	-inf
0.1	0.3	0.3	0.3

Source: https://www.revistek.com/posts/transformer-architecture

Final Classifier



Linear Classifier

Softmax outputs probabilities

The decoder can be stacked N layers

Learn to focus on different combinations of attention from its heads

Attention is all you need

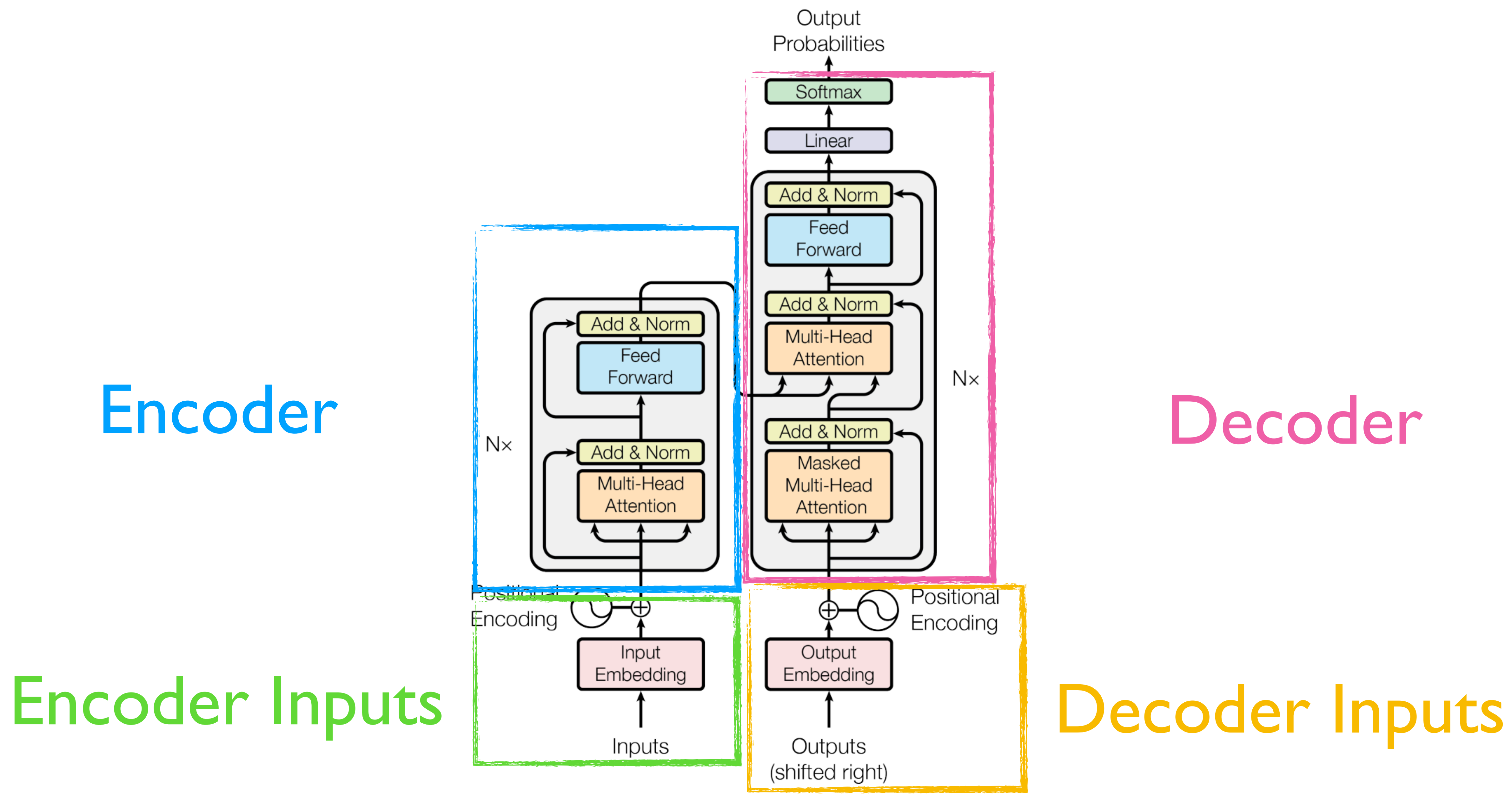


Figure 1: The Transformer - model architecture.

Vaswani, Ashish, et al. "Attention is all you need." *Advances in neural information processing systems* 30 (2017).

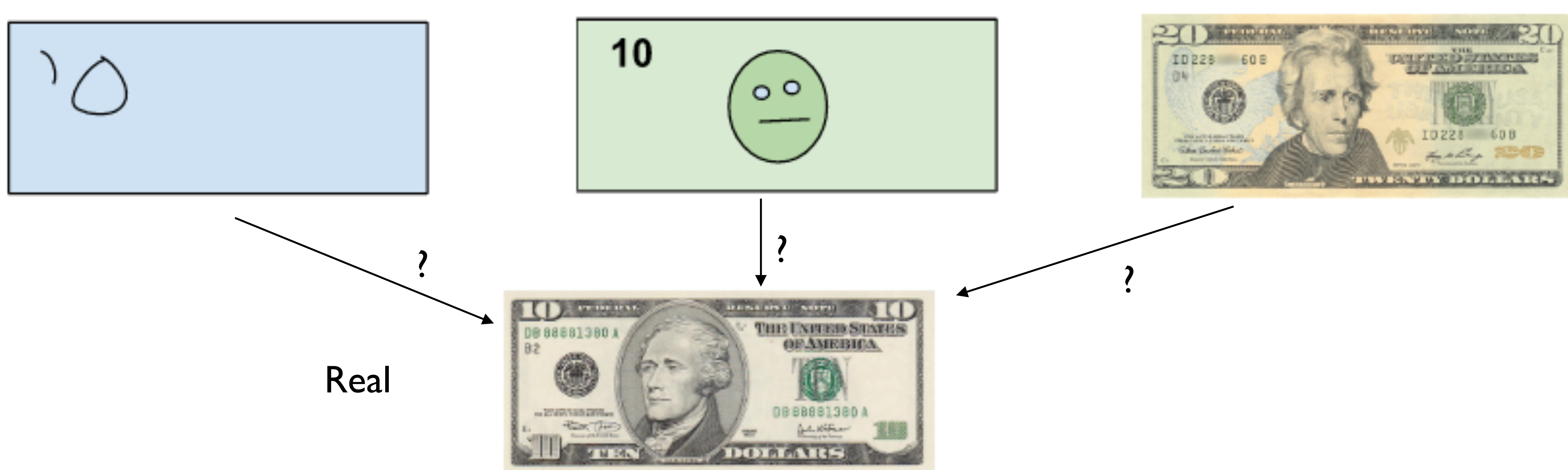
GANs

Deep generative models learn to represent probability distributions over the training data and generate new data with the same statistics.

Challenges

- High-dimensional and large dataset
- Complex probability distributions
- Intractable likelihoods

Generative Adversarial Network (GAN) is a framework for estimating generative models, a training scheme



Source: https://developers.google.com/machine-learning/gan/gan_structure

Key Components

Generator and discriminator

Generator: create fake samples
Discriminator: distinguish real from fake samples

When training begins, the generator produces obviously fake data, and the discriminator quickly learns to tell that it's fake:

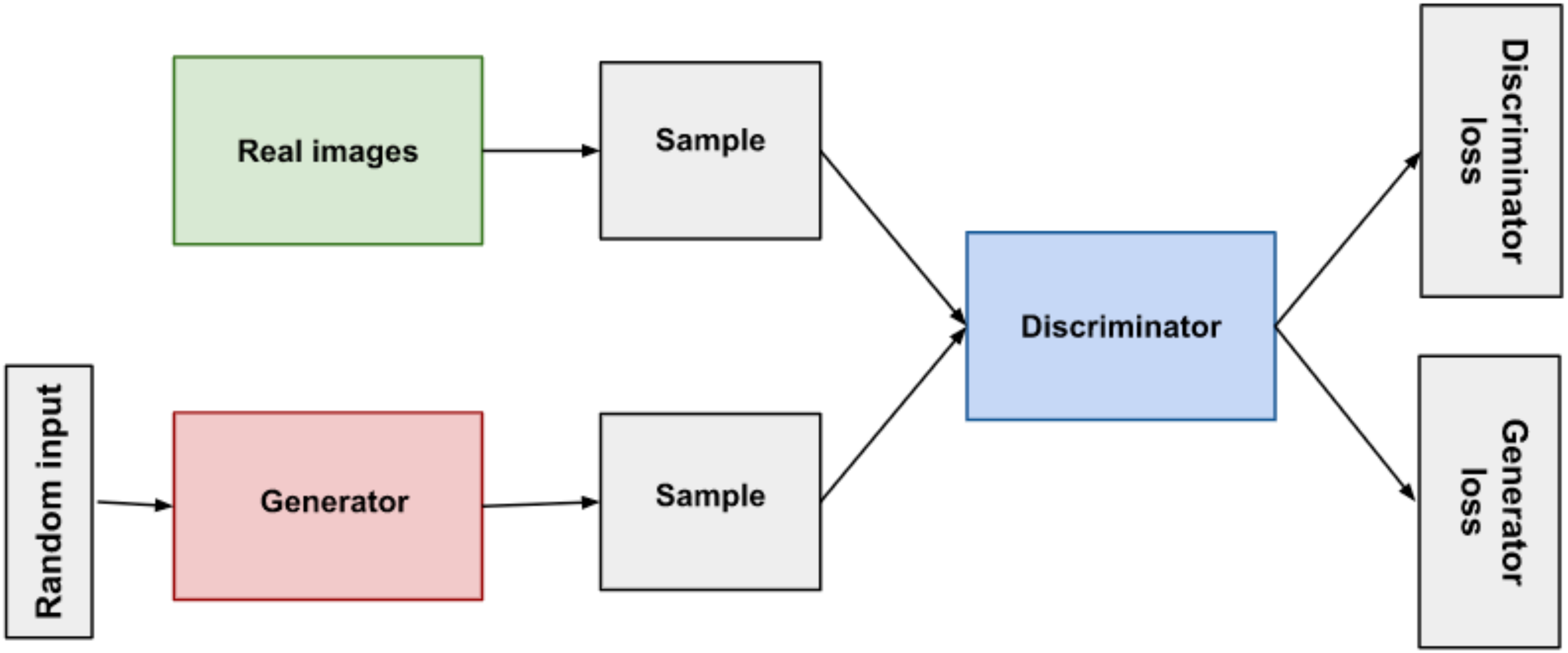
As training progresses, the generator gets closer to producing output that can fool the discriminator:

Finally, if generator training goes well, the discriminator gets worse at telling the difference between real and fake. It starts to classify fake data as real, and its accuracy decreases.

Source: https://developers.google.com/machine-learning/gan/gan_structure

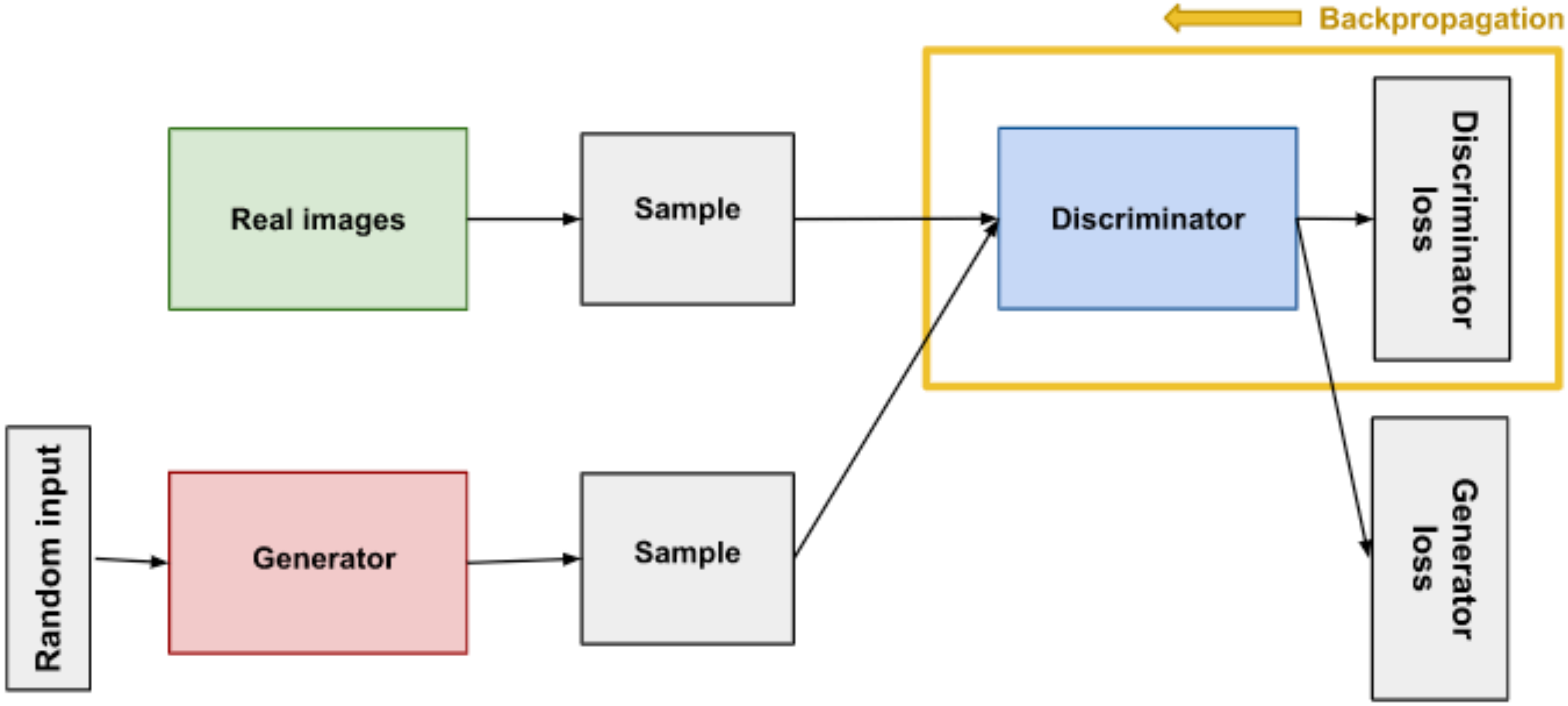
Minimax Game Concept

Generator aims to minimize the discriminator's ability to distinguish fake from real samples, while the discriminator aims to maximize its accuracy



Source: https://developers.google.com/machine-learning/gan/gan_structure

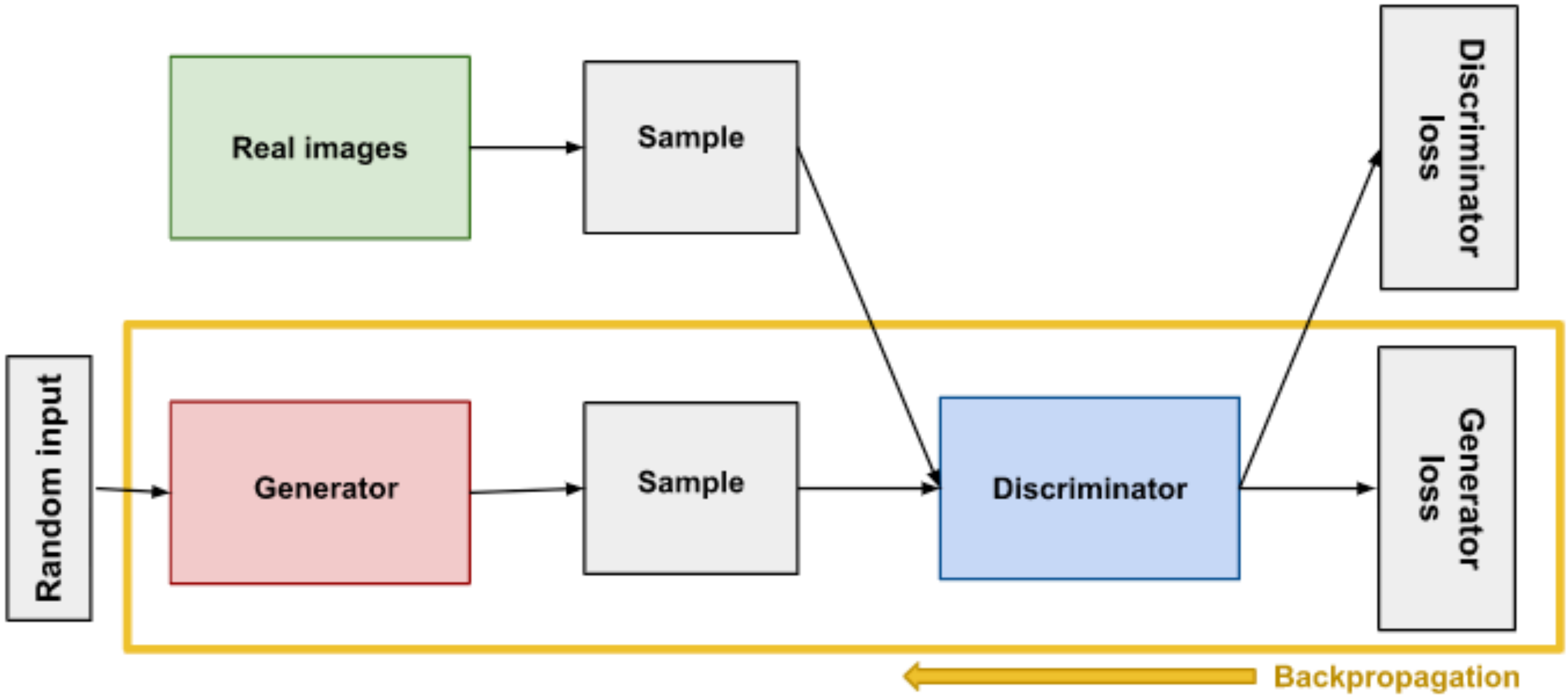
Discriminator Training



- Differentiates between real and fake data generated by the generator
- Loss penalizes the misclassification
- Weights updated via backpropagation using the discriminator loss within the discriminator

Source: https://developers.google.com/machine-learning/gan/gan_structure

Generator Training



- Sample random noise, generate output using noise
- Obtain "Real" or "Fake" classification from discriminator
- Compute loss based on classification
- Backpropagate to get gradients from both networks
- Update only generator weights with gradients

Source: https://developers.google.com/machine-learning/gan/gan_structure

Loss Functions

Minimax loss

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

E is the expected value.

For the generator, minimizing the loss is equivalent to minimizing `log(1 - D(G(z)))`

The formula derives from the cross-entropy between the real and generated distributions

Wasserstein Loss

Discriminator loss: maximize D(x) - D(G(z))

The difference between its output on real instances and its output on fake instances

Generator loss: maximize D(G(z))

Maximize the discriminator's output for its generated output

The formulas derives from the earth mover distance

The Iterative Training Process

1. The discriminator trains for one or more epochs
2. The generator trains for one or more epochs
3. Repeat 1 and 2

- GAN convergence is hard to identify.

Ideally discriminator reaches a 50% accuracy.

In practice, discriminator is getting weaker overtime and gives less meaningful feedback

Challenges and Limitations

- **Mode collapse**: generator produces a limited variety of samples
- **Vanishing gradients**: if your discriminator is too good
- **Training instability**: sensitivity to hyperparameters and architecture choices
- **Evaluation difficulties**: measuring the quality of generated samples
- …

Training tricks

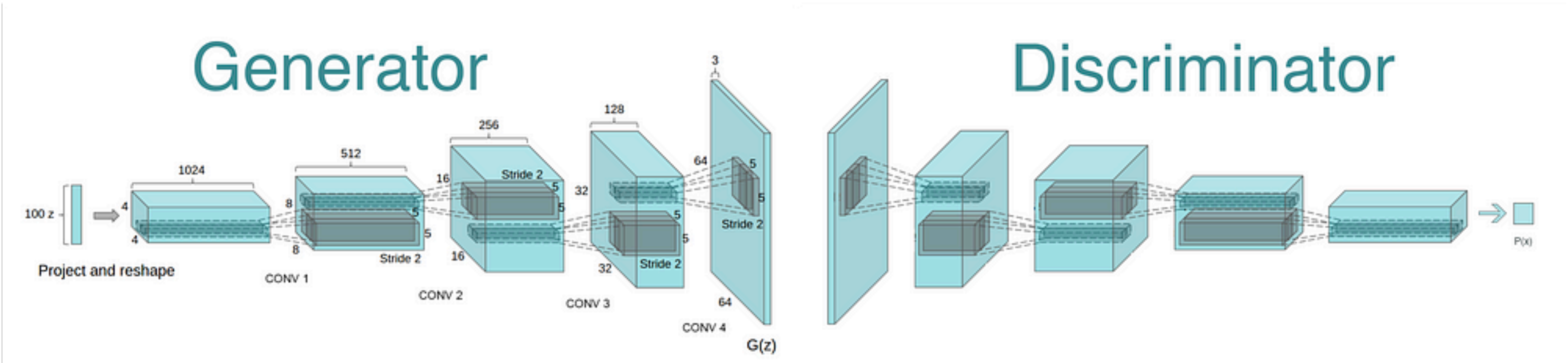
The key is: the balance between the generator and discriminator during training

If the discriminator is too good, it will return 0 or 1; if the generator is too good, it will exploit the weaknesses in the discriminator.

- Pre-train the discriminator on another dataset like MNIST before training the generator
- Pre-train the generator on the target dataset for several epochs
- 2:1 updates
- Babysit the training process
- Reweighing the losses
- Use other loss functions

…

1. **Deep Convolutional GAN (DCGAN)**: employs deep convolutional layers in both the generator and discriminator
Radford, Alec, Luke Metz, and Soumith Chintala. "Unsupervised representation learning with deep convolutional generative adversarial networks." arXiv preprint arXiv:1511.06434 (2015).



2. **Conditional GAN (cGAN):** the paper demonstrates the concept of conditioning both the generator and discriminator on some additional information, which allows the model to generate data samples with desired characteristics based on the given condition. Mirza, Mehdi, and Simon Osindero. "Conditional generative adversarial nets." arXiv preprint arXiv:1411.1784 (2014).

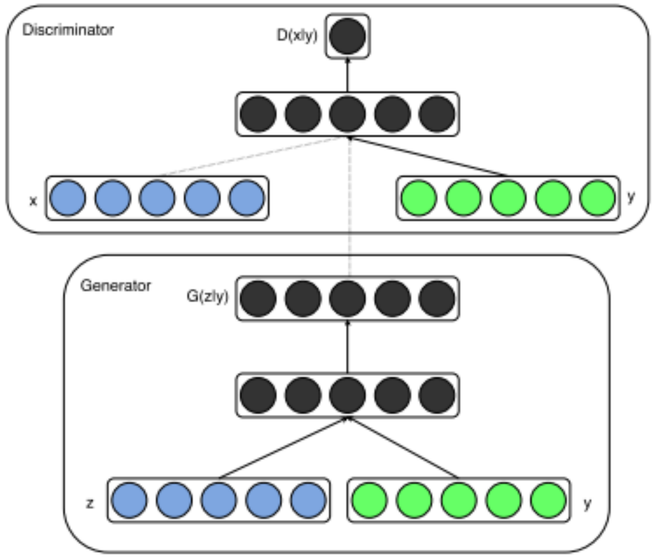
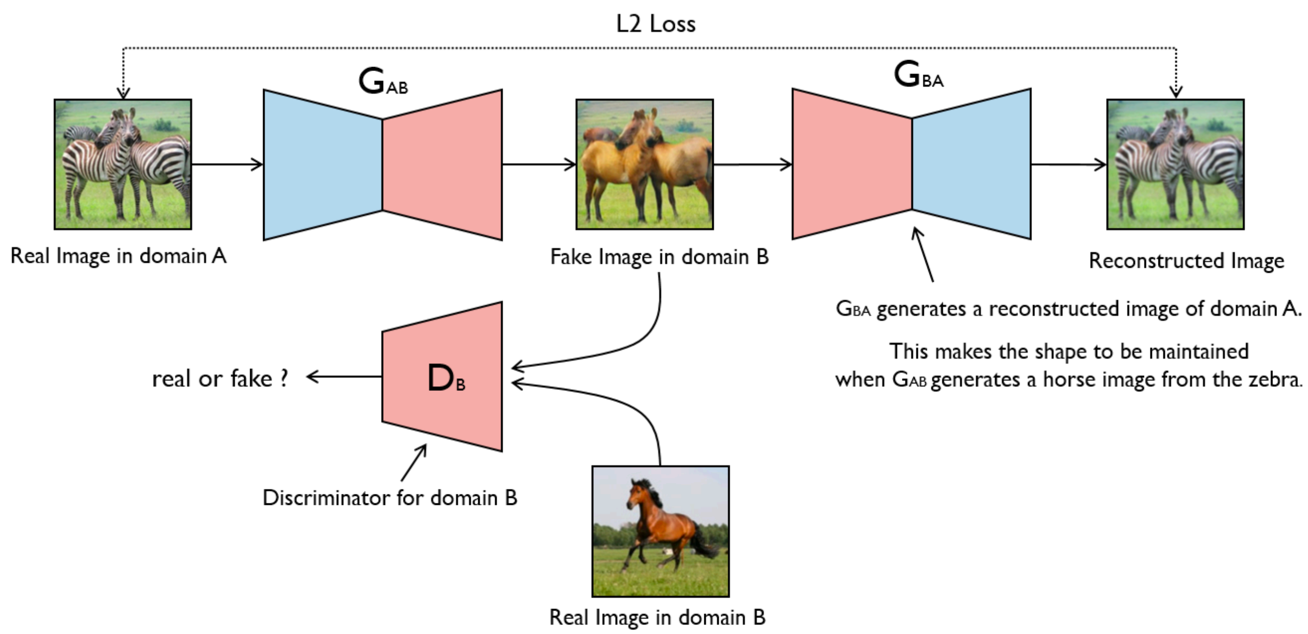


Figure 1: Conditional adversarial net

Some GAN Variants

1. **CycleGAN**: CycleGAN enables unpaired image-to-image translation by using a cycle consistency loss, which encourages the preservation of content while altering the style. Zhu, Jun-Yan, et al. "Unpaired image-to-image translation using cycle-consistent adversarial networks." Proceedings of the IEEE international conference on computer vision. 2017.



2. **HiFi-GAN** is a high-fidelity generative adversarial network for high-quality and efficient speech synthesis. Kong, Jungil, Jaehyeon Kim, and Jaekyoung Bae. "Hifi-gan: Generative adversarial networks for efficient and high fidelity speech synthesis." Advances in Neural Information Processing Systems 33 (2020): 17022-17033.