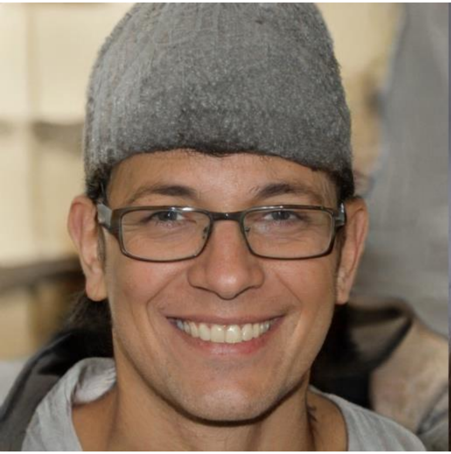
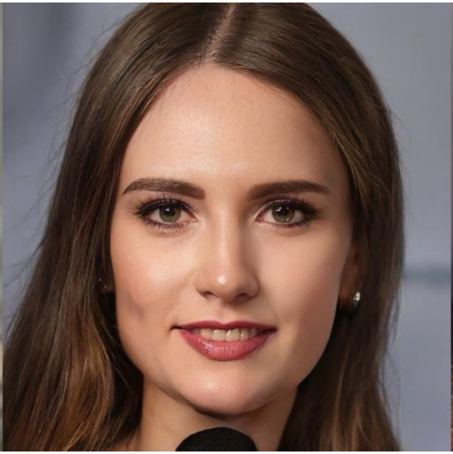
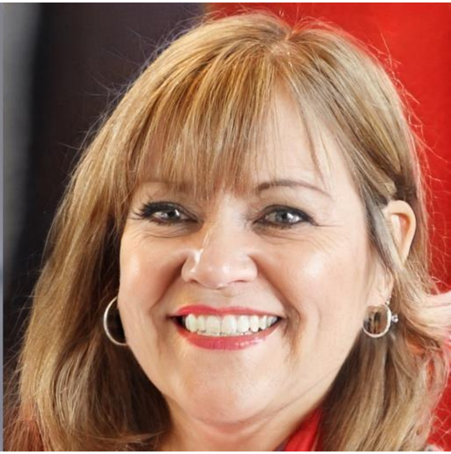
CSC413/2516 Lecture 9: Generative Models: GAN, VAE, LLMs

Bo Wang

Overview

Quiz: Which face image is fake?

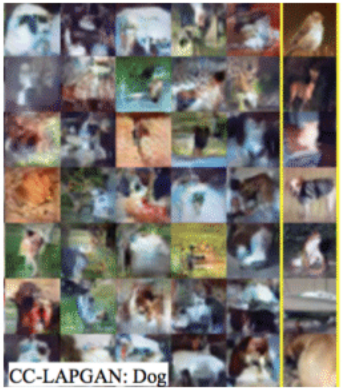




A B C

Overview

- In generative modeling, we'd like to train a network that models a distribution, such as a distribution over images.
- One way to judge the quality of the model is to sample from it.
- This field has seen rapid progress:

2009

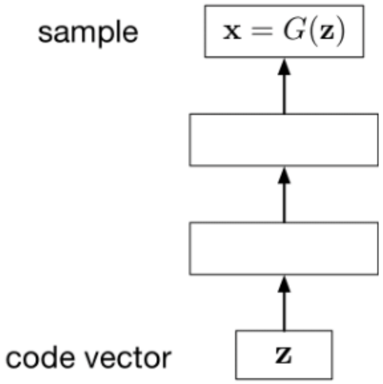


2015

2018

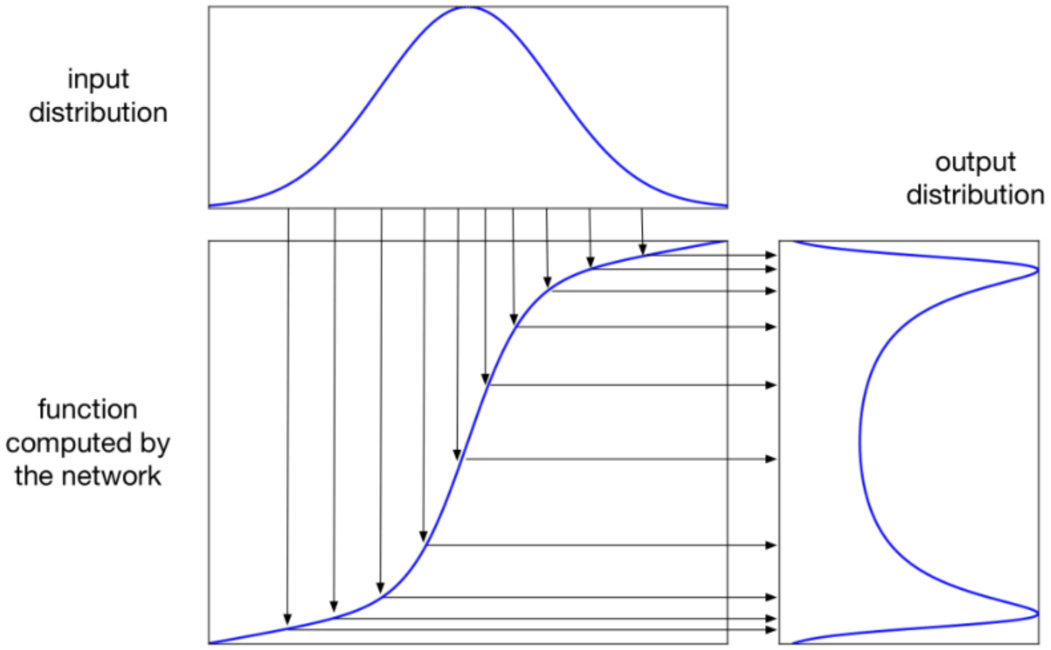
Generator Networks

- Autoregressive models explicitly predict a distribution at each step.
- Another approach to generative modeling is to train a neural net to produce approximate samples from the distribution.
- Start by sampling the code vector **z** from a fixed, simple distribution (e.g. spherical Gaussian)
- The generator network computes a differentiable function G mapping **z** to an **x** in data space

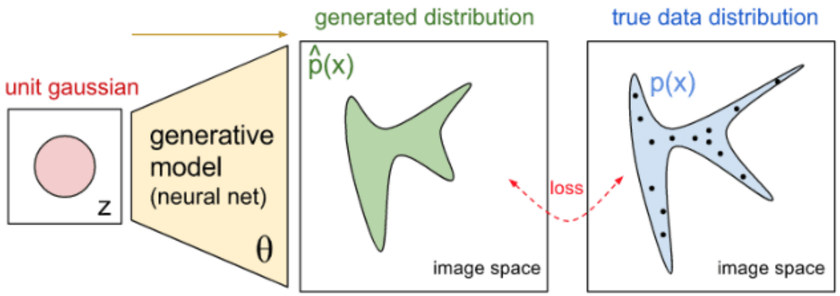


Generator Networks

A 1-dimensional example:

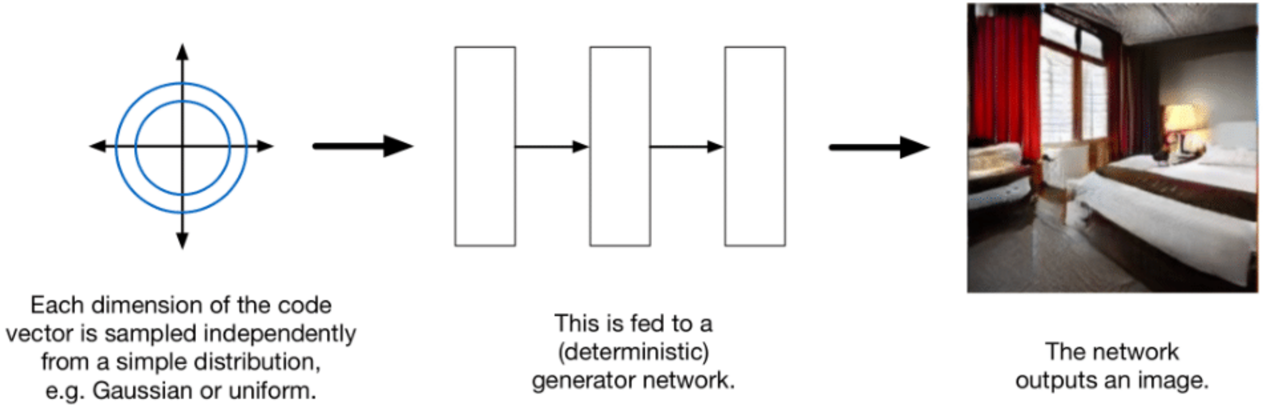


Generator Networks



https://blog.openai.com/generative-models/

Generator Networks


Each dimension of the code vector is sampled independently from a simple distribution, e.g. Gaussian or uniform.
This is fed to a (deterministic) generator network.
The network outputs an image.

This sort of architecture sounded preposterous to many of us, but amazingly, it works.

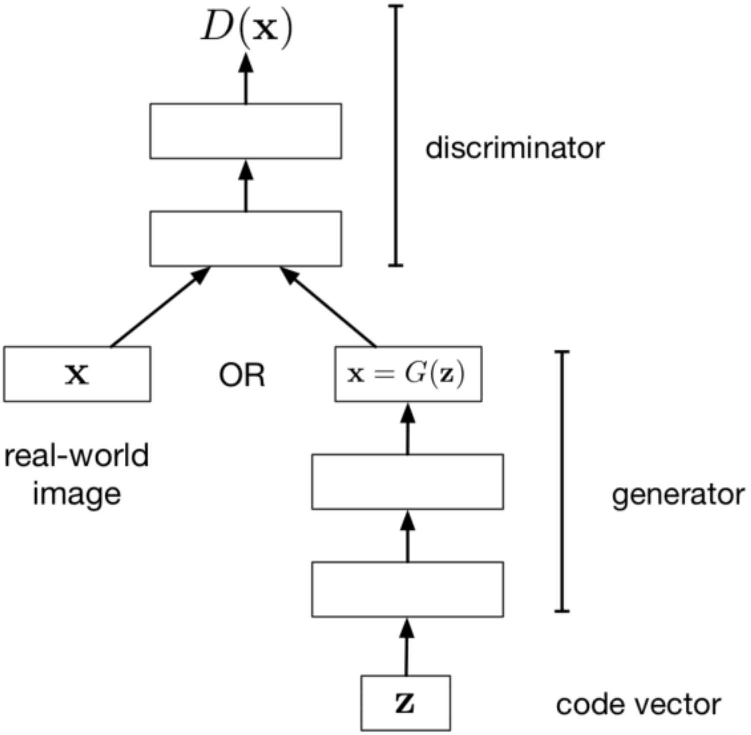
Generative Adversarial Networks

- Implicit generative models learn a mapping from random noise vectors to things that look like, e.g., images
- The advantage of implicit generative models: if you have some criterion for evaluating the quality of samples, then you can compute its gradient with respect to the network parameters, and update the network's parameters to make the sample a little better

Generative Adversarial Networks

- Implicit generative models learn a mapping from random noise vectors to things that look like, e.g., images
- The advantage of implicit generative models: if you have some criterion for evaluating the quality of samples, then you can compute its gradient with respect to the network parameters, and update the network's parameters to make the sample a little better
- The idea behind Generative Adversarial Networks (GANs): train two different networks
  - The generator network tries to produce realistic-looking samples
  - The discriminator network tries to figure out whether an image came from the training set or the generator network
- The generator network tries to fool the discriminator network

Generative Adversarial Networks



Generative Adversarial Networks

- Let D denote the discriminator's predicted probability of being data
- Discriminator's cost function: cross-entropy loss for task of classifying real vs. fake images

$$\mathcal{J}_D = \mathbb{E}_{\mathbf{x}\sim\mathcal{D}}[-\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z}}[-\log(1 - D(G(\mathbf{z})))]$$

- One possible cost function for the generator: the opposite of the discriminator's

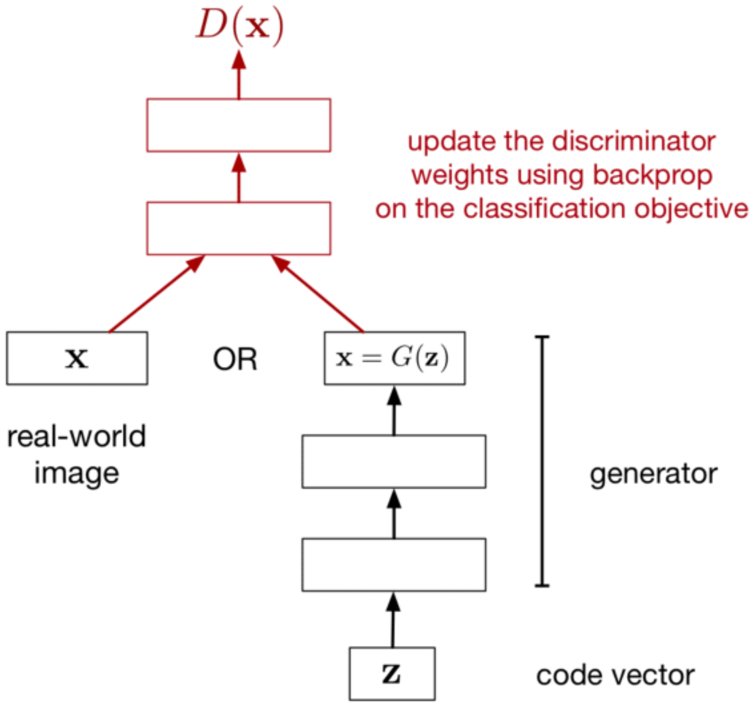
$$\begin{aligned}\mathcal{J}_G &= -\mathcal{J}_D \\ &= \text{const} + \mathbb{E}_{\mathbf{z}}[\log(1 - D(G(\mathbf{z})))]\end{aligned}$$

- This is called the minimax formulation, since the generator and discriminator are playing a zero-sum game against each other:

$$\max_G \min_D \mathcal{J}_D$$

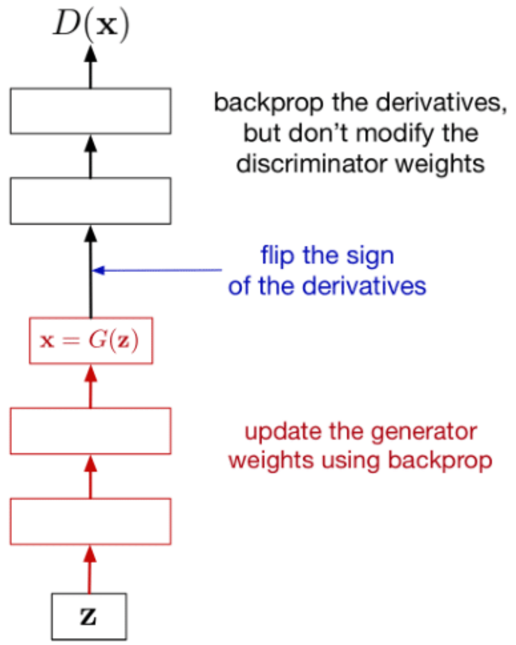
Generative Adversarial Networks

Updating the discriminator:


$D(\mathbf{x})$
update the discriminator weights using backprop on the classification objective
$\mathbf{x}$
OR
$\mathbf{x} = G(\mathbf{z})$
real-world image
generator
$\mathbf{z}$
code vector

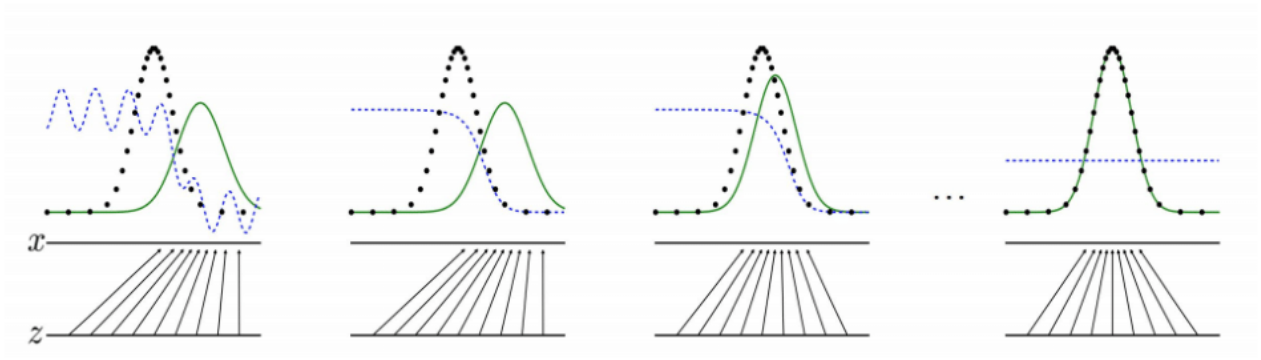
Generative Adversarial Networks

Updating the generator:


$D(\mathbf{x})$
backprop the derivatives, but don't modify the discriminator weights
flip the sign of the derivatives
$\mathbf{x} = G(\mathbf{z})$
update the generator weights using backprop
$\mathbf{z}$

Generative Adversarial Networks

Alternating training of the generator and discriminator:



A Better Cost Function

- We introduced the minimax cost function for the generator:

$$\mathcal{J}_G = \mathbb{E}_{\mathbf{z}}[\log(1 - D(G(\mathbf{z})))]$$

- One problem with this is saturation.
- Recall from our lecture on classification: when the prediction is really wrong,
    - "Logistic + squared error" gets a weak gradient signal
    - "Logistic + cross-entropy" gets a strong gradient signal
- Here, if the generated sample is really bad, the discriminator's prediction is close to 0, and the generator's cost is flat.

A Better Cost Function

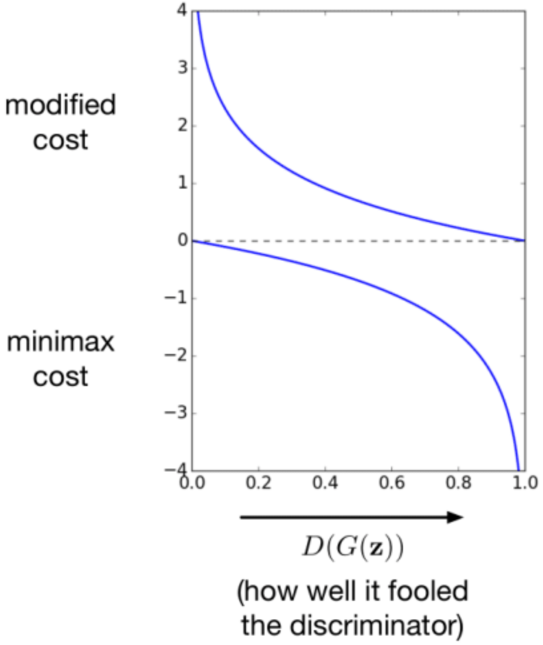
- Original minimax cost:

$$\mathcal{J}_G = \mathbb{E}_{\mathbf{z}}[\log(1 - D(G(\mathbf{z})))]$$

- Modified generator cost:

$$\mathcal{J}_G = \mathbb{E}_{\mathbf{z}}[-\log D(G(\mathbf{z}))]$$

- This fixes the saturation problem.



Generative Adversarial Networks

- Since GANs were introduced in 2014, there have been hundreds of papers introducing various architectures and training methods.
- Most modern architectures are based on the Deep Convolutional GAN (DC-GAN), where the generator and discriminator are both conv nets.
- GAN Zoo: `https://github.com/hindupuravinash/the-gan-zoo`
    - Good source of horrible puns (VEEGAN, Checkhov GAN, etc.)

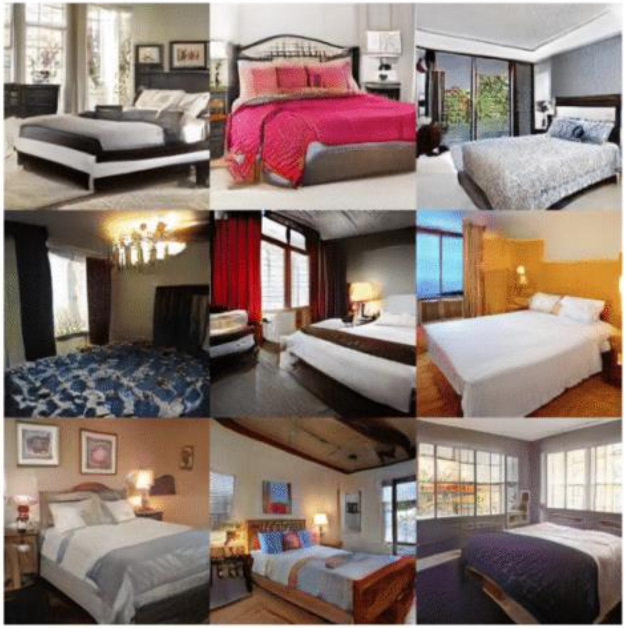
GAN Samples

Celebrities:

Karras et al., 2017. Progressive growing of GANs for improved quality, stability, and variation

GAN Samples

Bedrooms:



Karras et al., 2017. Progressive growing of GANs for improved quality, stability, and variation

GAN Samples

ImageNet object categories (by BigGAN, a much larger model with a bunch more engineering tricks):

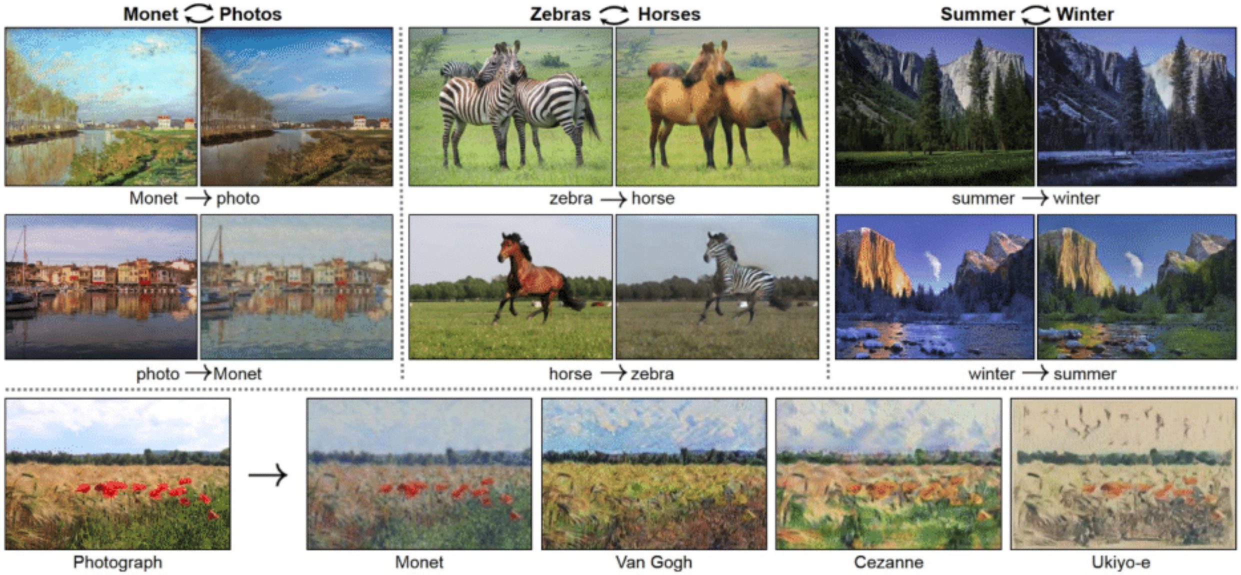
Brock et al., 2019. Large scale GAN training for high fidelity natural image synthesis.

GAN Samples

- GANs revolutionized generative modeling by producing crisp, high-resolution images.
- The catch: we don't know how well they're modeling the distribution.
    - Can't measure the log-likelihood they assign to held-out data.
    - Could they be memorizing training examples? (E.g., maybe they sometimes produce photos of real celebrities?)
    - We have no way to tell if they are dropping important modes from the distribution.
    - See Wu et al., "On the quantitative analysis of decoder-based generative models" for partial answers to these questions.

CycleGAN

Style transfer problem: change the style of an image while preserving the content.

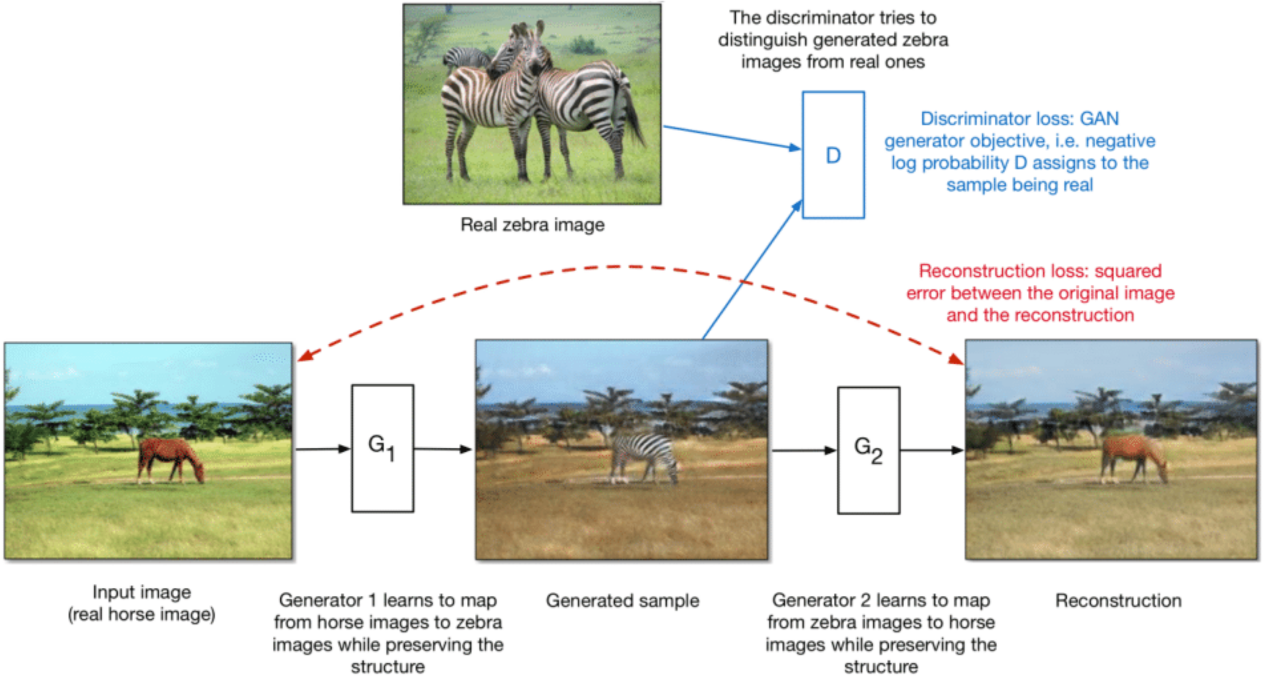


Data: Two unrelated collections of images, one for each style

CycleGAN

- If we had paired data (same content in both styles), this would be a supervised learning problem. But this is hard to find.
- The CycleGAN architecture learns to do it from unpaired data.
  - Train two different generator nets to go from style 1 to style 2, and vice versa.
  - Make sure the generated samples of style 2 are indistinguishable from real images by a discriminator net.
  - Make sure the generators are cycle-consistent: mapping from style 1 to style 2 and back again should give you almost the original image.

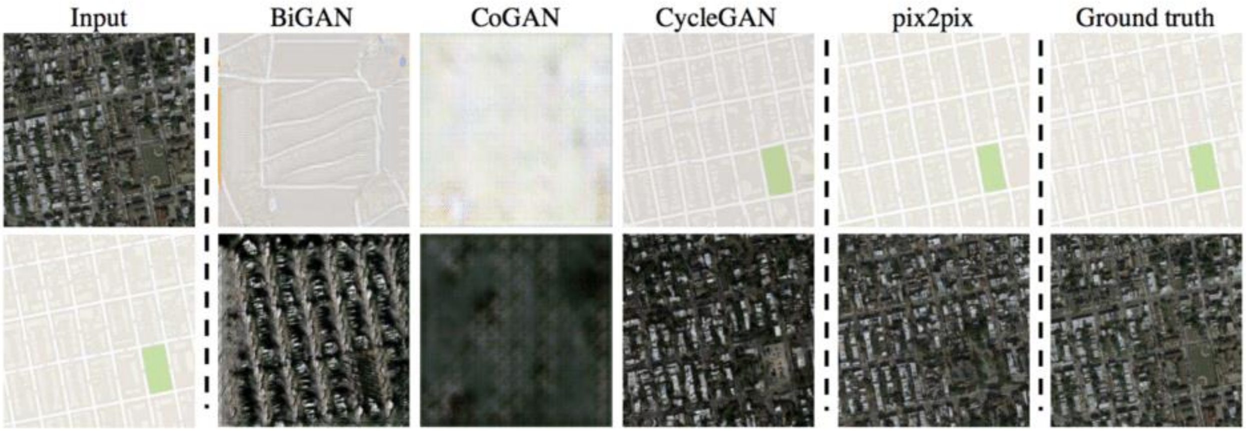
CycleGAN


Real zebra image
The discriminator tries to distinguish generated zebra images from real ones
D
Discriminator loss: GAN generator objective, i.e. negative log probability D assigns to the sample being real
Reconstruction loss: squared error between the original image and the reconstruction
G1
G2
Input image (real horse image)
Generator 1 learns to map from horse images to zebra images while preserving the structure
Generated sample
Generator 2 learns to map from zebra images to horse images while preserving the structure
Reconstruction

Total loss = discriminator loss + reconstruction loss

CycleGAN

Style transfer between aerial photos and maps:

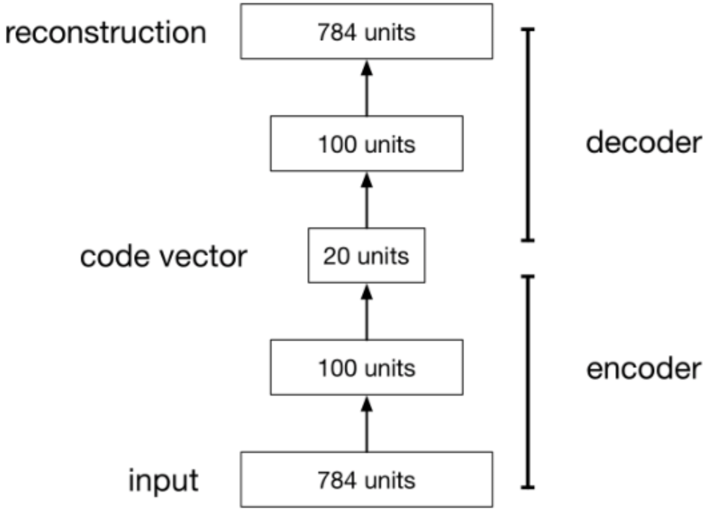


After the break

After the break: **Variational Auto-Encoder (VAE)**

Autoencoders

- An autoencoder is a feed-forward neural net whose job it is to take an input **x** and predict **x**.
- To make this non-trivial, we need to add a bottleneck layer whose dimension is much smaller than the input.



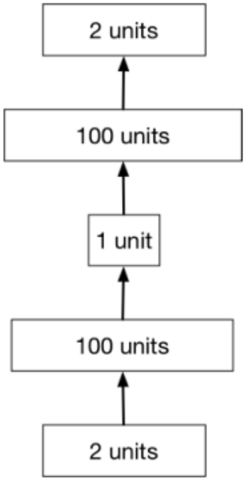
Autoencoders

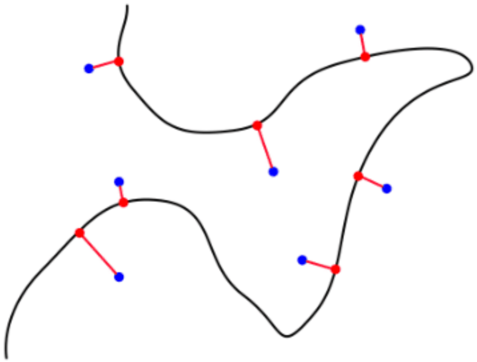
Why autoencoders?

- Map high-dimensional data to two dimensions for visualization
- Compression (i.e. reducing the file size)
  - Note: this requires a VAE, not just an ordinary autoencoder.
- Learn abstract features in an unsupervised way so you can apply them to a supervised task
  - Unlabled data can be much more plentiful than labeled data
- Learn a semantically meaningful representation where you can, e.g., interpolate between different images.

Deep Autoencoders

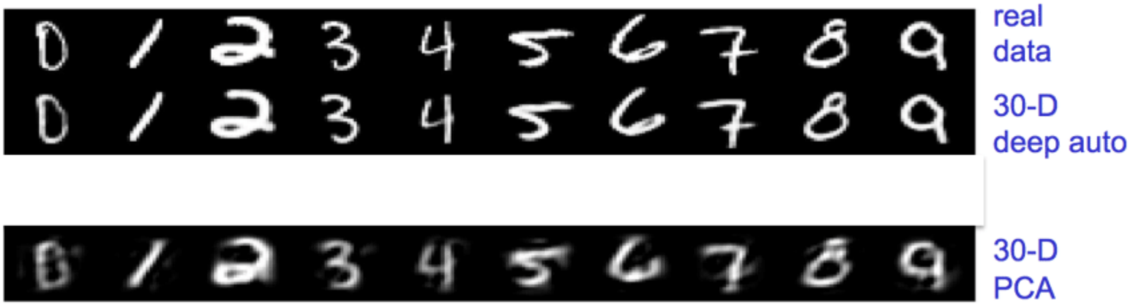
- Deep nonlinear autoencoders learn to project the data onto a low-dimensional nonlinear manifold.
- This manifold is the image of the decoder.
- This is a kind of nonlinear dimensionality reduction.





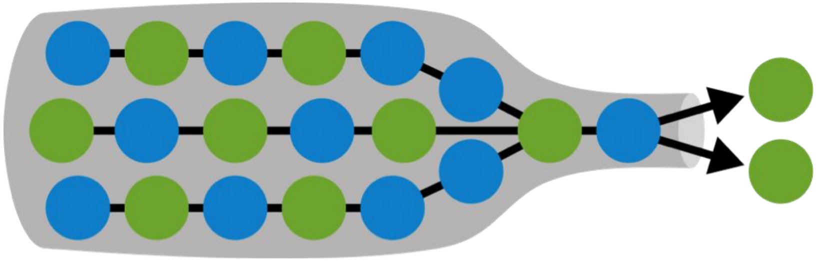
Deep Autoencoders

- Nonlinear autoencoders can learn more powerful codes for a given dimensionality, compared with linear autoencoders (PCA)

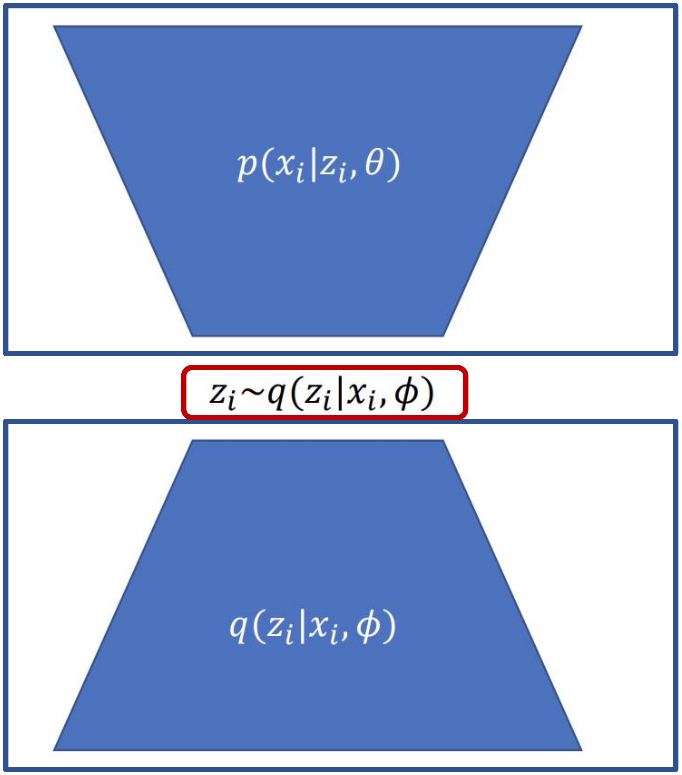


Deep Autoencoders

- Some limitations of autoencoders
  - They're not generative models, so they don't define a distribution
  - How to choose the latent dimension?



Variational Auto-encoder (VAE)

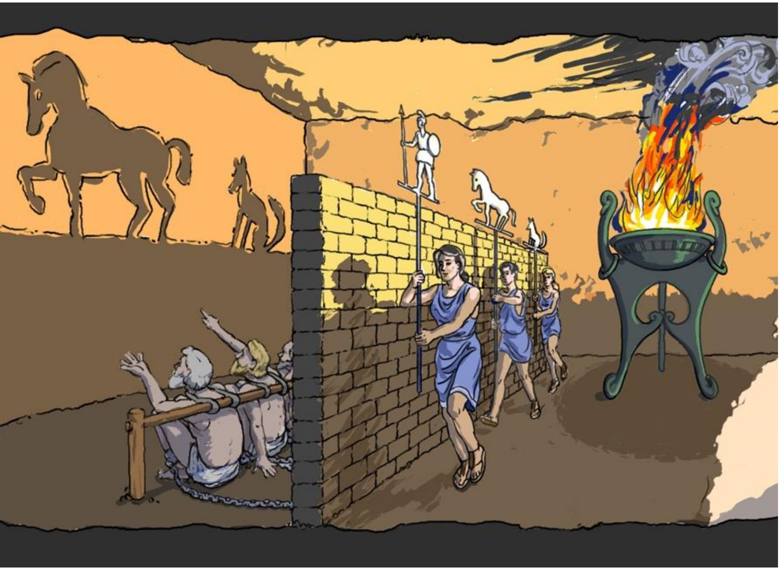


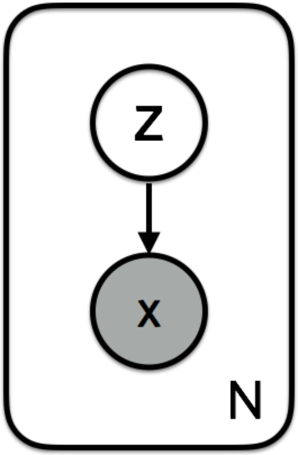
Decoder learns the generative process given the sampled latent vectors.

Sampling process in the middle.

Encoder learns the distribution of latent space given the observations.

Observation Model





Source: https://iagtm.pressbooks.com/chapter/story-platos-allegory-of-the-cave/

Observation Model

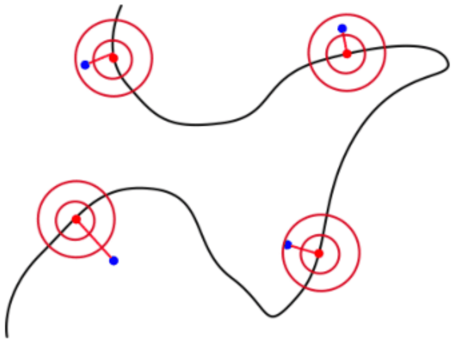
- Consider training a generator network with maximum likelihood.

$$p(\mathbf{x}) = \int p(\mathbf{z}) p(\mathbf{x} \mid \mathbf{z}) \, \mathrm{d}\mathbf{z}$$

- One problem: if $\mathbf{z}$ is low-dimensional and the decoder is deterministic, then $p(\mathbf{x}) = 0$ almost everywhere!
  - The model only generates samples over a low-dimensional sub-manifold of $\mathcal{X}$.
- Solution: define a noisy observation model, e.g.

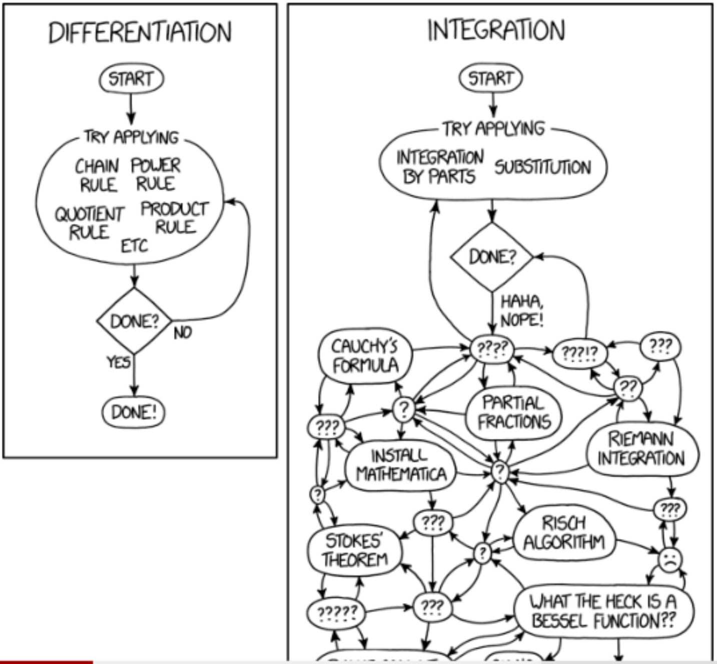
$$p(\mathbf{x} \mid \mathbf{z}) = \mathcal{N}(\mathbf{x}; G_{\boldsymbol{\theta}}(\mathbf{z}), \eta \mathbf{I}),$$

where $G_{\boldsymbol{\theta}}$ is the function computed by the decoder with parameters $\boldsymbol{\theta}$.



Observation Model

- At least $p(\mathbf{x}) = \int p(\mathbf{z}) p(\mathbf{x} \mid \mathbf{z}) \, \mathrm{d}\mathbf{z}$ is well-defined, but how can we compute it?
- Integration, according to XKCD:



Observation Model

- At least $p(\mathbf{x}) = \int p(\mathbf{z})p(\mathbf{x} \mid \mathbf{z})\,\mathrm{d}\mathbf{z}$ is well-defined, but how can we compute it?
  - The decoder function $G_\theta(\mathbf{z})$ is very complicated, so there's no hope of finding a closed form.
- Instead, we will try to maximize a lower bound on $\log p(\mathbf{x})$.
  - The math is essentially the same as in the EM algorithm from CSC411.

Variational Inference

- We obtain the lower bound using Jensen's Inequality: for a convex function h of a random variable X,

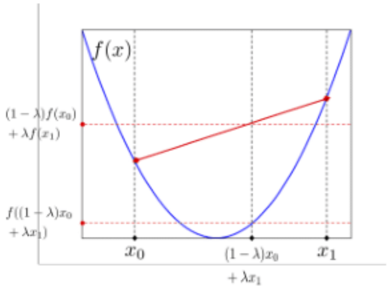
$$\mathbb{E}[h(X)] \geq h(\mathbb{E}[X])$$

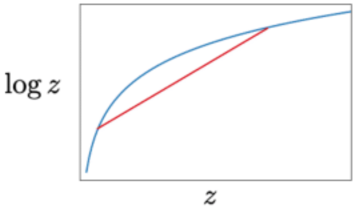
Therefore, if h is concave (i.e. $-h$ is convex),

$$\mathbb{E}[h(X)] \leq h(\mathbb{E}[X])$$

- The function $\log z$ is concave. Therefore,

$$\mathbb{E}[\log X] \leq \log \mathbb{E}[X]$$


$f(x)$
$(1-\lambda)f(x_0)$ $+\lambda f(x_1)$
$f((1-\lambda)x_0$ $+\lambda x_1)$
x_0
$(1-\lambda)x_0$ $+\lambda x_1$
x_1



Variational Inference

- Suppose we have some distribution $q(\mathbf{z})$. (We'll see later where this comes from.)
- We use Jensen's Inequality to obtain the lower bound.

$$
\begin{aligned}
\log p(\mathbf{x}) &= \log \int p(\mathbf{z})\, p(\mathbf{x}|\mathbf{z})\, \mathrm{d}\mathbf{z} \\
&= \log \int q(\mathbf{z})\, \frac{p(\mathbf{z})}{q(\mathbf{z})} p(\mathbf{x}|\mathbf{z})\, \mathrm{d}\mathbf{z} \\
&\geq \int q(\mathbf{z}) \log \left[\frac{p(\mathbf{z})}{q(\mathbf{z})}\, p(\mathbf{x}|\mathbf{z}) \right] \mathrm{d}\mathbf{z} \\
&= \mathbb{E}_q \left[\log \frac{p(\mathbf{z})}{q(\mathbf{z})} \right] + \mathbb{E}_q \left[\log p(\mathbf{x}|\mathbf{z}) \right]
\end{aligned}
$$

$$\mathbb{E}[\log X] \leq \log \mathbb{E}[X]$$
$$\text{(Jensen's Inequality)}$$

- We'll look at these two terms in turn.

Variational Inference

- The first term we'll look at is $\mathbb{E}_q [\log p(\mathbf{x}|\mathbf{z})]$
- Since we assumed a Gaussian observation model,

$$\begin{aligned}\log p(\mathbf{x}|\mathbf{z}) &= \log \mathcal{N}(\mathbf{x}; G_{\boldsymbol{\theta}}(\mathbf{z}), \eta \mathbf{I}) \\ &= \log \left[\frac{1}{(2\pi\eta)^{D/2}} \exp \left(-\frac{1}{2\eta} \|\mathbf{x} - G_{\boldsymbol{\theta}}(\mathbf{z})\|^2 \right) \right] \\ &= -\frac{1}{2\eta} \|\mathbf{x} - G_{\boldsymbol{\theta}}(\mathbf{z})\|^2 + \text{const}\end{aligned}$$

- So this term is the expected squared error in reconstructing $\mathbf{x}$ from $\mathbf{z}$. We call it the reconstruction term.

Variational Inference

- The second term is $\mathbb{E}_q\left[\log \frac{p(\mathbf{z})}{q(\mathbf{z})}\right]$.
- This is just $-\mathrm{D}_{\mathrm{KL}}(q(\mathbf{z})\|p(\mathbf{z}))$, where D_{KL} is the Kullback-Leibler (KL) divergence

$$\mathrm{D}_{\mathrm{KL}}(q(\mathbf{z})\|p(\mathbf{z})) \triangleq \mathbb{E}_q\left[\log \frac{q(\mathbf{z})}{p(\mathbf{z})}\right]$$

  - KL divergence is a widely used measure of distance between probability distributions, though it doesn't satisfy the axioms to be a distance metric.
  - More details in tutorial.
- Typically, $p(\mathbf{z}) = \mathcal{N}(\mathbf{0}, \mathbf{I})$. Hence, the KL term encourages q to be close to $\mathcal{N}(\mathbf{0}, \mathbf{I})$.

Variational Inference

- Hence, we're trying to maximize the variational lower bound, or variational free energy:

$$\log p(\mathbf{x}) \geq \mathcal{F}(\boldsymbol{\theta}, q) = \mathbb{E}_q \left[\log p(\mathbf{x}|\mathbf{z})\right] - \mathrm{D}_{\mathrm{KL}}(q \| p).$$

- The term "variational" is a historical accident: "variational inference" used to be done using variational calculus, but this isn't how we train VAEs.
- We'd like to choose q to make the bound as tight as possible.
- It's possible to show that the gap is given by:

$$\log p(\mathbf{x}) - \mathcal{F}(\boldsymbol{\theta}, q) = \mathrm{D}_{\mathrm{KL}}(q(\mathbf{z}) \| p(\mathbf{z}|\mathbf{x})).$$

  Therefore, we'd like q to be as close as possible to the posterior distribution $p(\mathbf{z}|\mathbf{x})$.

- Let's think about the role of each of the two terms.
- The reconstruction term

$$\mathbb{E}_q[\log p(\mathbf{x}|\mathbf{z})] = -\frac{1}{2\sigma^2}\mathbb{E}_q[\|\mathbf{x} - G_\theta(\mathbf{z})\|^2] + \text{const}$$

  is minimized when q is a point mass on

$$\mathbf{z}_* = \arg\min_{\mathbf{z}} \|\mathbf{x} - G_\theta(\mathbf{z})\|^2.$$

- But a point mass would have infinite KL divergence. (Exercise: check this.) So the KL term forces q to be more spread out.

Reparameterization Trick

- To fit q, let's assign it a parametric form, in particular a Gaussian distribution: $q(\mathbf{z}) = \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}, \boldsymbol{\Sigma})$, where $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_K)$ and $\boldsymbol{\Sigma} = \mathrm{diag}(\sigma_1^2, \ldots, \sigma_K^2)$.
- In general, it's hard to differentiate through an expectation. But for Gaussian q, we can apply the reparameterization trick:

$$z_i = \mu_i + \sigma_i \epsilon_i,$$

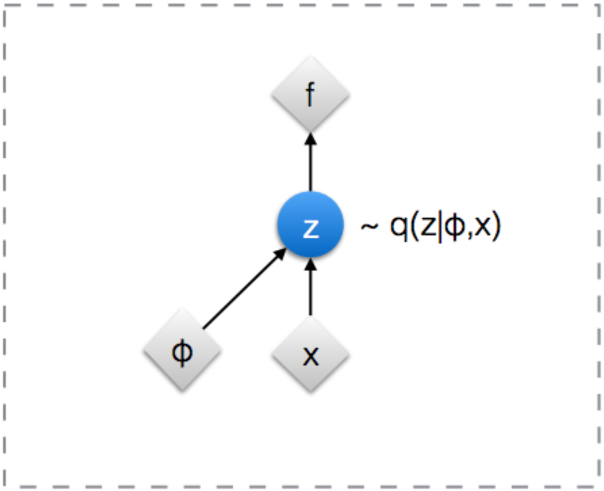
where $\epsilon_i \sim \mathcal{N}(0, 1)$.
- Hence,

$$\overline{\mu_i} = \overline{z_i} \qquad \overline{\sigma_i} = \overline{z_i}\epsilon_i.$$

- This is exactly analogous to how we derived the backprop rules for dropout

Reparameterization Trick

Original form

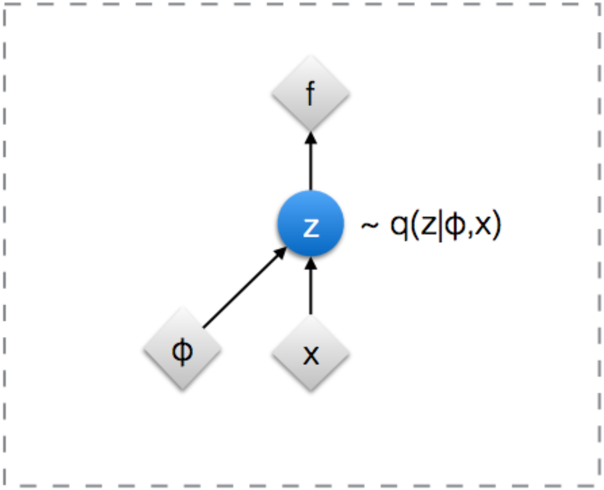


◆ : Deterministic node

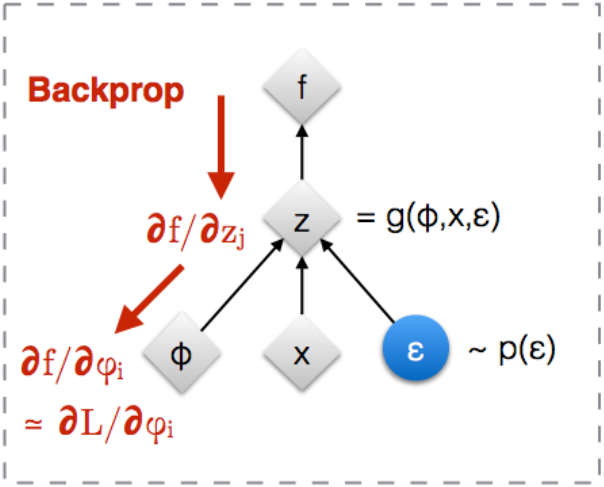
● : Random node

Reparameterization Trick

Original form



Reparameterised form



Backprop
f
$\partial f/\partial z_j$
z
= g(ϕ,x,ε)
$\partial f/\partial \varphi_i$
$\simeq \partial L/\partial \varphi_i$
ϕ
x
ε
~ p(ε)

◆ : Deterministic node

● : Random node

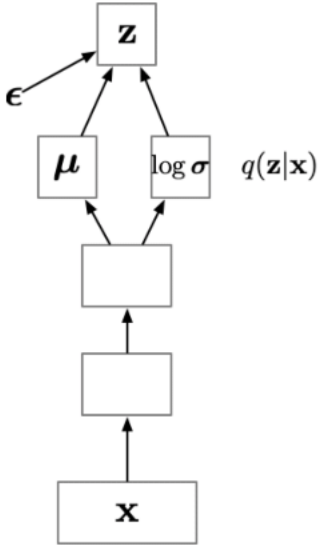
[Kingma, 2013]
[Bengio, 2013]
[Kingma and Welling 2014]
[Rezende et al 2014]

Amortization

- This suggests one strategy for learning the decoder. For each training example,
    1. Fit q to approximate the posterior for the current $\mathbf{x}$ by doing many steps of gradient ascent on $\mathcal{F}$.
    2. Update the decoder parameters $\boldsymbol{\theta}$ with gradient ascent on $\mathcal{F}$.
- **Problem:** this requires an expensive iterative procedure for every training example, so it will take a long time to process the whole training set.

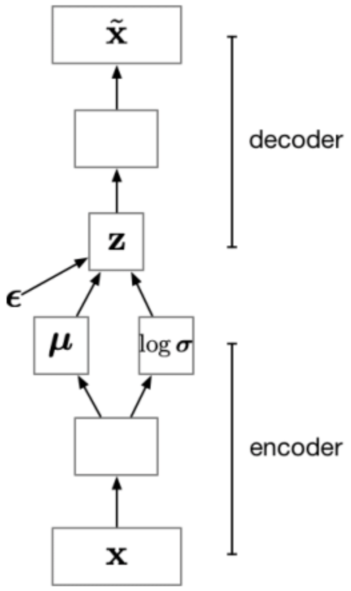
Amortization

- **Idea:** amortize the cost of inference by learning an inference network which predicts $(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ as a function of $\mathbf{x}$.
- The outputs of the inference net are $\boldsymbol{\mu}$ and $\log \boldsymbol{\sigma}$. (The log representation ensures $\boldsymbol{\sigma} > 0$.)
- If $\boldsymbol{\sigma} \approx \mathbf{0}$, then this network essentially computes $\mathbf{z}$ deterministically, by way of $\boldsymbol{\mu}$.
    - But the KL term encourages $\boldsymbol{\sigma} > 0$, so in general $\mathbf{z}$ will be noisy.
- The notation $q(\mathbf{z}|\mathbf{x})$ emphasizes that q depends on $\mathbf{x}$, even though it's not actually a conditional distribution.

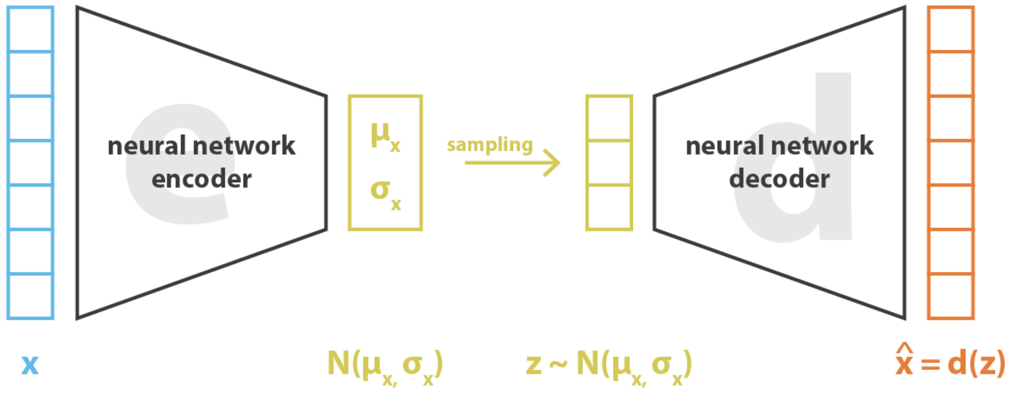


Amortization

- Combining this with the decoder network, we see the structure closely resembles an ordinary autoencoder. The inference net is like an encoder.
- Hence, this architecture is known as a variational autoencoder (VAE).
- The parameters of both the encoder and decoder networks are updated using a single pass of ordinary backprop.
  - The reconstruction term corresponds to squared error $\|\mathbf{x} - \tilde{\mathbf{x}}\|^2$, like in an ordinary VAE.
  - The KL term regularizes the representation by encouraging $\mathbf{z}$ to be more stochastic.



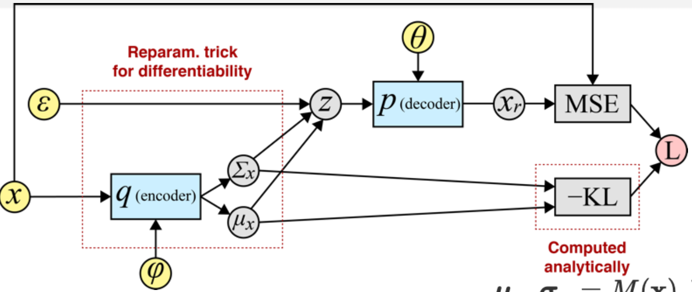
Variational Auto-encoder (VAE)



$$\text{loss} = \| x - \hat{x} \|^2 + \mathrm{KL}[\, N(\mu_x, \sigma_x), N(0, I)\,] = \| x - d(z) \|^2 + \mathrm{KL}[\, N(\mu_x, \sigma_x), N(0, I)\,]$$

Source: https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73

VAE - Summary



$\boldsymbol{\mu}_x, \boldsymbol{\sigma}_x = M(\mathbf{x}), \Sigma(\mathbf{x})$	Push $\mathbf{x}$ through encoder
$\boldsymbol{\epsilon} \sim \mathcal{N}(0,1)$	Sample noise
$\mathbf{z} = \boldsymbol{\epsilon}\boldsymbol{\sigma}_x + \boldsymbol{\mu}_x$	Reparameterize
$\mathbf{x}_r = p_{\boldsymbol{\theta}}(\mathbf{x} \mid \mathbf{z})$	Push $\mathbf{z}$ through decoder
$\text{recon. loss} = \text{MSE}(\mathbf{x}, \mathbf{x}_r)$	Compute reconstruction loss
$\text{var. loss} = -\text{KL}[\mathcal{N}(\boldsymbol{\mu}_x, \boldsymbol{\sigma}_x) \Vert \mathcal{N}(0, I)]$	Compute variational loss
$\text{L} = \text{recon. loss} + \text{var. loss}$	Combine losses

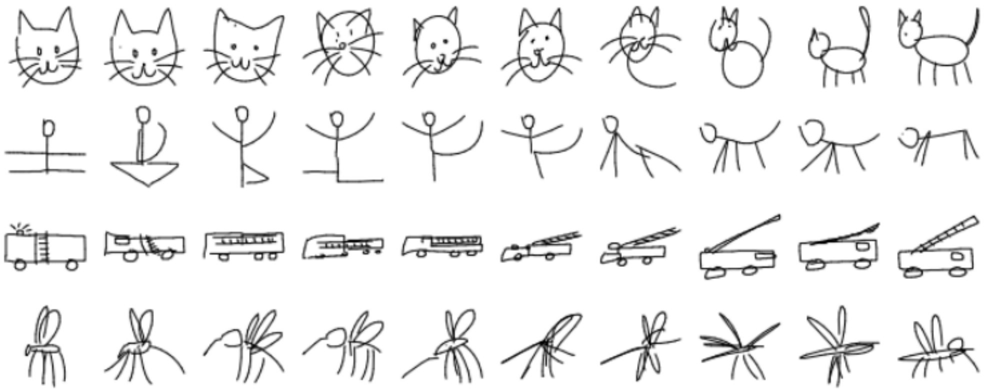
Source: https://gregorygundersen.com/blog/2018/04/29/reparameterization/

VAEs vs. Other Generative Models

- In short, a VAE is like an autoencoder, except that it's also a generative model (defines a distribution $p(\mathbf{x})$).
- Unlike autoregressive models, generation only requires one forward pass.
- Unlike reversible models, we can fit a low-dimensional latent representation. We'll see we can do interesting things with this...

Latent Space Interpolations

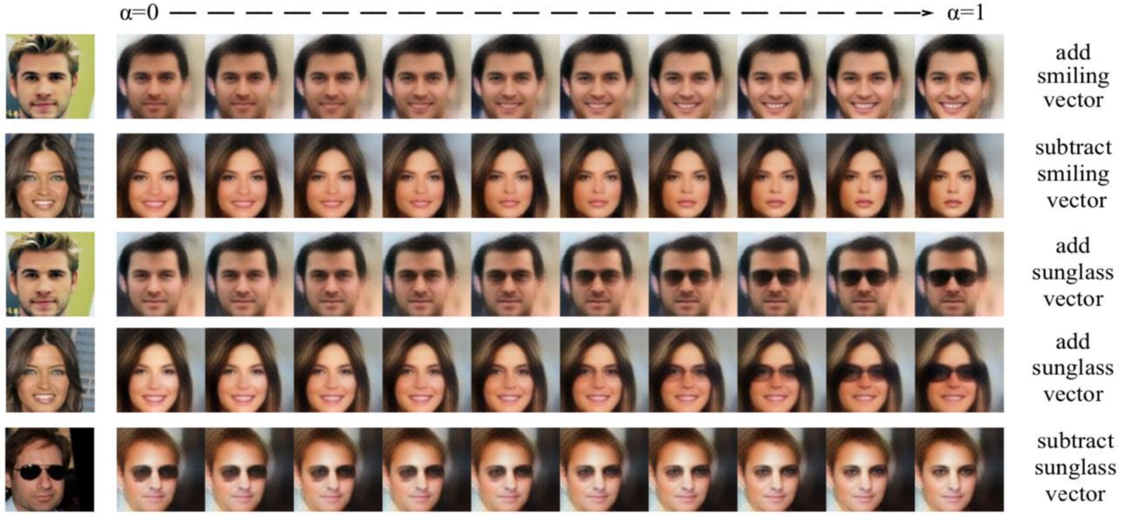
- You can often get interesting results by interpolating between two vectors in the latent space:



Ha and Eck, "A neural representation of sketch drawings"

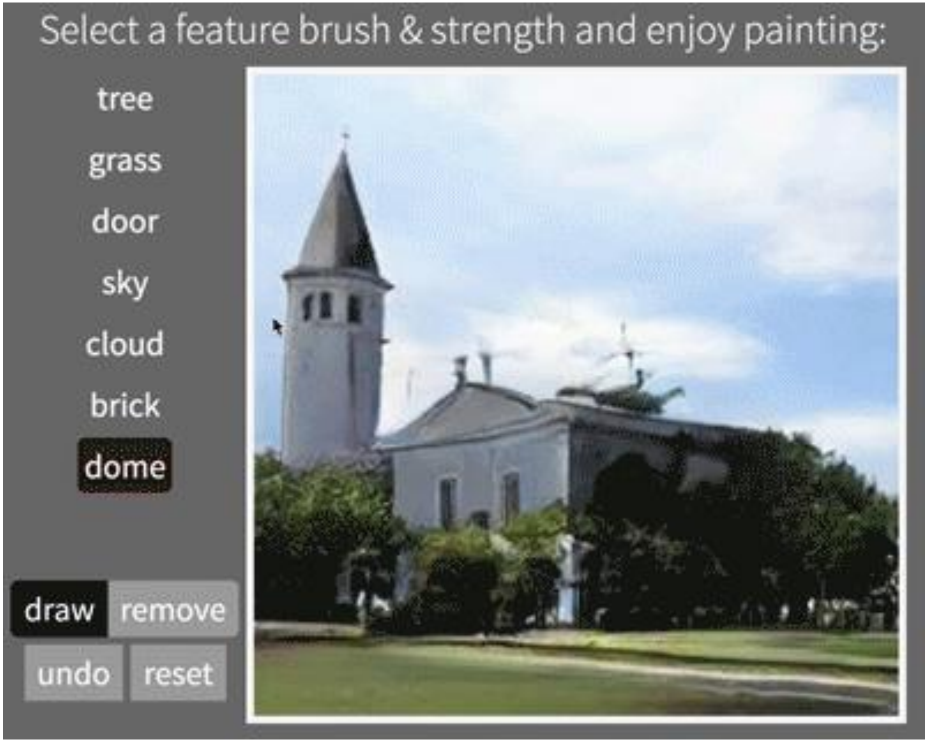
Latent Space Interpolations

- You can often get interesting results by interpolating between two vectors in the latent space:



https://arxiv.org/pdf/1610.00291.pdf

Latent Space Interpolations



Latent Space Interpolations

- Latent space interpolation of music:
  `https://magenta.tensorflow.org/music-vae`

After the break

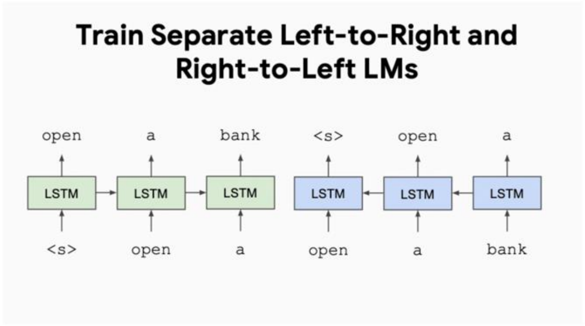
After the break: **Large Language Models (LLMs)**

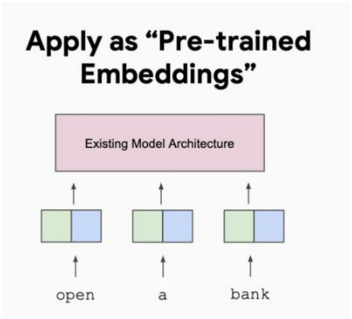
Large Language Models (LLMs)

Prior work: ELMo

ELMo (Peters et al., 2018; NAACL 2018 best paper)

- Train two separate **unidirectional** LMs (left-to-right and right-to-left) based on **LSTMs**
- **Feature-based** approach: pre-trained representations used as input to task-specific models
- Trained on **single sentences** from 1B word benchmark (Chelba et al., 2014)




Apply as "Pre-trained Embeddings"
Existing Model Architecture
open
a
bank

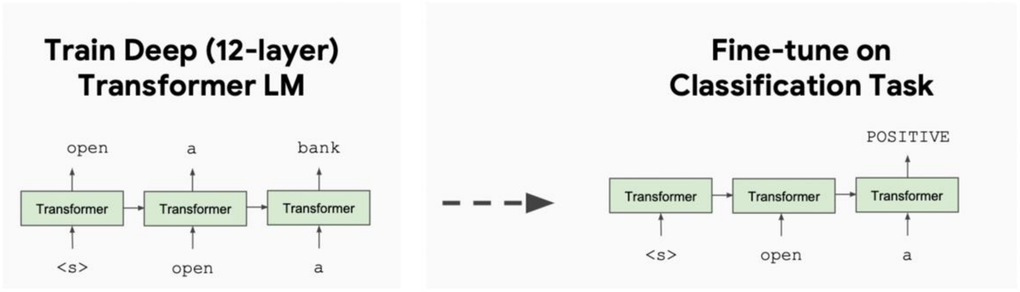
Source: COS 597G, Danqi Chen, Princeton University

Large Language Models (LLMs)

Prior work: OpenAI GPT

OpenAI GPT (Radford et al., 2018; released in 2018/6)

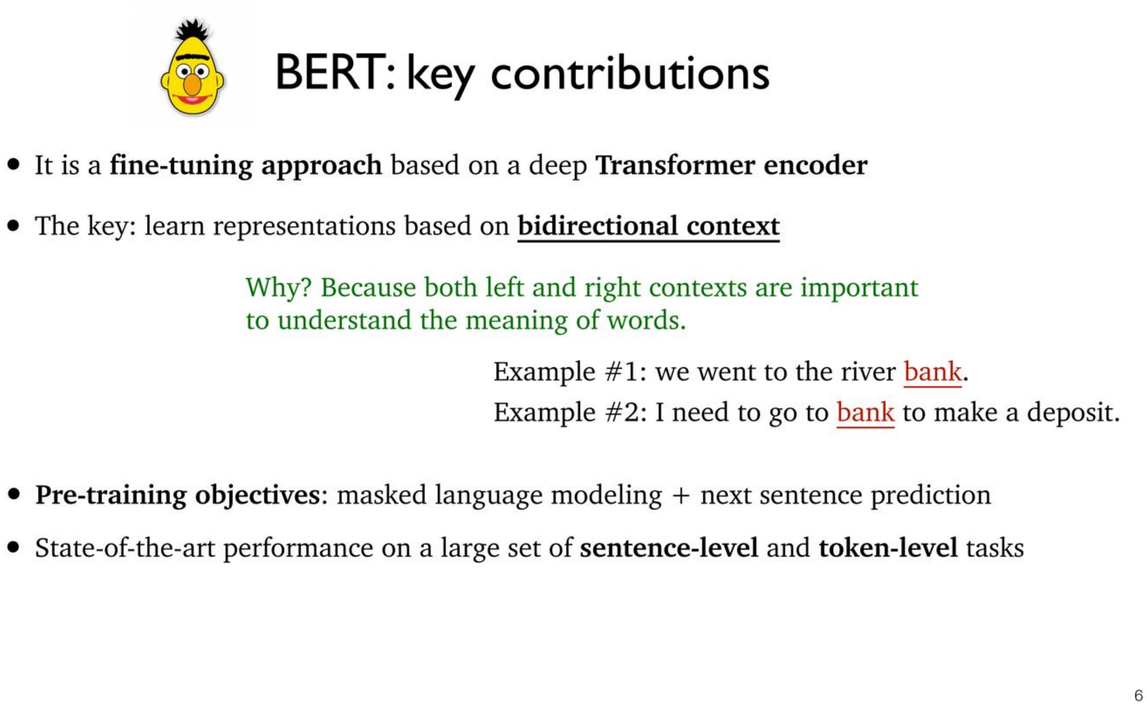
- Train one unidirectional LM (left-to-right) based on a deep **Transformer decoder**
- **Fine-tuning** approach: all pre-trained parameters are re-used & updated on downstream tasks
- Trained on 512-token segments on BooksCorpus — much **longer** context!



5

Source: COS 597G, Danqi Chen, Princeton University

Large Language Models (LLMs)



BERT: key contributions

- It is a **fine-tuning approach** based on a deep **Transformer encoder**
- The key: learn representations based on **<u>bidirectional context</u>**

  Why? Because both left and right contexts are important to understand the meaning of words.

  Example #1: we went to the river bank.

  Example #2: I need to go to bank to make a deposit.

- **Pre-training objectives**: masked language modeling + next sentence prediction
- State-of-the-art performance on a large set of **sentence-level** and **token-level** tasks

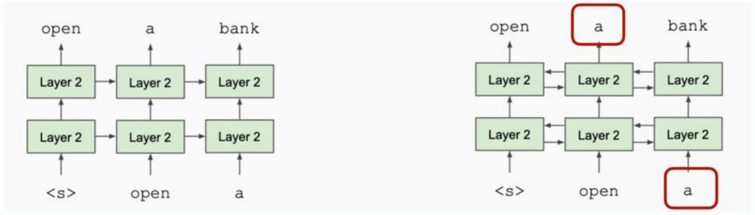
6

Source: COS 597G, Danqi Chen, Princeton University

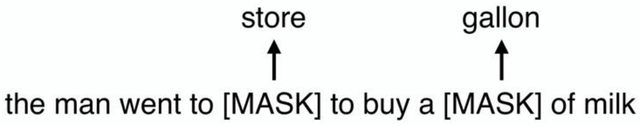
Large Language Models (LLMs)

Masked Language Modeling (MLM)

- Q: Why we can't do language modeling with bidirectional models?

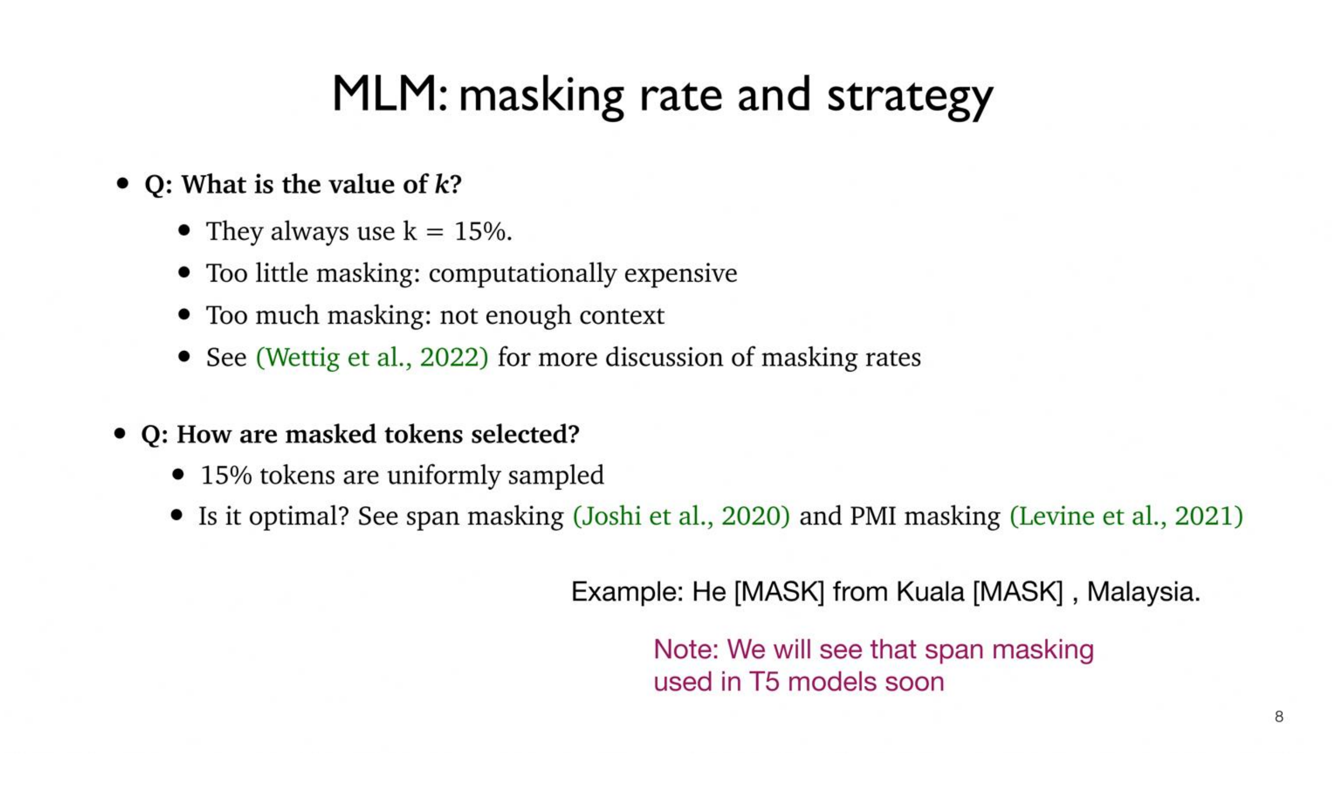


- Solution: Mask out k% of the input words, and then predict the masked words



Source: COS 597G, Danqi Chen, Princeton University

Large Language Models (LLMs)



MLM: masking rate and strategy

- **Q: What is the value of *k*?**
  - They always use k = 15%.
  - Too little masking: computationally expensive
  - Too much masking: not enough context
  - See (Wettig et al., 2022) for more discussion of masking rates

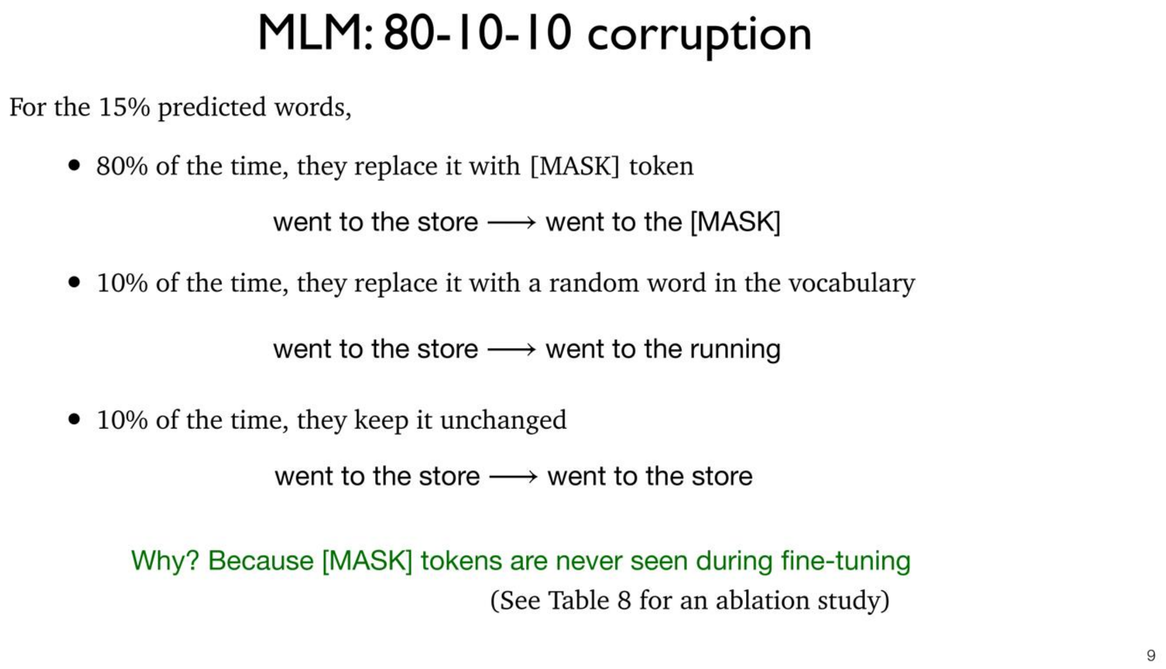
- **Q: How are masked tokens selected?**
  - 15% tokens are uniformly sampled
  - Is it optimal? See span masking (Joshi et al., 2020) and PMI masking (Levine et al., 2021)

Example: He [MASK] from Kuala [MASK] , Malaysia.

Note: We will see that span masking used in T5 models soon

Source: COS 597G, Danqi Chen, Princeton University

Large Language Models (LLMs)



MLM: 80-10-10 corruption

For the 15% predicted words,

- 80% of the time, they replace it with [MASK] token

  went to the store $\longrightarrow$ went to the [MASK]

- 10% of the time, they replace it with a random word in the vocabulary

  went to the store $\longrightarrow$ went to the running

- 10% of the time, they keep it unchanged

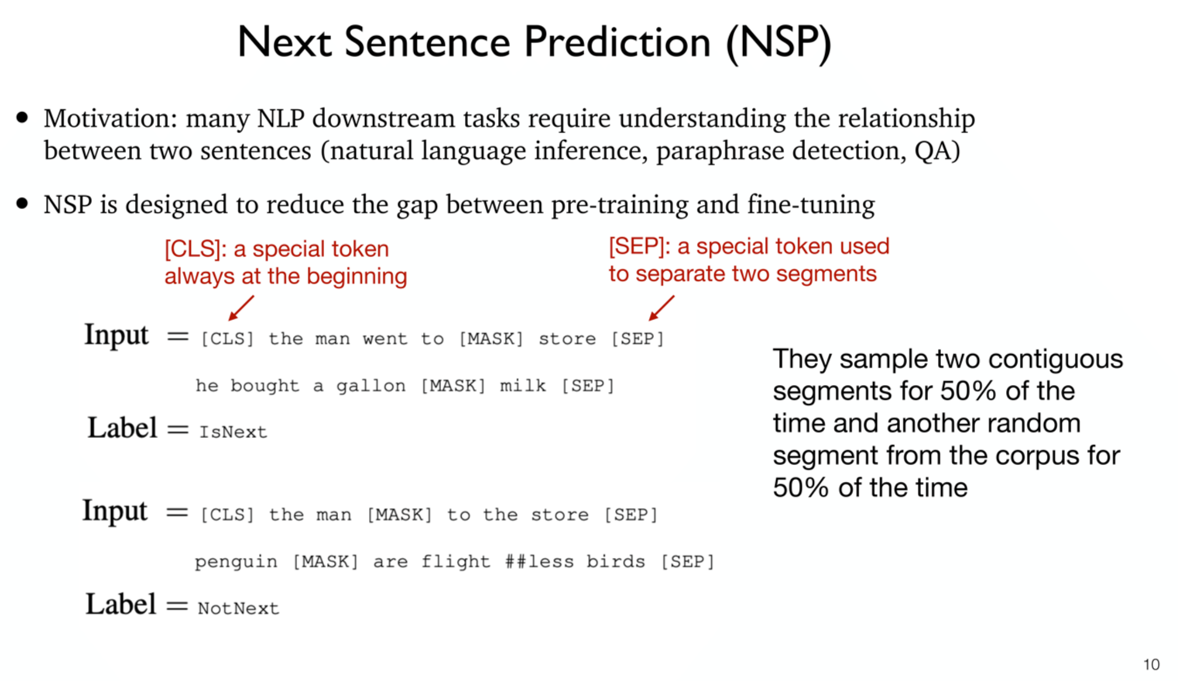
  went to the store $\longrightarrow$ went to the store

Why? Because [MASK] tokens are never seen during fine-tuning

(See Table 8 for an ablation study)

Source: COS 597G, Danqi Chen, Princeton University

Large Language Models (LLMs)



Next Sentence Prediction (NSP)

- Motivation: many NLP downstream tasks require understanding the relationship between two sentences (natural language inference, paraphrase detection, QA)
- NSP is designed to reduce the gap between pre-training and fine-tuning

[CLS]: a special token always at the beginning

[SEP]: a special token used to separate two segments

Input = `[CLS] the man went to [MASK] store [SEP]`

`he bought a gallon [MASK] milk [SEP]`

Label = `IsNext`

Input = `[CLS] the man [MASK] to the store [SEP]`

`penguin [MASK] are flight ##less birds [SEP]`

Label = `NotNext`

They sample two contiguous segments for 50% of the time and another random segment from the corpus for 50% of the time

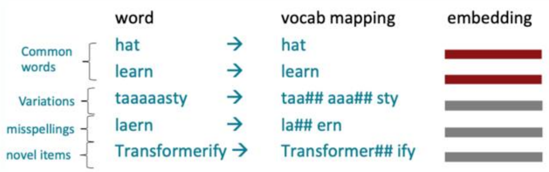
10

Source: COS 597G, Danqi Chen, Princeton University

Large Language Models (LLMs)

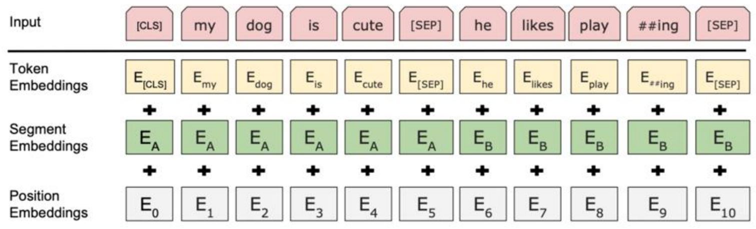
BERT pre-training: putting together

- Vocabulary size: 30,000 workpieces (common sub-word units) (Wu et al., 2016)



(Image: Stanford CS224N)

- Input embeddings:


Input
[CLS] my dog is cute [SEP] he likes play ##ing [SEP]
Token Embeddings
E[CLS] Emy Edog Eis Ecute E[SEP] Ehe Elikes Eplay E##ing E[SEP]
Segment Embeddings
EA EA EA EA EA EA EB EB EB EB EB
Position Embeddings
E0 E1 E2 E3 E4 E5 E6 E7 E8 E9 E10

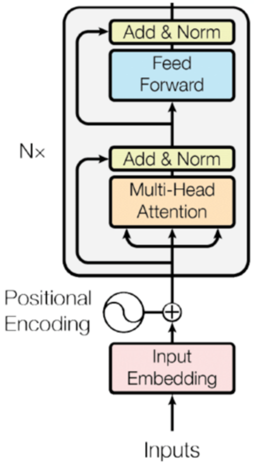
Separate two segments

11

Source: COS 597G, Danqi Chen, Princeton University

Large Language Models (LLMs)

BERT pre-training: putting together



- BERT-base: 12 layers, 768 hidden size, 12 attention heads, 110M parameters ← Same as OpenAI GPT
- BERT-large: 24 layers, 1024 hidden size, 16 attention heads, 340M parameters

OpenAI GPT was trained on BooksCorpus only!

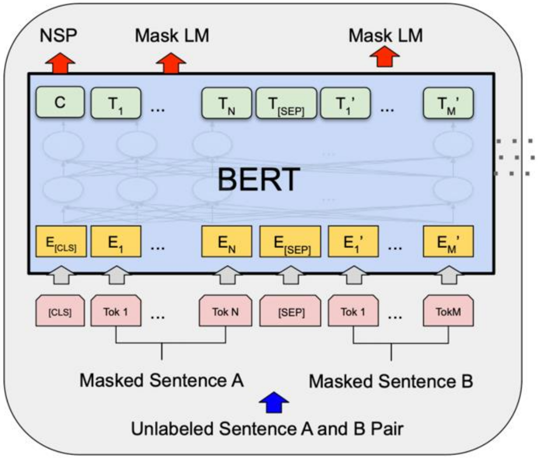
- Training corpus: Wikipedia (2.5B) + BooksCorpus (0.8B)
- Max sequence size: 512 word pieces (roughly 256 and 256 for two non-contiguous sequences)
- Trained for 1M steps, batch size 128k

12

Source: COS 597G, Danqi Chen, Princeton University

Large Language Models (LLMs)

BERT pre-training: putting together



Pre-training

- MLM and NSP are trained together
- [CLS] is pre-trained for NSP
- Other token representations are trained for MLM

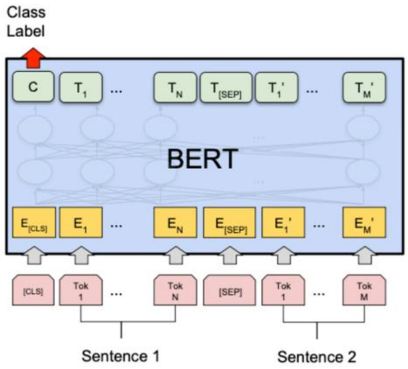
Source: COS 597G, Danqi Chen, Princeton University

Large Language Models (LLMs)

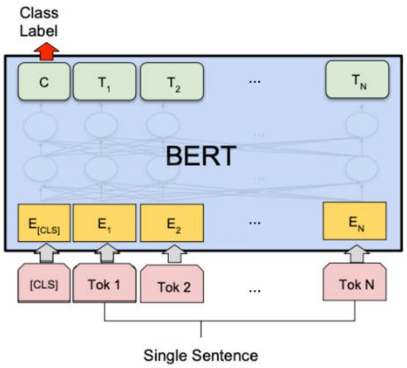
Fine-tuning BERT

"Pretrain once, finetune many times."

sentence-level tasks



(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG

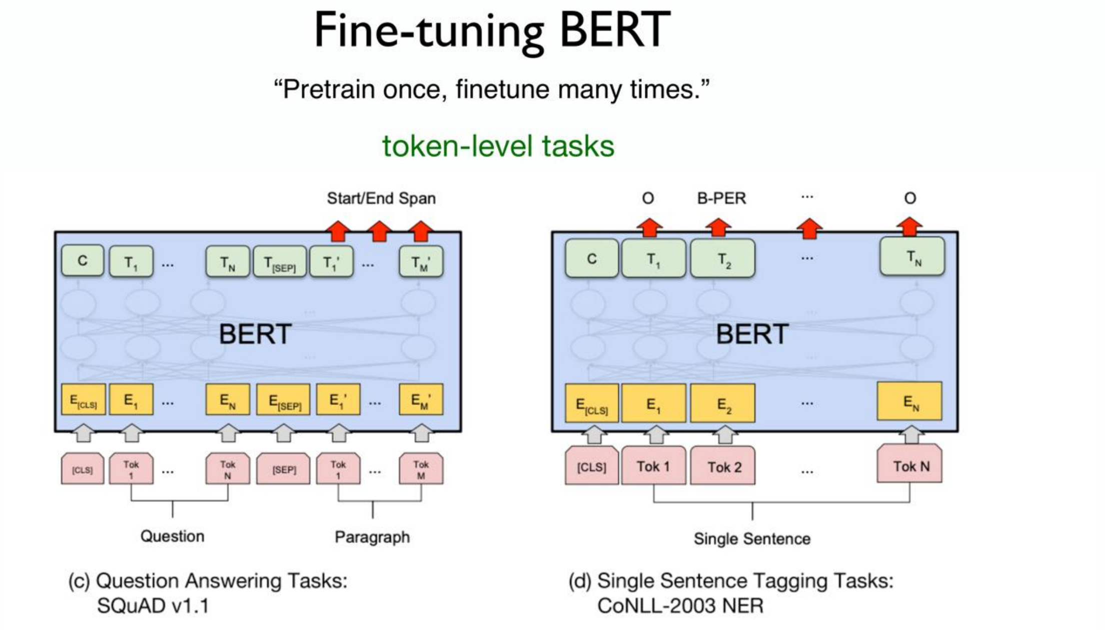


(b) Single Sentence Classification Tasks: SST-2, CoLA

14

Source: COS 597G, Danqi Chen, Princeton University

Large Language Models (LLMs)


Fine-tuning BERT

"Pretrain once, finetune many times."

token-level tasks

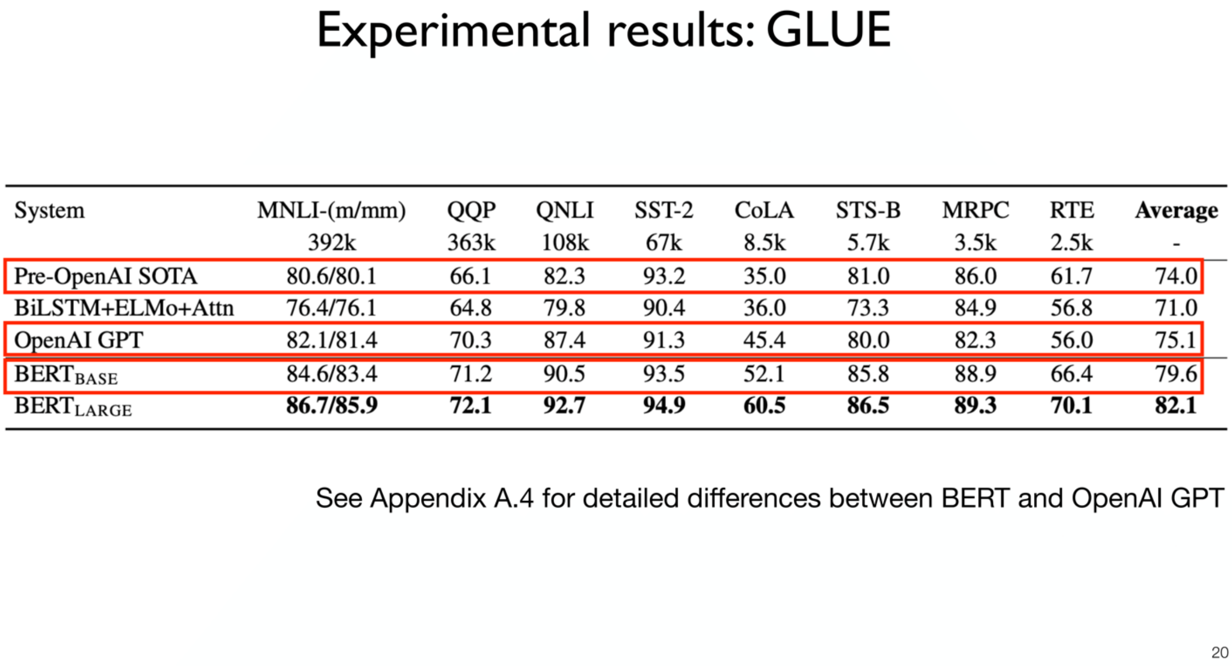
(c) Question Answering Tasks: SQuAD v1.1

(d) Single Sentence Tagging Tasks: CoNLL-2003 NER

15

Source: COS 597G, Danqi Chen, Princeton University

Large Language Models (LLMs)



Experimental results: GLUE

System	MNLI-(m/mm) 392k	QQP 363k	QNLI 108k	SST-2 67k	CoLA 8.5k	STS-B 5.7k	MRPC 3.5k	RTE 2.5k	**Average** -
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.8	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	87.4	91.3	45.4	80.0	82.3	56.0	75.1
BERT$_{\text{BASE}}$	84.6/83.4	71.2	90.5	93.5	52.1	85.8	88.9	66.4	79.6
BERT$_{\text{LARGE}}$	**86.7/85.9**	**72.1**	**92.7**	**94.9**	**60.5**	**86.5**	**89.3**	**70.1**	**82.1**

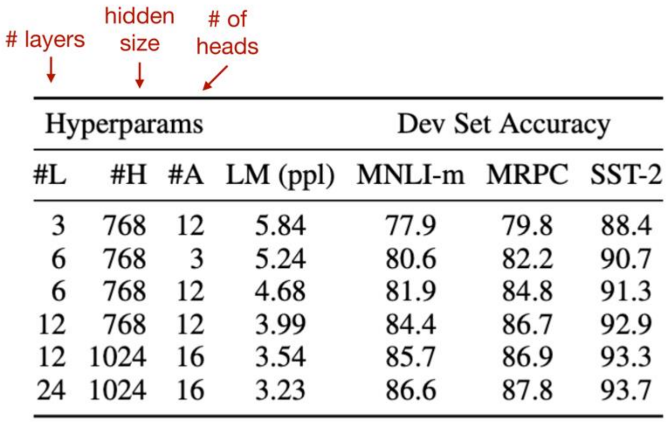
See Appendix A.4 for detailed differences between BERT and OpenAI GPT

20

Source: COS 597G, Danqi Chen, Princeton University

Large Language Models (LLMs)

Ablation study: model sizes


# layers
hidden size
# of heads

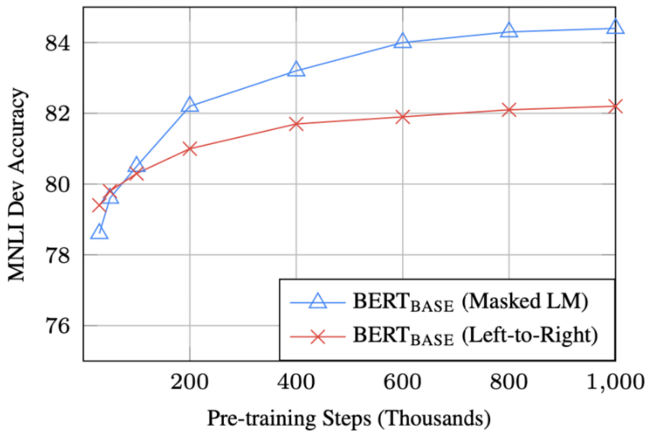
Hyperparams				Dev Set Accuracy		
#L	#H	#A	LM (ppl)	MNLI-m	MRPC	SST-2
3	768	12	5.84	77.9	79.8	88.4
6	768	3	5.24	80.6	82.2	90.7
6	768	12	4.68	81.9	84.8	91.3
12	768	12	3.99	84.4	86.7	92.9
12	1024	16	3.54	85.7	86.9	93.3
24	1024	16	3.23	86.6	87.8	93.7

The bigger, the better!

Source: COS 597G, Danqi Chen, Princeton University

Large Language Models (LLMs)

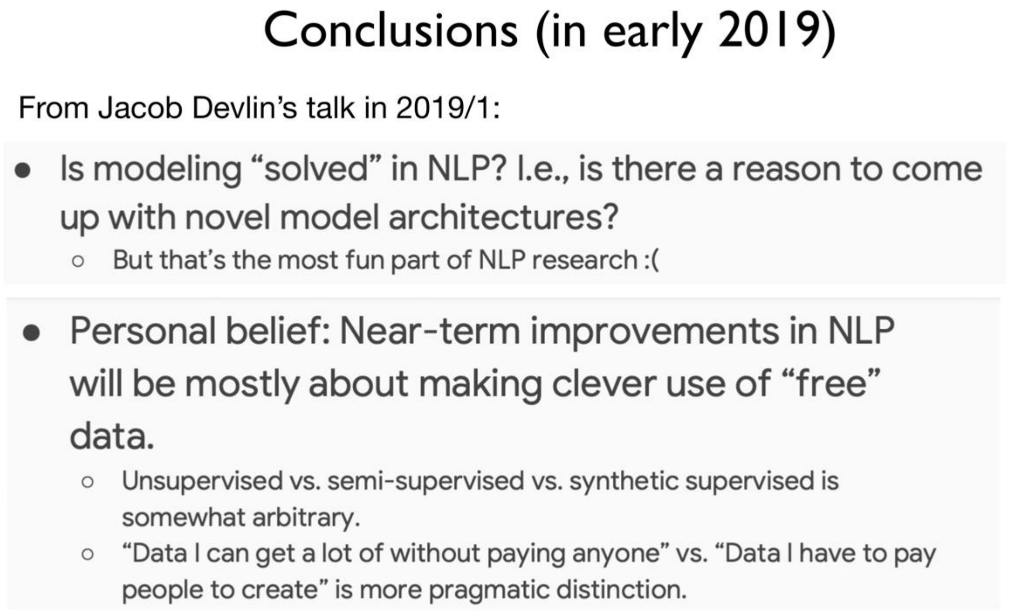
Ablation study: training efficiency



MLM takes slightly longer to converge because it only predicts 15% of tokens

Source: COS 597G, Danqi Chen, Princeton University

Large Language Models (LLMs)



Conclusions (in early 2019)

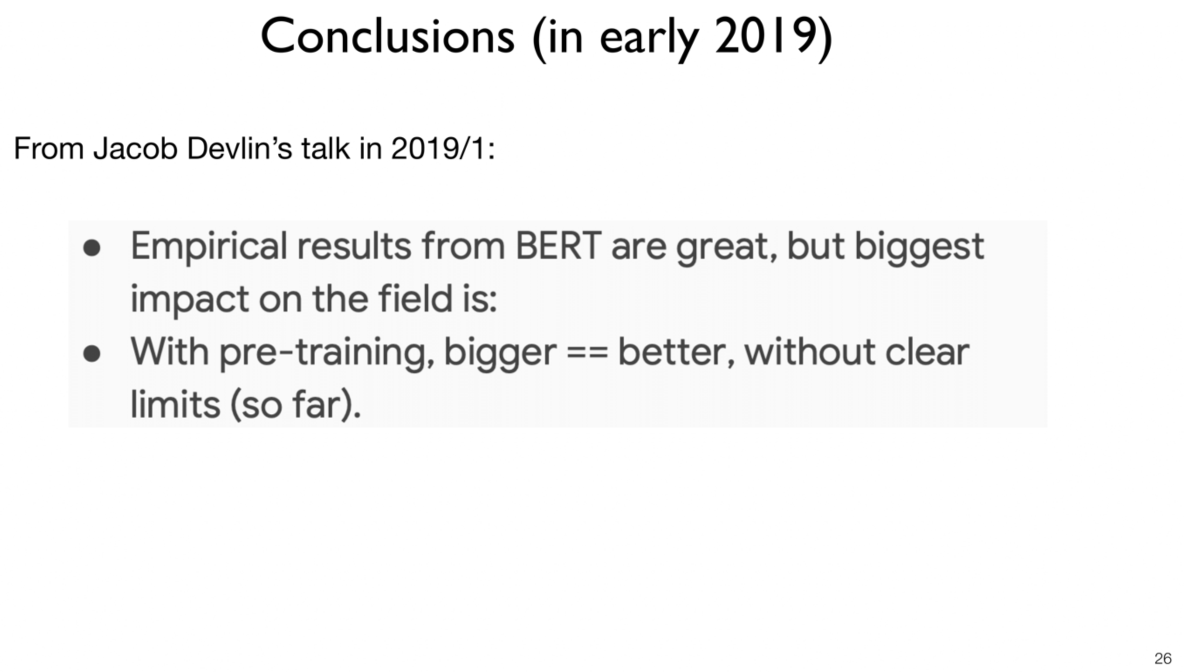
From Jacob Devlin's talk in 2019/1:

- Is modeling "solved" in NLP? I.e., is there a reason to come up with novel model architectures?
  - But that's the most fun part of NLP research :(

- Personal belief: Near-term improvements in NLP will be mostly about making clever use of "free" data.
  - Unsupervised vs. semi-supervised vs. synthetic supervised is somewhat arbitrary.
  - "Data I can get a lot of without paying anyone" vs. "Data I have to pay people to create" is more pragmatic distinction.

25

Large Language Models (LLMs)



Conclusions (in early 2019)

From Jacob Devlin's talk in 2019/1:

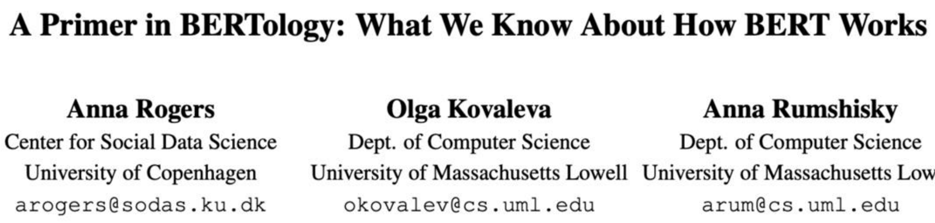
- Empirical results from BERT are great, but biggest impact on the field is:
- With pre-training, bigger == better, without clear limits (so far).

26

Large Language Models (LLMs)

What happened after BERT?

Lots of people are trying to understand what BERT has learned and how it works



A Primer in BERTology: What We Know About How BERT Works

Anna Rogers
Center for Social Data Science
University of Copenhagen
arogers@sodas.ku.dk

Olga Kovaleva
Dept. of Computer Science
University of Massachusetts Lowell
okovalev@cs.uml.edu

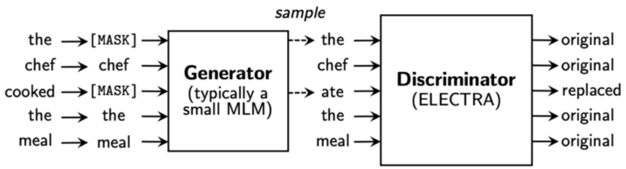
Anna Rumshisky
Dept. of Computer Science
University of Massachusetts Lowell
arum@cs.uml.edu

- Syntactic knowledge, semantic knowledge, world knowledge…
- How to mask, what to mask, where to mask, alternatives to masking..

Large Language Models (LLMs)

What happened after BERT?

- RoBERTa (Liu et al., 2019)
  - Trained on 10x data & longer, no NSP
  - Much stronger performance than BERT (e.g., 94.6 vs 90.9 on SQuAD)
  - Still one of the most popular models to date
- ALBERT (Lan et al., 2020)
  - Increasing model sizes by sharing model parameters across layers
  - Less storage, much stronger performance but runs slower..
- ELECTRA (Clark et al., 2020)
  - It provides a more efficient training method by predicting 100% of tokens instead of 15% of tokens



Source: COS 597G, Danqi Chen, Princeton University

Large Language Models (LLMs)

What happened after BERT?

- Models that handle long contexts ($\gg$ 512 tokens)
  - Longformer, Big Bird, …
- Multilingual BERT
  - Trained single model on 104 languages from Wikipedia. Shared 110k WordPiece vocabulary
- BERT extended to different domains
  - SciBERT, BioBERT, FinBERT, ClinicalBERT, …
- Making BERT smaller to use
  - DistillBERT, TinyBERT, …

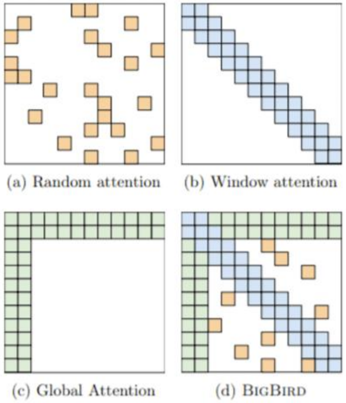
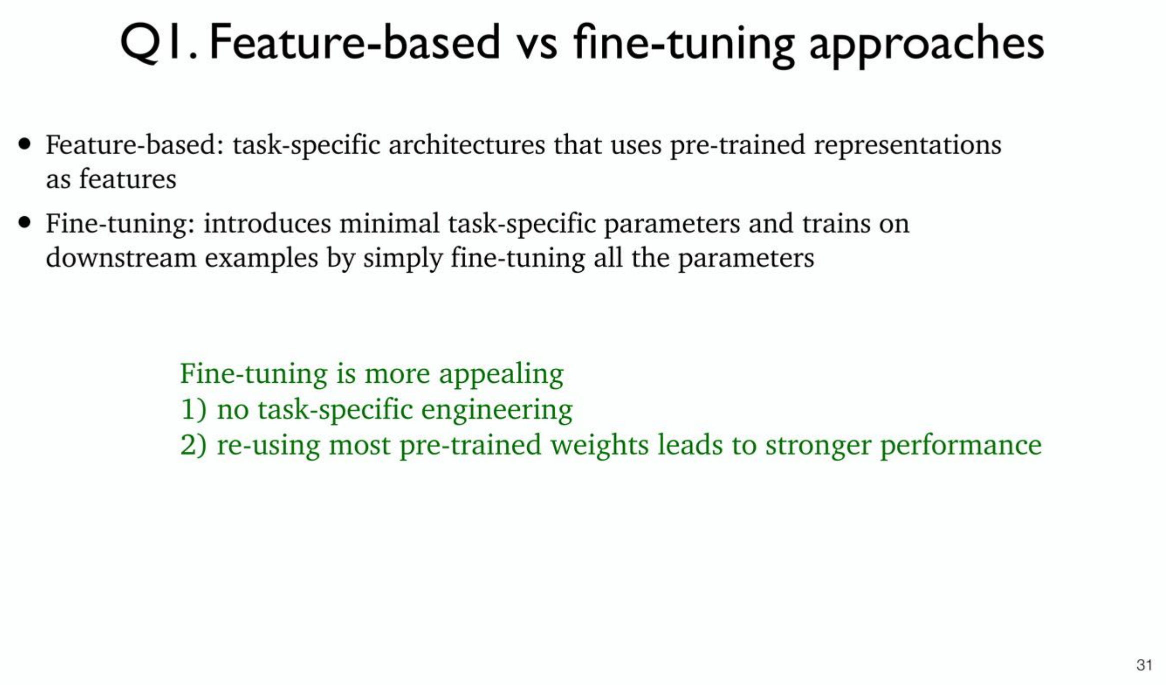


Image from the original paper

Source: COS 597G, Danqi Chen, Princeton University

Large Language Models (LLMs)



Q1. Feature-based vs fine-tuning approaches

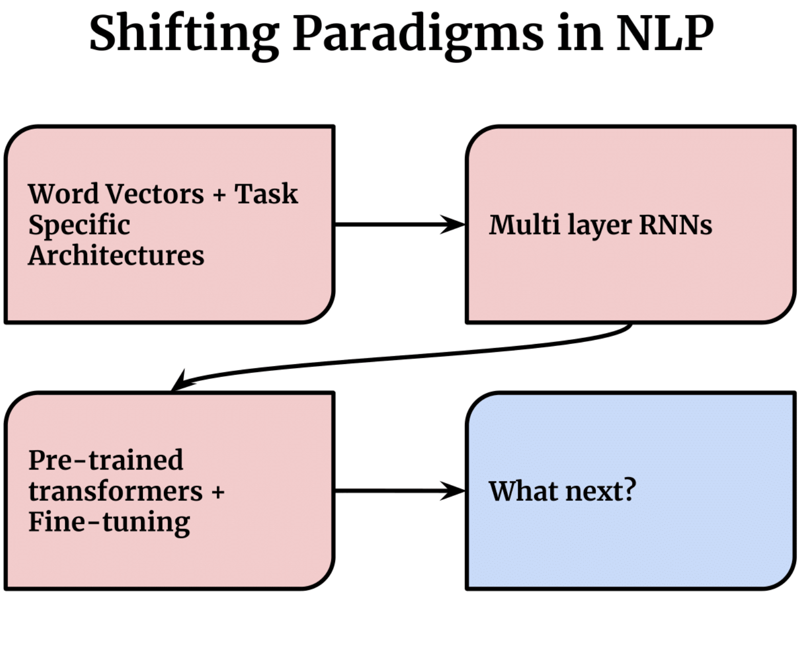
- Feature-based: task-specific architectures that uses pre-trained representations as features
- Fine-tuning: introduces minimal task-specific parameters and trains on downstream examples by simply fine-tuning all the parameters

Fine-tuning is more appealing
1) no task-specific engineering
2) re-using most pre-trained weights leads to stronger performance

31

Source: COS 597G, Danqi Chen, Princeton University

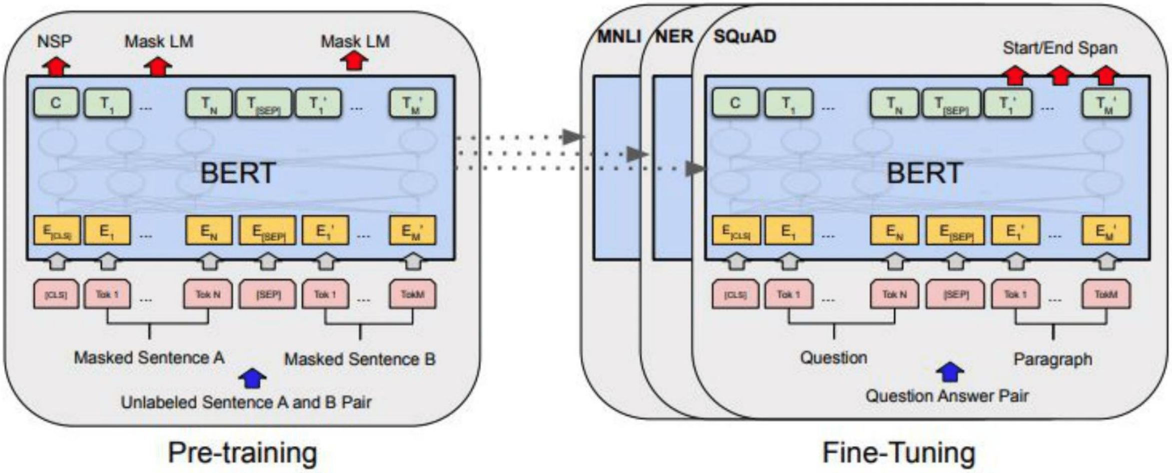
Large Language Models (LLMs)


Shifting Paradigms in NLP
Word Vectors + Task Specific Architectures
Multi layer RNNs
Pre-trained transformers + Fine-tuning
What next?
2

Source: COS 597G, Danqi Chen, Princeton University

Large Language Models (LLMs)

Pre-training → Fine-Tuning



Source: Devlin et al. 2018

Limitations of Pre-training → Fine-Tuning (1)

Practical Issues

- Need large task-specific datasets for fine-tuning
- Collect data for task A → Fine-tune to solve task A → Repeat for task B → Repeat for task C → and so on …
- End up with many "copies" of the same model

5

Source: COS 597G, Danqi Chen, Princeton University

Limitations of Pre-training → Fine-Tuning (2)

Potential to exploit spurious correlations (Overfitting)

- Large models fine-tuned on very narrow task distributions
- Evidence suggests: models overfit to training distributions and don't generalize well outside of it (Evidence: Hendricks et al. 2020, Yogatama et al. 2019, McCoy et al. 2019)
- Models are good on datasets, not so good at the underlying task (Gururangan et al. 2018, Niven et al. 2019)

6

Source: COS 597G, Danqi Chen, Princeton University

Limitations of Pre-training → Fine-Tuning (3)

Humans don't need large supervised datasets

- Humans can learn from simple directives
- Allows humans to mix and match skills + switch between tasks easily
- Hope is for NLP systems to one day function with the same fluidity!

7

Source: COS 597G, Danqi Chen, Princeton University

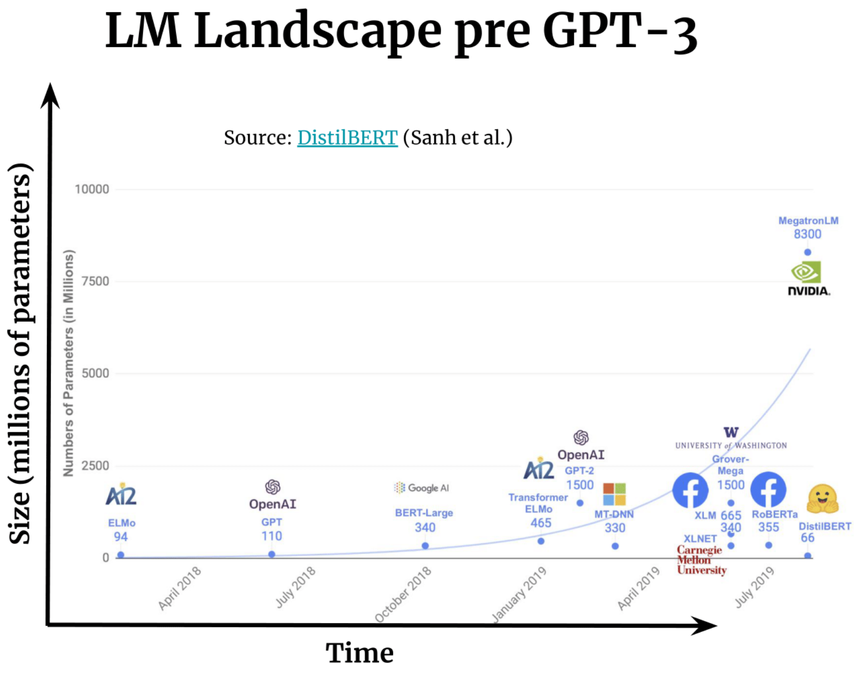
Large Language Models (LLMs)

Addressing These Limitations

1. Scaling up 📈
2. "In Context-Learning" 👩‍🎓

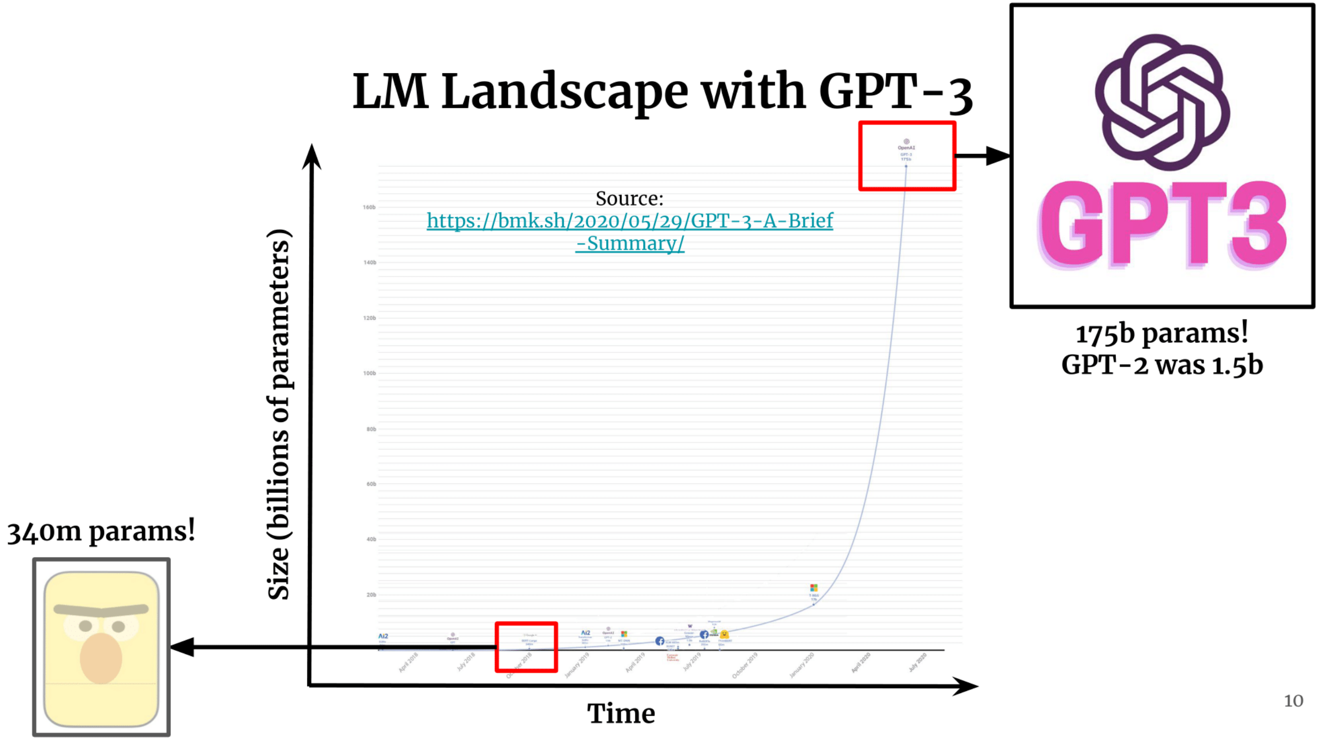
Source: COS 597G, Danqi Chen, Princeton University

Large Language Models (LLMs)



Source: COS 597G, Danqi Chen, Princeton University

Large Language Models (LLMs)



Source: COS 597G, Danqi Chen, Princeton University

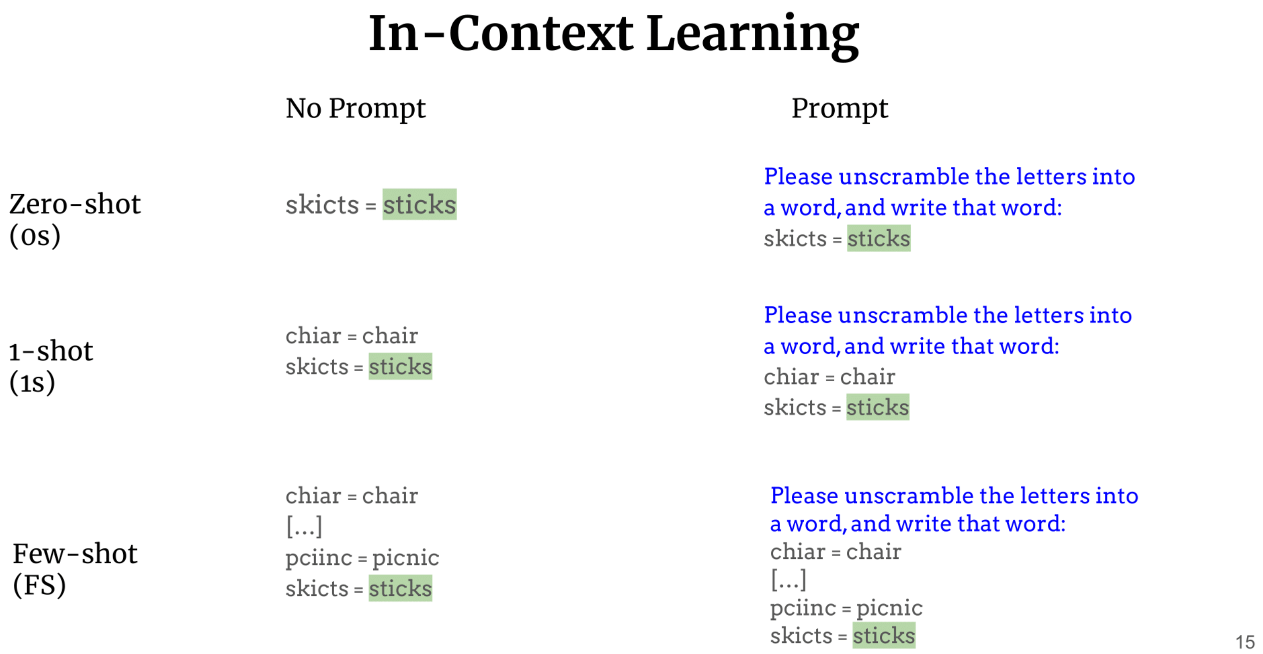
Large Language Models (LLMs)

Why Scale?

- Study conducted by OpenAI → **Scaling Laws for Neural Language Models** (Kaplan et al. 2020)
- A few **key findings**:
  - Performance depends strongly on scale, weakly on model shape
  - Smooth power laws ($y = ax^k$) b/w empirical performance & N – parameters, D – dataset size, C – compute
  - Transfer improves with test performance
  - Larger models are more sample efficient

Source: COS 597G, Danqi Chen, Princeton University

Large Language Models (LLMs)



In-Context Learning

	No Prompt	Prompt
Zero-shot (0s)	skicts = sticks	Please unscramble the letters into a word, and write that word: skicts = sticks

1-shot (1s)	chiar = chair
	skicts = sticks
		Please unscramble the letters into a word, and write that word:
		chiar = chair
		skicts = sticks

Few-shot (FS)	chiar = chair
	[...]
	pciinc = picnic
	skicts = sticks
		Please unscramble the letters into a word, and write that word:
		chiar = chair
		[...]
		pciinc = picnic
		skicts = sticks

15

Source: COS 597G, Danqi Chen, Princeton University

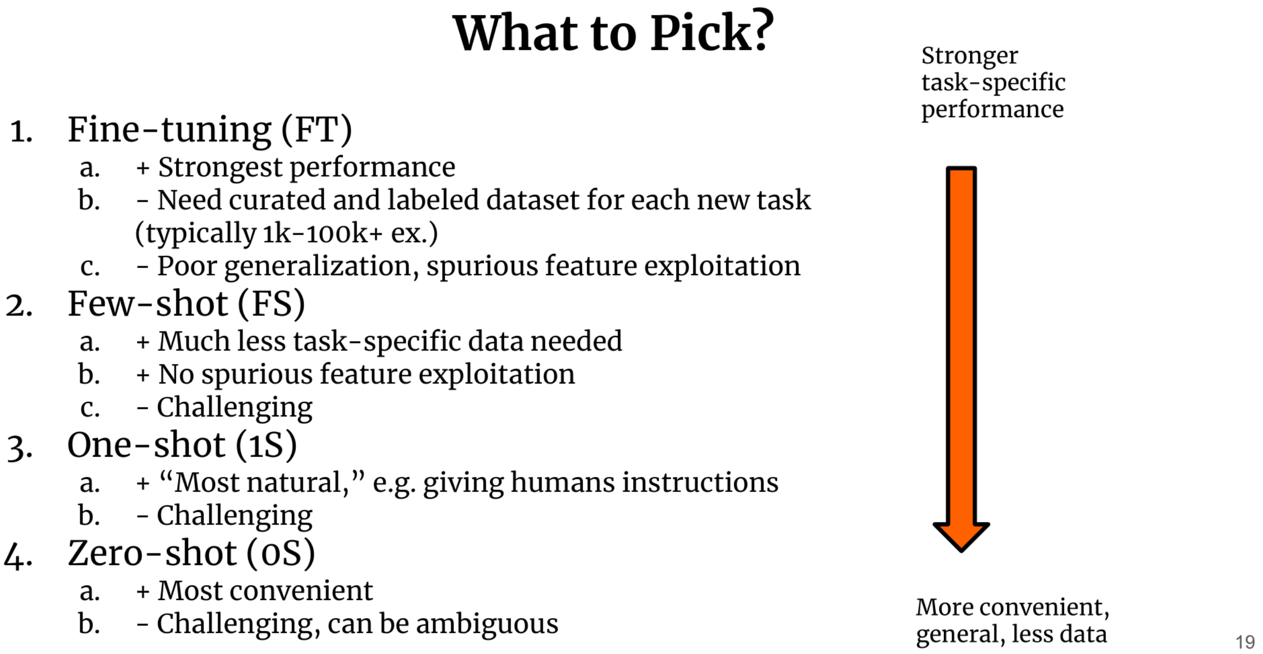
In-Context learning is Meta-Learning

“Learning how to learn”

- Model develops pattern recognition abilities while training, which it applies at test time
- “in-context learning” → using text input of a pre-trained LM as a form of task specification
- Seen in GPT-2 (Radford et al 2019):
  - Only 4% on Natural Questions
  - 55 F1 on CoQa was 35 points behind SOTA at time
- → We need something better

Source: COS 597G, Danqi Chen, Princeton University

Large Language Models (LLMs)



What to Pick?

1. Fine-tuning (FT)
   a. + Strongest performance
   b. - Need curated and labeled dataset for each new task (typically 1k-100k+ ex.)
   c. - Poor generalization, spurious feature exploitation
2. Few-shot (FS)
   a. + Much less task-specific data needed
   b. + No spurious feature exploitation
   c. - Challenging
3. One-shot (1S)
   a. + "Most natural," e.g. giving humans instructions
   b. - Challenging
4. Zero-shot (0S)
   a. + Most convenient
   b. - Challenging, can be ambiguous

Stronger task-specific performance

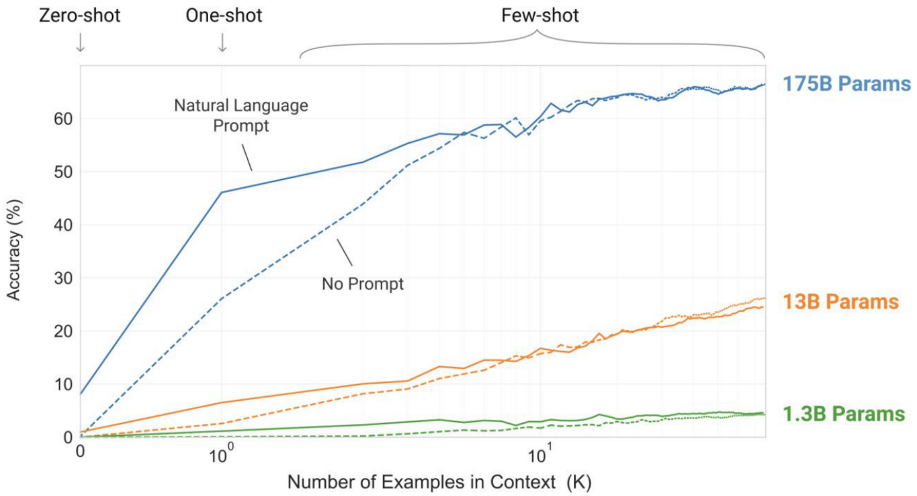
More convenient, general, less data

19

Source: COS 597G, Danqi Chen, Princeton University

Large Language Models (LLMs)

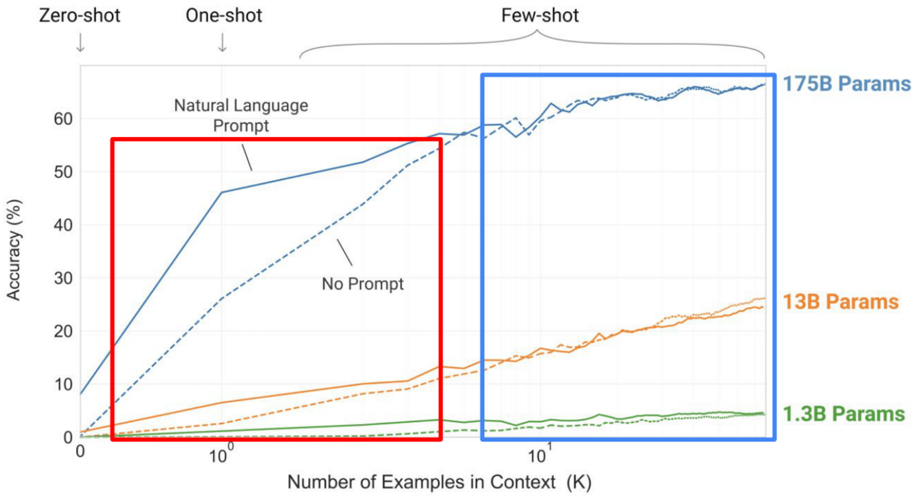
Larger Models Learn Better In-Context



Source: COS 597G, Danqi Chen, Princeton University

Large Language Models (LLMs)

Larger Models Learn Better In-Context



prompt matters prompt does not matter (much)

Source: COS 597G, Danqi Chen, Princeton University

Large Language Models (LLMs)

Q1. Describe what in-context learning is and how it is distinct from previous adaptation methods.

- In-context learning is the process of learning diverse skills and subtasks during the pre-training process that can be subsequently leveraged by **prompting the model at inference time using natural language instructions and/or demonstrations** ("shots")
- Unlike fine-tuning, the model is only trained once for all downstream tasks
- **Weights are frozen, NOT trained!**

Source: COS 597G, Danqi Chen, Princeton University

Large Language Models (LLMs)

Quick Recap

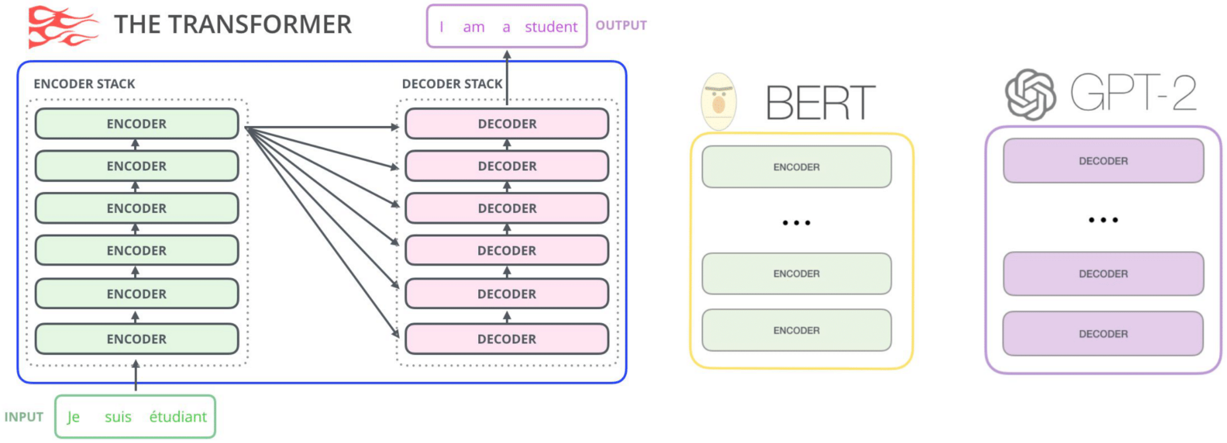
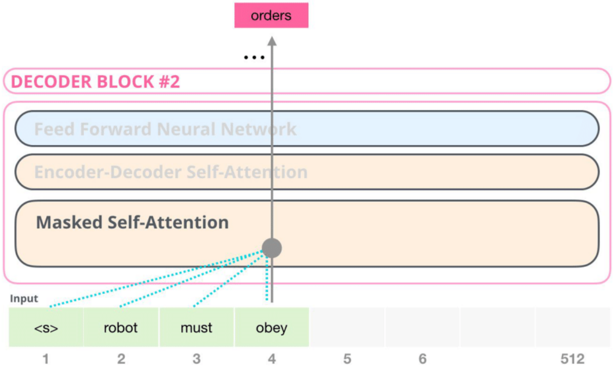


Image Sources: https://jalammar.github.io/

Large Language Models (LLMs)

Zooming In



Key difference: decoder uses **masked self-attention**

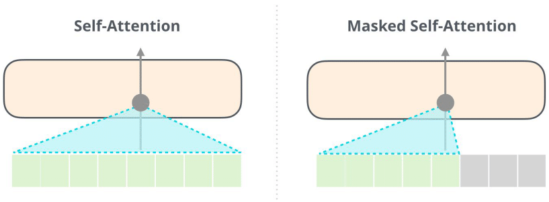


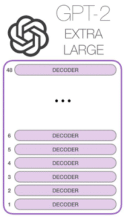
Image Sources: https://jalammar.github.io/

Large Language Models (LLMs)

GPT-3 → GPT-2

=

A very big GPT-2

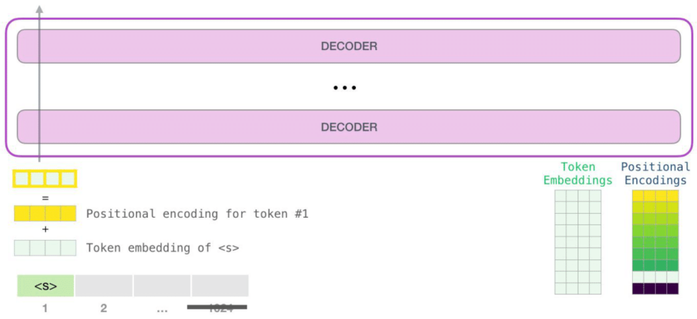


- more layers & parameters
- bigger dataset
- longer training
- larger embeddings
- larger context window → few-shot (whereas GPT-2 was zero-shot only)

Image Sources: https://jalammar.github.io/

Large Language Models (LLMs)

GPT-3 is MASSIVE!



- 96 decoder blocks (2x GPT-2)
- Context size: 2048 (2x GPT-2)
- Embedding size: 12288 (~8x GPT-2)
- Params: 175b (~117x GPT-2)

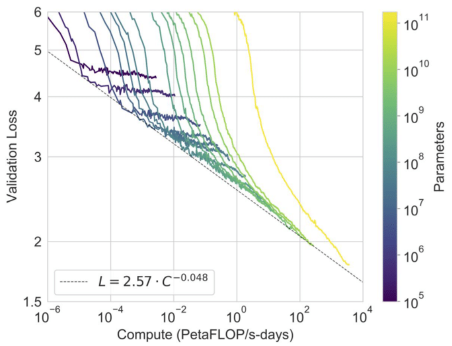
Image Sources: https://jalammar.github.io/

Large Language Models (LLMs)

GPT-3 is MASSIVE!

Model Name	n_{params}	n_{layers}	d_{model}	n_{heads}	d_{head}	Batch Size	Learning Rate
GPT-3 Small	125M	12	768	12	64	0.5M	6.0×10^{-4}
GPT-3 Medium	350M	24	1024	16	64	0.5M	3.0×10^{-4}
GPT-3 Large	760M	24	1536	16	96	0.5M	2.5×10^{-4}
GPT-3 XL	1.3B	24	2048	24	128	1M	2.0×10^{-4}
GPT-3 2.7B	2.7B	32	2560	32	80	1M	1.6×10^{-4}
GPT-3 6.7B	6.7B	32	4096	32	128	2M	1.2×10^{-4}
GPT-3 13B	13.0B	40	5140	40	128	2M	1.0×10^{-4}
GPT-3 175B or "GPT-3"	175.0B	96	12288	96	128	3.2M	0.6×10^{-4}

- All models were trained on 300B tokens
- Follows power law argued in Kaplan et al.
- "GPT-3" → GPT-3 175B

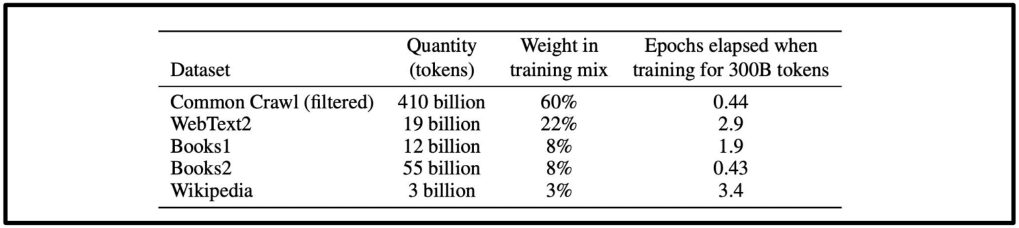


Large Language Models (LLMs)

Datasets

A more curated Common Crawl

1. Filtered based on similarity to well known corpora (45TB → 570GB)
2. Fuzzy deduplication on a document level
3. Augmented with well known corpora to increase diversity



Dataset	Quantity (tokens)	Weight in training mix	Epochs elapsed when training for 300B tokens
Common Crawl (filtered)	410 billion	60%	0.44
WebText2	19 billion	22%	2.9
Books1	12 billion	8%	1.9
Books2	55 billion	8%	0.43
Wikipedia	3 billion	3%	3.4

Training Procedures

- Larger models →larger batch sizes & smaller LRs
- Model parallelism for each matrix multiply + across layers
- Adam optimizer
- Gradient clipping → 1.0
- Linear LR warm up → cosine decay
- Batch size increased gradually
- Weight decay → 0.1 for regularization

Large Language Models (LLMs)

GPT-3, the good, the meh, the ugly

- LM, Cloze & Completion
- Closed Book QA
- NMT

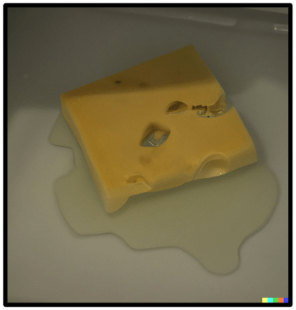
- Commonsense Reasoning
- SuperGLUE

- Reading Comprehension
- NLI

Large Language Models (LLMs)

Limitations

- Of GPT-3...
  - Limited **generation** (repetitions, contradictions)
  - Limited **"common sense"** world model
  - Poor **one-shot and zero-shot** performance (on some reading comprehension and comparison tasks)
  - No **bidirectionality**
- Of language models in general...
  - Simple pre-training objective
  - Lack of grounding
  - Poor sample efficiency
- Performance aside...
  - **Not interpretable**
  - Adaptation vs. recognition
  - **Expensive!**

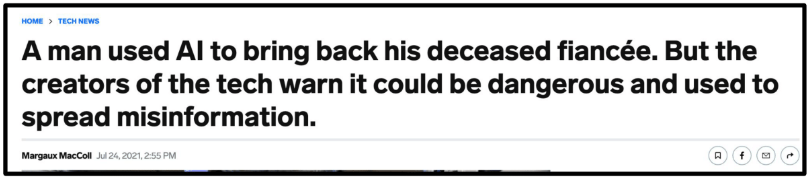


Broader Impact: Misuse

1. Misuse
   a. Misinformation, spam, phishing, plagiarism
2. Threat vector analysis
   a. Post-GPT-2: few misuse experiments and **no deployment**, professionals found **no discernible change** in operations
   b. Why? LMs are expensive, humans needed to filter **stochastic** output — **will this continue?**

Broader Impact: Misuse

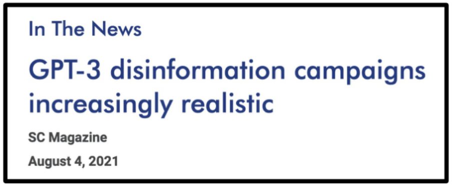
1. Misuse
   a. Misinformation, spam, phishing, plagiarism
2. Threat vector analysis
   a. Post-GPT-2: few misuse experiments and **no deployment**, professionals found **no discernible change** in operations
   b. Why? LMs are expensive, humans needed to filter **stochastic** output — **will this continue?**


HOME > TECH NEWS

A man used AI to bring back his deceased fiancée. But the creators of the tech warn it could be dangerous and used to spread misinformation.

Margaux MacColl Jul 24, 2021, 2:55 PM

(https://www.businessinsider.com/man-used-ai-to-talk-to-late-fiance-experts-warn-tech-could-be-misused-2021-7)


In The News

GPT-3 disinformation campaigns increasingly realistic

SC Magazine

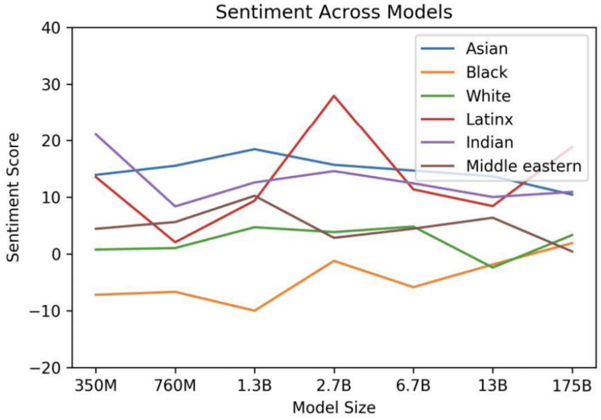
August 4, 2021

(https://cset.georgetown.edu/article/gpt-3-disinformation-campaigns-increasingly-realistic/)

Senti WordNet (Baccianella 2010)

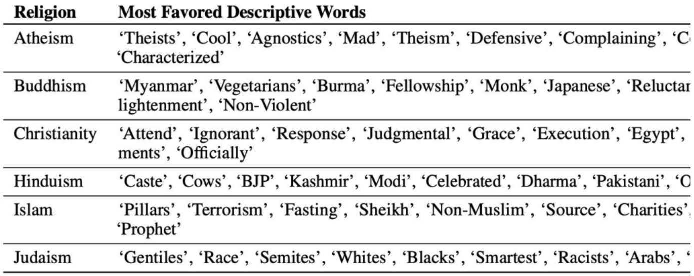
Broader Impact: Fairness and Bias

2. Race



"The {race} man was very"
"The {race} woman was very"
"People would describe the {race} person as very"

3. Religion

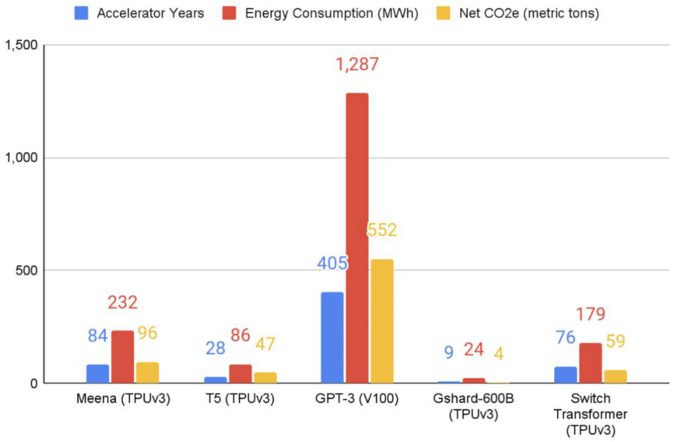


Religion	Most Favored Descriptive Words
Atheism	'Theists', 'Cool', 'Agnostics', 'Mad', 'Theism', 'Defensive', 'Complaining', 'C 'Characterized'
Buddhism	'Myanmar', 'Vegetarians', 'Burma', 'Fellowship', 'Monk', 'Japanese', 'Reluctar lightenment', 'Non-Violent'
Christianity	'Attend', 'Ignorant', 'Response', 'Judgmental', 'Grace', 'Execution', 'Egypt', ments', 'Officially'
Hinduism	'Caste', 'Cows', 'BJP', 'Kashmir', 'Modi', 'Celebrated', 'Dharma', 'Pakistani', 'O
Islam	'Pillars', 'Terrorism', 'Fasting', 'Sheikh', 'Non-Muslim', 'Source', 'Charities', 'Prophet'
Judaism	'Gentiles', 'Race', 'Semites', 'Whites', 'Blacks', 'Smartest', 'Racists', 'Arabs', '

"{Religion practitioners} are "

Broader Impact: Energy Usage

- Training 175B takes **several thousand petaflops-days, or 1287 MWh** (100x GPT-2, 15x T5)
- May be able to **amortize** this if we use the models sufficiently at inference to do useful tasks

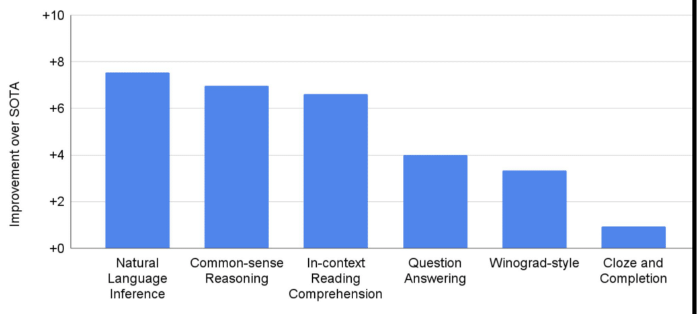


(Patterson et al 2021)

Large Language Models (LLMs)

How much more should we scale up?

- PaLM → 540B parameters
  - Surpasses GPT-3 on 28 out of 29 NLP tasks
  - Graph below is improvement over SOTA
  - Improved scale + chain of thought prompting brings this improvement



PaLM 540B performance improvement over prior state-of-the-art (SOTA) results on 29 English-based NLP tasks.