Machine Learning Department
School of Computer Science
Carnegie Mellon University

Vision Transformers

+

Generative Adversarial Networks (GANs)

Matt Gormley
Lecture 5
Jan. 31, 2024

Reminders

- Homework 1: Generative Models of Text
  - Out: Thu, Jan 25
  - Due: Wed, Feb 7 at 11:59pm
- Matt's office hours on GCal

Q&A

Q: Does pre-training always involve layer-by-layer unsupervised training?

A: No! That's just where it started in 2006 for standard feed forward neural networks.

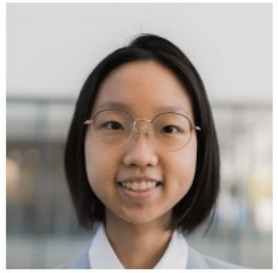
- To pretrain a CNN or Vision Transformer, we typically train the entire model on a supervised classification problem (i.e. image classification)
- To pretrain an LLM, we typically train the entire model on the likelihood of unlabeled sentences.

Course Staff

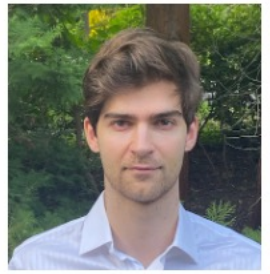

Meghana


Advaith Sridhar


Afreen Shaikh

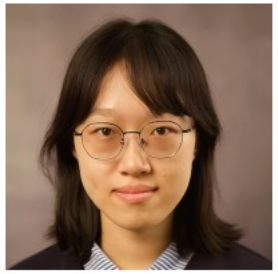

Haohui Liu


Jacob Rast


Samuel Sokota


Ifigeneia Apostolopoulou


Qin Han


Jing Gao


Ritu Gala


Asmita


Haoyang He

Tiancheng Zhao


Joshmin Ray


Yuanzhi Li


Matt Gormley

*take two

ROTARY POSITION EMBEDDINGS (ROPE)

Rotary Position Embeddings (RoPE)

Q: Why does this slide have so many typos?

A: I'm really not sure. I very meticulously type up the latex for my slides myself and think carefully about all the things I put in them.

RoPE attention:

$$f_q(\mathbf{x}_t, m) \triangleq \mathbf{R}_{\Theta,m} \mathbf{W}_q^T \mathbf{x}_t$$

$$f_k(\mathbf{x}_j, m) \triangleq \mathbf{R}_{\Theta,m} \mathbf{W}_k^T \mathbf{x}_j$$

$$s_{t,j} = f_k(\mathbf{x}_j, m)^T f_q(\mathbf{x}_t, m) / \sqrt{|\mathbf{k}|},$$

$$\forall j, t \text{ where } m = t - j$$

$$\mathbf{a}_t = \text{softmax}(\mathbf{s}_t), \forall t$$

wrong

wrong

wrong

wrong

where $\mathbf{W}_k, \mathbf{W}_q \in \mathbb{R}^{d_{model} \times d_k}$, and the rotary matrix $\mathbf{R}_{\Theta,m} \in \mathbb{R}^{d_k \times d_k}$ is given by:

$$R_{\Theta,m} = \begin{pmatrix} \cos m\theta_1 & -\sin m\theta_1 & 0 & 0 & \dots & 0 & 0 \\ \sin m\theta_1 & \cos m\theta_1 & 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & \cos m\theta_2 & -\sin m\theta_2 & \dots & 0 & 0 \\ 0 & 0 & \sin m\theta_2 & \cos m\theta_2 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \dots & \cos m\theta_{d_k/2} & -\sin m\theta_{d_k/2} \\ 0 & 0 & 0 & 0 & \dots & \sin m\theta_{d_k/2} & \cos m\theta_{d_k/2} \end{pmatrix}$$

The θ_i parameters are fixed ahead of time and defined as below:

$$\Theta = \{\theta_i = 10000^{-2^{i-1}/d}, i \in [1, 2, \dots, d/2]\}$$

wrong

Rotary Position Embeddings (RoPE)

Q: Why does this slide have so many typos?

A: I'm not really sure. I very meticulously type up the latex for my slides myself and think carefully about all the things I put in them.

Rotary Position Embeddings (RoPE)

- Rotary position embeddings are a kind of **relative** position embeddings
- Key idea:
  - break each d-dimensional input vector into d/2 vectors of length 2
  - rotate each of the d/2 vectors by an amount scaled by m
  - m is the absolute position of the query or the key

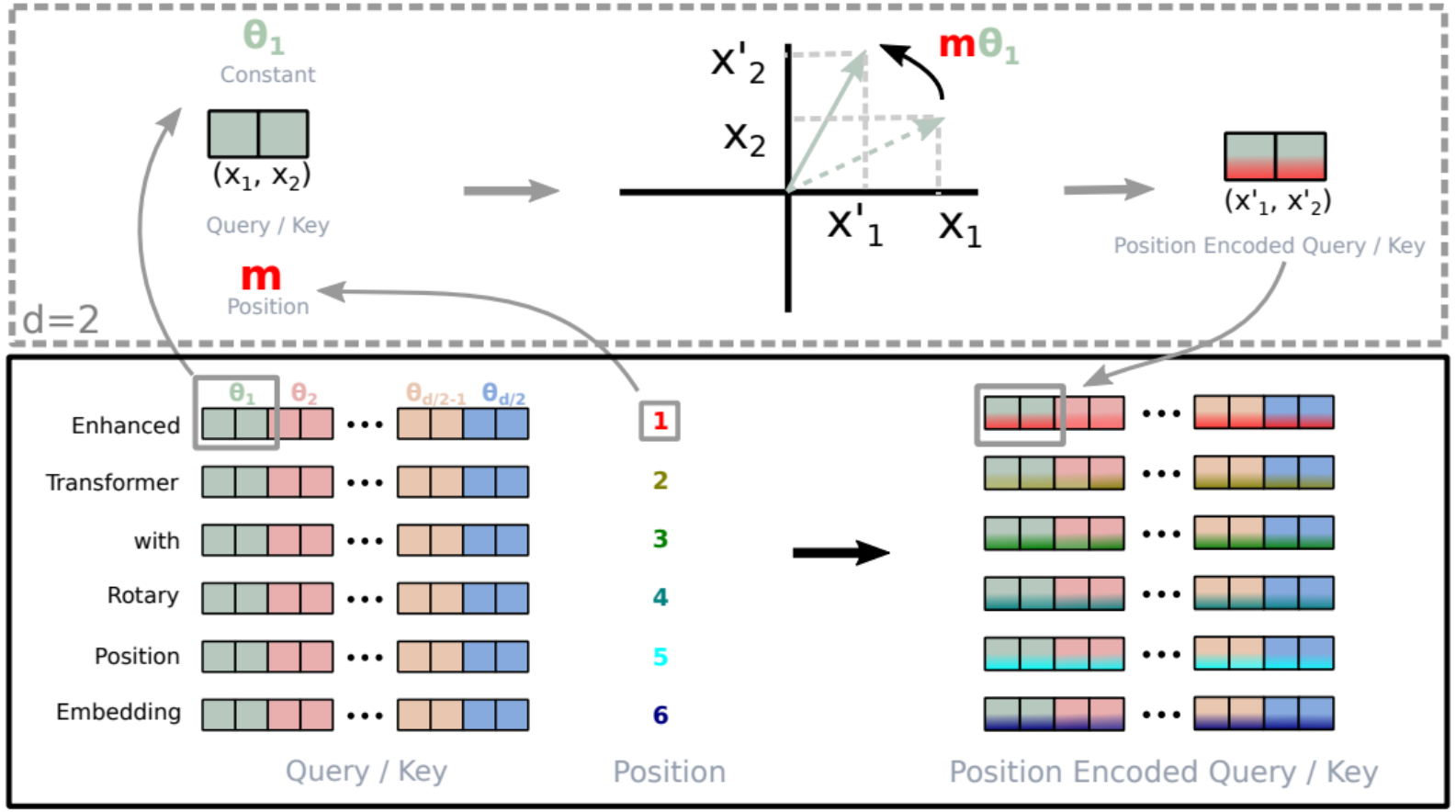


Figure from http://arxiv.org/abs/2104.09864

Rotary Position Embeddings (RoPE)

Standard attention:

$$\mathbf{q}_j = \mathbf{W}_q^T \mathbf{x}_j, \forall j$$
$$\mathbf{k}_j = \mathbf{W}_k^T \mathbf{x}_j, \forall j$$
$$s_{t,j} = \mathbf{k}_j^T \mathbf{q}_t / \sqrt{|\mathbf{k}|}, \forall j, t$$
$$\mathbf{a}_t = \text{softmax}(\mathbf{s}_t), \forall t$$

RoPE attention:

$$\mathbf{q}_j = \mathbf{W}_q^T \mathbf{x}_j, \forall j \qquad \mathbf{k}_j = \mathbf{W}_k^T \mathbf{x}_j, \forall j$$
$$\tilde{\mathbf{q}}_j = \mathbf{R}_{\Theta,j} \mathbf{q}_j \qquad \tilde{\mathbf{k}}_j = \mathbf{R}_{\Theta,j} \mathbf{k}_j$$
$$s_{t,j} = \tilde{\mathbf{k}}_j^T \tilde{\mathbf{q}}_t / \sqrt{d_k}, \forall j, t$$
$$\mathbf{a}_t = \text{softmax}(\mathbf{s}_t), \forall t$$

where $d = d_k/2$, $\mathbf{W}_k, \mathbf{W}_q \in \mathbb{R}^{d_{model} \times d_k}$.
For some fixed absolute position m, the rotary matrix $\mathbf{R}_{\Theta,m} \in \mathbb{R}^{d_k \times d_k}$ is given by:

$$R_{\Theta,m} = \begin{pmatrix} \cos m\theta_1 & -\sin m\theta_1 & 0 & 0 & \dots & 0 & 0 \\ \sin m\theta_1 & \cos m\theta_1 & 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & \cos m\theta_2 & -\sin m\theta_2 & \dots & 0 & 0 \\ 0 & 0 & \sin m\theta_2 & \cos m\theta_2 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \dots & \cos m\theta_{d_k/2} & -\sin m\theta_{d_k/2} \\ 0 & 0 & 0 & 0 & \dots & \sin m\theta_{d_k/2} & \cos m\theta_{d_k/2} \end{pmatrix}$$

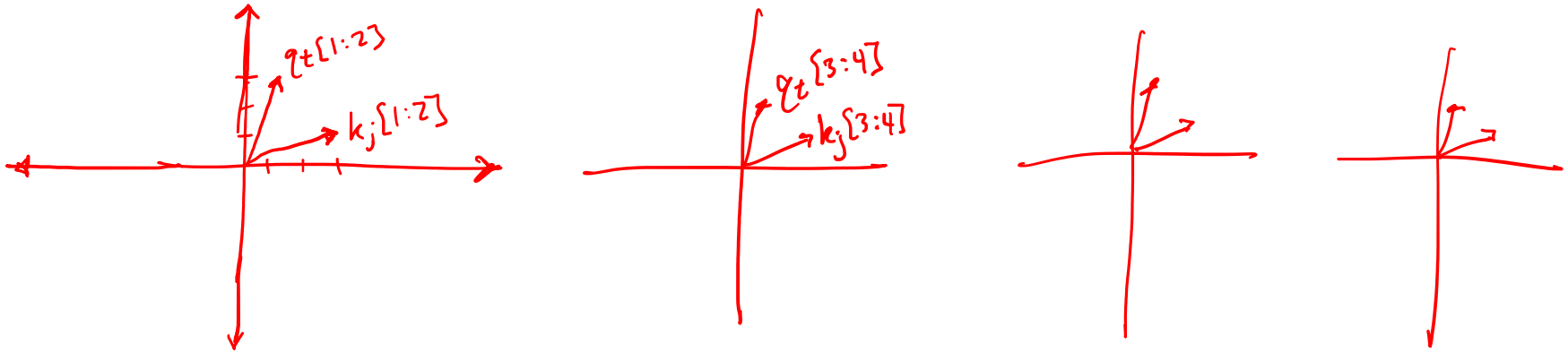
The θ_i parameters are fixed ahead of time and defined as below.

$$\Theta = \{\theta_i = 10000^{-2(i-1)/d}, i \in [1, 2, \dots, d/2]\}$$

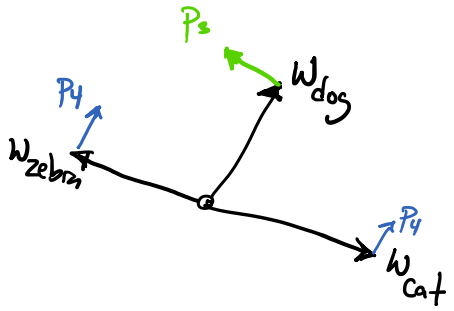
Rotary Position Embeddings (RoPE)

$q_t = W_q^T x_t = [1, 3, 1, 3, 1, 3, 1, 3]$

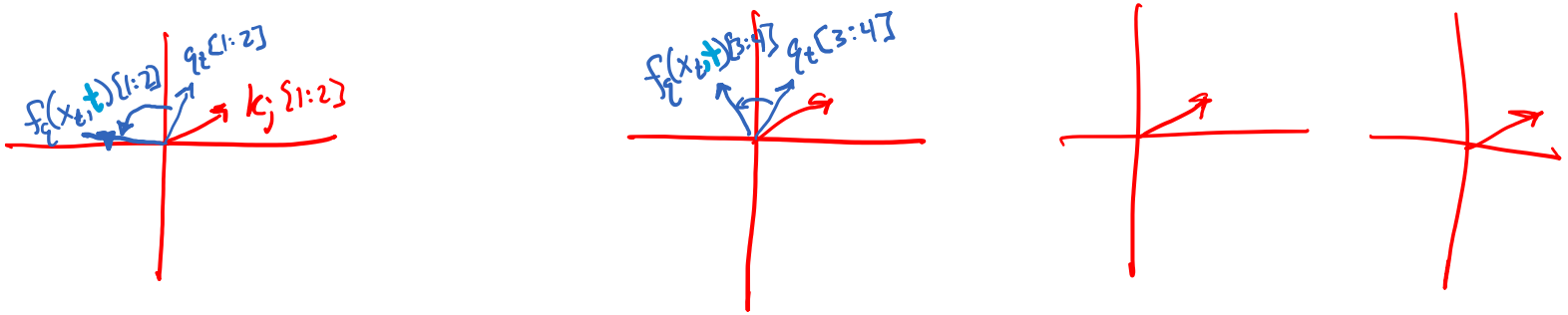
$k_j = W_k^T x_j = [3, 1, 3, 1, 3, 1, 3, 1]$



Absolute PE.



$f_q(x_t, t) = R_{\theta,t}\, q_t$



Rotary Position Embeddings (RoPE)

Standard attention:

$$\mathbf{q}_j = \mathbf{W}_q^T \mathbf{x}_j, \forall j$$
$$\mathbf{k}_j = \mathbf{W}_k^T \mathbf{x}_j, \forall j$$
$$s_{t,j} = \mathbf{k}_j^T \mathbf{q}_t / \sqrt{|\mathbf{k}|}, \forall j, t$$
$$\mathbf{a}_t = \text{softmax}(\mathbf{s}_t), \forall t$$

RoPE attention:

$$\mathbf{q}_j = \mathbf{W}_q^T \mathbf{x}_j, \forall j \qquad \mathbf{k}_j = \mathbf{W}_k^T \mathbf{x}_j, \forall j$$
$$\tilde{\mathbf{q}}_j = \mathbf{R}_{\Theta,j} \mathbf{q}_j \qquad \tilde{\mathbf{k}}_j = \mathbf{R}_{\Theta,j} \mathbf{k}_j$$
$$s_{t,j} = \tilde{\mathbf{k}}_j^T \tilde{\mathbf{q}}_t / \sqrt{d_k}, \forall j, t$$
$$\mathbf{a}_t = \text{softmax}(\mathbf{s}_t), \forall t$$

Because of the block sparse pattern in $\mathbf{R}_{\theta,m}$, we can efficiently compute the matrix-vector product of $\mathbf{R}_{\theta,m}$ with some arbitrary vector $\mathbf{y}$ in a more efficient manner:

$$\mathbf{R}_{\Theta,m}\mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ \vdots \\ y_{d-1} \\ y_d \end{pmatrix} \otimes \begin{pmatrix} \cos m\theta_1 \\ \cos m\theta_1 \\ \cos m\theta_2 \\ \cos m\theta_2 \\ \vdots \\ \cos m\theta_{d/2} \\ \cos m\theta_{d/2} \end{pmatrix} + \begin{pmatrix} -y_2 \\ y_1 \\ -y_4 \\ y_3 \\ \vdots \\ -y_d \\ y_{d-1} \end{pmatrix} \otimes \begin{pmatrix} \sin m\theta_1 \\ \sin m\theta_1 \\ \sin m\theta_2 \\ \sin m\theta_2 \\ \vdots \\ \sin m\theta_{d/2} \\ \sin m\theta_{d/2} \end{pmatrix}$$

Matrix Version of RoPE

RoPE attention:

$$\mathbf{q}_j = \mathbf{W}_q^T \mathbf{x}_j, \forall j \qquad \mathbf{k}_j = \mathbf{W}_k^T \mathbf{x}_j, \forall j$$

$$\tilde{\mathbf{q}}_j = \mathbf{R}_{\Theta,j} \mathbf{q}_j \qquad \tilde{\mathbf{k}}_j = \mathbf{R}_{\Theta,j} \mathbf{k}_j$$

$$s_{t,j} = \tilde{\mathbf{k}}_j^T \tilde{\mathbf{q}}_t / \sqrt{d_k}, \forall j, t$$

$$\mathbf{a}_t = \text{softmax}(\mathbf{s}_t), \forall t$$

Matrix Version:

$$\mathbf{Q} = \mathbf{X}\mathbf{W}_q \qquad \mathbf{K} = \mathbf{X}\mathbf{W}_k$$

$$\tilde{\mathbf{Q}} = g(\mathbf{Q}; \Theta) \qquad \tilde{\mathbf{K}} = g(\mathbf{K}; \Theta)$$

$$\mathbf{S} = \tilde{\mathbf{Q}}\tilde{\mathbf{K}}^T / \sqrt{d_k}$$

$$\mathbf{A} = \text{softmax}(\mathbf{S})$$

Goal: to construct a new matrix $\tilde{\mathbf{Y}} = g(\mathbf{Y}; \Theta)$ such that $\tilde{\mathbf{Y}}_{m,.} = \mathbf{R}_{\Theta,m} \mathbf{y}_m$

$$\mathbf{C} = \left[\begin{array}{ccc|ccc} 1\theta_1 & \cdots & 1\theta_{\frac{d}{2}} & 1\theta_1 & \cdots & 1\theta_{\frac{d}{2}} \\ \vdots & & \vdots & \vdots & & \vdots \\ N\theta_1 & \cdots & N\theta_{\frac{d}{2}} & N\theta_1 & \cdots & N\theta_{\frac{d}{2}} \end{array}\right]$$

$$\begin{aligned} \tilde{\mathbf{Y}} &= g(\mathbf{Y}; \Theta) \\ &= \left[\begin{array}{c|c} \mathbf{Y}_{.,1:d/2} & \mathbf{Y}_{.,d/2+1:d} \end{array}\right] \otimes \cos(\mathbf{C}) \\ &\quad + \left[\begin{array}{c|c} -\mathbf{Y}_{.,d/2+1:d} & \mathbf{Y}_{.,1:d/2} \end{array}\right] \otimes \sin(\mathbf{C}) \end{aligned}$$

Matrix Version of RoPE

Q: Is this slide correct?

A: I'm really not sure.

But I did write it myself!

$$\mathbf{k}_j = \mathbf{W}_k^T \mathbf{x}_j, \forall j$$

$$\tilde{\mathbf{k}}_j = \mathbf{R}_{\Theta,j} \mathbf{k}_j$$

Matrix Version:

$$\mathbf{Q} = \mathbf{X}\mathbf{W}_q \qquad \mathbf{K} = \mathbf{X}\mathbf{W}_k$$

$$\tilde{\mathbf{Q}} = g(\mathbf{Q}; \Theta) \qquad \tilde{\mathbf{K}} = g(\mathbf{K}; \Theta)$$

$$\mathbf{S} = \tilde{\mathbf{Q}}\tilde{\mathbf{K}}^T / \sqrt{d_k}$$

$$\mathbf{A} = \text{softmax}(\mathbf{S})$$

Goal: to construct a new matrix $\tilde{\mathbf{Y}} = g(\mathbf{Y}; \Theta)$ such that $\tilde{\mathbf{Y}}_{m,.} = \mathbf{R}_{\Theta,m}\mathbf{y}_m$

$$\mathbf{C} = \left[\begin{array}{ccc|ccc} 1\theta_1 & \cdots & 1\theta_{\frac{d}{2}} & 1\theta_1 & \cdots & 1\theta_{\frac{d}{2}} \\ \vdots & & \vdots & \vdots & & \vdots \\ N\theta_1 & \cdots & N\theta_{\frac{d}{2}} & N\theta_1 & \cdots & N\theta_{\frac{d}{2}} \end{array}\right]$$

$$\begin{aligned} \tilde{\mathbf{Y}} &= g(\mathbf{Y}; \Theta) \\ &= \left[\begin{array}{c|c} \mathbf{Y}_{\cdot,1:d/2} & \mathbf{Y}_{\cdot,d/2+1:d} \end{array}\right] \otimes \cos(\mathbf{C}) \\ &\quad + \left[\begin{array}{c|c} -\mathbf{Y}_{\cdot,d/2+1:d} & \mathbf{Y}_{\cdot,1:d/2} \end{array}\right] \otimes \sin(\mathbf{C}) \end{aligned}$$

COMPUTER VISION

Common Tasks in Computer Vision

1. Image Classification
2. Image Classification + Localization
3. Human Pose Estimation
4. Semantic Segmentation
5. Object Detection
6. Instance Segmentation
7. Image Captioning
8. Image Generation

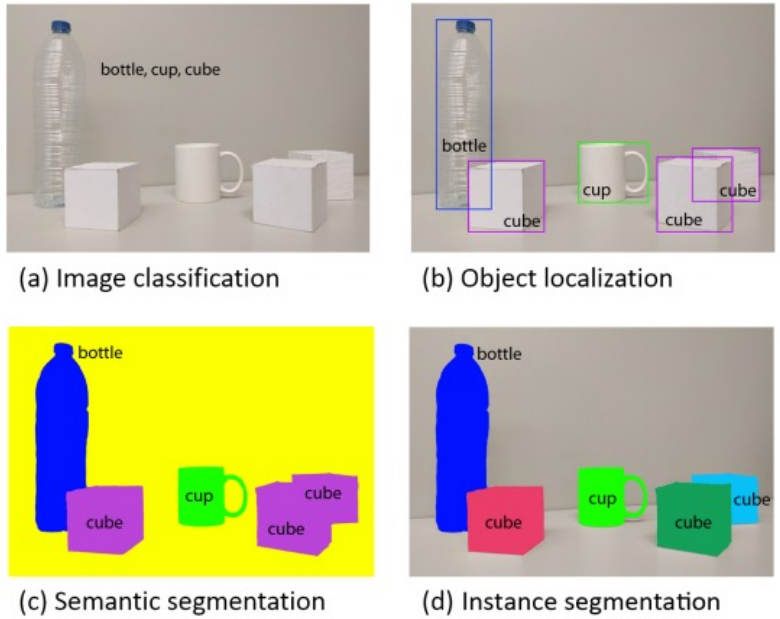
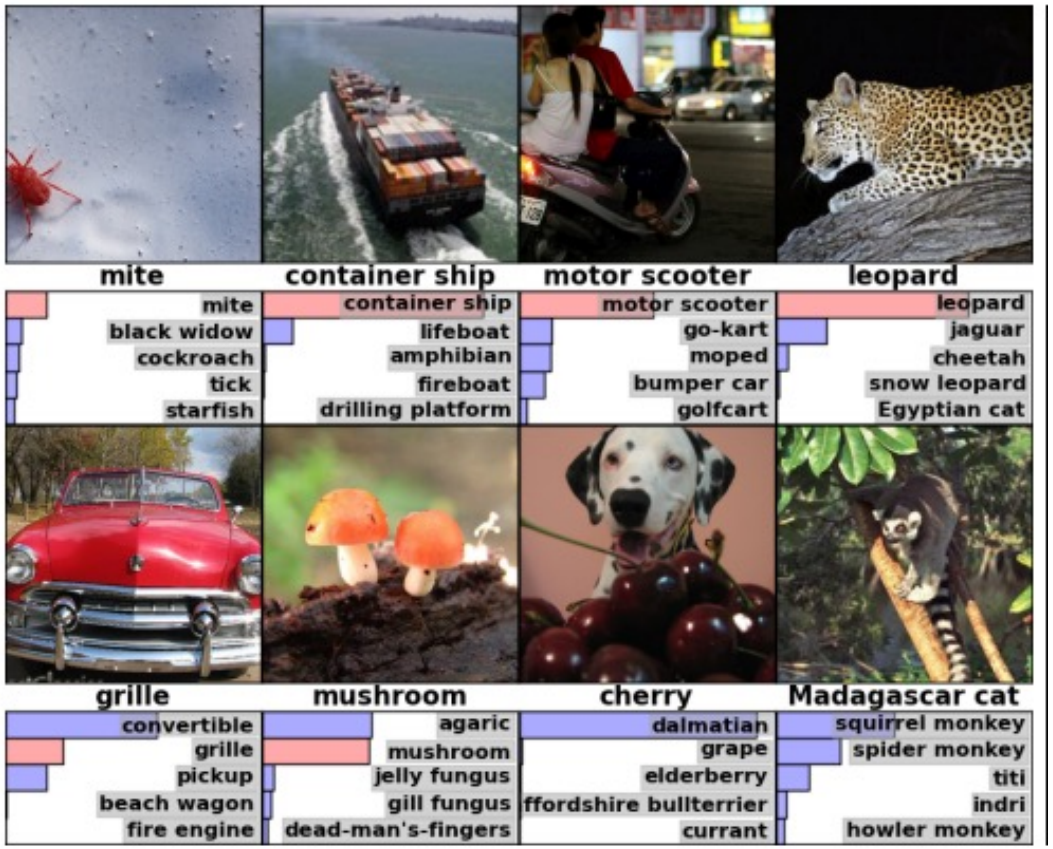


Figure from https://arxiv.org/pdf/1704.06857.pdf

Image Classification

- Given an image, predict a single label
- A multi-class classification problem



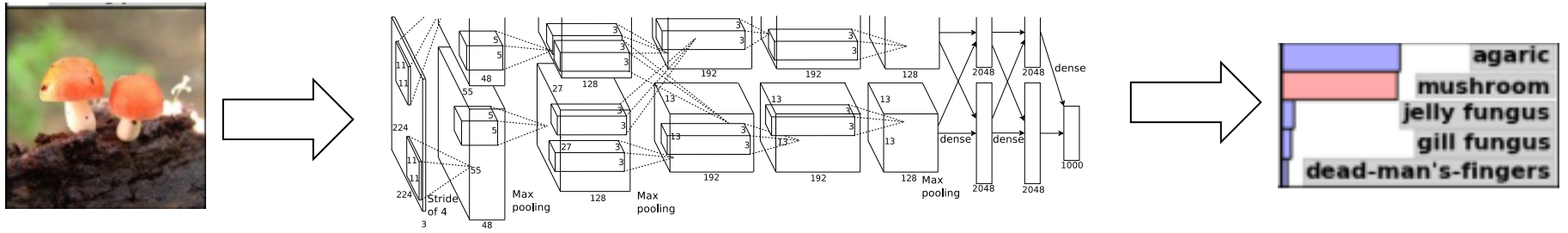


Figure from https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf

Image Classification + Localization

- Given an image, predict a single label and a bounding box for the object
- Bounding box is represented as (x, y, h, w), position (x,y) and height/width (h,w)

Figure from https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=5459257

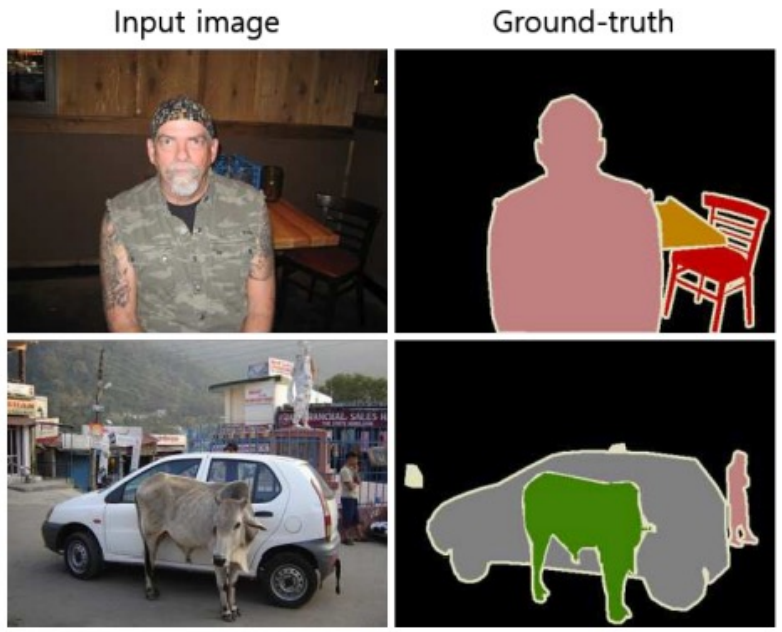
Human Pose Estimation

- Given an image of a human, predict the position of several keypoints (left hand, right hand, left elbow, …, right foot)
- This is a multiple regression problem, where each keypoint has a corresponding position (x_i,y_i)

Figure from https://openaccess.thecvf.com/content_cvpr_2014/papers/Toshev_DeepPose_Human_Pose_2014_CVPR_paper.pdf

Semantic Segmentation

- Given an image, predict a label for every pixel in the image
- Not merely a classification problem, because there are strong correlations between pixel-specific labels



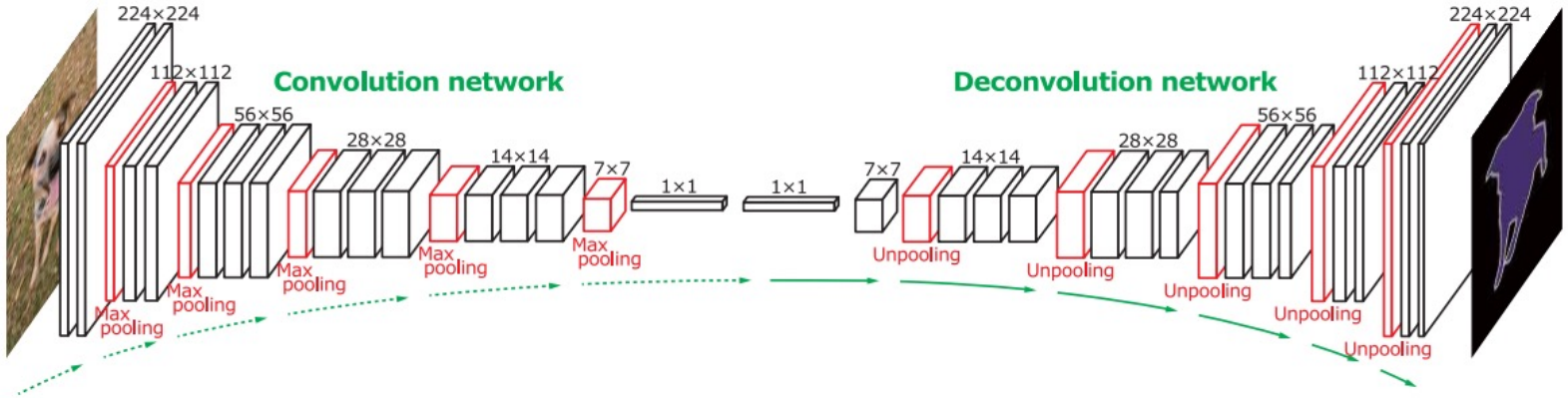


Figure from https://openaccess.thecvf.com/content_iccv_2015/papers/Noh_Learning_Deconvolution_Network_ICCV_2015_paper.pdf

Object Detection

- Given an image, for each object predict a bounding box and a label (x,y,w,h,l)
- Example: R-CNN
  - (x=110, y=13, w=50, h=72, l=person)
  - (x=90, y=55, w=81, h=87, l=horse)
  - (x=421, y=533, w=24, h=30, l=chair)
  - (x=2, y=25, w=51, h=121, l=gate)

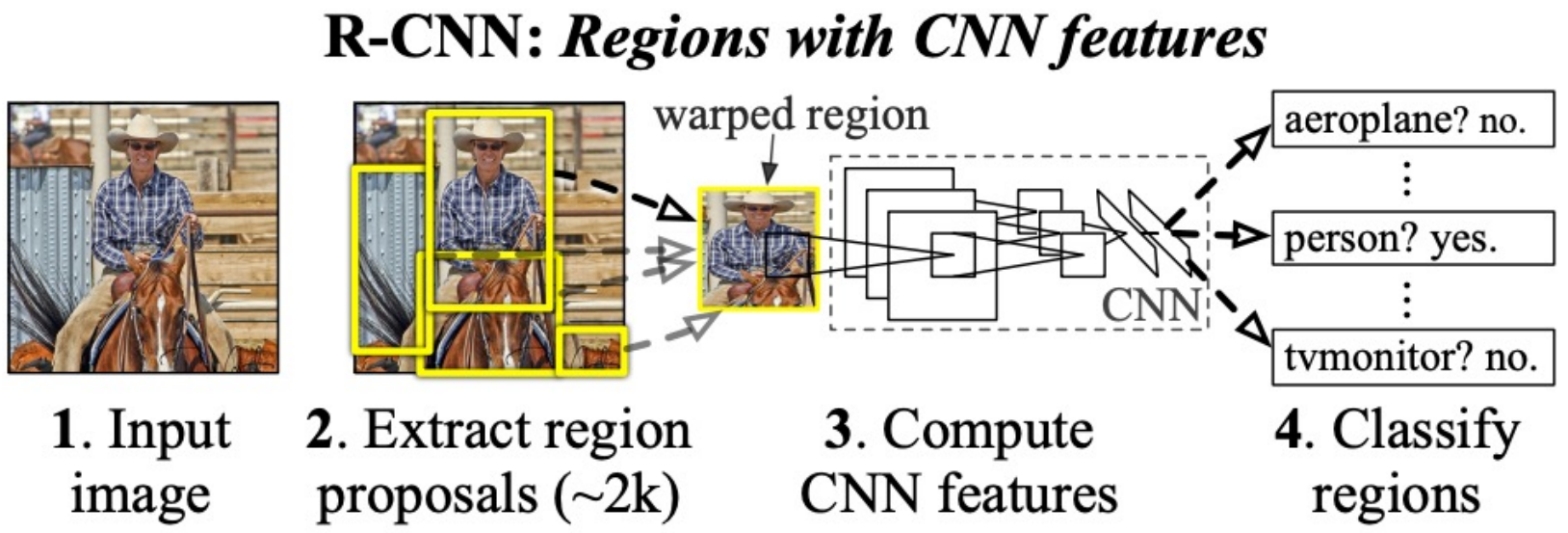


Figure from https://openaccess.thecvf.com/content_cvpr_2014/papers/Girshick_Rich_Feature_Hierarchies_2014_CVPR_paper.pdf

Instance Segmentation

- Predict per-pixel labels as in semantic segmentation, but differentiate between different instances of the same label
- *Example*: if there are two people in the image, one person should be labeled **person-1** and one should be labeled **person-2**

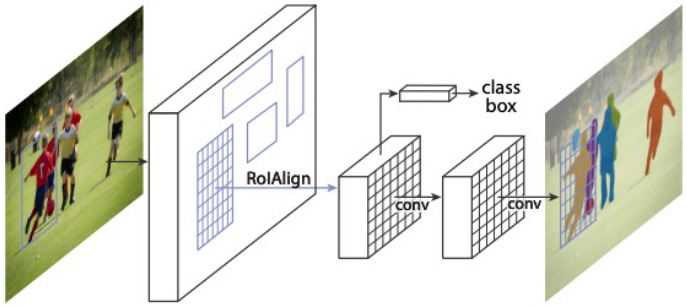
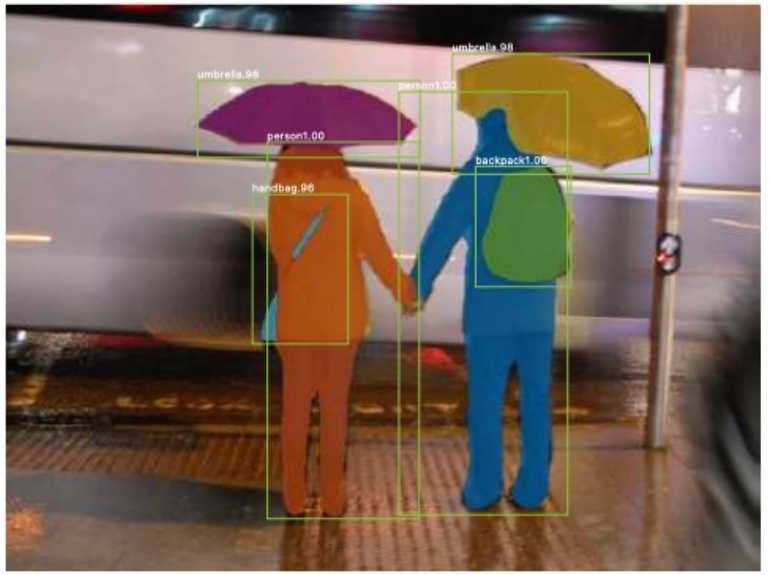


Figure 1. The **Mask R-CNN** framework for instance segmentation.



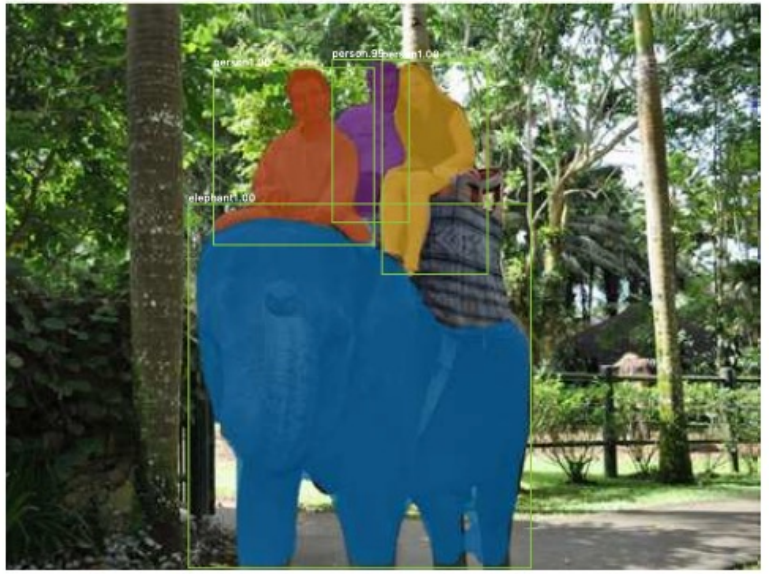
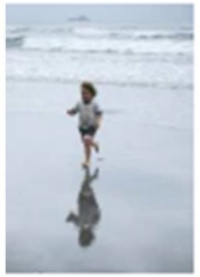


Figure from https://openaccess.thecvf.com/content_ICCV_2017/papers/He_Mask_R-CNN_ICCV_2017_paper.pdf

Image Captioning



Ground Truth Caption: A little boy runs away from the approaching waves of the ocean.

Generated Caption: A young boy is running on the beach.

Ground Truth Caption: A brunette girl wearing sunglasses and a yellow shirt.

Generated Caption: A woman in a black shirt and sunglasses smiles.

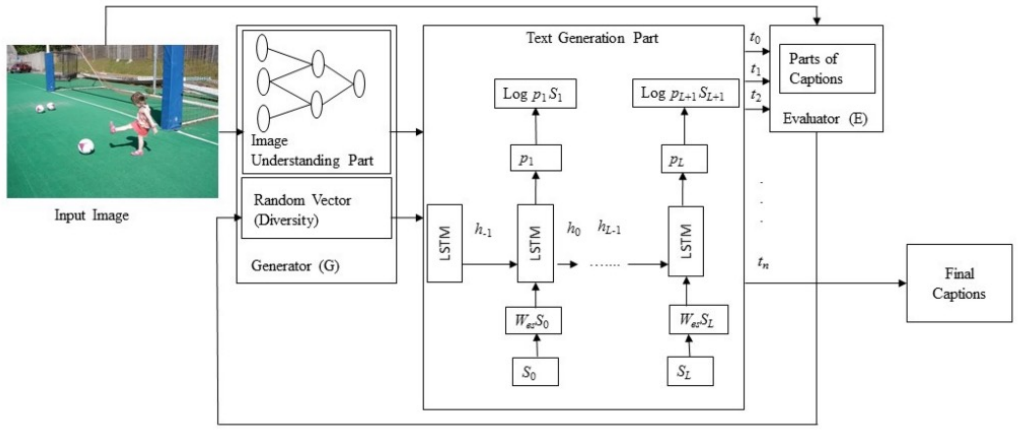


Fig. 3. A block diagram of other deep-learning-based captioning.

- Take an image as input, and generate a sentence describing it as output (i.e. the caption)
- Typical methods include a deep CNN/transformer and a RNN-like language model
- (The task of *Dense Captioning* is to generate one caption per bounding box)

Figure from https://dl.acm.org/doi/pdf/10.1145/3295748

Image Captioning

Table 1. An Overview of the Deep-Learning-Based Approaches for Image Captioning

Reference	Image Encoder	Language Model	Category
Kiros et al. 2014 [69]	AlexNet	LBL	MS, SL, WS, EDA
Kiros et al. 2014 [70]	AlexNet, VGGNet	1. LSTM 2. SC-NLM	MS, SL, WS, EDA
Mao et al. 2014 [95]	AlexNet	RNN	MS, SL, WS
Karpathy et al. 2014 [66]	AlexNet	DTR	MS, SL, WS, EDA
Mao et al. 2015 [94]	AlexNet, VGGNet	RNN	MS, SL, WS
Chen et al. 2015 [23]	VGGNet	RNN	VS, SL, WS, EDA
Fang et al. 2015 [33]	AlexNet, VGGNet	MELM	VS, SL, WS, CA
Jia et al. 2015 [59]	VGGNet	LSTM	VS, SL, WS, EDA
Karpathy et al. 2015 [65]	VGGNet	RNN	MS, SL, WS, EDA
Vinyals et al. 2015 [142]	GoogLeNet	LSTM	VS, SL, WS, EDA
Xu et al. 2015 [152]	AlexNet	LSTM	VS, SL, WS, EDA, AB
Jin et al. 2015 [61]	VGGNet	LSTM	VS, SL, WS, EDA, AB
Wu et al. 2016 [151]	VGGNet	LSTM	VS, SL, WS, EDA, AB
Sugano et at. 2016 [129]	VGGNet	LSTM	VS, SL, WS, EDA, AB
Mathews et al. 2016 [97]	GoogLeNet	LSTM	VS, SL, WS, EDA, SC
Wang et al. 2016 [144]	AlexNet, VGGNet	LSTM	VS, SL, WS, EDA
Johnson et al. 2016 [62]	VGGNet	LSTM	VS, SL, DC, EDA
Mao et al. 2016 [92]	VGGNet	LSTM	VS, SL, WS, EDA
Wang et al. 2016 [146]	VGGNet	LSTM	VS, SL, WS, CA
Tran et al. 2016 [135]	ResNet	MELM	VS, SL, WS, CA
Ma et al. 2016 [90]	AlexNet	LSTM	VS, SL, WS, CA
You et al. 2016 [156]	GoogLeNet	RNN	VS, SL, WS, EDA, SCB
Yang et al. 2016 [153]	VGGNet	LSTM	VS, SL, DC, EDA
Anne et al. 2016 [6]	VGGNet	LSTM	VS, SL, WS, CA, NOB
Yao et al. 2017 [155]	GoogLeNet	LSTM	VS, SL, WS, EDA, SCB
Lu et al. 2017 [88]	ResNet	LSTM	VS, SL, WS, EDA, AB
Chen et al. 2017 [21]	VGGNet, ResNet	LSTM	VS, SL, WS, EDA, AB
Gan et al. 2017 [41]	ResNet	LSTM	VS, SL, WS, CA, SCB
Pedersoli et al. 2017 [112]	VGGNet	RNN	VS, SL, WS, EDA, AB
Ren et al. 2017 [119]	VGGNet	LSTM	VS, ODL, WS, EDA
Park et al. 2017 [111]	ResNet	LSTM	VS, SL, WS, EDA, AB
Wang et al. 2017 [148]	ResNet	LSTM	VS, SL, WS, EDA
Tavakoli et al. 2017 [134]	VGGNet	LSTM	VS, SL, WS, EDA, AB
Liu et al. 2017 [84]	VGGNet	LSTM	VS, SL, WS, EDA, AB
Gan et al. 2017 [39]	ResNet	LSTM	VS, SL, WS, EDA, SC
Dai et al. 2017 [26]	VGGNet	LSTM	VS, ODL, WS, EDA
Shetty et al. 2017 [126]	GoogLeNet	LSTM	VS, ODL, WS, EDA
Liu et al. 2017 [85]	Inception-V3	LSTM	VS, ODL, WS, EDA
Gu et al. 2017 [51]	VGGNet	1. Language CNN 2. LSTM	VS, SL, WS, EDA
Yao et al. 2017 [154]	VGGNet	LSTM	VS, SL, WS, CA, NOB

(Continued)

- Take an image as input, and generate a sentence describing it as output (i.e. the caption)
- Typical methods include a deep CNN/transformer and a RNN-like language model
- (The task of *Dense Captioning* is to generate one caption per bounding box)

Table from https://dl.acm.org/doi/pdf/10.1145/3295748

Medical Image Analysis

Notice that **most** of these tasks are structured prediction problems, not merely classification

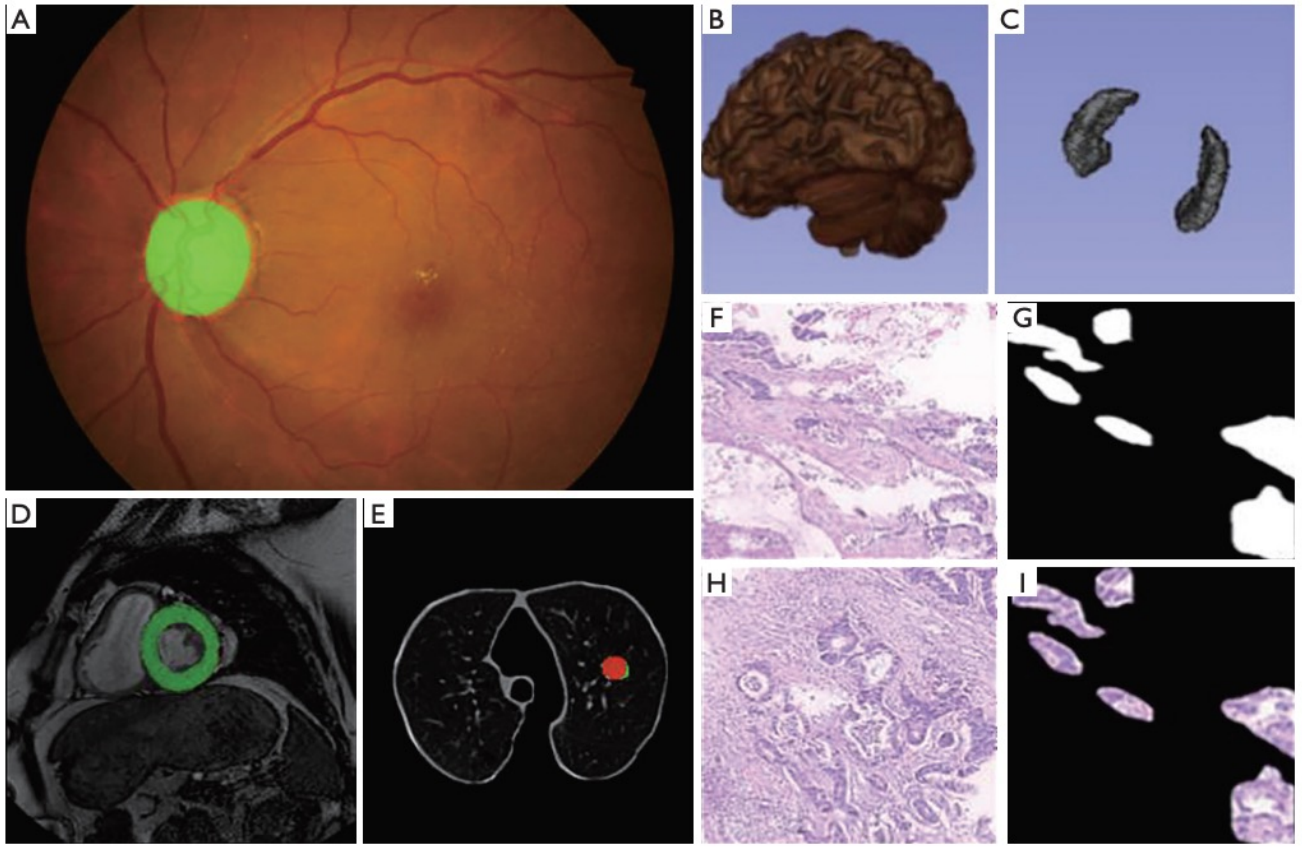


Figure 2 Deep learning application in medical image analysis. (A) Fundus detection; (B,C) hippocampus segmentation; (D) left ventricular segmentation; (E) pulmonary nodule classification; (F,G,H,I) gastric cancer pathology segmentation. The staining method is H&E, and the magnification is ×40.

Figure from https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7327346/

(aka. BERT-style models)

TRANSFORMER (ENCODER ONLY VERSION)

Autoregressive Language Model

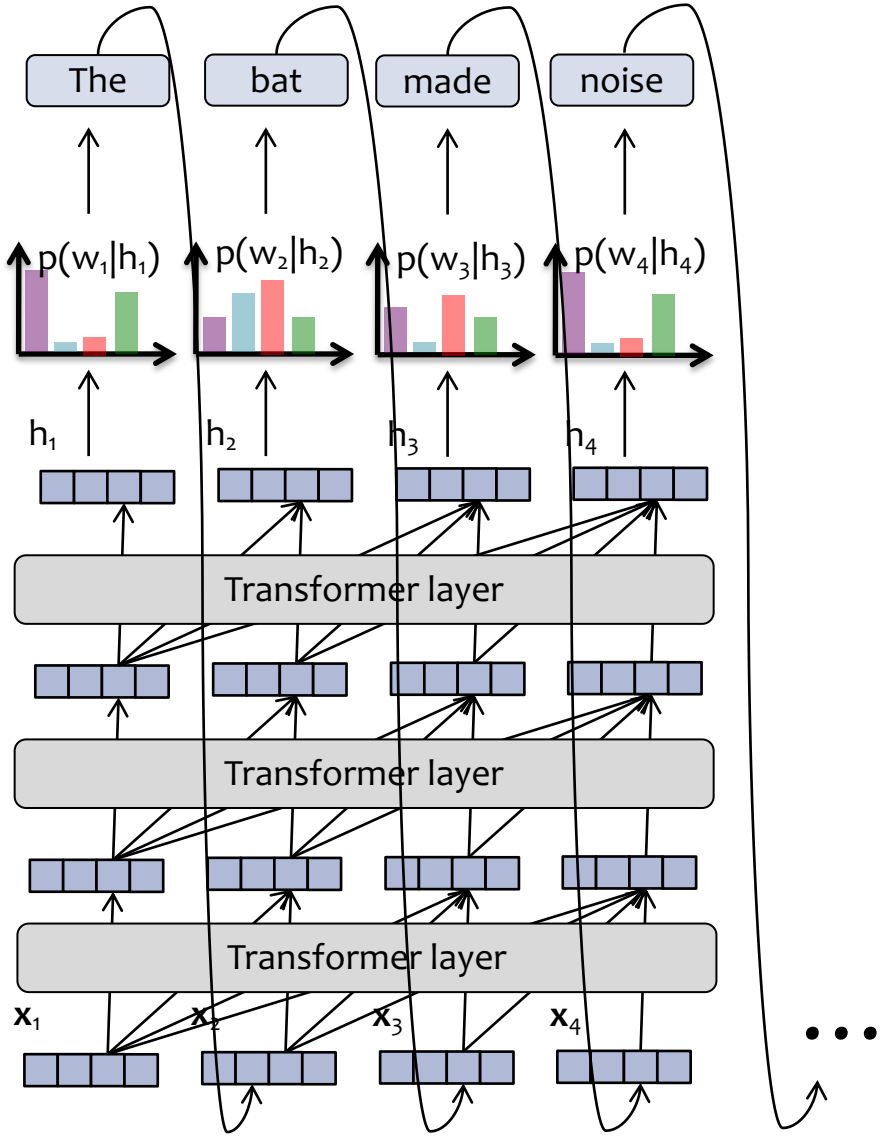
Definition: An *autoregressive language model* defines a probability distribution over sequences $\mathbf{x}_{1:T}$ of the form:

$$p(\mathbf{x}_{1:T}) = \prod_{t=1}^{T} p(x_t \mid x_1, \ldots, x_{t-1})$$

Decoder-only Transformer

Also called a

Transformer language model



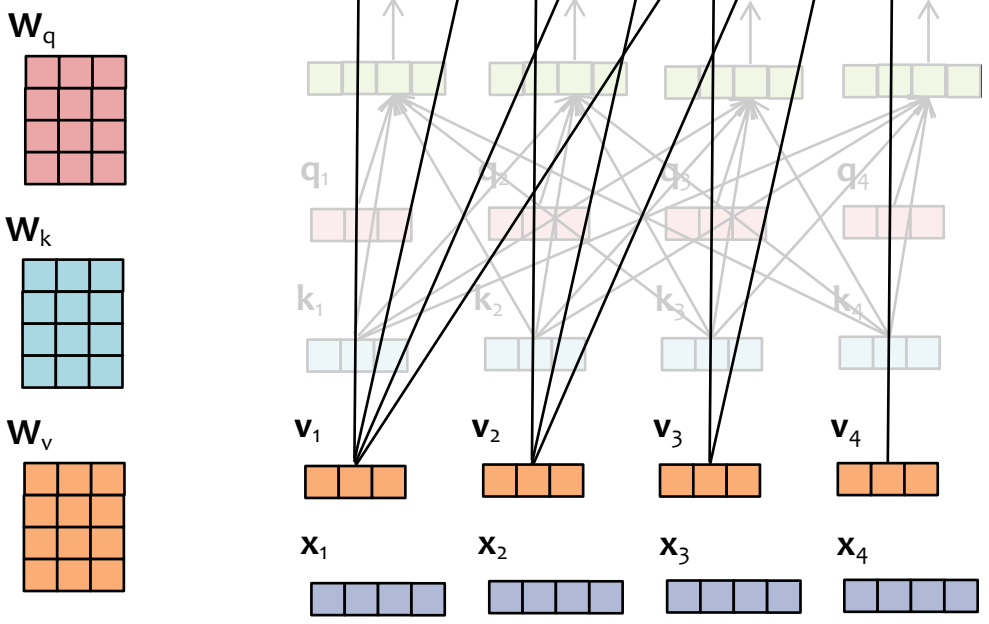
Each layer of a Transformer LM consists of several **sublayers:**

1. causal attention
2. feed-forward neural network
3. layer normalization
4. residual connections

Each hidden vector looks back at the hidden vectors of the **current and previous timesteps in the previous layer.**

The language model part is just like an RNN-LM.

Single-Headed (Causal) Attention



$$\mathbf{X}' = \mathbf{AV} = \text{softmax}(\mathbf{QK}^T/\sqrt{d_k} + \mathbf{M})\mathbf{V}$$

$$\mathbf{A}_{\text{causal}} = \text{softmax}(\mathbf{S} + \mathbf{M})$$

$$\mathbf{S} = \mathbf{QK}^T/\sqrt{d_k}$$

$$\mathbf{Q} = \mathbf{XW}_q$$

$$\mathbf{K} = \mathbf{XW}_k$$

$$\mathbf{V} = \mathbf{XW}_v$$

$$\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_4]^T$$

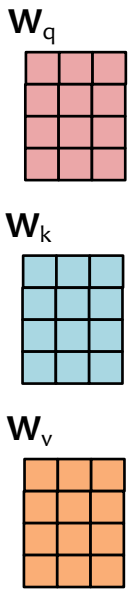
In practice, the attention weights are computed for all time steps T, then we mask out (by setting to –inf) all the inputs to the softmax that are for the timesteps to the right of the query

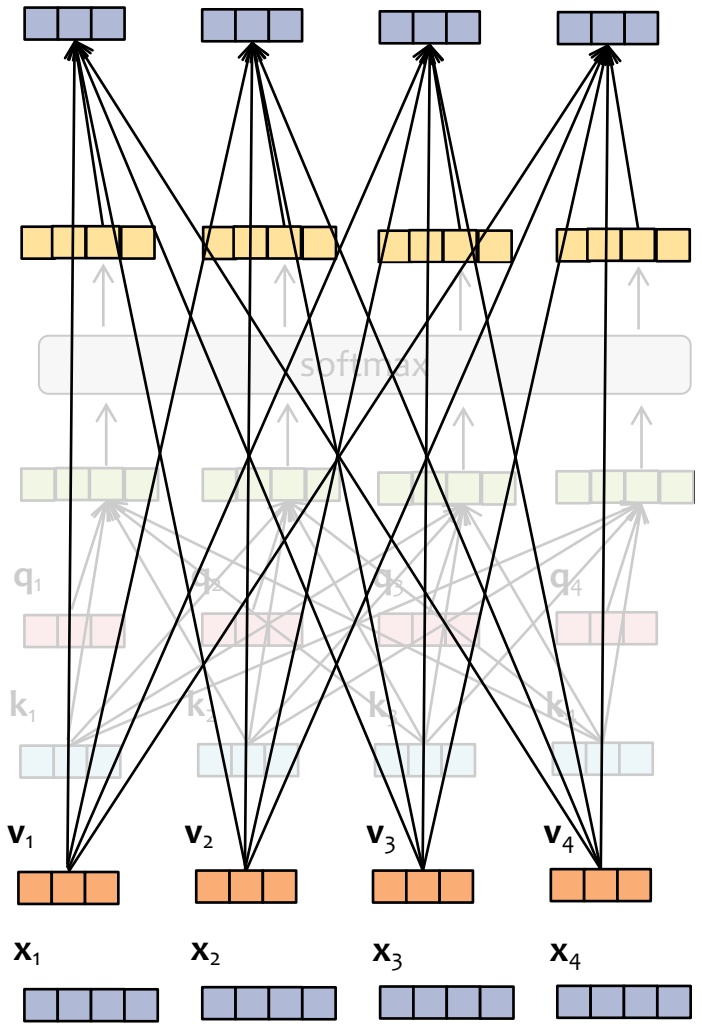
$$\mathbf{M} = \begin{bmatrix} 0 & -\infty & -\infty & -\infty \\ 0 & 0 & -\infty & -\infty \\ 0 & 0 & 0 & -\infty \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Single-Headed Attention

So what is this model?

- Notice that each token can attend to all other tokens on the left and right
- It is **not** an autoregressive language model





$$\mathbf{X}' = \mathbf{A}\mathbf{V} = \text{softmax}(\mathbf{Q}\mathbf{K}^T/\sqrt{d_k})\mathbf{V}$$

$$\mathbf{A} = \text{softmax}(\mathbf{S})$$

$$\mathbf{S} = \mathbf{Q}\mathbf{K}^T/\sqrt{d_k}$$

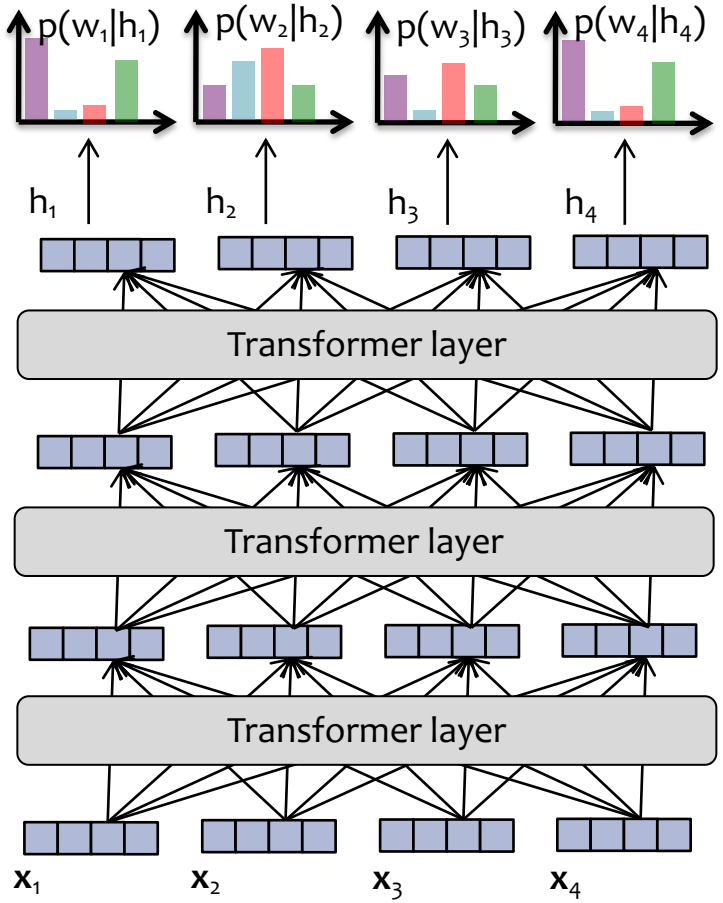
$$\mathbf{Q} = \mathbf{X}\mathbf{W}_q$$

$$\mathbf{K} = \mathbf{X}\mathbf{W}_k$$

$$\mathbf{V} = \mathbf{X}\mathbf{W}_v$$

$$\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_4]^T$$

Encoder-only Transformer



...

Each layer of an encoder-only Transformer consists of several **sublayers:**

1. non-causal attention
2. feed-forward neural network
3. layer normalization
4. residual connections

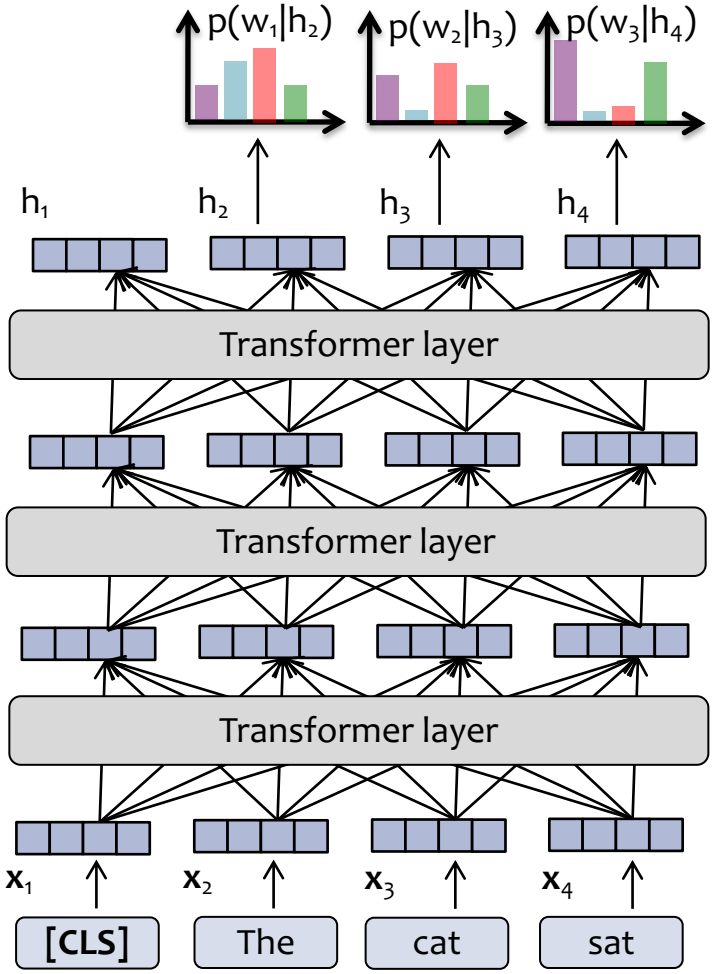
Each hidden vector looks at the the hidden vectors of **all timesteps in the previous layer.**

Encoder-only Transformer

BERT popularized this encoder-only Transformer architecture and style of pretraining

MLM Pretraining:

- Rather than trying to predict the next word from the previous ones…
- …mask out a word (or a few words) and predict the missing words from the remaining ones



…

Each layer of an encoder-only Transformer consists of several **sublayers:**

1. non-causal attention
2. feed-forward neural network
3. layer normalization
4. residual connections

Each hidden vector looks at the the hidden vectors of **all timesteps in the previous layer.**

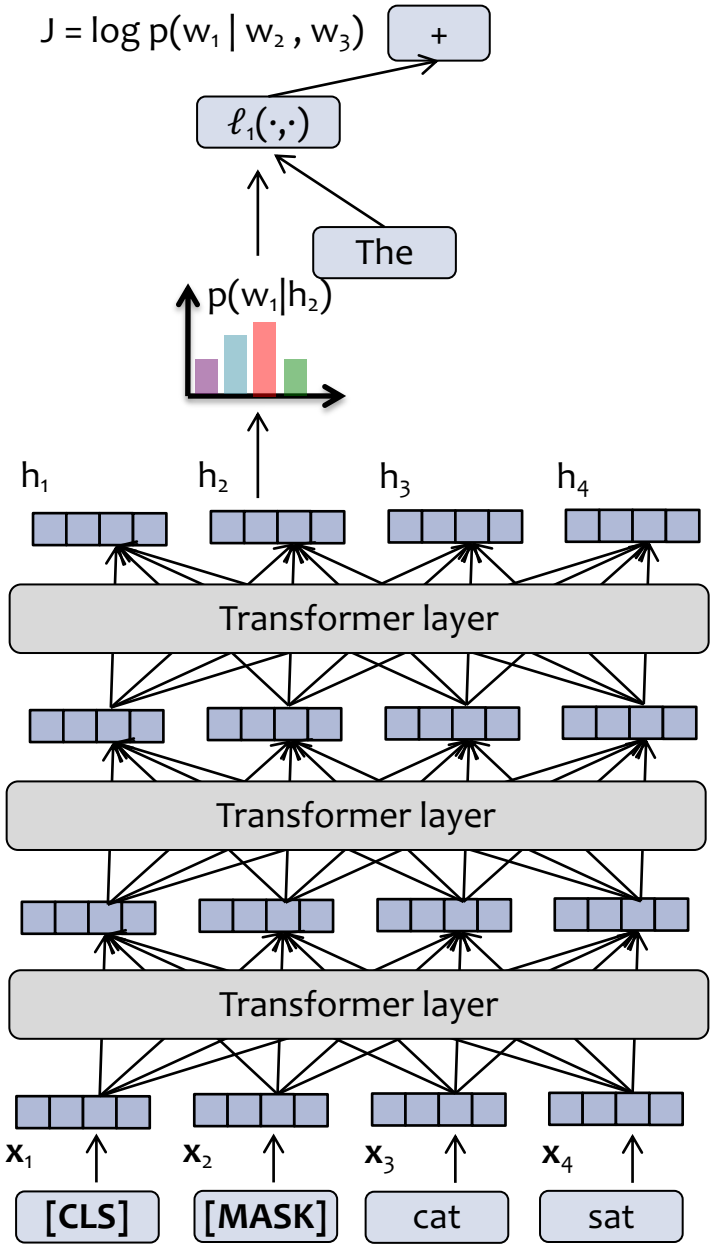
The distribution over words is used for **masked language model (MLM) pre-training** (cf. BERT)

Encoder-only Transformer

BERT popularized this encoder-only Transformer architecture and style of pretraining

MLM Pretraining:

- Rather than trying to predict the next word from the previous ones…
- …mask out a word (or a few words) and predict the missing words from the remaining ones



…

Each layer of an encoder-only Transformer consists of several **sublayers:**

1. non-causal attention
2. feed-forward neural network
3. layer normalization
4. residual connections

Each hidden vector looks at the the hidden vectors of **all timesteps in the previous layer.**

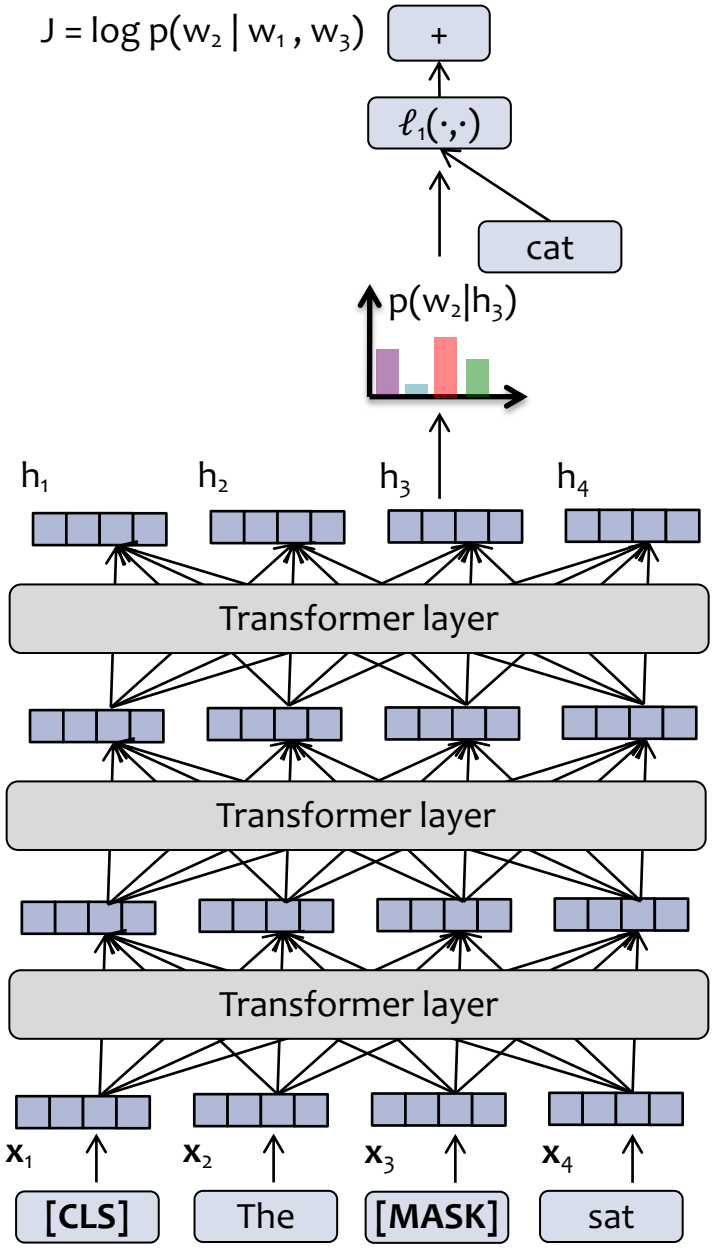
The distribution over words is used for **masked language model (MLM) pre-training** (cf. BERT)

Encoder-only Transformer

BERT popularized this encoder-only Transformer architecture and style of pretraining

MLM Pretraining:

- Rather than trying to predict the next word from the previous ones…
- …mask out a word (or a few words) and predict the missing words from the remaining ones



…

Each layer of an encoder-only Transformer consists of several **sublayers:**

1. non-causal attention
2. feed-forward neural network
3. layer normalization
4. residual connections

Each hidden vector looks at the the hidden vectors of **all timesteps in the previous layer.**

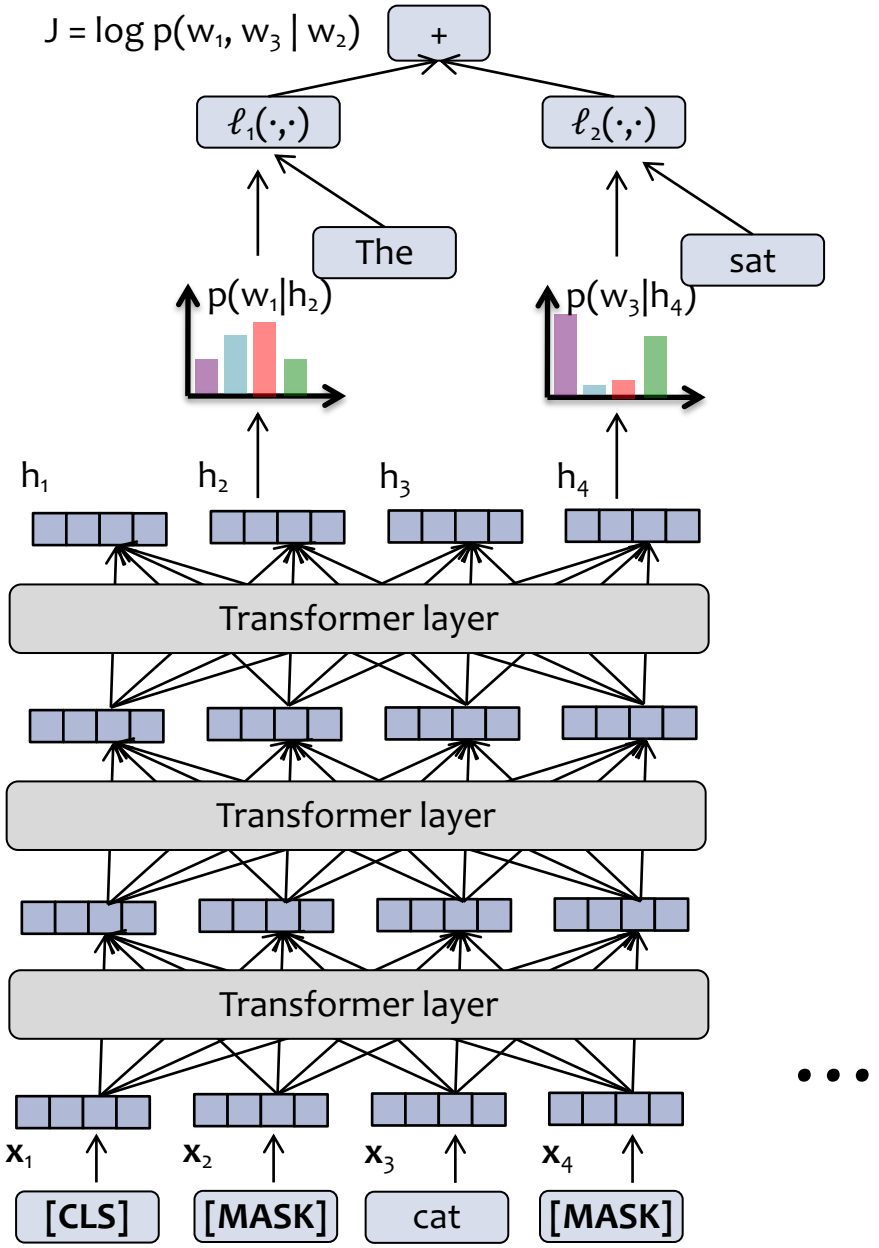
The distribution over words is used for **masked language model (MLM) pre-training** (cf. BERT)

Encoder-only Transformer

BERT popularized this encoder-only Transformer architecture and style of pretraining

MLM Pretraining:

- Rather than trying to predict the next word from the previous ones…
- …mask out a word (or a few words) and predict the missing words from the remaining ones



…

Each layer of an encoder-only Transformer consists of several **sublayers:**

1. non-causal attention
2. feed-forward neural network
3. layer normalization
4. residual connections

Each hidden vector looks at the the hidden vectors of **all timesteps in the previous layer.**

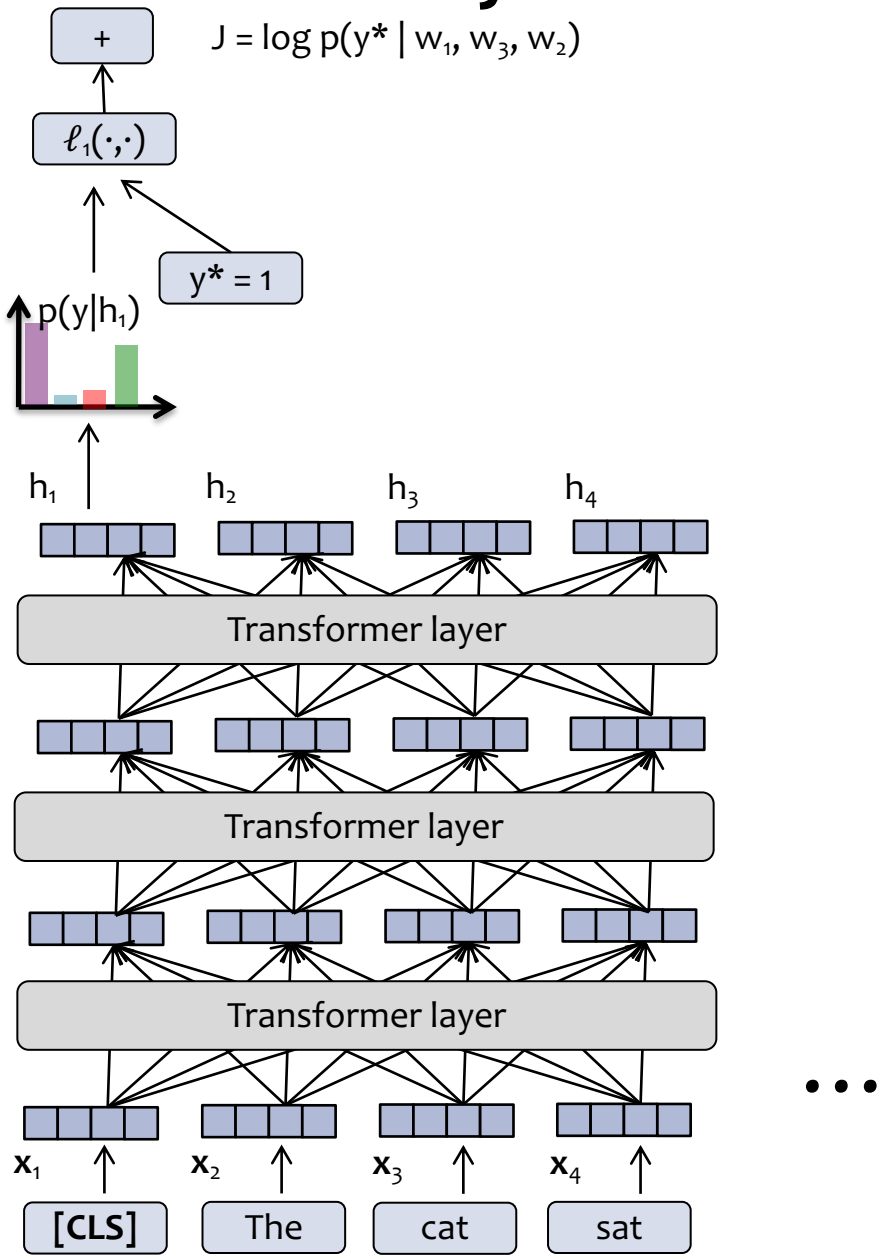
The distribution over words is used for **masked language model (MLM) pre-training** (cf. BERT)

Encoder-only Transformer

BERT popularized this encoder-only Transformer architecture and style of pretraining

MLM Pretraining:

- Rather than trying to predict the next word from the previous ones…
- …mask out a word (or a few words) and predict the missing words from the remaining ones



…

Supervised Fine-tuning:

- How to fine-tune this model for a classification task?
- Predict the class label given the embedding for the special [CLS] token
- Fine-tune the model to be good at classification in this way

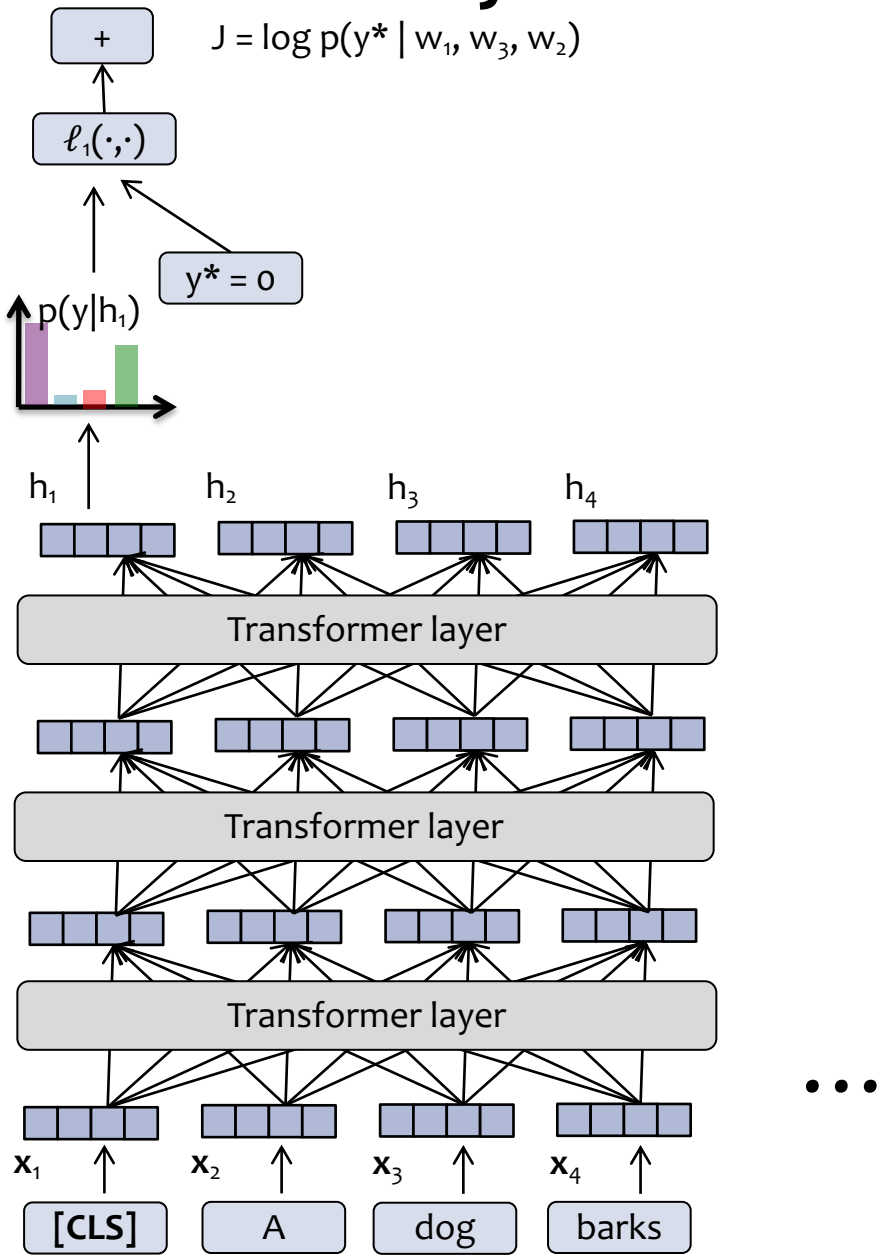
Although this is not a generative language model, it can be used very effectively as a discriminator

Encoder-only Transformer

BERT popularized this encoder-only Transformer architecture and style of pretraining

MLM Pretraining:

- Rather than trying to predict the next word from the previous ones…
- …mask out a word (or a few words) and predict the missing words from the remaining ones



…

Supervised Fine-tuning:

- How to fine-tune this model for a classification task?
- Predict the class label given the embedding for the special [CLS] token
- Fine-tune the model to be good at classification in this way

Although this is not a generative language model, it can be used very effectively as a discriminator

VISION TRANSFORMER

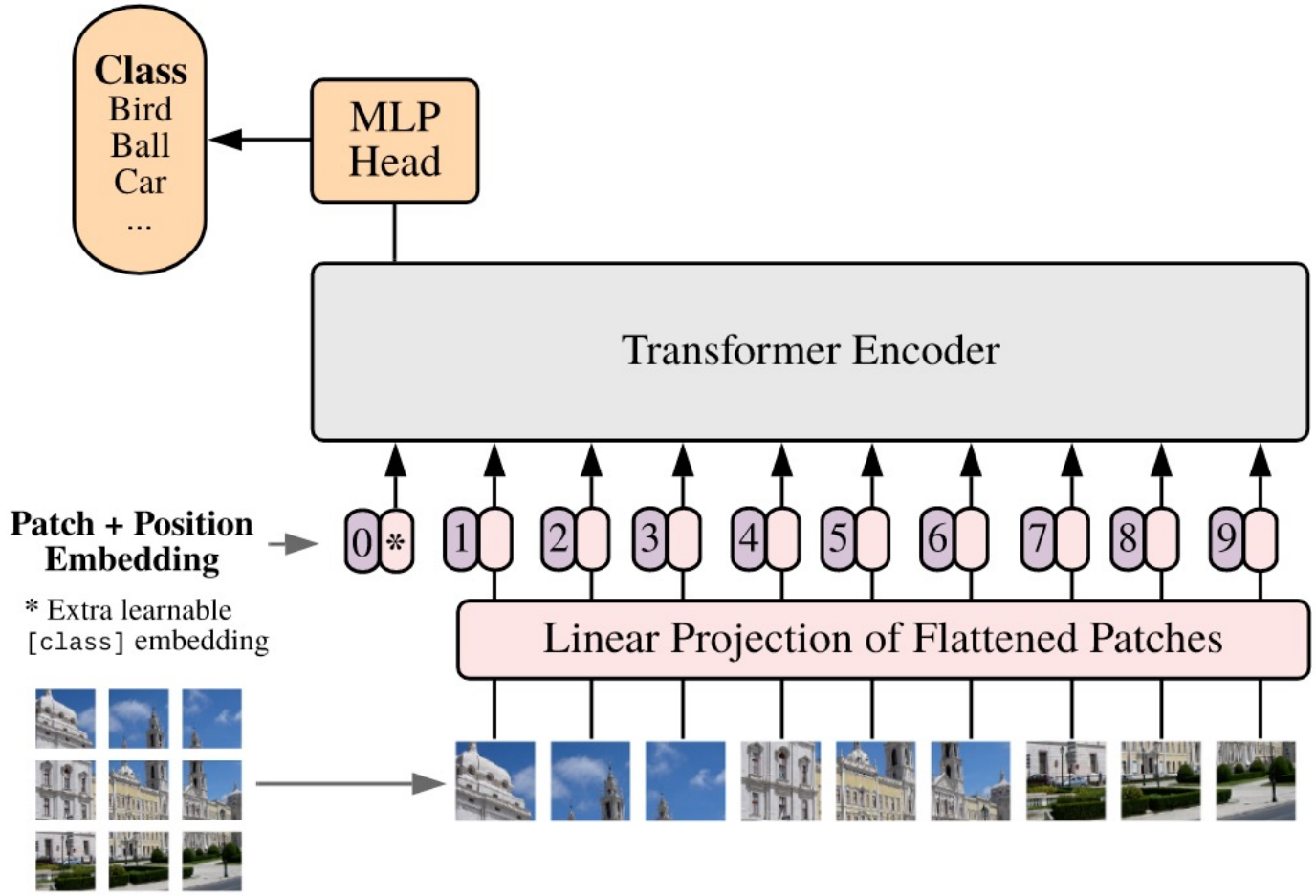
Vision Transformer (ViT)

Model:

- model is almost identical to BERT
- instead of words as input the inputs are PxP pixel image patches, P ∈ {14, 16, 32} (no overlap)
- each patch is embedded linearly into a vector of size 1024
- 1D positional embeddings

Training:

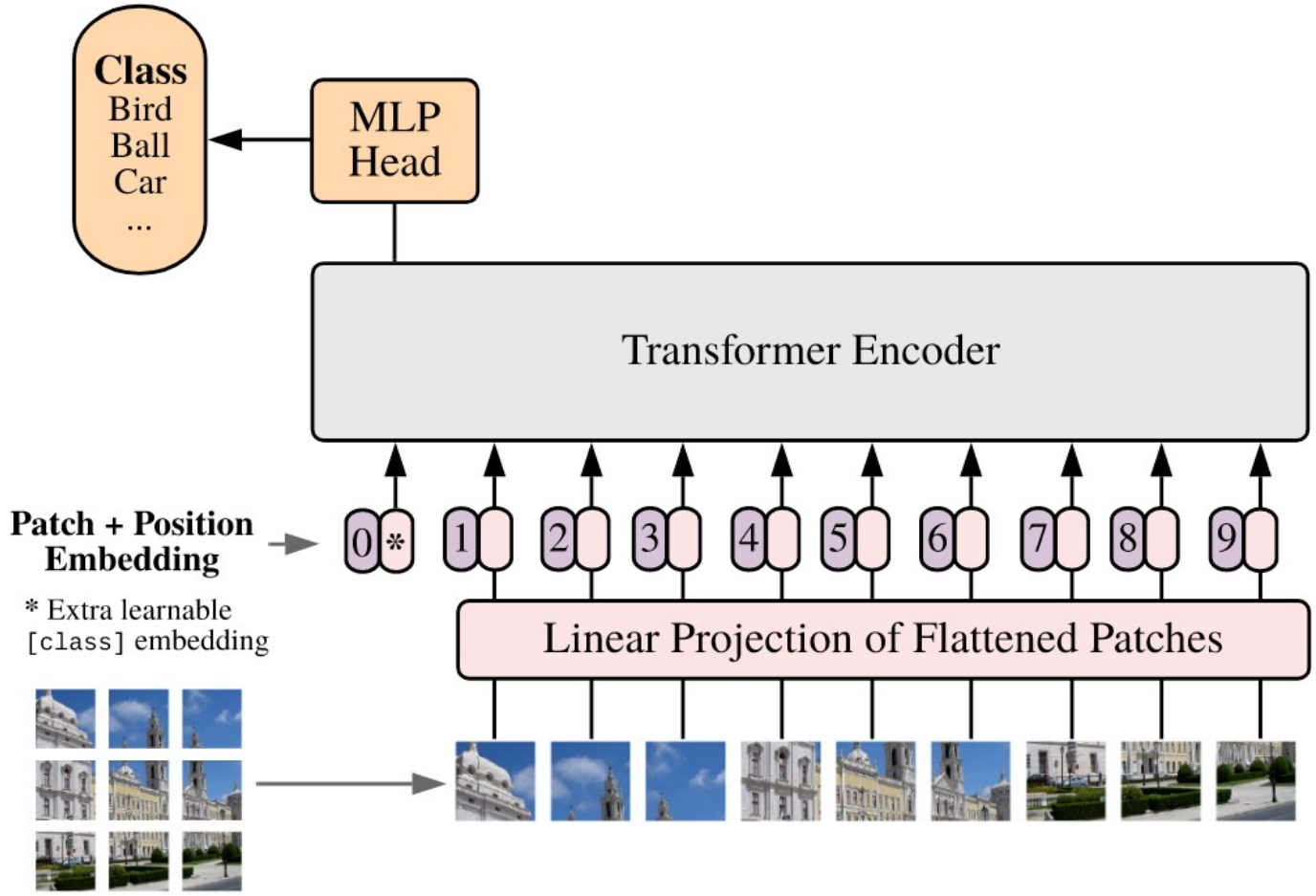
- for pre-training, optimize for image classification on large supervised dataset (e.g. ImageNet 21K, JFT-300M)—same setup as a CNN
- for fine-tuning, learn a new classification head on a small dataset (e.g. CIFAR-100)



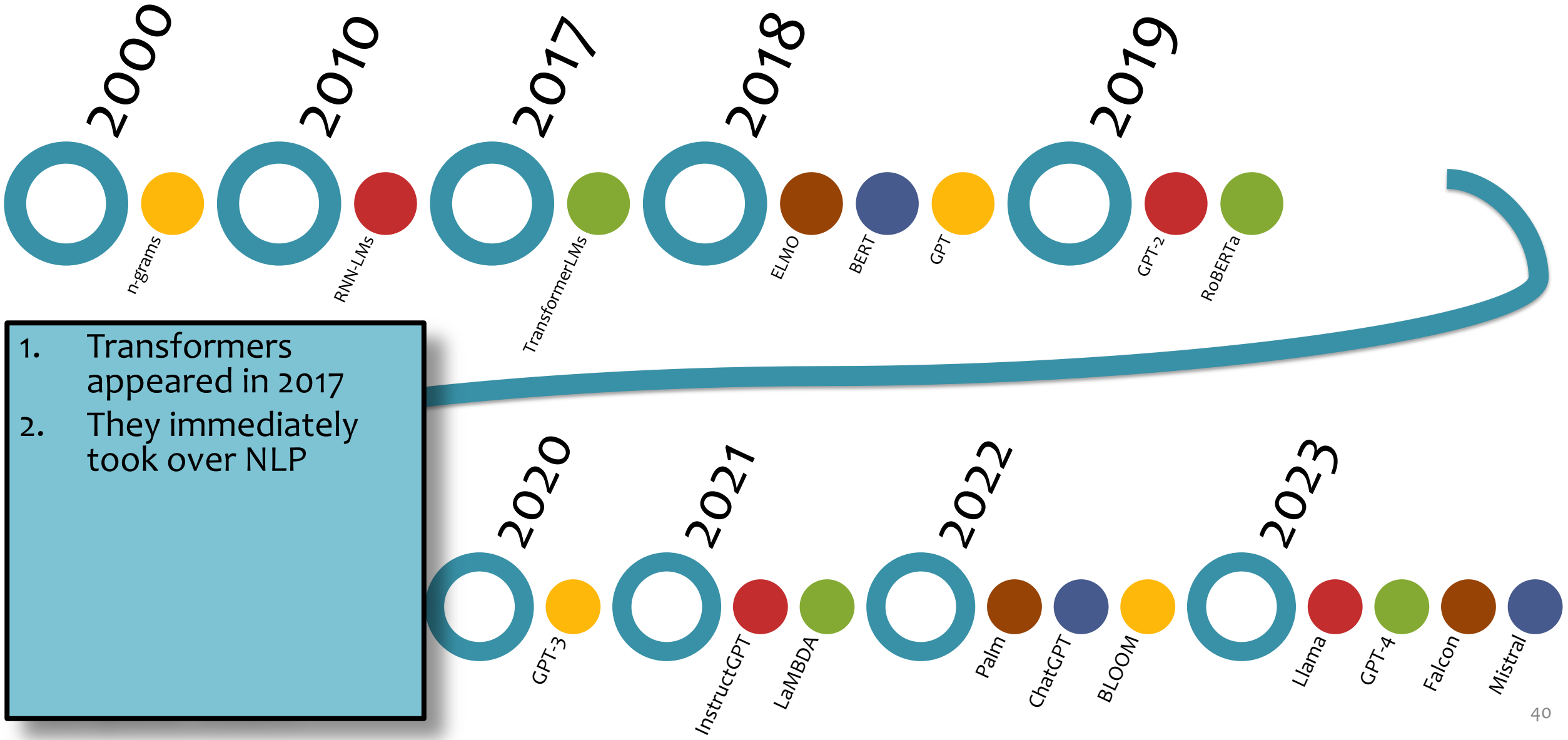
Vision Transformer (ViT)

Question: how can a ViT learn 2D positional information from 1D position embeddings?

Answer:

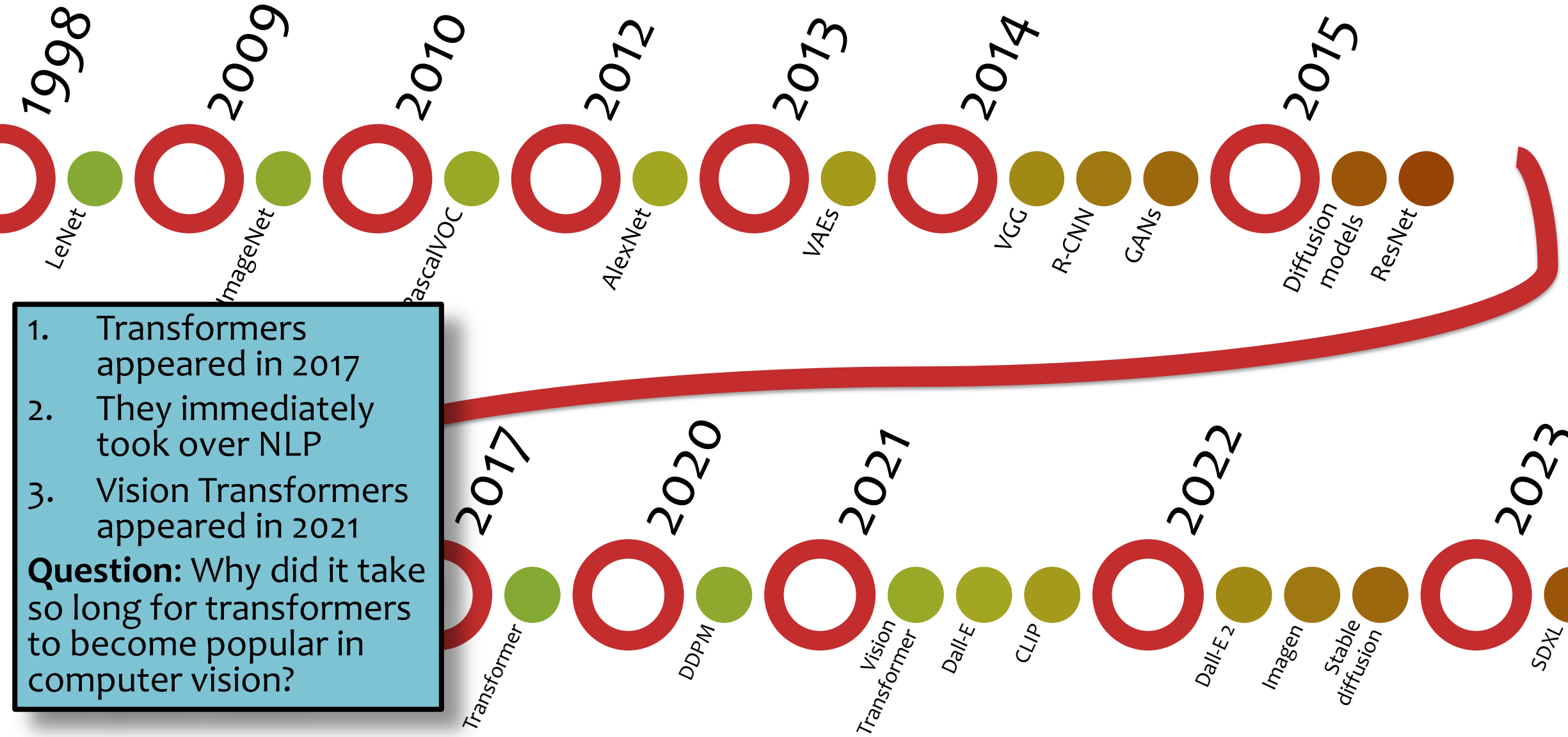


Timeline: Language Modeling



1. Transformers appeared in 2017
2. They immediately took over NLP

Timeline: Image Generation



1. Transformers appeared in 2017
2. They immediately took over NLP
3. Vision Transformers appeared in 2021

Question: Why did it take so long for transformers to become popular in computer vision?

Timeline: Image Generation

1. Transformers appeared in 2017
2. They immediately took over NLP
3. Vision Transformers appeared in 2021

Question: Why did it take so long for transformers to become popular in computer vision?

AN IMAGE IS WORTH 16X16 WORDS: TRANSFORMERS FOR IMAGE RECOGNITION AT SCALE

Alexey Dosovitskiy[*,†], **Lucas Beyer**[*], **Alexander Kolesnikov**[*], **Dirk Weissenborn**[*], **Xiaohua Zhai**[*], **Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, Neil Houlsby**[*,†]

[*]equal technical contribution, [†]equal advising

Google Research, Brain Team

{adosovitskiy, neilhoulsby}@google.com

When trained on mid-sized datasets such as ImageNet without strong regularization, these models yield modest accuracies of a few percentage points below ResNets of comparable size. This seemingly discouraging outcome may be expected: Transformers lack some of the inductive biases inherent to CNNs, such as translation equivariance and locality, and therefore do not generalize well when trained on insufficient amounts of data.

However, the picture changes if the models are trained on larger datasets (14M-300M images). We find that large scale training trumps inductive bias. Our Vision Transformer (ViT) attains excellent results when pre-trained at sufficient scale and transferred to tasks with fewer datapoints. When pre-trained on the public ImageNet-21k dataset or the in-house JFT-300M dataset, ViT approaches or beats state of the art on multiple image recognition benchmarks. In particular, the best model reaches the accuracy of 88.55% on ImageNet, 90.72% on ImageNet-ReaL, 94.55% on CIFAR-100, and 77.63% on the VTAB suite of 19 tasks.

Timeline: Image Generation

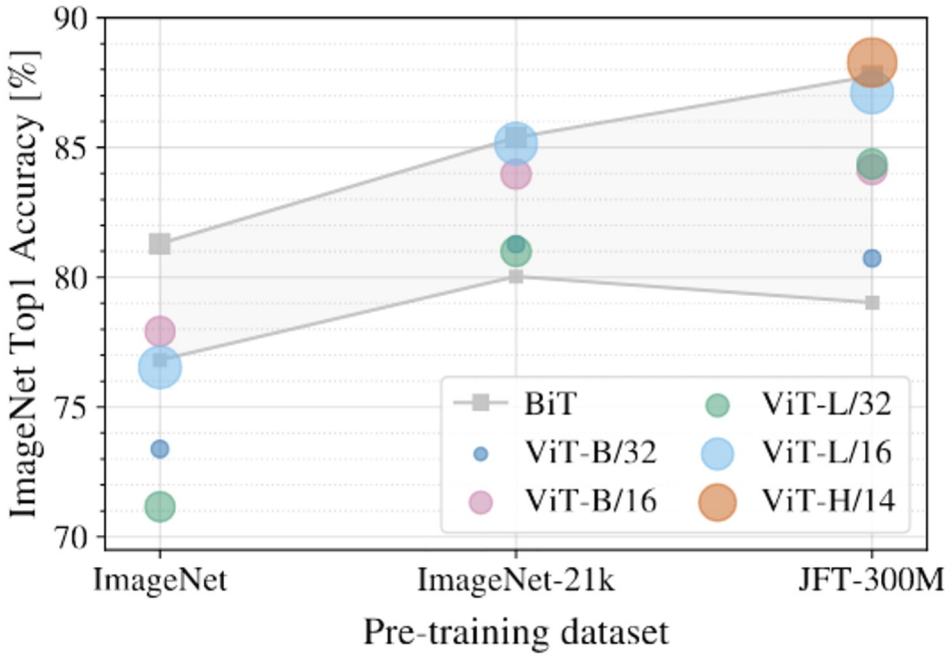


Figure 3: Transfer to ImageNet. While large ViT models perform worse than BiT ResNets (shaded area) when pre-trained on small datasets, they shine when pre-trained on larger datasets. Similarly, larger ViT variants overtake smaller ones as the dataset grows.

Comparison of two model families:

1. BiT – large CNNs based on ResNet
2. ViT – vision transformers of various sizes

1. Transformers appeared in 2017
2. They immediately took over NLP
3. Vision Transformers appeared in 2021

Question: Why did it take so long for transformers to become popular in computer vision?

Vision Transformer (ViT)

- The original Vision Transformer models were quite small compared to the Large Language Models (LLMs) of the time
- By 2023, ViT had been scaled to 22 billion parameters with good success

Model	Layers	Hidden size D	MLP size	Heads	Params
ViT-Base	12	768	3072	12	86M
ViT-Large	24	1024	4096	16	307M
ViT-Huge	32	1280	5120	16	632M

Table 1: Details of Vision Transformer model variants.

Table 1: ViT-22B model architecture details.

Name	Width	Depth	MLP	Heads	Params [M]
ViT-G	1664	48	8192	16	1843
ViT-e	1792	56	15360	16	3926
ViT-22B	6144	48	24576	48	21743

TASK: IMAGE GENERATION

Image Generation

- Class-conditional generation
- Super resolution
- Image Editing
- Style transfer
- Text-to-image (TTI) generation

Figure from Razavi et al. (2019)

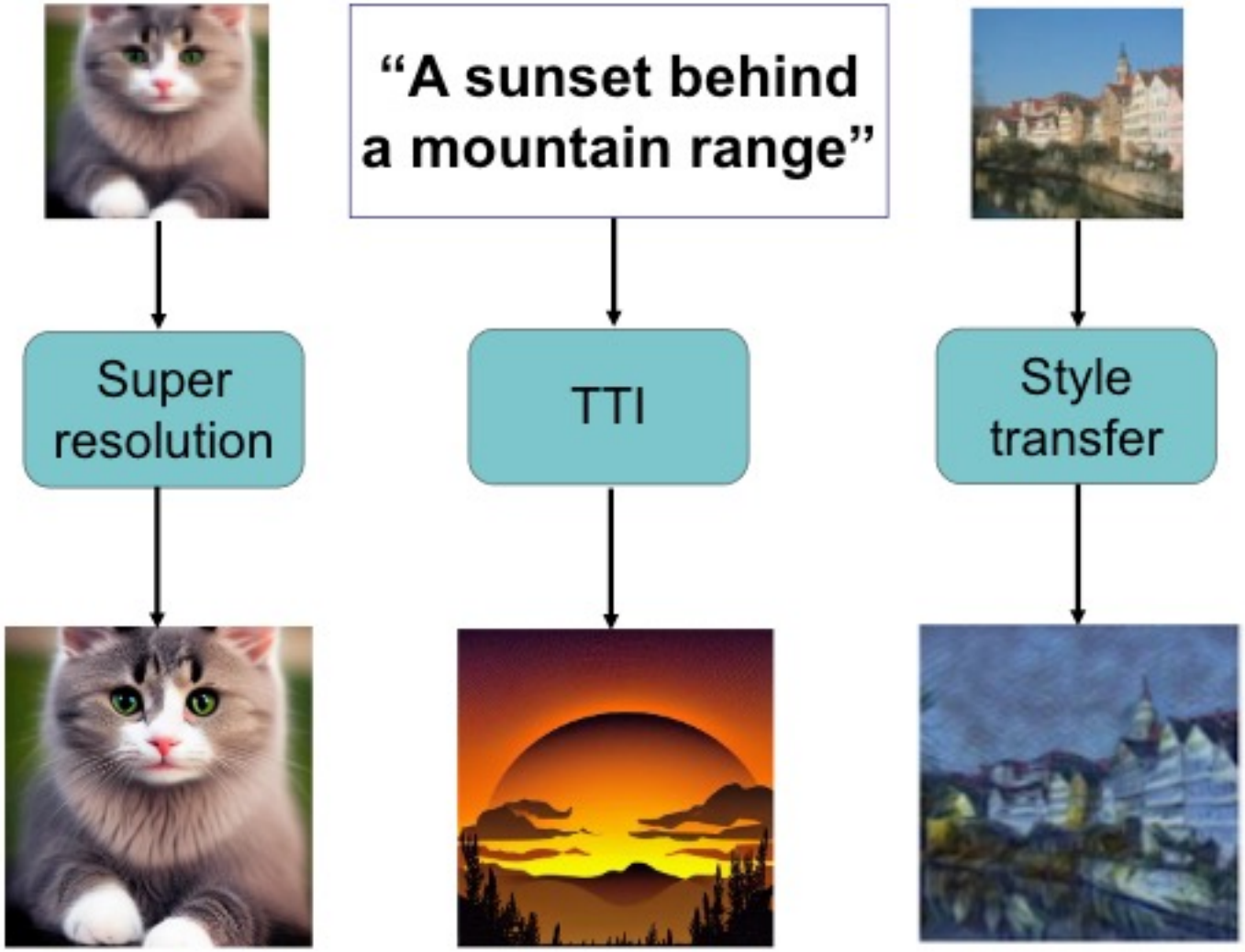


Figure from Bie et al. (2023)

Class Conditional Generation

- **Task**: Given a class label indicating the image type, sample a new image from the model with that type
- Image classification is the problem of taking in an image and predicting its label p(y|x)
- Class conditional generation is doing this in reverse p(x|y)

sea anemone

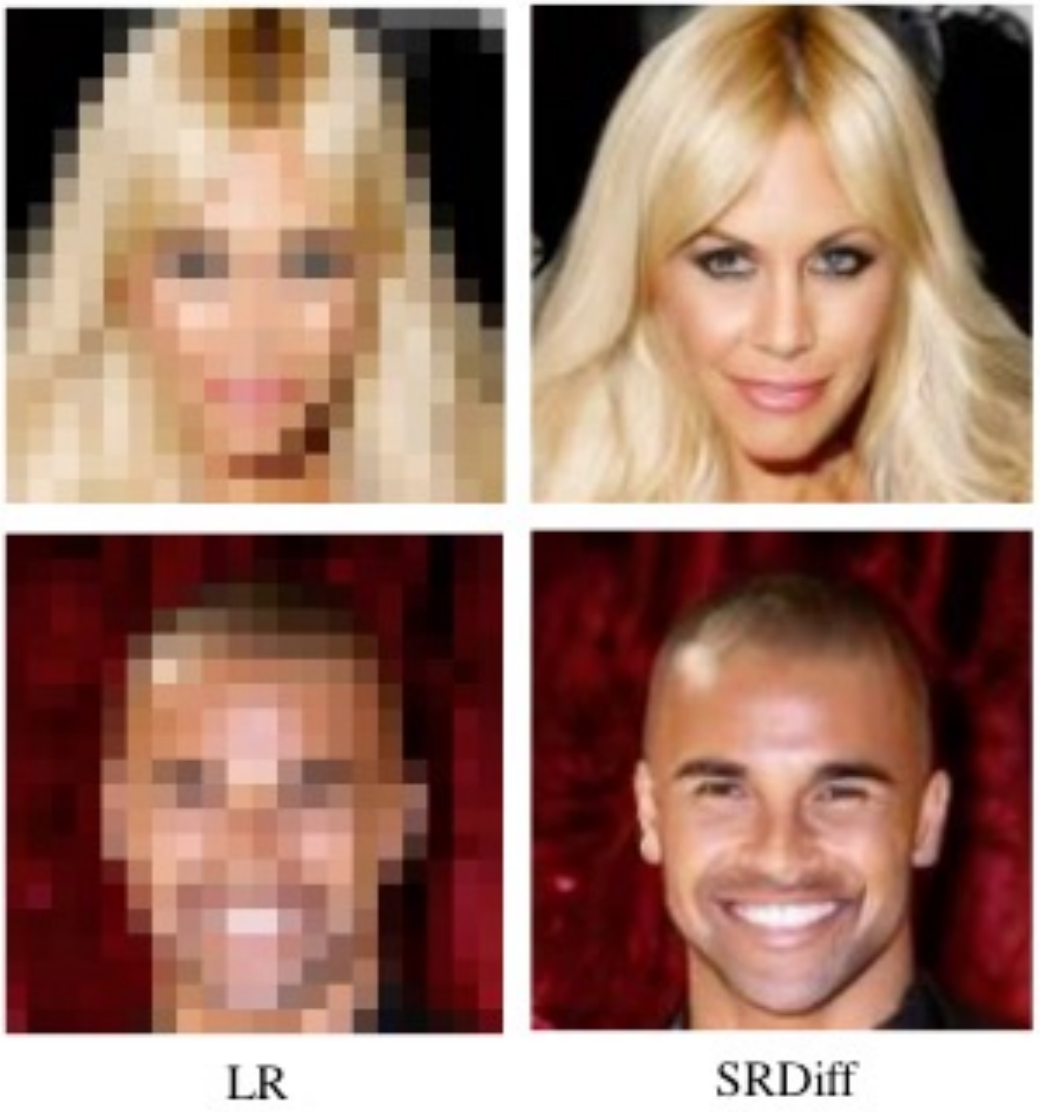
brain coral

slug

goldfinch

Figure from Razavi et al. (2019)

Super Resolution



- Given a low resolution image, generate a high resolution reconstruction of the image
- Compelling on low resolution inputs (see example to the left) but also effective on high resolution inputs

Figure from Li et al. (2021)

Image Editing

A variety of tasks involve automatic editing of an image:

- **Inpainting** fills in the (pre-specified) missing pixels
- **Colorization** restores color to a greyscale image
- **Uncropping** creates a photo-realistic reconstruction of a missing side of an image

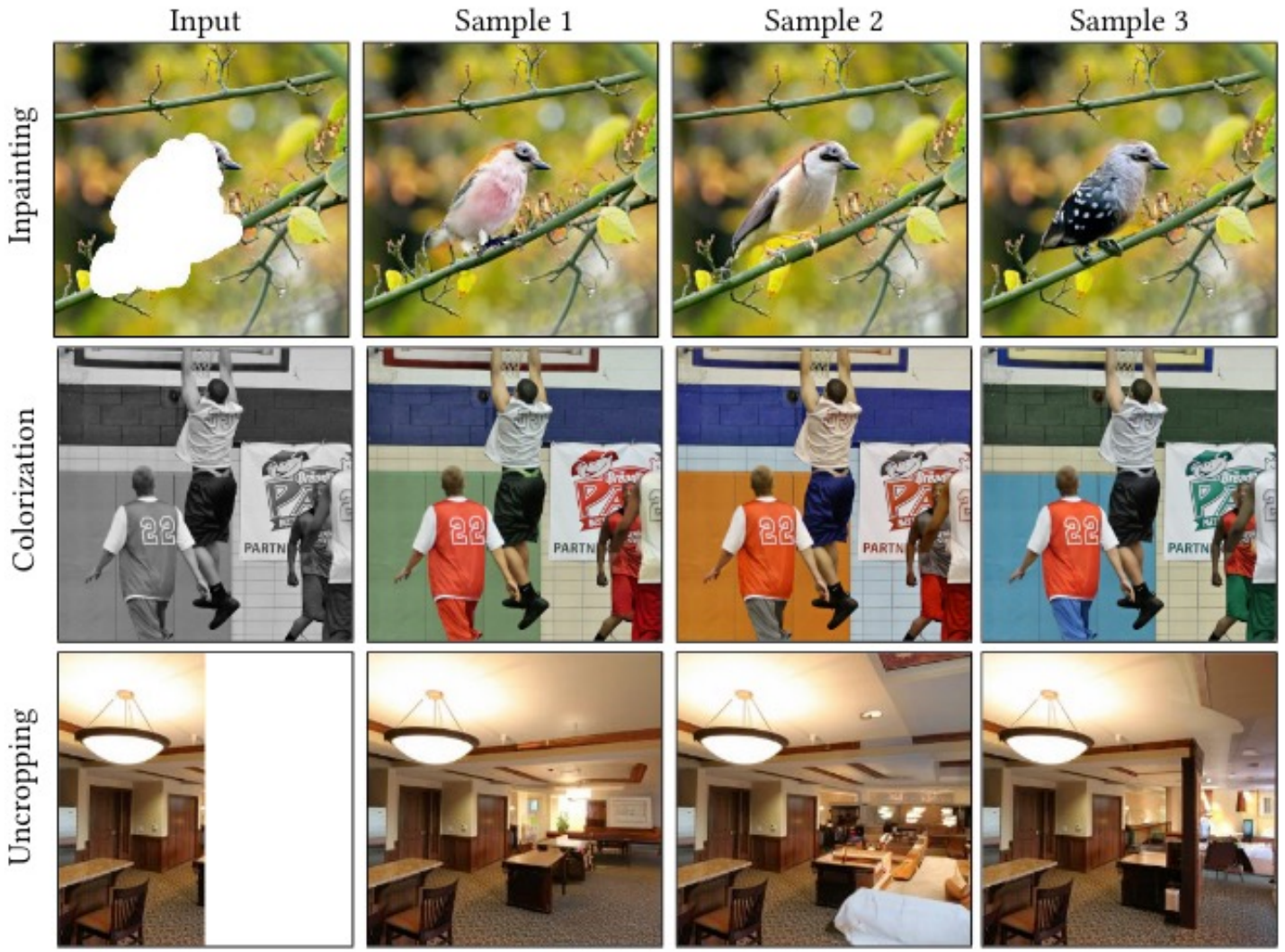


Figure from Saharia et al. (2022)

Style Transfer

- The goal of style transfer is to blend two images
- Yet, the blend should retain the semantic content of the source image presented in the style of another image

Figure from Gatys et al. (2016)

Figure 3. Images that combine the content of a photograph with the style of several well-known artworks. The images were created by finding an image that simultaneously matches the content representation of the photograph and the style representation of the artwork. The original photograph depicting the Neckarfront in Tübingen, Germany, is shown in **A** (Photo: Andreas Praefcke). The painting that provided the style for the respective generated image is shown in the bottom left corner of each panel. **B** *The Shipwreck of the Minotaur* by J.M.W. Turner, 1805. **C** *The Starry Night* by Vincent van Gogh, 1889. **D** *Der Schrei* by Edvard Munch, 1893. **E** *Femme nue assise* by Pablo Picasso, 1910. **F** *Composition VII* by Wassily Kandinsky, 1913.

Text-to-Image Generation

- Given a text description, sample an image that depicts the prompt
- The following images are samples from SDXL with refinement

Prompt: A propaganda poster depicting a cat dressed as french emperor napoleon holding a piece of cheese.

Figure from Podell et al. (2023)

Text-to-Image Generation

- Given a text description, sample an image that depicts the prompt
- The following images are samples from SDXL with refinement

Prompt: Epic long distance cityscape photo of New York City flooded by the ocean and overgrown buildings and jungle ruins in rainforest, at sunset, cinematic shot, highly detailed, 8k, golden light

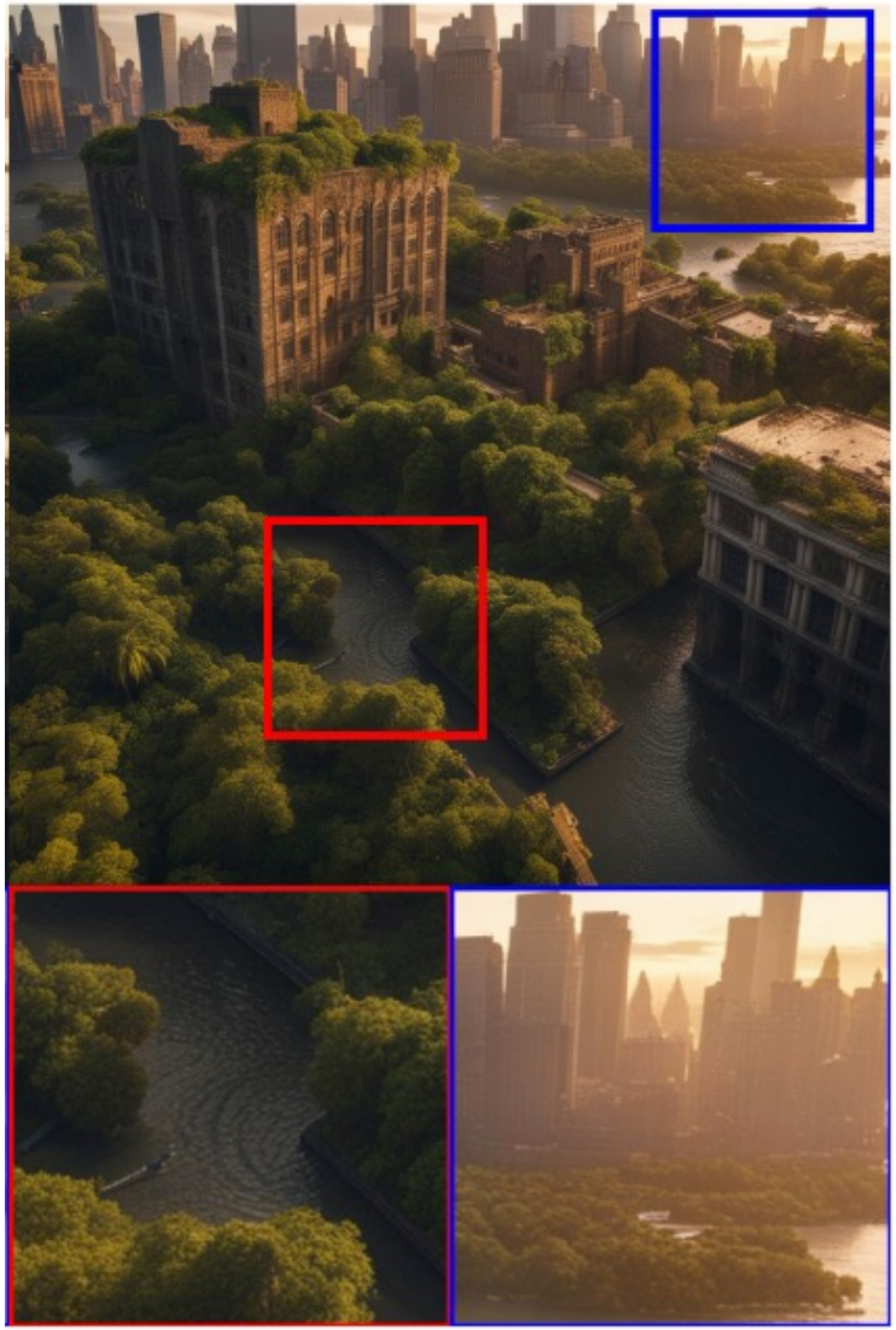


Figure from Podell et al. (2023)

Text-to-Image Generation

- Given a text description, sample an image that depicts the prompt
- The following images are samples from SDXL with refinement

Prompt: close up headshot, futuristic young woman, wild hair sly smile in front of gigantic UFO, dslr, sharp focus, dynamic composition

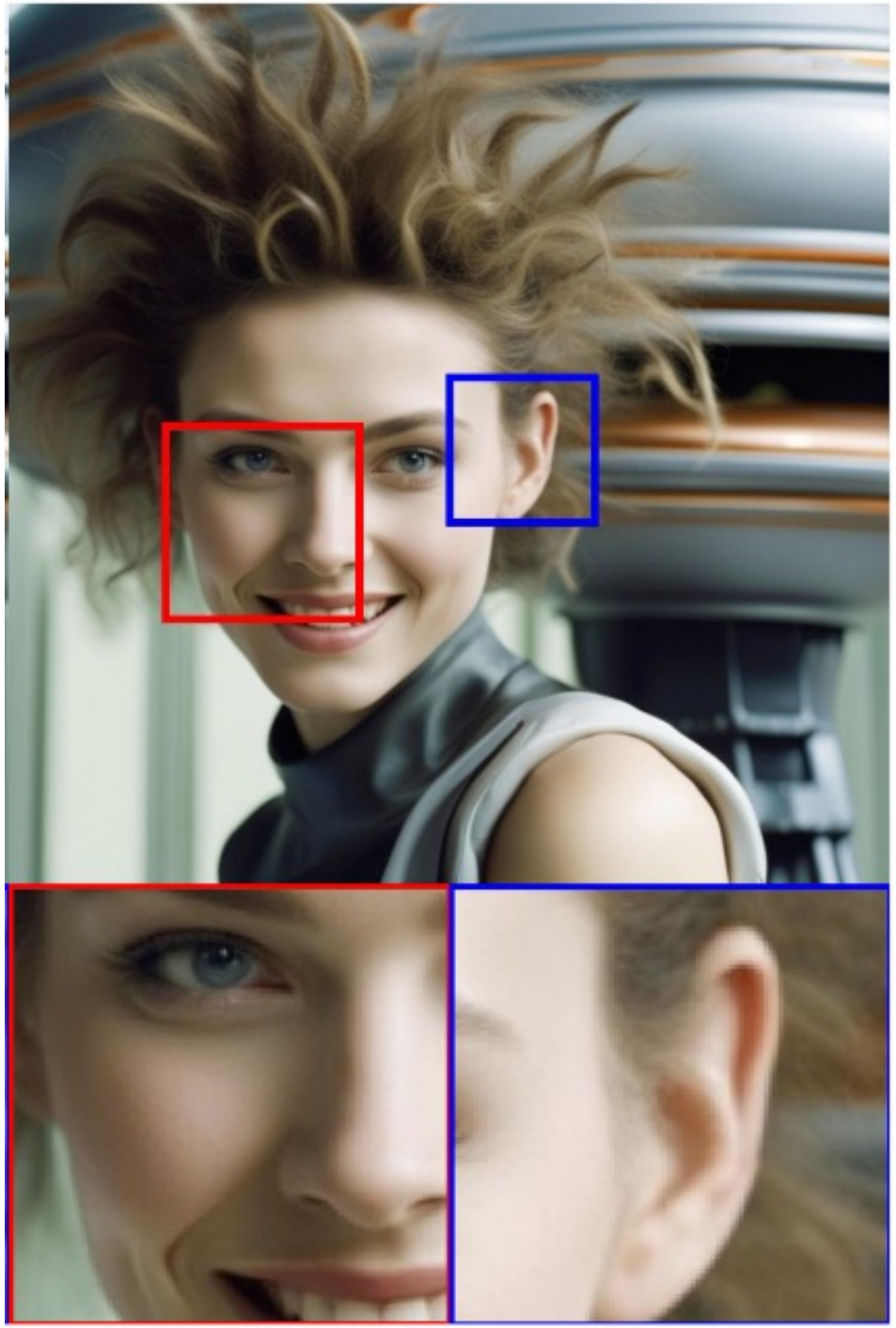


Figure from Podell et al. (2023)

Text-to-Image Generation

- Given a text description, sample an image that depicts the prompt
- The following images are samples from SDXL with refinement

Prompt: close up headshot, futuristic **old man**, wild hair sly smile in front of gigantic UFO, dslr, sharp focus, dynamic composition, **rule of thirds**

Figure from https://stablediffusionweb.com/

In-Class Poll

Question:

What are the potential societal impacts of image generation?

Answer:

Summary

- Computer Vision
- Task: Image Generation
- Model: Generative Adversarial Network (GAN)
- Learning for GANs
- Scaling Up the Model Size
- Societal Impacts of Image Generation

MODEL: GENERATIVE ADVERSARIAL NETWORK (GAN)

Stable Diffusion still can’t explain GANs

Prompt: slide explaining Generative Adversarial Networks (GANs) for Intro to Machine Learning course, carefully designed, easy to follow

Negative Prompt: boring, unclear, nontechnical

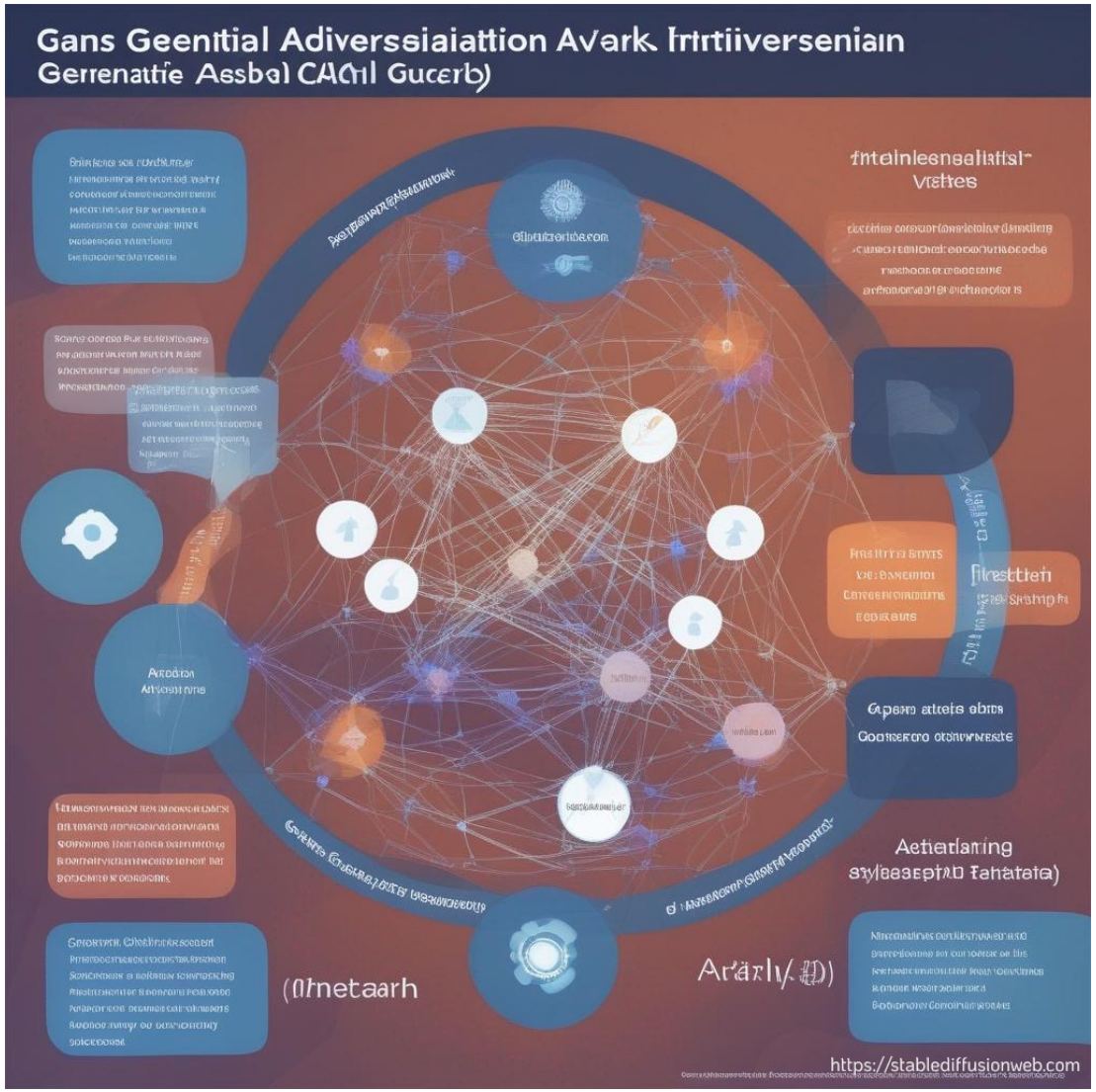


Figure from https://stablediffusionweb.com/

Generative Adversarial Networks (GANs)

A GAN consists of two deterministic neural network models:

1) the Generator

takes a vector of random noise as input, and generates an image

2) the Discriminator

takes in an image classifies whether it is real (label 1) or fake (label 0)

Generator Model

1) the Generator

takes a vector of random noise as input, and generates an image

Example Generator: DCGAN

- An inverted CNN with four **fractionally-strided** convolution layers (not deconvolution)
- These fractional strides grow the size of the image from layer to layer
- The final layer has three channels for red/green/blue

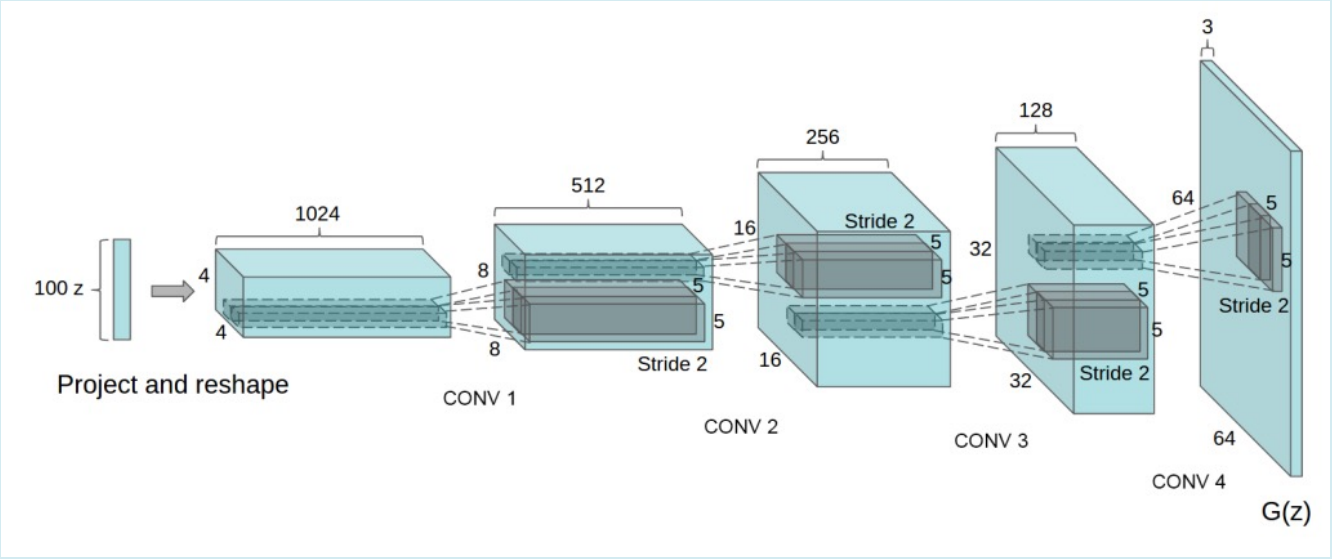


Figure from Radford et al. (2016)

Generative Adversarial Networks (GANs)

A GAN consists of two deterministic neural network models:

1) the Generator

takes a vector of random noise as input, and generates an image

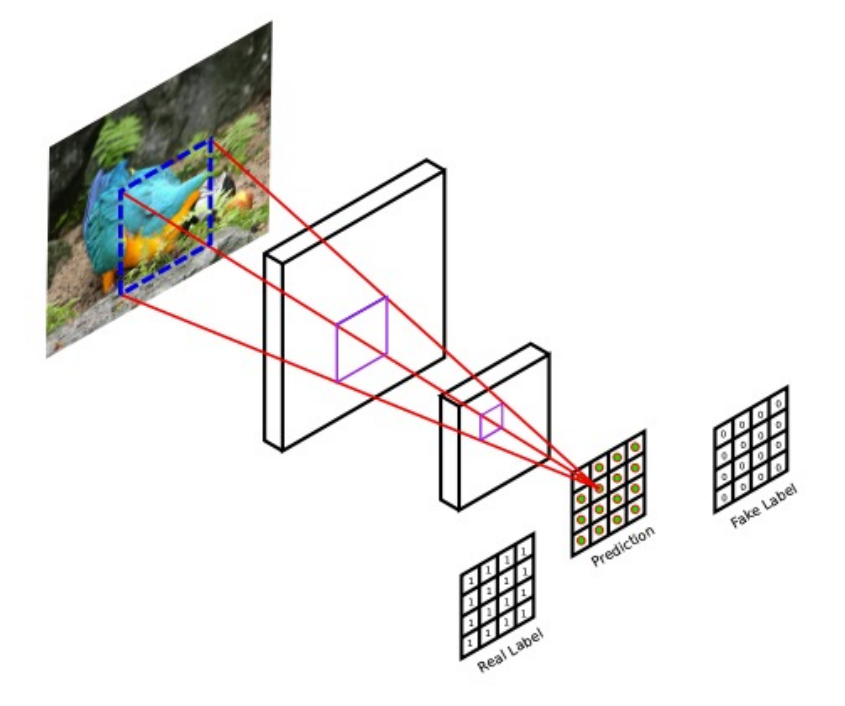
2) the Discriminator

takes in an image classifies whether it is real (label 1) or fake (label 0)

Discriminator Model

Example Discriminator: PatchGAN

- Convolutional neural network
- Looks at each patch of the image and tries to predict whether it is real or fake
- Helps avoid producing blurry images



2) the Discriminator

takes in an image classifies whether it is real (label 1) or fake (label 0)

Figure from Demir & Unal (2018)

Generative Adversarial Networks (GANs)

A GAN consists of two deterministic neural network models:

1) the Generator

takes a vector of random noise as input, and generates an image

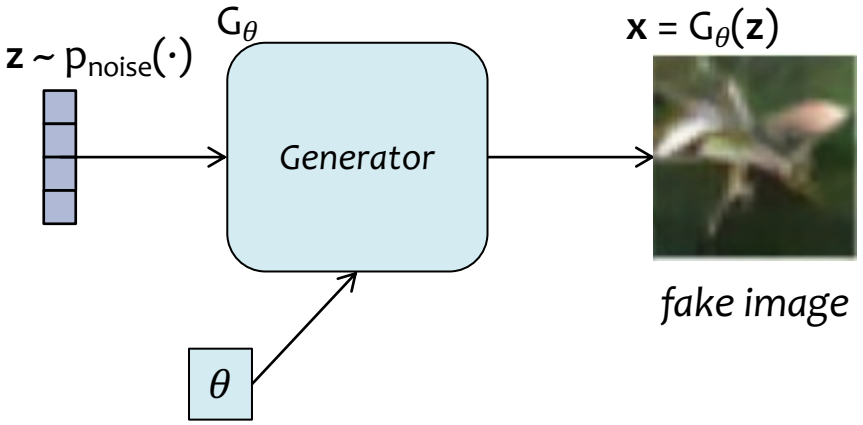
2) the Discriminator

takes in an image classifies whether it is real (label 1) or fake (label 0)

In training, the GAN plays a two player minimax game:

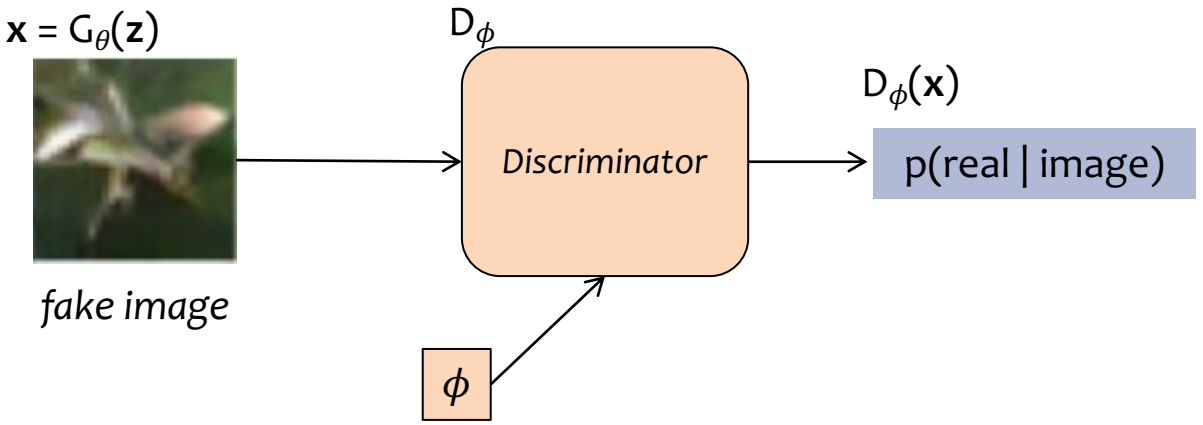
1. the Generator tries to create realistic images to fool the Discriminator into thinking they are real
2. the Discriminator tries to identify the real images from the fake

Generative Adversarial Networks (GANs)



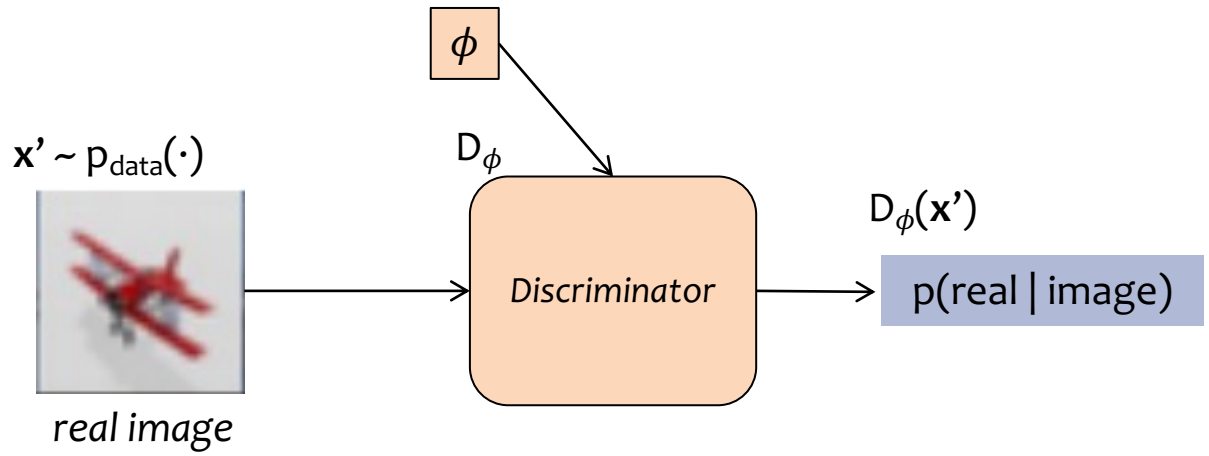
Real/fake images from Huang et al. (2017)

Generative Adversarial Networks (GANs)



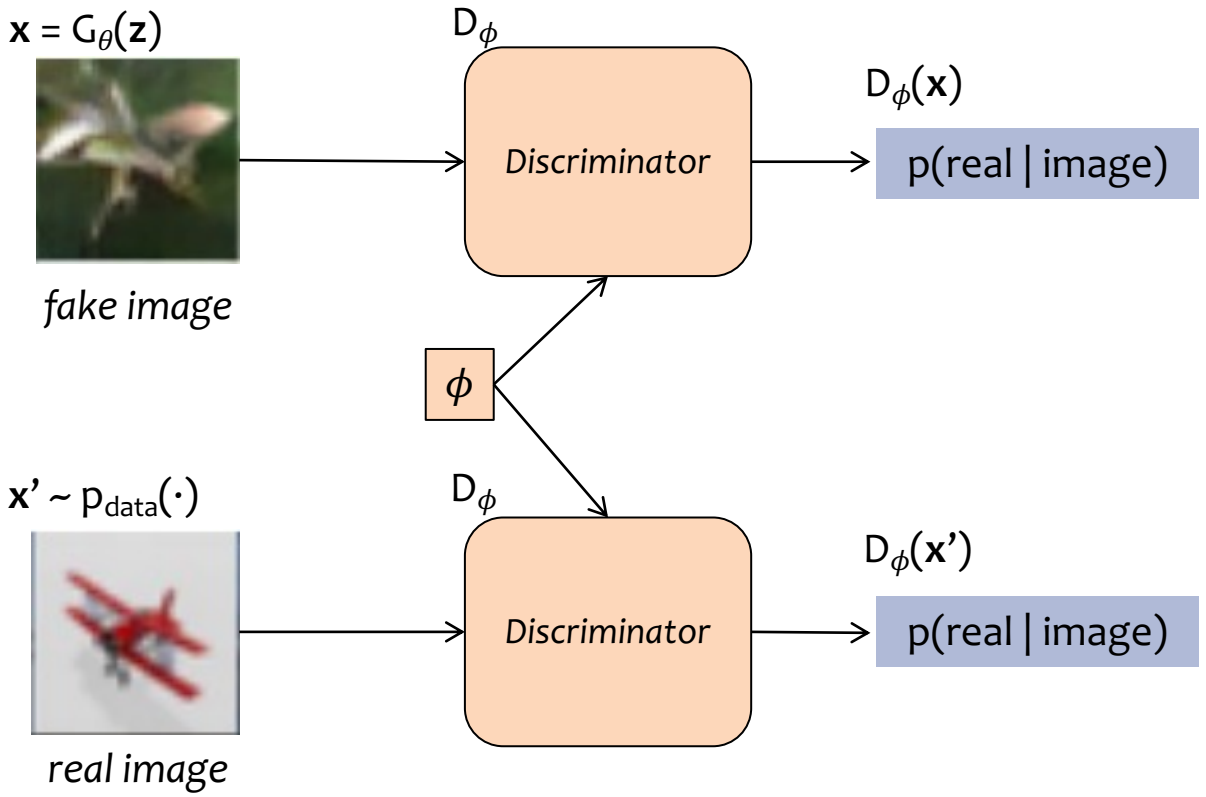
Real/fake images from Huang et al. (2017)

Generative Adversarial Networks (GANs)



Real/fake images from Huang et al. (2017)

Generative Adversarial Networks (GANs)



Real/fake images from Huang et al. (2017)

LEARNING FOR GANS

Generative Adversarial Networks (GANs)

A GAN consists of two deterministic neural network models:

1) the Generator

takes a vector of random noise as input, and generates an image

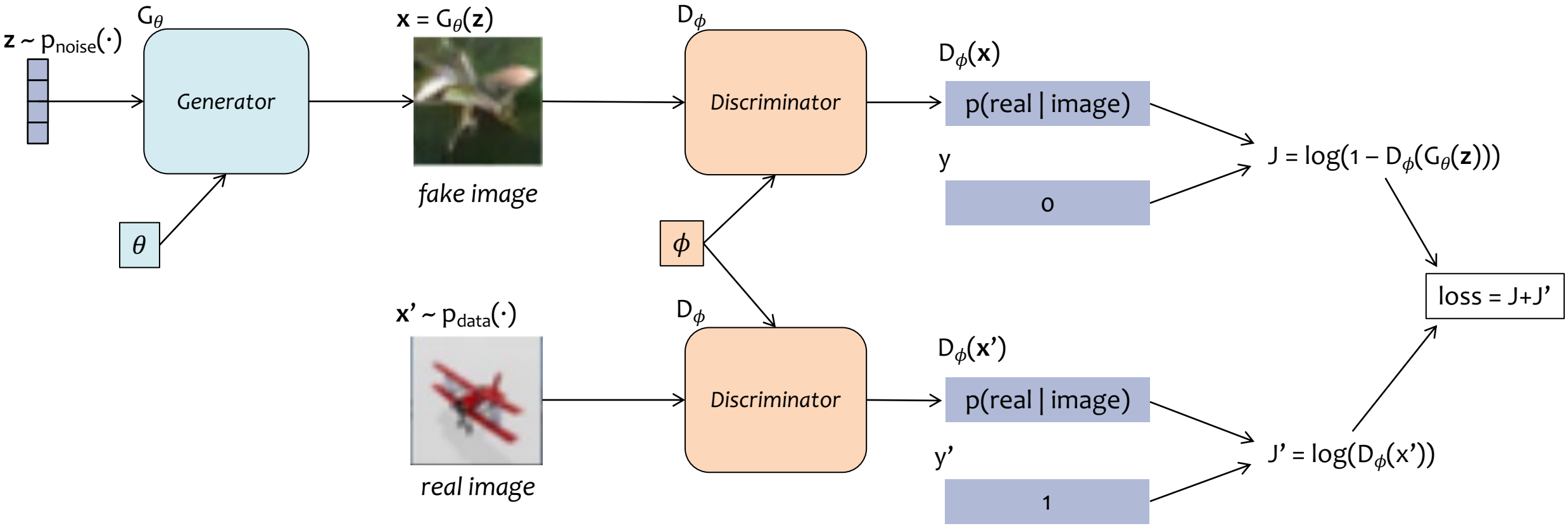
2) the Discriminator

takes in an image classifies whether it is real (label 1) or fake (label 0)

In training, the GAN plays a two player minimax game:

1. the Generator tries to create realistic images to fool the Discriminator into thinking they are real
2. the Discriminator tries to identify the real images from the fake

Generative Adversarial Networks (GANs)



Real/fake images from Huang et al. (2017)

Generative Adversarial Networks (GANs)

$$\max_{\phi} \log\left(D_\phi(\mathbf{x}^{(i)})\right) + \log\left(1 - D_\phi(G_\theta(\mathbf{z}^{(i)}))\right)$$

The discriminator is trying to maximize the likelihood of a binary classifier with labels {real = 1, fake = 0}, on the fixed output of the generator

$$\min_{\theta} \log\left(1 - D_\phi(G_\theta(\mathbf{z}^{(i)}))\right)$$

The generator is trying to minimize the likelihood of its generated (fake) image being classified as fake, according to a fixed discriminator

In training, the GAN plays a two player minimax game:

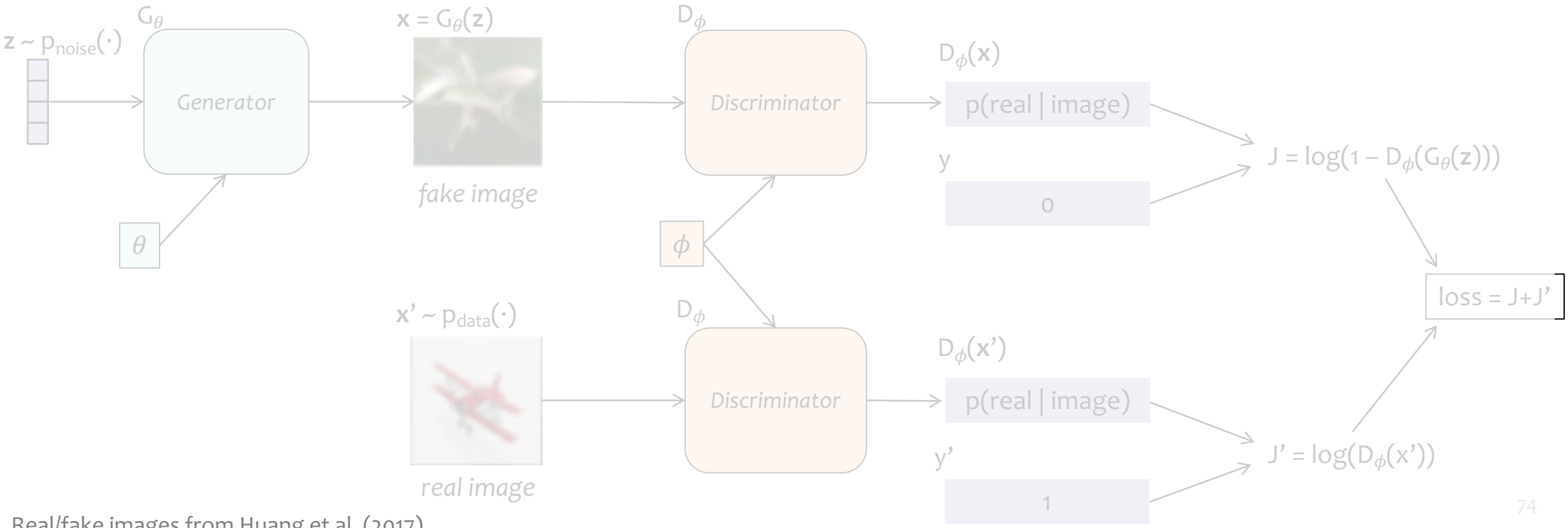
1. the Generator tries to create realistic images to fool the Discriminator into thinking they are real
2. the Discriminator tries to identify the real images from the fake

Learning a GAN

- Objective function is a simple differentiable function
- We chose G and D to be differentiable neural networks

Training alternates between:

- Keep G_θ fixed and backprop through D_ϕ
- Keep D_ϕ fixed and backprop through G_θ



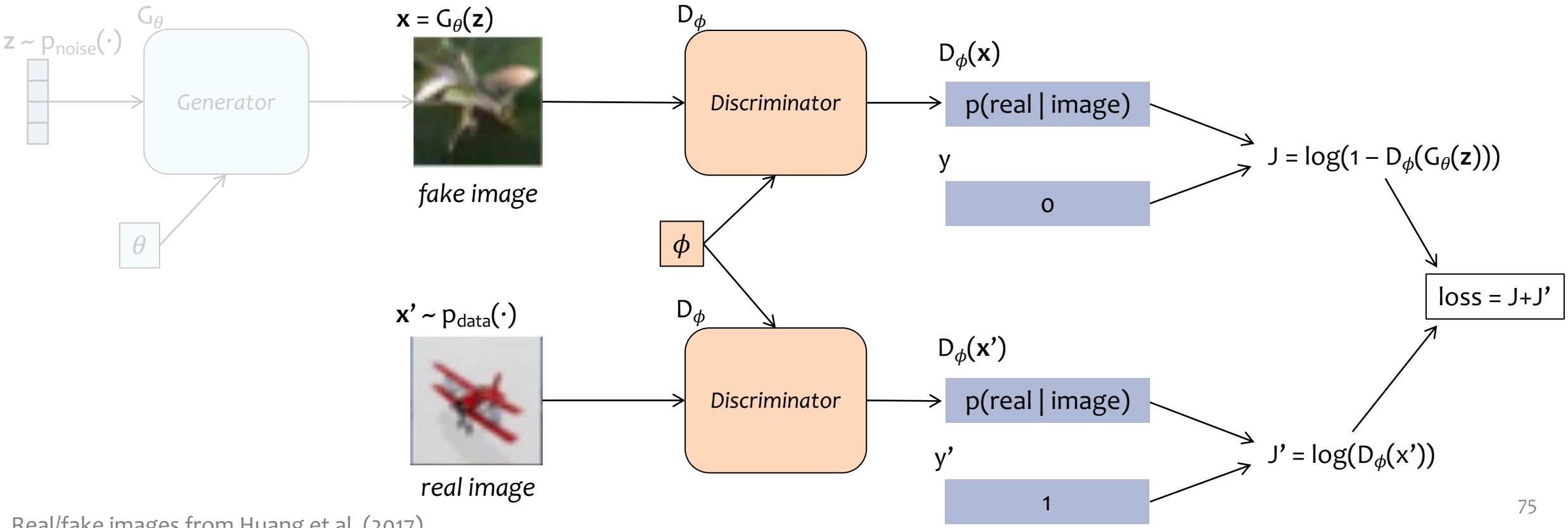
Real/fake images from Huang et al. (2017)

Learning a GAN

- Objective function is a simple differentiable function
- We chose G and D to be differentiable neural networks

Training alternates between:

- **Keep G_θ fixed and backprop through D_ϕ**
- Keep D_ϕ fixed and backprop through G_θ



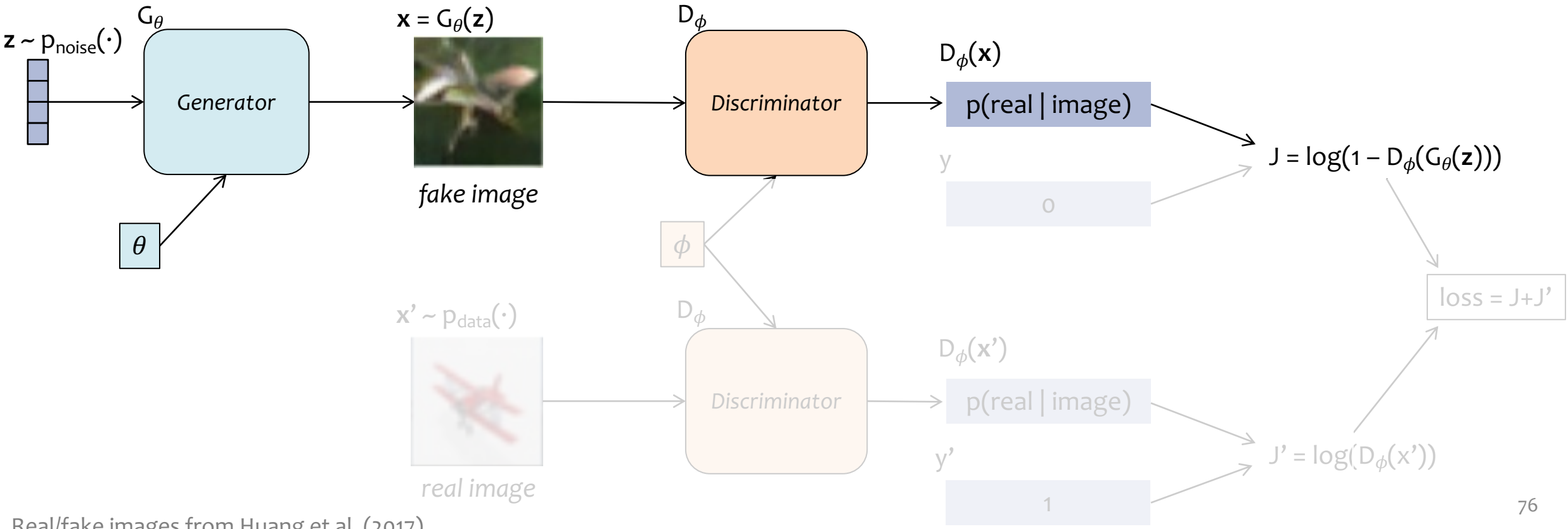
Real/fake images from Huang et al. (2017)

Learning a GAN

- Objective function is a simple differentiable function
- We chose G and D to be differentiable neural networks

Training alternates between:

- Keep G_θ fixed and backprop through D_ϕ
- **Keep D_ϕ fixed and backprop through G_θ**



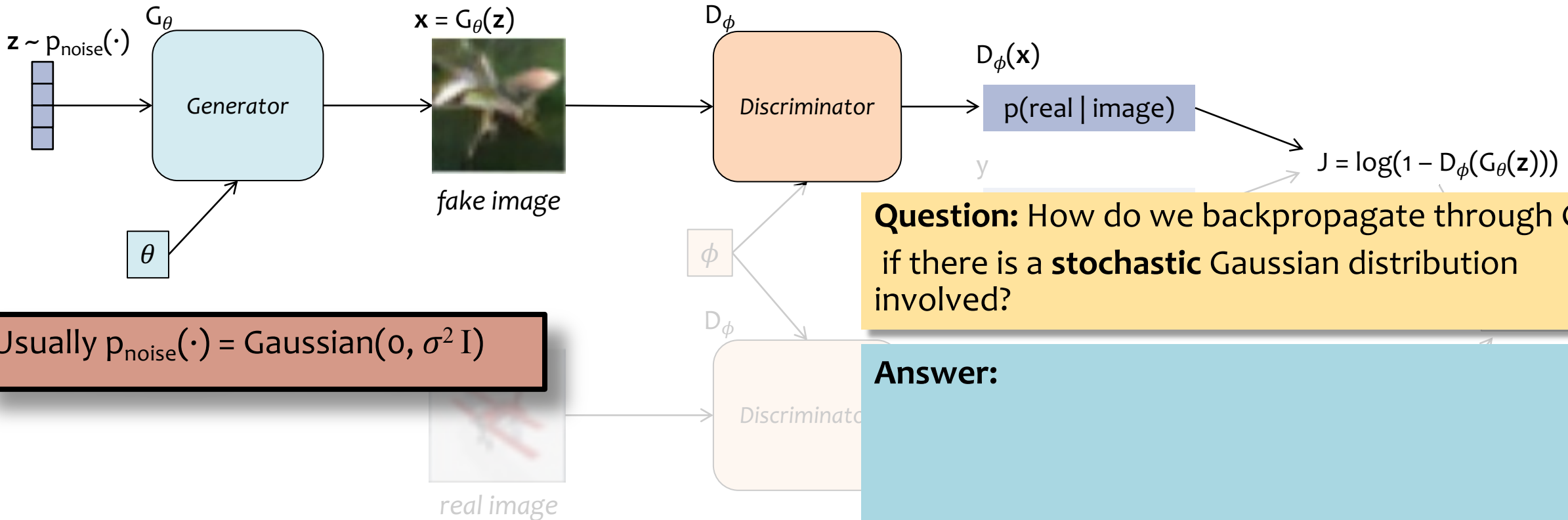
Real/fake images from Huang et al. (2017)

Learning a GAN

- Objective function is a simple differentiable function
- We chose G and D to be differentiable neural networks

Training alternates between:

- Keep G_θ fixed and backprop through D_ϕ
- **Keep D_ϕ fixed and backprop through G_θ**



Usually $p_{noise}(\cdot) = \text{Gaussian}(0, \sigma^2 I)$

Question: How do we backpropagate through G_θ if there is a **stochastic** Gaussian distribution involved?

Answer:

Real/fake images from Huang et al. (2017)

Learning a GAN

- Training data consists of a collection of m unlabeled images $x^{(1)}, \ldots, x^{(m)}$
- Optimization is similar to block coordinate descent
- But instead of exactly solving the min/max problem, we take a step of mini-batch SGD

Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k, is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

for number of training iterations **do**

  for k steps **do**

  - Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
  - Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
  - Update the discriminator by ascending its stochastic gradient:

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

  end for

  - Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
  - Update the generator by descending its stochastic gradient:

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

end for

The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

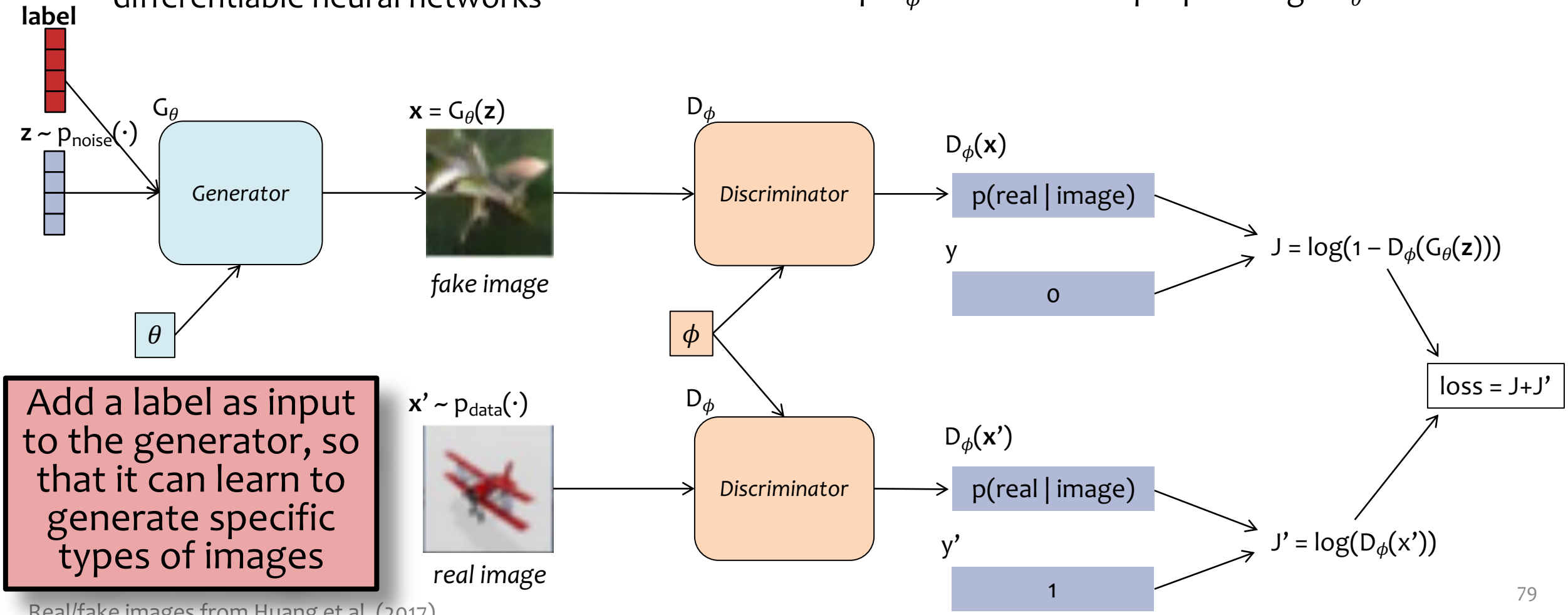
Figure from https://arxiv.org/pdf/1406.2661.pdf

Class-conditional GANs

- Objective function is a simple differentiable function
- We chose G and D to be differentiable neural networks

Training alternates between:

- Keep G_θ fixed and backprop through D_ϕ
- Keep D_ϕ fixed and backprop through G_θ



Add a label as input to the generator, so that it can learn to generate specific types of images

Real/fake images from Huang et al. (2017)

SCALING UP THE MODEL SIZE

Scaling Up the Model Size

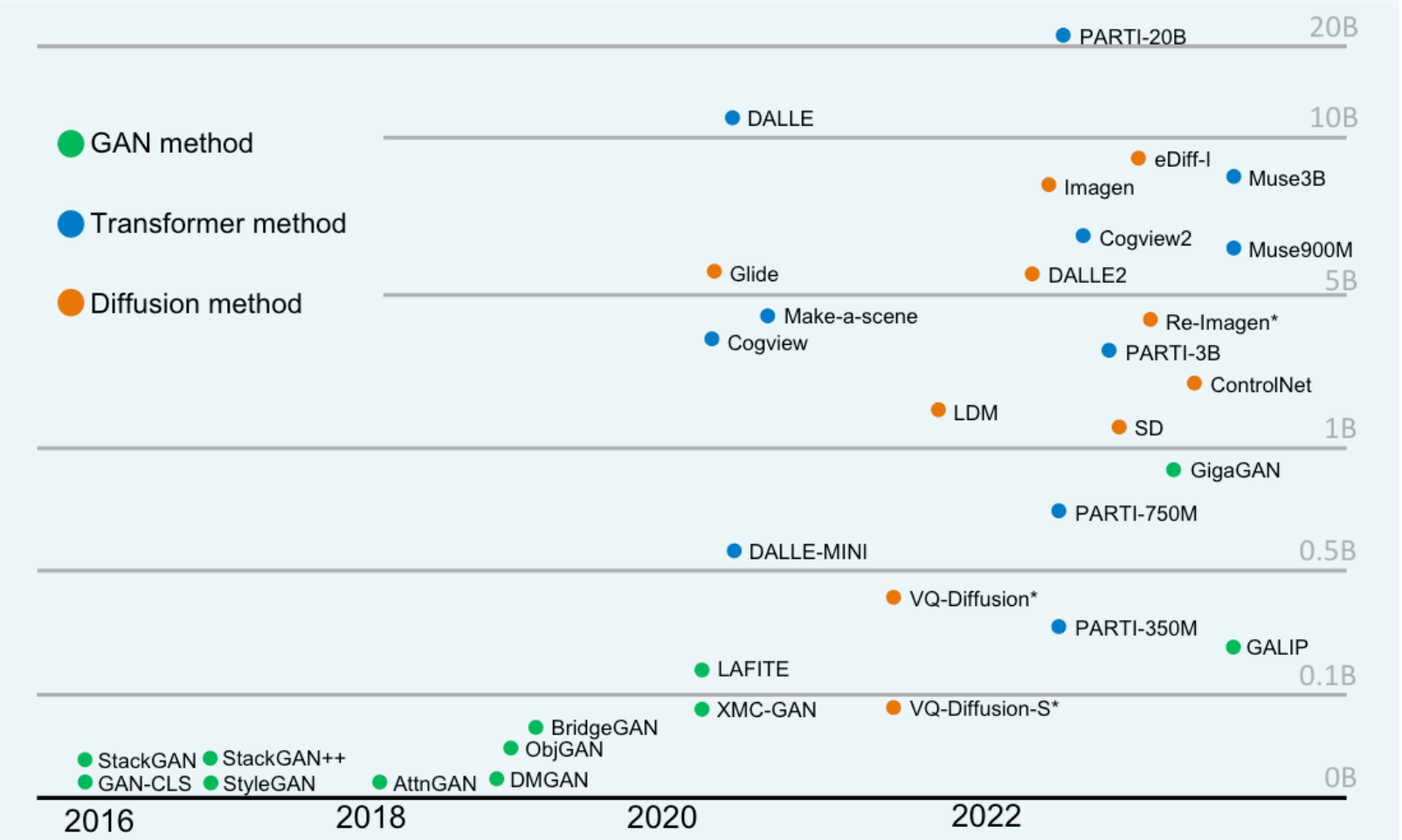


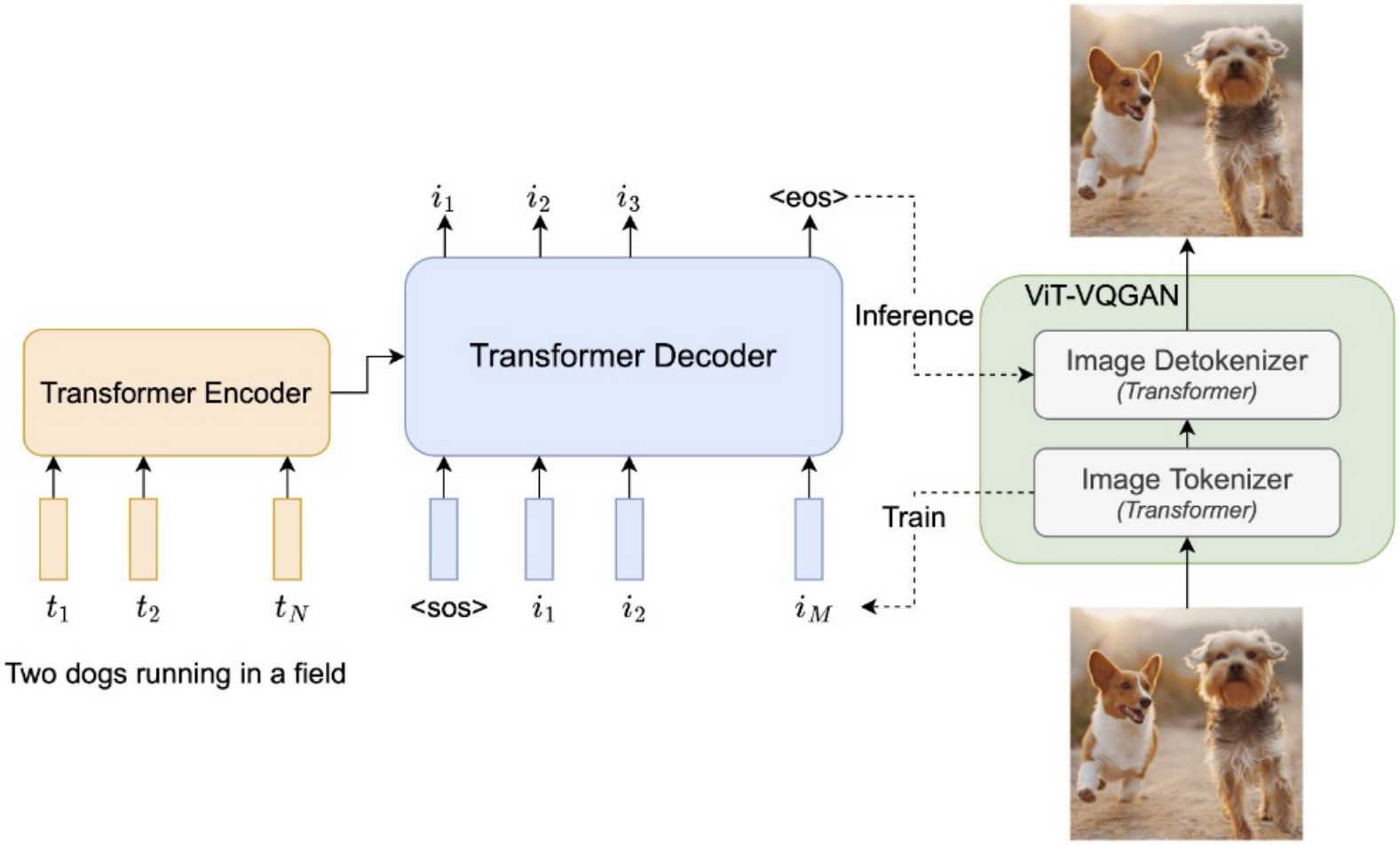
Fig. 5. Timeline of TTI model development, where green dots are GAN TTI models, blue dots are autoregressive Transformers and orange dots are Diffusion TTI models. Models are separated by their parameter, which are in general counted for all their components. Models with asterisk are calculated without the involvement of their text encoders.

Figure from Bie et al. (2023)

Scaling Up the Model Size

The Pathways Autoregressive Text-to-Image (Parti) model:

- treat image generation as a sequence-to-sequence problem
- text prompt is input to encoder
- sequence of image tokens is output of decoder
- ViT-VQGAN takes in the image tokens and generates a high-quality image



Scaling Up the Model Size

Prompt: A portrait photo of a kangaroo wearing an orange hoodie and blue sunglasses standing on the grass in front of the Sydney Opera House holding a sign on the chest that says Welcome Friends!

Parti with different model sizes

Figure from https://sites.research.google/parti/

Watermarking & Attribution

- **Watermarking**
  - A digital watermark allows one to identify when an image has been created by a model
  - Most methods for image generation (GANs, VAEs, stable diffusion) can be augmented with watermarking
- **Fake-image Detection**
  - Goal: identify fakes even without a watermark
- **Model Attribution**
  - Identify which generative model created an image (e.g. Dalle-2 vs. SDXL)
  - Very successful (natural watermarks)
- **Image Attribution**
  - Goal: identify the source images that led to the generation of a new image
  - Extremely challenging

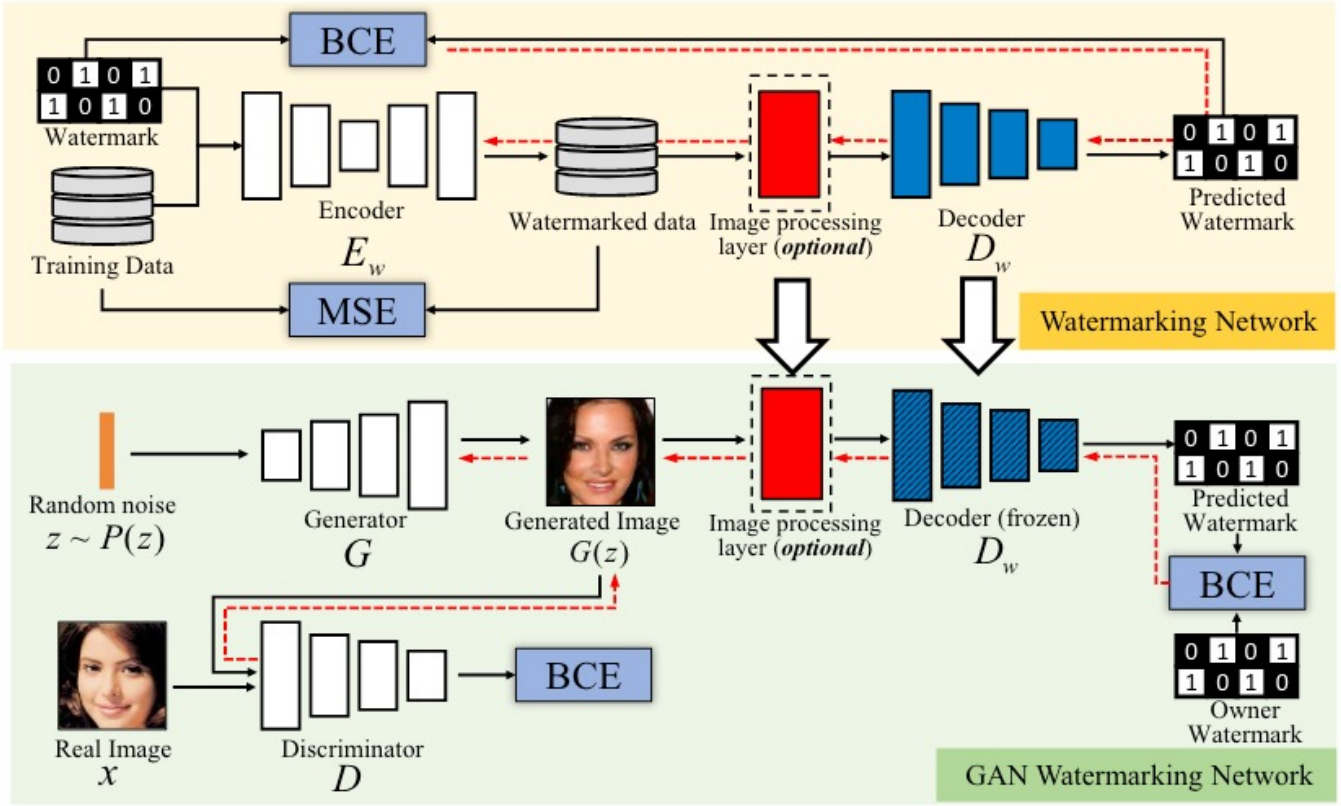


Figure from Fei et al. (2022)

SOCIETAL IMPACTS OF IMAGE GENERATION

Societal Impacts of Image Generation

Pros

- New tools for artists
- Faster creation of memes

Cons

- Copyright infringement / loss of work for artists
- Societal decrease in creativity
- Potential to create dehumanizing content
- Fake news / false realities / increased difficulty of fact checking
- Not rooted in reality
- Video generation is around the corner

https://www.bbcearth.com/flying-draco-lizard