Deep Learning: Advanced Topics

Hualou Liang @ Drexel

Final Project, 06/06/2023

10-15 pages report due 6/6/23, max 30 min

(5) **Formatting** (i.e., how well did you follow the instructions above; clarity of equations that you present, graphs that have an appropriate form, etc.)
(10) **Problem identification**: Clearly explain the problem you try to solve (e.g. classify bacterial and viral pneumonia on chest X-rays, predict movie ratings etc.), and why it's interesting.
(10) **Dataset**: Identify the data set(s) that you will be using. You should give a clear description of the characteristics of the data (how many examples, what kinds of features do we have for each example, are there issues with missing data or bad data, etc.).
(10) **Methods**: What methods do you use to solve the problem? Methods include data preprocessing, feature extraction, and the deep learning models you use.
(25) **Implementation, results and interpretations**: This is a major part of your project. You need to provide the implementation details, the results obtained and interpretations.
(15) **Performance evaluation**: How will you evaluate performance? In certain settings, you may want to try a few different performance measures and/or identify a few "baseline algorithms". Ideally, you will be able to report the performance of a couple baseline algorithms. The goal will be to beat the baseline, so if the baseline is already quite high, you will have a challenge.
(5) Description of **follow-up experiments** to improve the results
(20) **30-min presentation**

Final Project, 06/06/2023

- Nick Fioravanti, Justin Serwinski; Jalen Winfield, Deep Learning in Neural Networks with Neural Data Analysis: EEG
- Alexander Wang, A.I. Philosophy and Ethical Alignment
- Josh Miller; Marc Mounzer; Kevin Chavez, Kasonde Chewe, Computational Neuroscience
- Christopher Campbell: Application in Biomedicine
- Eleni Alexakis, Hadis Jami, Tony Okeke, Medical Imaging

Roadmap

Deep Learning in NLP

Fundamentals

- Autoencoders
- Generative Adversarial Networks (GAN)

Applications

- Images, Text, Music

AI Generated Content (AIGC)

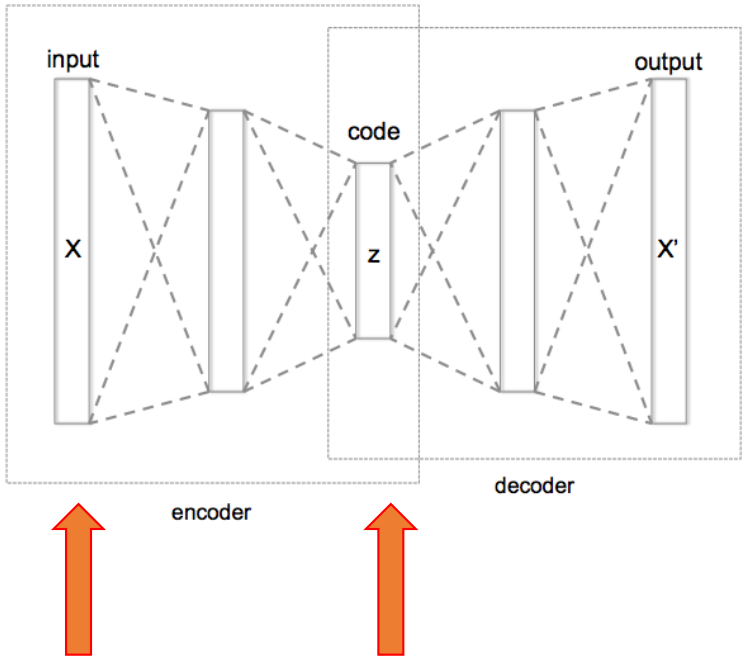
- **AI image**: text-to-image diffusion model (Stable Diffusion)
- **AI music**: music generation with stable diffusion (Riffusion)
- **AI text**: ChatGPT - AI chatbot writes smart essays (ai.com)
- **AI code**: GitHub Copilot: an AI pair programmer that offers autocomplete-style suggestions as you code

Autoencoders

An Autoencoder is a feedforward neural network that learns to predict the input itself in the output.

$$y^{(i)} = x^{(i)}$$

- The input-to-hidden part corresponds to an encoder
- The hidden-to-output part corresponds to a decoder.



Deep Autoencoders

- A deep Autoencoder is constructed by extending the encoder and decoder of autoencoder with multiple hidden layers.
- Gradient vanishing problem: the gradient becomes too small as it passes back through many layers

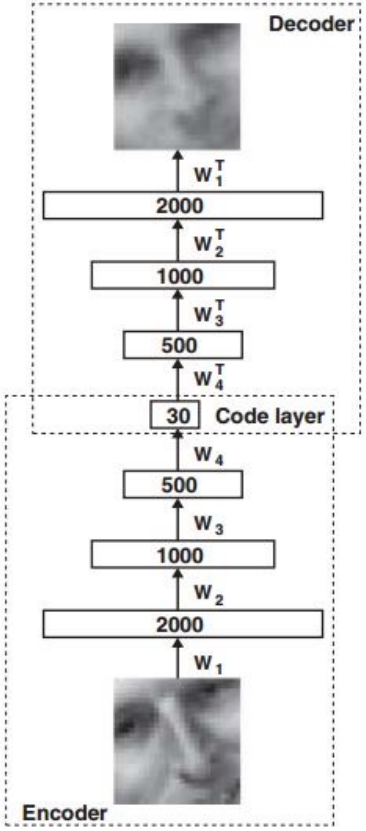


Diagram from (Hinton and Salakhutdinov, 2006)

Denoising Autoencoders

By adding stochastic noise to the data, it can force Autoencoder to learn more robust features.

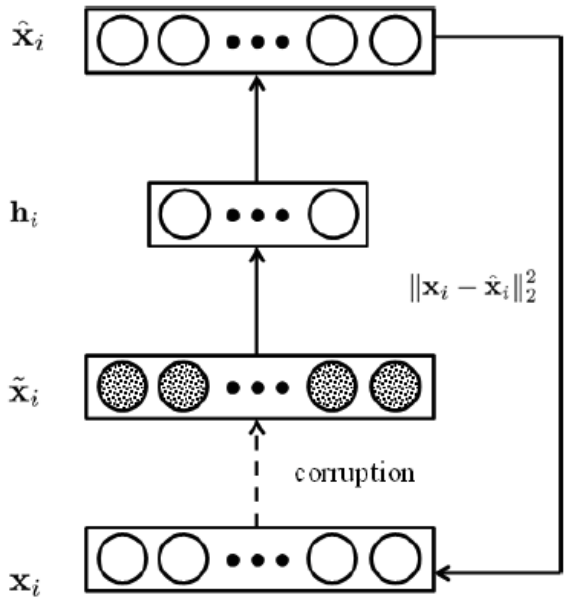
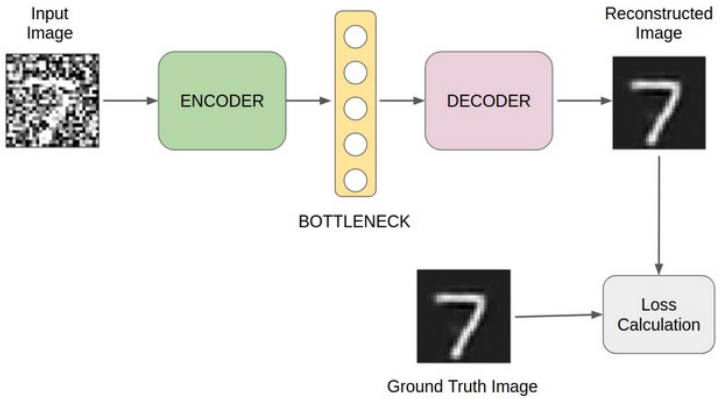
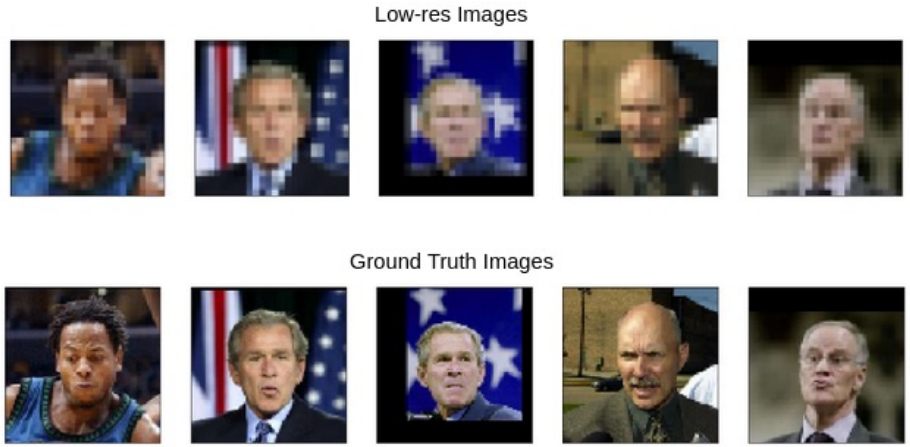


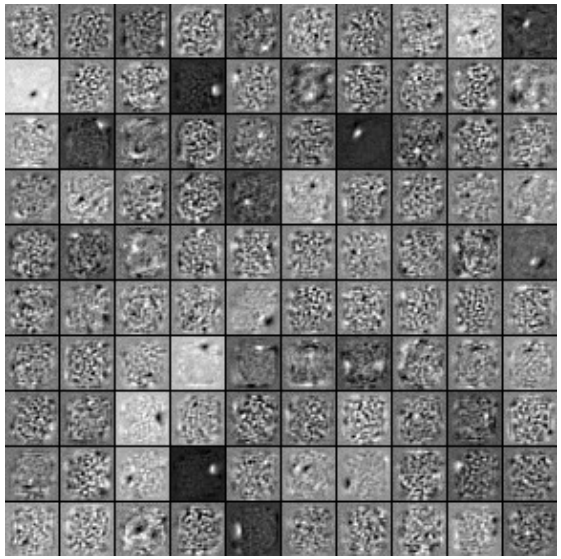
Image Denoising Autoencoder



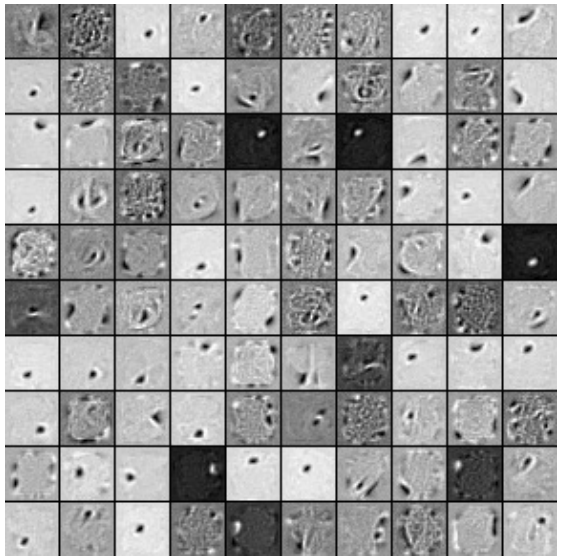


Training Denoising Autoencoder on MNIST

The following pictures show the difference between the resulting filters of Denoising Autoencoder trained on MNIST with different noise ratios.



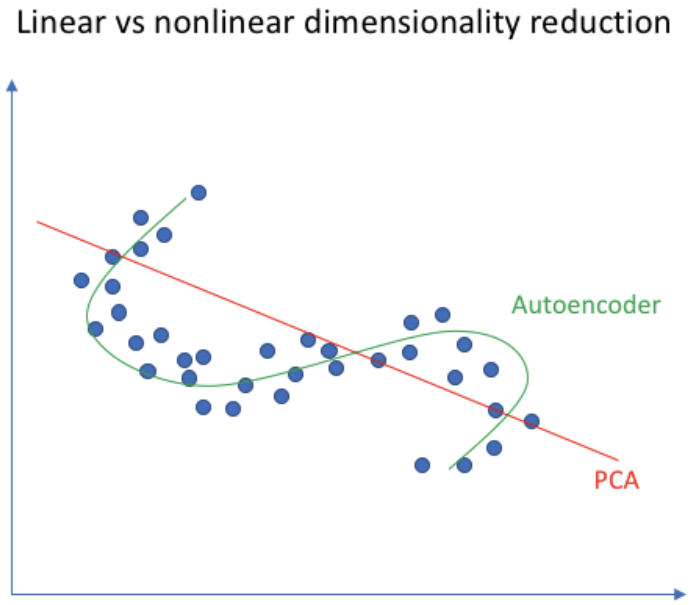
No noise (noise ratio=0%)



noise ratio=30%

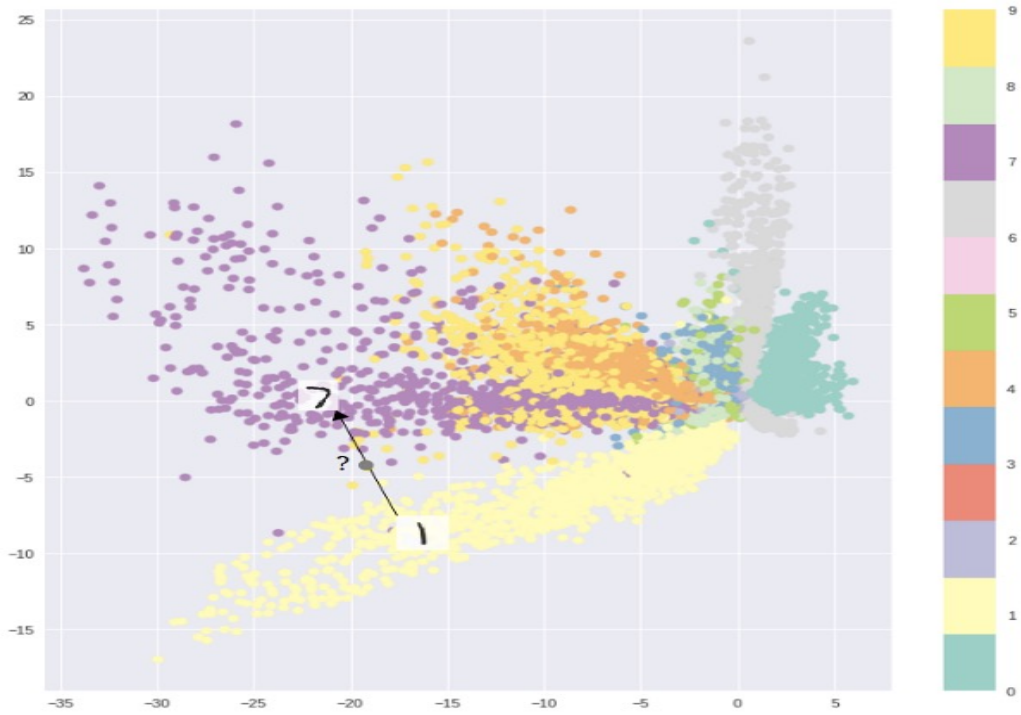
Diagram from (Hinton and Salakhutdinov, 2006)

Autoencoder vs. PCA



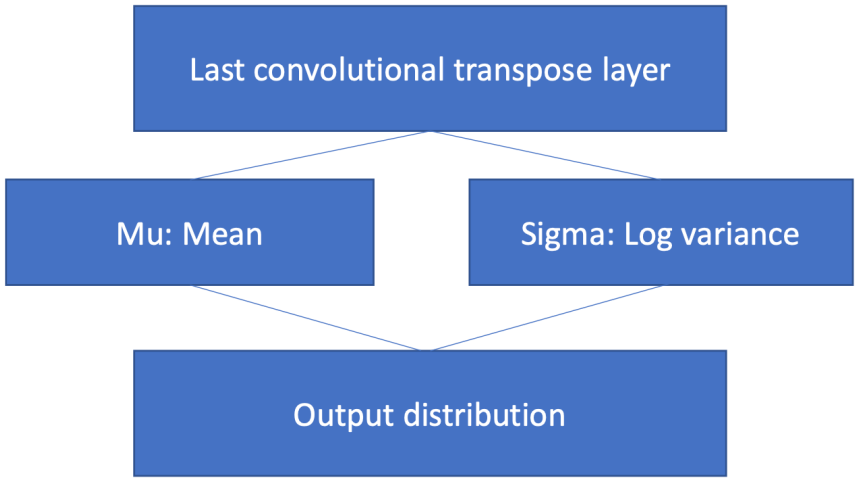
PCA: use linear function rather than ReLU

Problem with Autoencoder: 2D latent space of MNIST



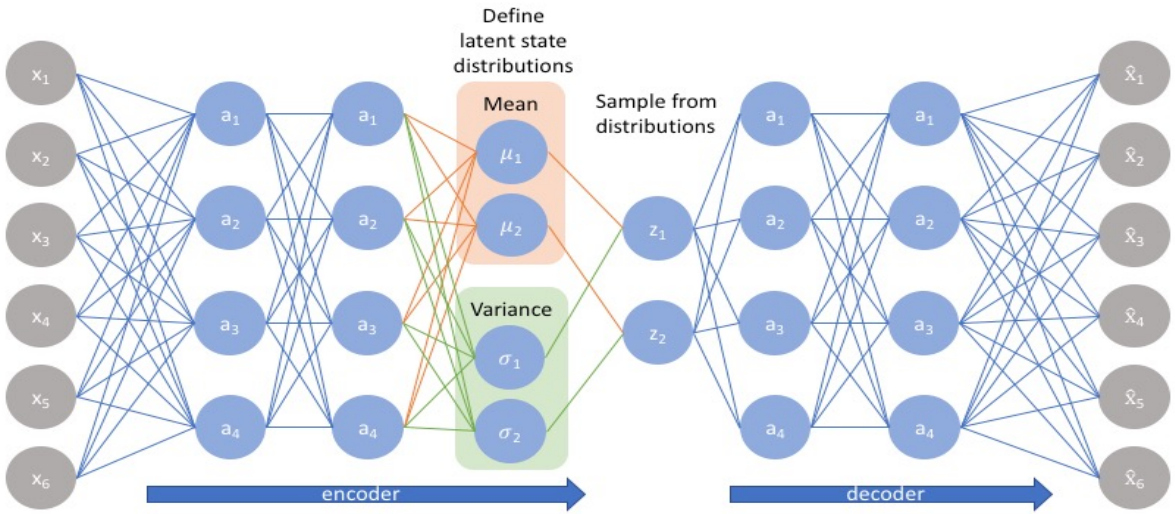
No idea to generate outputs coming from the gaps between clusters due to the discontinuities in the region of latent space

Variational Autoencoders



- Solve key problems with basic autoencoders
- Output distribution is described by mu and sigma
- Use a loss function that measures distance from output distribution to standard normal
- Tune parameter: r_loss_factor

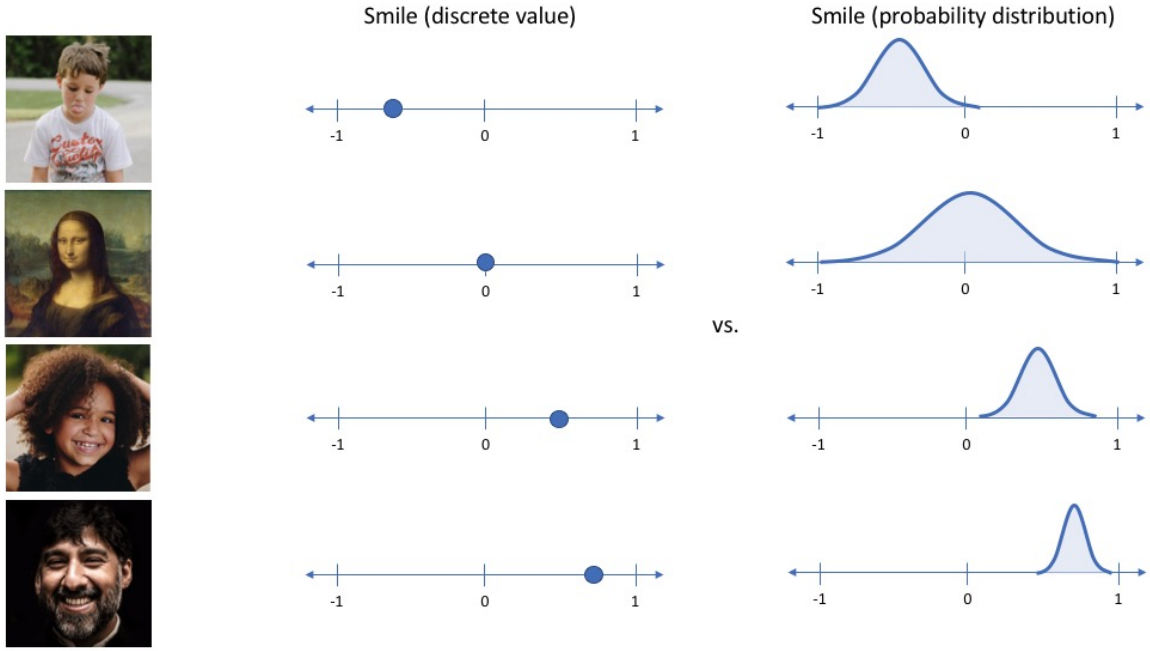
Variational Autoencoder (VAE)



- Provide a *probabilistic* manner for describing an observation in latent space
- Not output an encoding vector of size *n,* rather, outputting two vectors of size n: a vector of means**, μ**, and another vector of standard deviations, **σ**.
- Their latent spaces are, *by design,* continuous, allowing easy random sampling and interpolation

https://www.jeremyjordan.me/autoencoders/

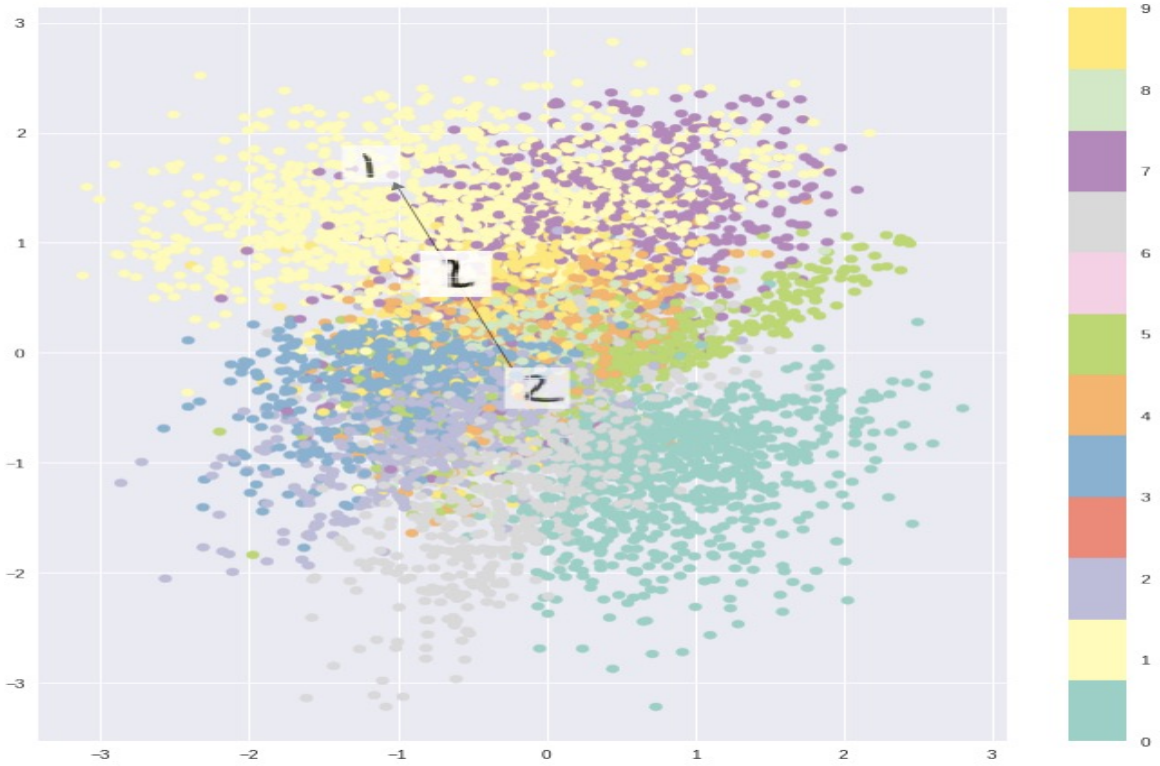
VAE vs. Autoencoder



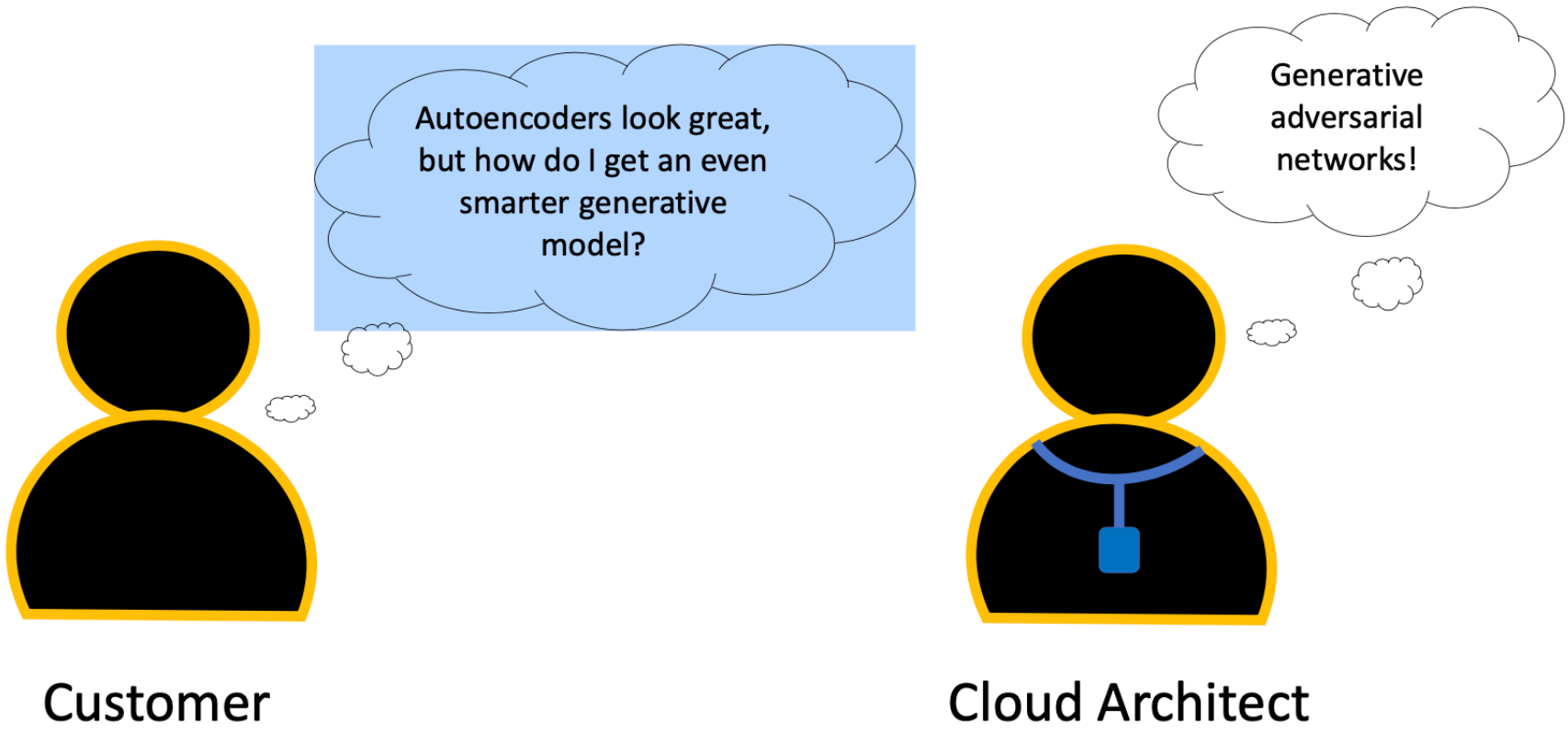
VAE: Provide a *probabilistic* manner for describing an observation in latent space
Autoencoder: output a *single value* to describe each latent state attribute

(https://www.jeremyjordan.me/variational-autoencoders/)

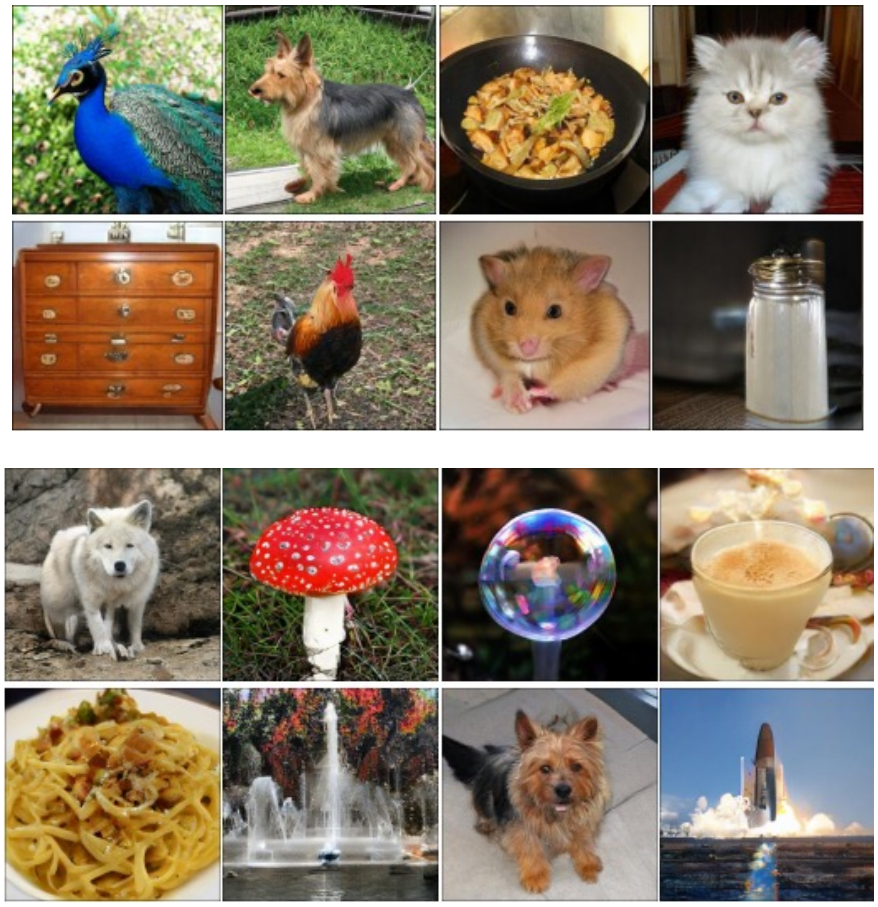
Variational Autoencoder (VAE)



their latent spaces are, *by design,* continuous, allowing easy random sampling and interpolation

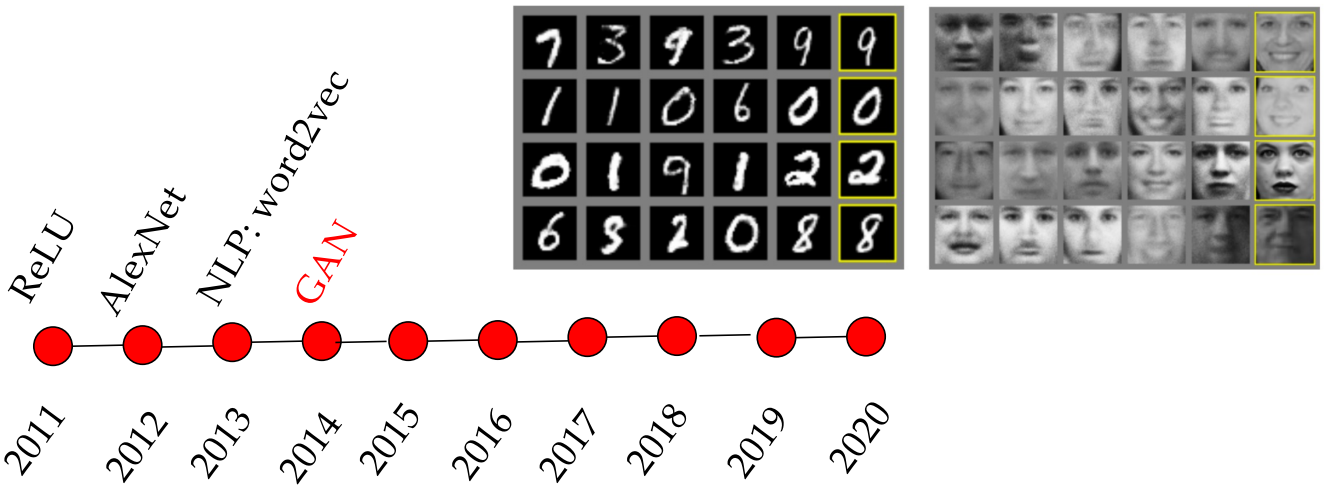

Autoencoders look great, but how do I get an even smarter generative model?
Generative adversarial networks!
Customer
Cloud Architect

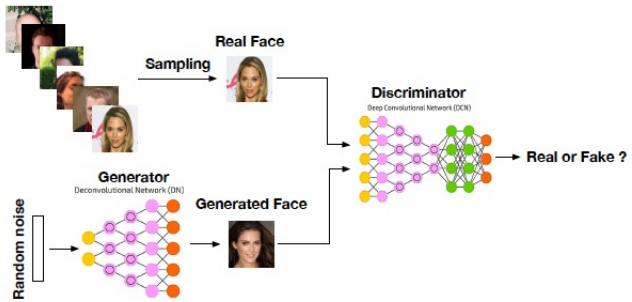
Can you pick out what's odd in the below collection of images?



All of the objects and animals in these images generated by GANs

Generative Adversarial Network (GAN)

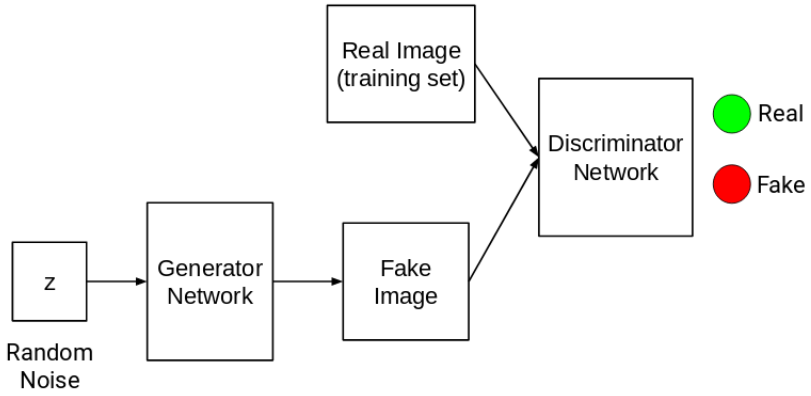




Related: Wasserstein GAN, StyleGAN; [Adam, Attention]

Goodefellow et al. Generative Adversarial Network

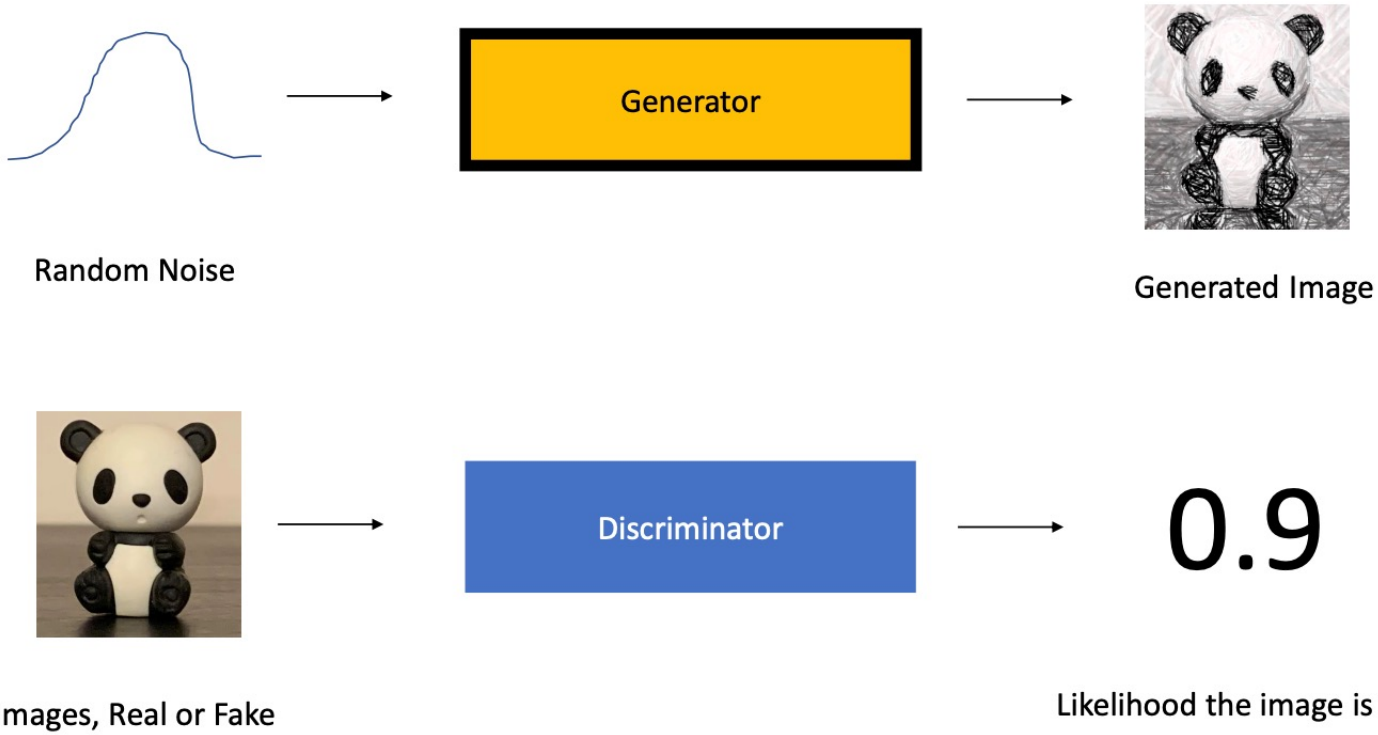
Generative Adversarial Networks



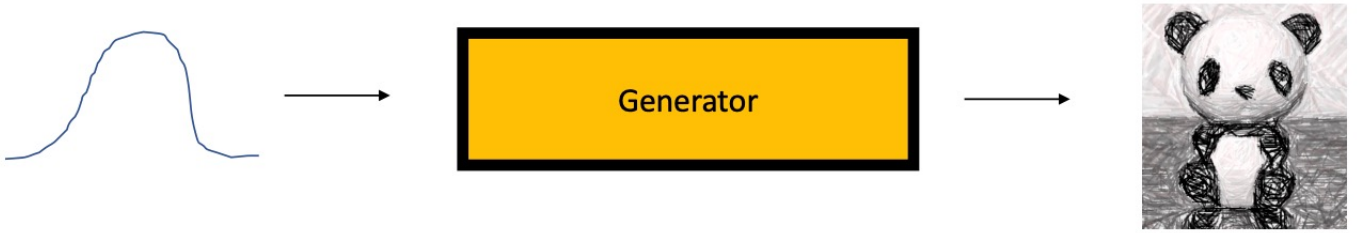
GANs are composed of two different networks:

- Generator Network
- Discriminator Network

Generative Adversarial Networks



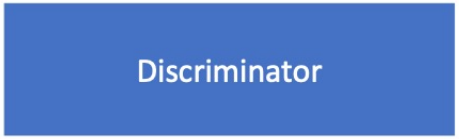
Generator



- Similar to variational autoencoder
- Uses upsampling instead of convolutional transpose, which has some problems creating checkerboard pixilation
- Binary cross-entropy loss function

Discriminator

Images, Real or Fake

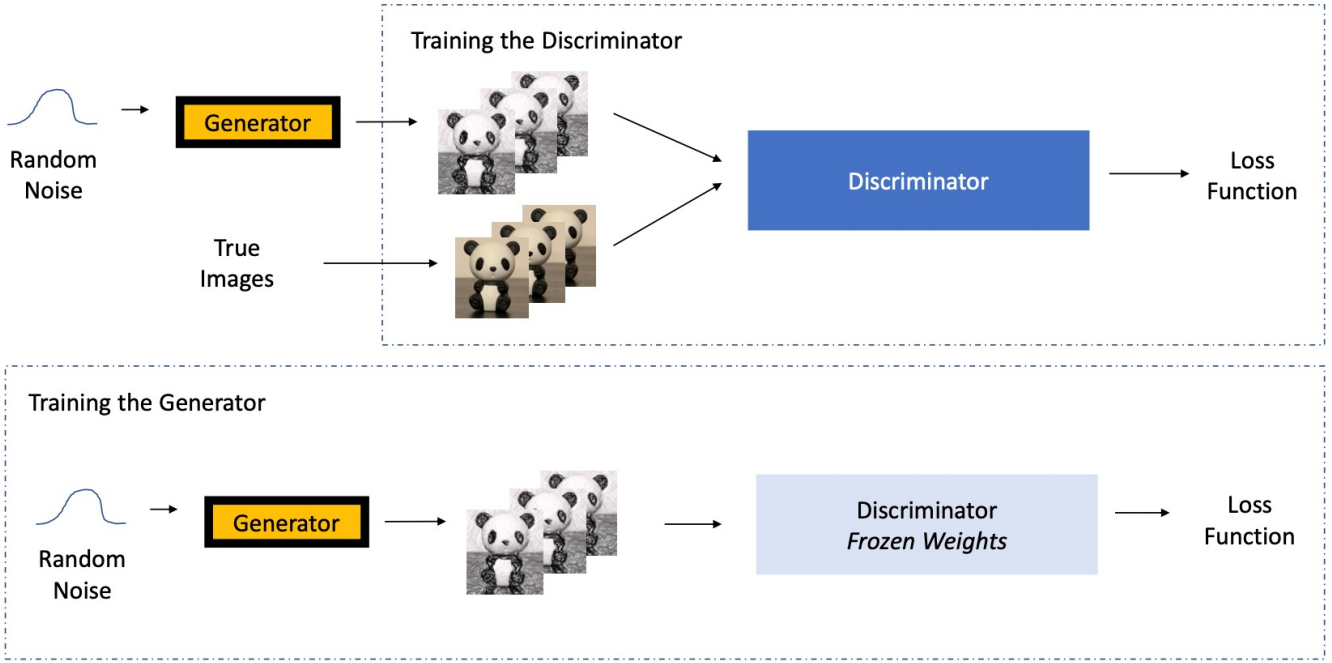


0.9

Likelihood the image is real

- Learning the difference between real and fake images, ie supervised
- Is trained before jointly training both models
- Uses a special deep convolutional layer, for a model called DCGAN

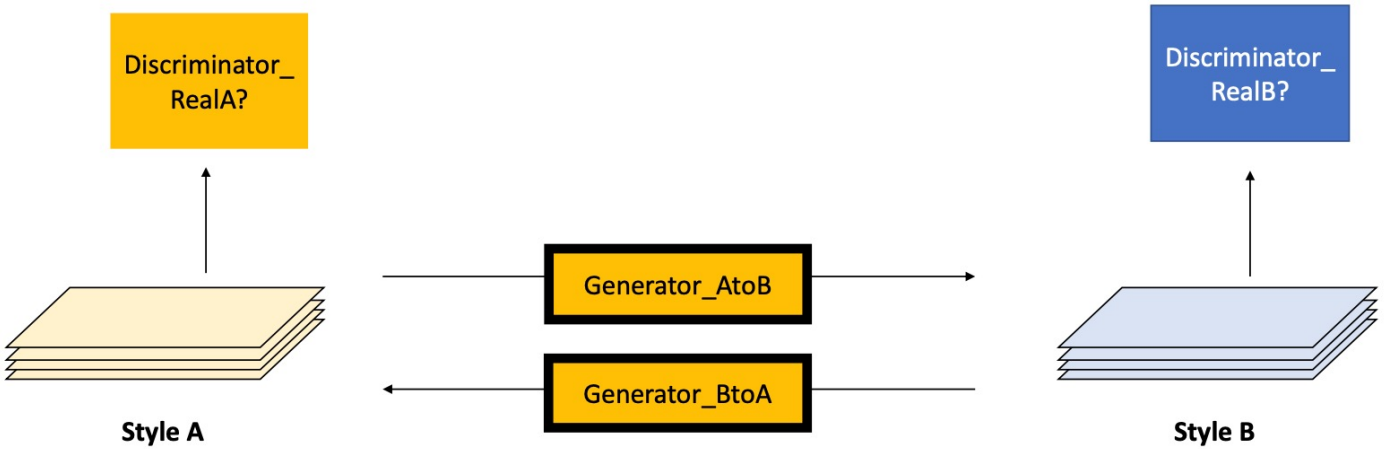
Dual-Step GAN Training



Generating Images

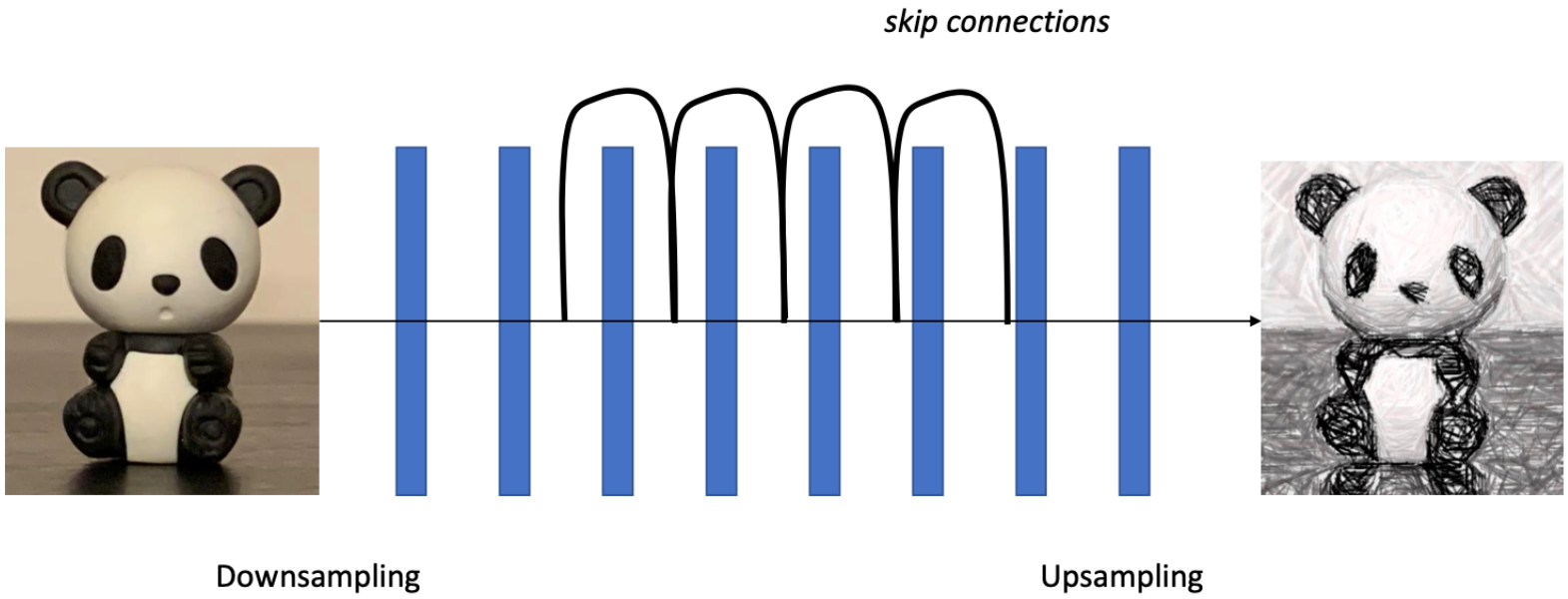
For fun, profit, and science

CycleGAN

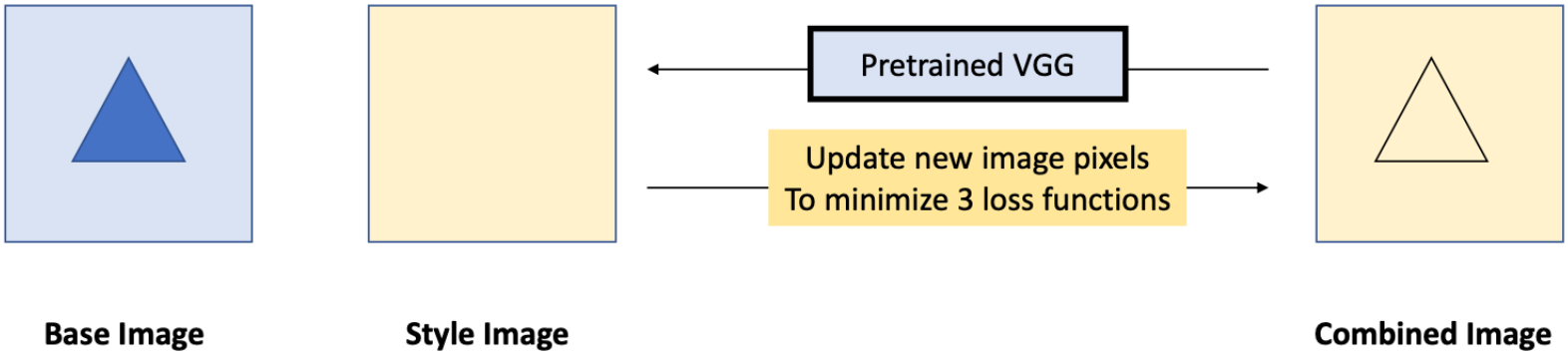


- Have 2 sets of images, eg Monet (A) and basic photographs (B)
- Use 4 models, two generators and two discriminators
- Train them to satisfy validity, reconstruction, and identity

ResNet For Image Generation



Neural Style Transfer



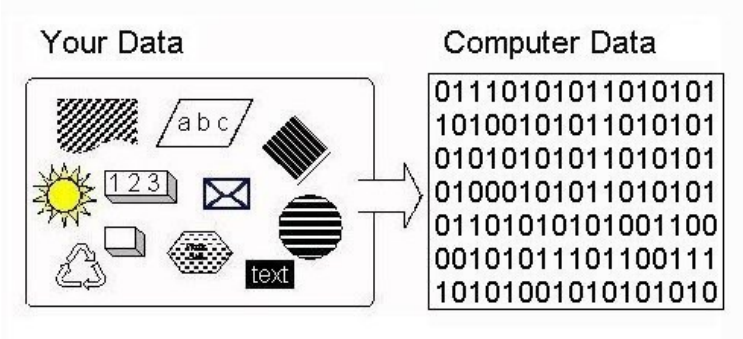
- Only 2 images, the base and the style image, with pretrained VGG
- Perform backpropagation base image pixels, updating to transfer style
- 3 loss functions: content, style, total variance

Generating Text

For fun, profit, and science

Representing text is not obvious

Representation is the key to Artificial Intelligence



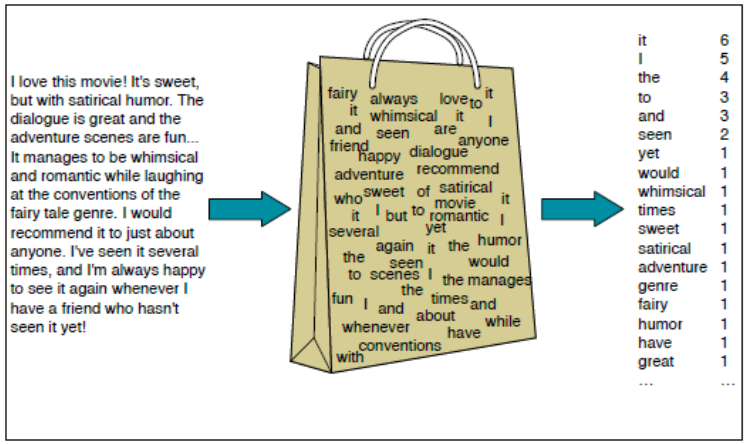
The Problem with Text: unstructured, messy, …

Machine learning algorithms cannot work with raw text directly; the text must be converted into numbers. Specifically, vectors of numbers.

In a nutshell, word embeddings are a type of word representation that allows words with similar meaning to have a similar representation.

LDA input: text in in matrix format: Document-term matrix or Document-word matrix

Bag of Words (BoW)



Bag of Words (BoW)

A bag-of-words is a representation of text that describes the occurrence of words within a document.

- Document/Sentence (Text from the book "A Tale of Two Cities"):

  It was the best of times,
  it was the worst of times,
  it was the age of wisdom,
  it was the age of foolishness,

- Design Vocabulary (unique words):

  "it" "was" "the" "best" "of" "times" "worst" "age" "wisdom" "foolishness"

- Document Vectors

It was the best of times,	*[1, 1, 1, 1, 1, 1, 0, 0, 0, 0]*
it was the worst of times,	*[1, 1, 1, 0, 1, 1, 1, 0, 0, 0]*
it was the age of wisdom,	*[1, 1, 1, 0, 1, 0, 0, 1, 1, 0]*
it was the age of foolishness,	*[1, 1, 1, 0, 1, 0, 0, 1, 0, 1]*

  BoW in in matrix format: Document-term matrix or Document-word matrix

Bag of Words (BoW)

Text cleaning or preprocessing:

- Ignoring case
- Ignoring punctuation
- Ignoring frequent words that don't contain much information, called stop words, like "a," "of," etc.
- Fixing misspelled words.
- Reducing words to their stem (e.g. "play" from "playing") using stemming algorithms.

N-gram: each word or token is called a "gram". N-gram is an N-token sequence of words:
2-gram: a bigram; 3-gram: trigram; …
Example: bigrams for "*It was the best of times*":
"it was" "was the" "the best" "best of" "of times"

Scoring Words: word count or word frequency

Term Frequency – Inverse Document Frequency: TF-IDF

- Problem with scoring word frequency: highly frequent words start to dominate in the document (e.g. larger score), yet may not contain as much "informational content" to the model as rarer but perhaps domain specific words.

- One approach is to rescale the frequency of words by how often they appear in all documents, so that the scores for frequent words like "the" that are also frequent across all documents are penalized.

TF-IDF: The idea is that high frequency may not able to provide much information gain

- **Term Frequency**: is a scoring of the frequency of the word in the current document.
- **Inverse Document Frequency**: is a scoring of how rare the word is across documents.

$$w_{x,y} = tf_{x,y} \times \log\left(\frac{N}{df_x}\right)$$

TF-IDF

Term x within document y

$tf_{x,y}$ = frequency of x in y

df_x = number of documents containing x

N = total number of documents

If you are still using bag of words, you should really be thinking about embeddings

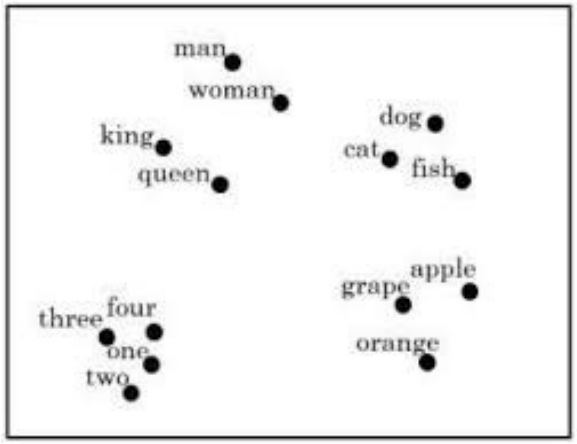
Is this word present?

What does this word mean?

Embeddings are *tunable*

Word Embeddings

In a nutshell, word embeddings are a type of word representation that allows words with similar meaning to have a similar representation.



- Traditional Word Embedding (Context-Free Representation)
  - Bag of Words or One Hot Encoding
  - TF-IDF Representation
- Pretrained Language Models (Contextualized representation)
  - Word2Vec
  - ELMo, ULMFiT, Open AI Transformer (GPT) and BERT

Pre-training in AI refers to training a model with one task to help it form parameters that can be used in other tasks.

Word Embeddings

Word2Vec: Google, 2013. A shallow neural network with each word's one-hot embedding as its input and output. Magical ability to measure word-to-word similarity. static word embedding: no matter the context, it will not change. can't handle polysemous words.

ELMo: Embeddings from Language Models.
AllenNLP, 2013, dynamic word embedding with LSTM. Contextual, Deep and Character based or feature-based method

ULMFiT: Universal Language Model Fine-tuning

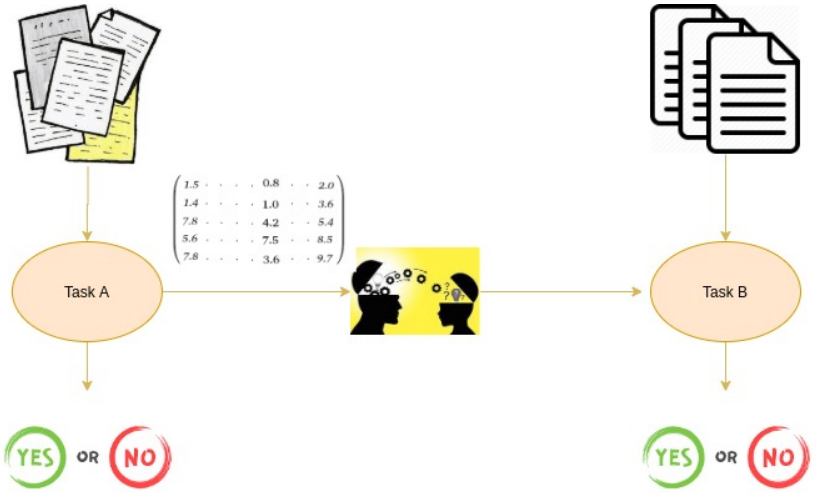
GPT: Generative Pre-Training
Transformer-decoder-based autoregressive language model. Fine-tuning-based method

BERT: Bidirectional Encoder Representations from Transformers
Transformer-encoder-based autoencoder language model.

XLNet: an AI technology that integrates GPT and BERT

Word2Vec

Pretrained Word Embeddings are the embeddings learned in one task that are used for solving another similar task, a form of **Transfer Learning**.



Why pretrained?

- Sparsity of training data
- Large number of trainable parameters

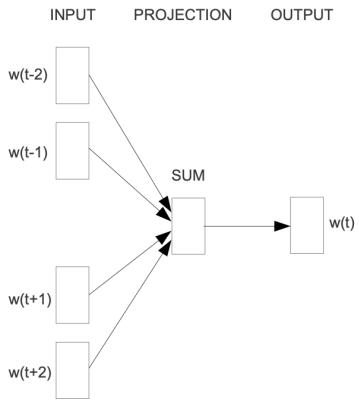
Word2Vec

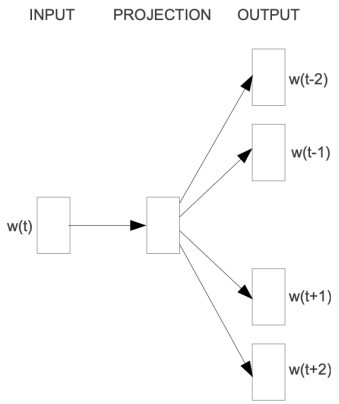
Word2Vec (by Google, ~ 100 billion words Google News dataset): the most popular by Mikolov et al. 2013. Other pre-trained word embedding models includes GloVe (by Stanford), fastText (by Facebook). word2vec uses a shallow neural network with each word's one-hot embedding as its input and output.

Two versions of Word2Vec: Continuous Bag of Words (CBOW) and Skip-Gram

Jay was hit by a red bus in …

input 1	input 2	input 3	input 4	output
by	a	bus	in	red





Jay was hit by a red bus in…

by	a	red	bus	in

input	output
red	by
red	a
red	bus
red	in

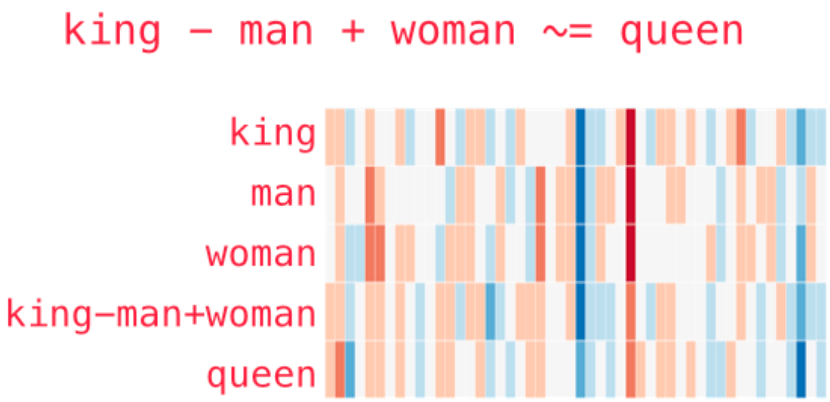
Figure 1: New model architectures. The CBOW architecture predicts the current word based on the context, and the Skip-gram predicts surrounding words given the current word.

Mikolov et al. Distributed Representations of Words and Phrases and their Compositionality, 2013

Word2Vec

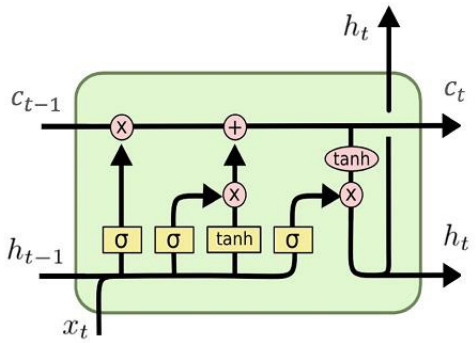
Word Embedding overcomes BoW's shortcomings by providing a dense distributed representation in low dimensional vector and expressive representation using contextual similarity.

Word2Vec is a learned representation (pre-trained, real-valued vectors) for text where words that have the same meaning have a similar representation.



The Illustrated Word2vec. http://jalammar.github.io/illustrated-word2vec/

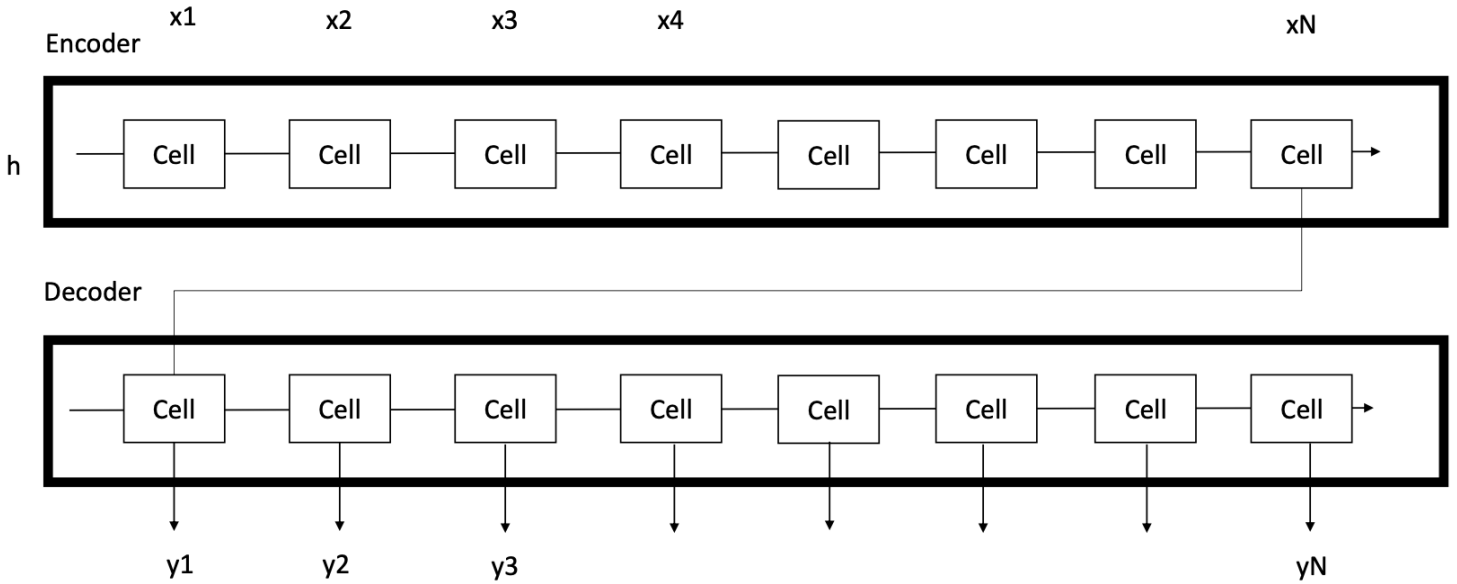
Generating the next word in a sequence – LSTM



LSTM
(Long-Short Term Memory)

- Stack an LSTM to learn deeper features
- Can optionally use a GRU cell
- Bidirectional layers can also be helpful

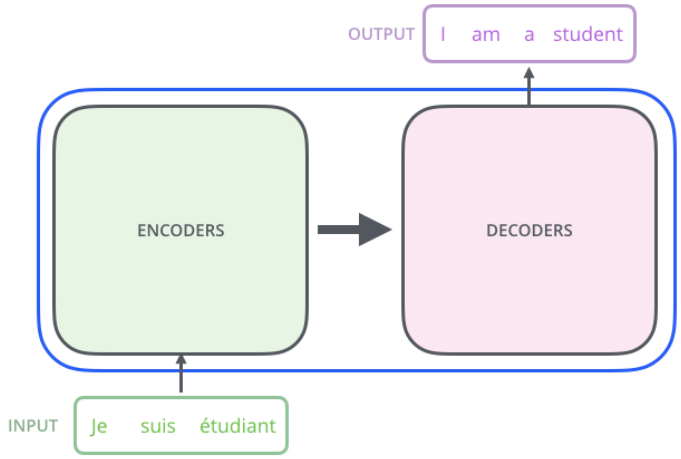
Generating totally new sequences: Encoder - Decoder



Useful for language translation, question generation, text summarization

Transformer

(Vaswani et al. Attention Is All You Need, 2017)



Google AI

June 2017

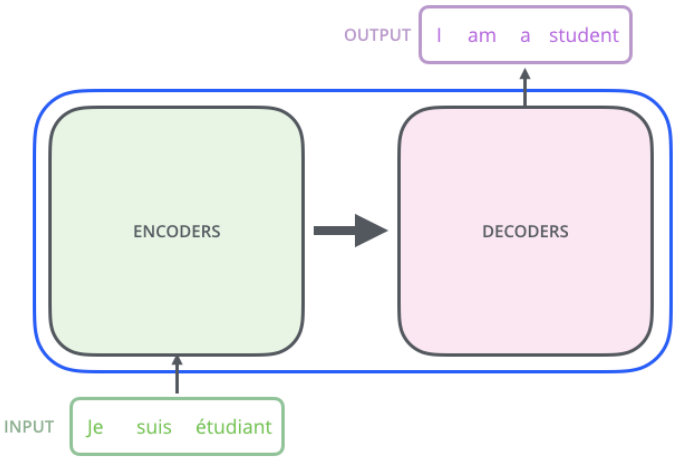
- Surpassed RNN's State-of-the-Art
- Transformer Architecture
  - A model architecture eschewing recurrence and instead relying entirely on an attention mechanism to draw global dependencies between input and output
  - Uses attention to boost the speed for significantly more parallelization
  - Use Multi-Head Attention

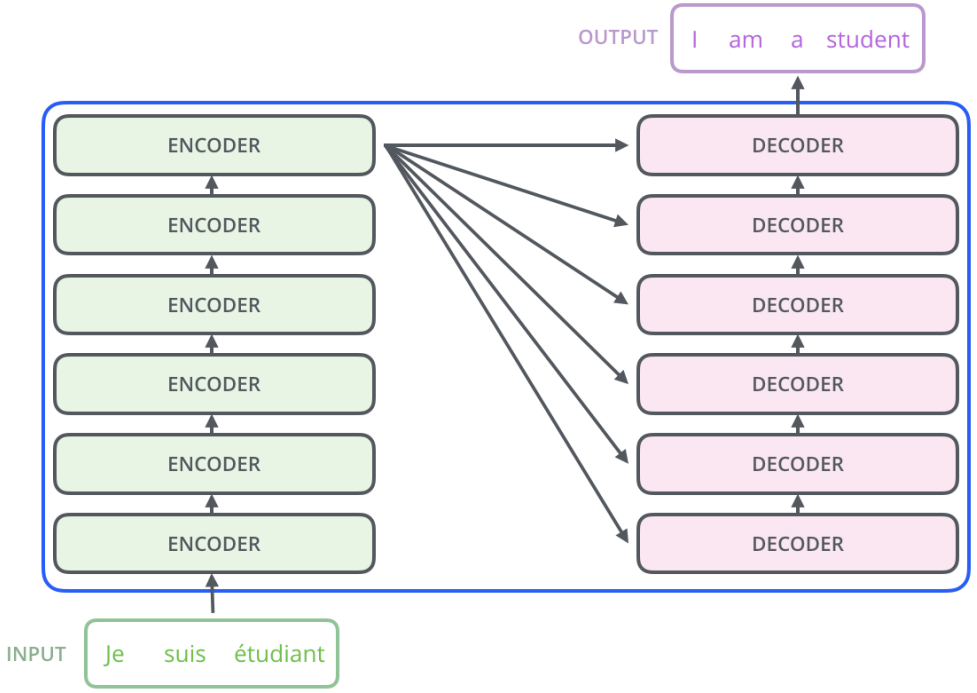
Transformer (Vaswani et al. Attention Is All You Need, 2017)

- RNN based architectures are **hard to parallelize** and can have difficulty learning **long-range dependencies** within the input and output sequences
- The Transformer models all these dependencies using **attention**
- Instead of using one sweep of attention, the Transformer uses multiple "**heads**" (multiple attention distributions and multiple outputs for a single input).
- In addition to attention, the Transformer uses layer normalization and residual connections to make optimization easier.
- Attention cannot utilize the positions of the inputs. To solve this, the Transformer uses **explicit position encodings** are added to the input embeddings
- This architecture achieves state-of-the-art performance on English-to-German and English-to-French translation and performs well on constituency parsing

Transformer

(Vaswani et al. Attention Is All You Need, 2017)





a stack of 6 encoders, a stack of 6 decoders

6 encoder layers, 512 hidden units, and 8 attention heads

BERT

Bidirectional **E**ncoder **R**epresentations from **T**ransformers

Dubbed ImageNet moment for NLP: mark the beginning of a new era in NLP

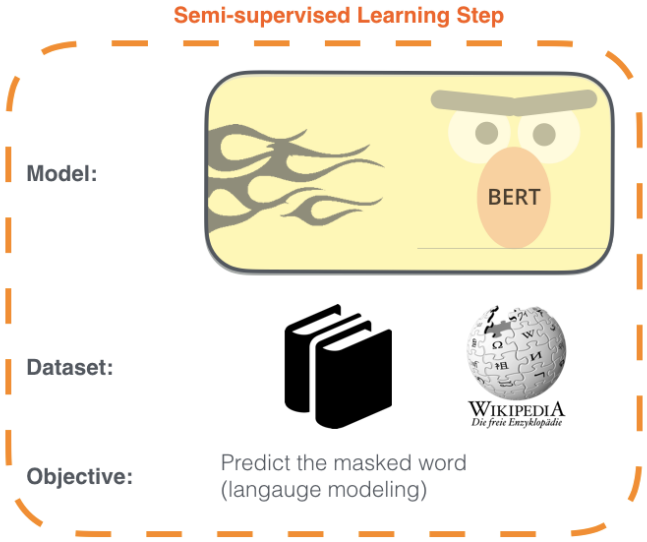
- State-of-the-Art Pre-training for Natural Language Processing
- Build upon recent work in pre-training contextual representations — including Semi-supervised Sequence Learning, Generative Pre-Training (GPT), ELMo, and ULMFit
- A bunch of **Transformer** encoders stacked together (not the whole Transformer architecture but just the encoder)
- The first deeply bidirectional, unsupervised language representation, pre-trained using only a plain text corpus, then it to "transfer learn" other tasks (Fine-tuning)
- Bidirectionality is the key differentiator between BERT and OpenAI GPT
- Pre-training: Masked Language Model and Next Sentence Prediction

Two steps of how BERT is developed

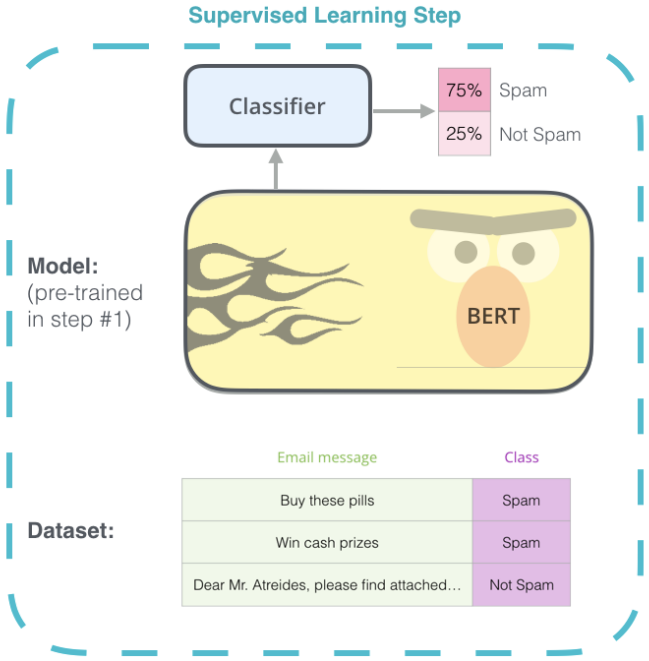
You can download the **model** pre-trained in step 1 (trained on un-annotated data), and only worry about fine-tuning **it for step 2**

1 - Semi-supervised training on large amounts of text (books, wikipedia..etc).

The model is trained on a certain task that enables it to grasp patterns in language. By the end of the training process, BERT has language-processing abilities capable of empowering many models we later need to build and train in a supervised way.



2 - Supervised training on a specific task with a labeled dataset.



GPT-2: Generative Pre-Training

- A successor to GPT, trained simply to predict the next word in 40GB of Internet text (WebText)
- GPT-2 is a large transformer-based language model with 1.5 billion parameters, trained on a dataset of 8 million web pages
- GPT-2 is a direct scale-up of GPT, with more than 10X the parameters and trained on more than 10X the amount of data.
- Too dangerous to release - fears that their technology would be misused for "malicious applications" such as spam or fake news

OpenAI has released the full 1.5 billion parameter version of GPT-2, a large-scale unsupervised language model for automatic text generation. 11/5/2019

(Radford et al. Language Models are Unsupervised Multitask Learners, 2019)

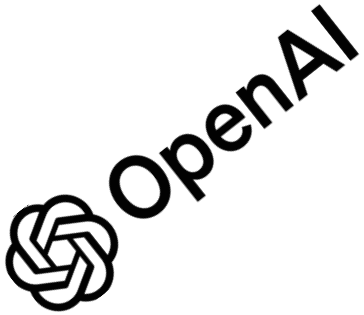
GPT-3: Generative Pre-Training

- 05/29/2020, OpenAI debuts gigantic GPT-3 language model with 175 billion parameters
- Strong performance in translation, question-answering, and cloze tasks, on-the-fly reasoning, arithmetic, reading comprehension etc
- No fine-tuning!
- Common Crawl (trained on 45TB text) + WebText + Books + Wikipedia
- $12M to train
- Energy use: several thousand petaflop/s-days
- Bert-large (0.3B), GPT-2 (1.5B), Megatron-LM (8.3B, NVIDIA), T5 (11B, Google), Turing NLG (17B, Microsoft), …

(Brown et al. Language Models are Few-Shot Learners, 2020)

GPT-4: OpenAI's Most Advanced System

March 14, 2023



- Large multimodal: image + text as inputs, text outputs
- Human-level performance: outperform ChatGPT by scoring in the top 10% of test takers (90th percentile on BAR exam, SAT, GRE etc)
- Surpass ChatGPT in its advanced reasoning capabilities
- 25,000 words context: allows full documents to fit *within a single prompt*
- More creative & collaborative: generate, edit, and iterate with users on writing tasks
- Models available via API: `gpt-4, gpt-4-0314, gpt-4-32k, and gpt-4-32k-0314`

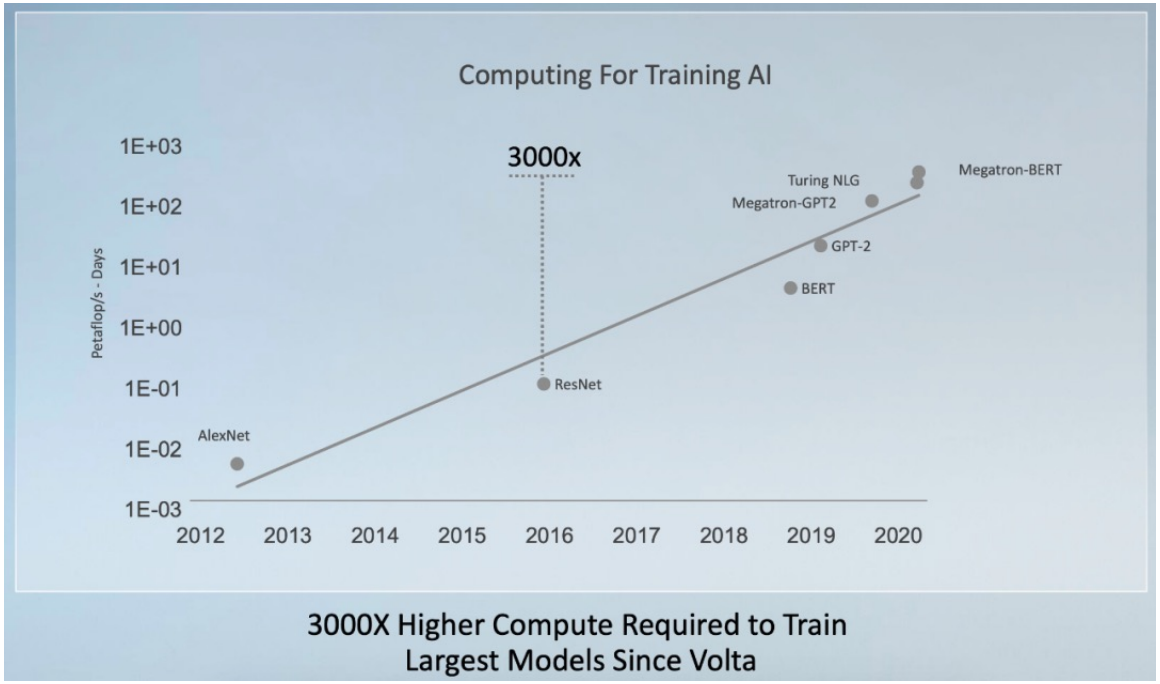
Model	Prompt Cost	Completion Cost	Context Window
gpt-4	$0.03/1K prompt	$0.06/1K completion	8K context window (~13 pages)
gpt-4-32k	$0.06/1K prompt	$0.12/1K completion	32K context window (~52 pages)

Switch Transformers

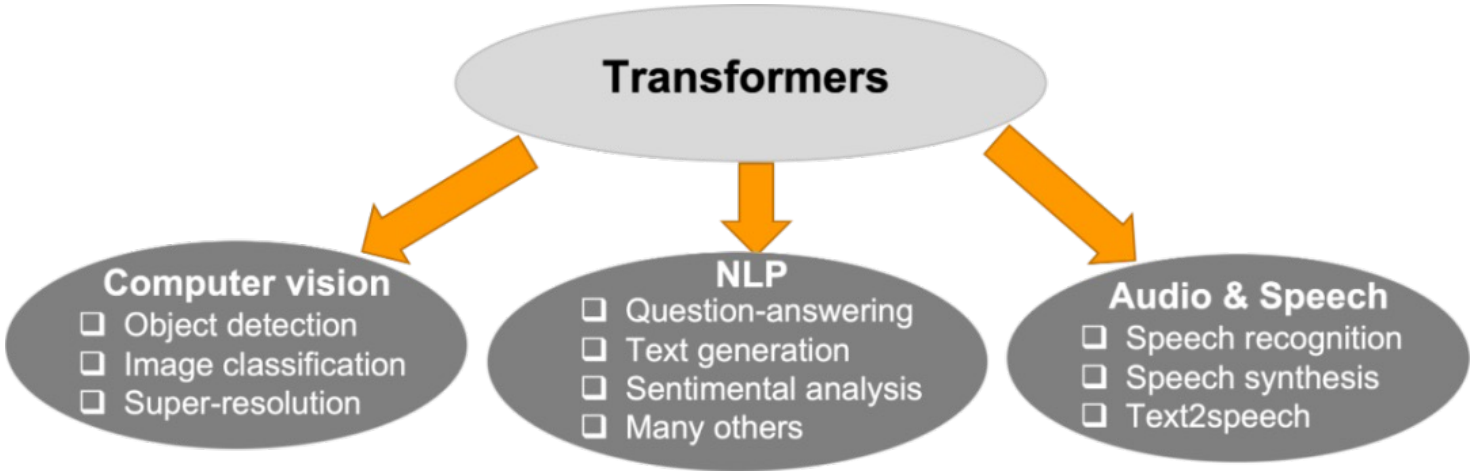
- 01/11/2021, 1.6-trillion-parameter model (9x GPT-3) by Google Brain
- Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity
- Mixture of Experts (MoE), sparsity, lower precision (bfloat16)
- Pre-trained on Colossal Clean Crawled Corpus - ~750GB of cleaned text from Common Crawl
- Significant speedup and state-of-the-art performance in multilingual settings (101 languages)

(Fedus et al. Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity, 2021)

From BERT to ALBERT: Pre-trained Language Models

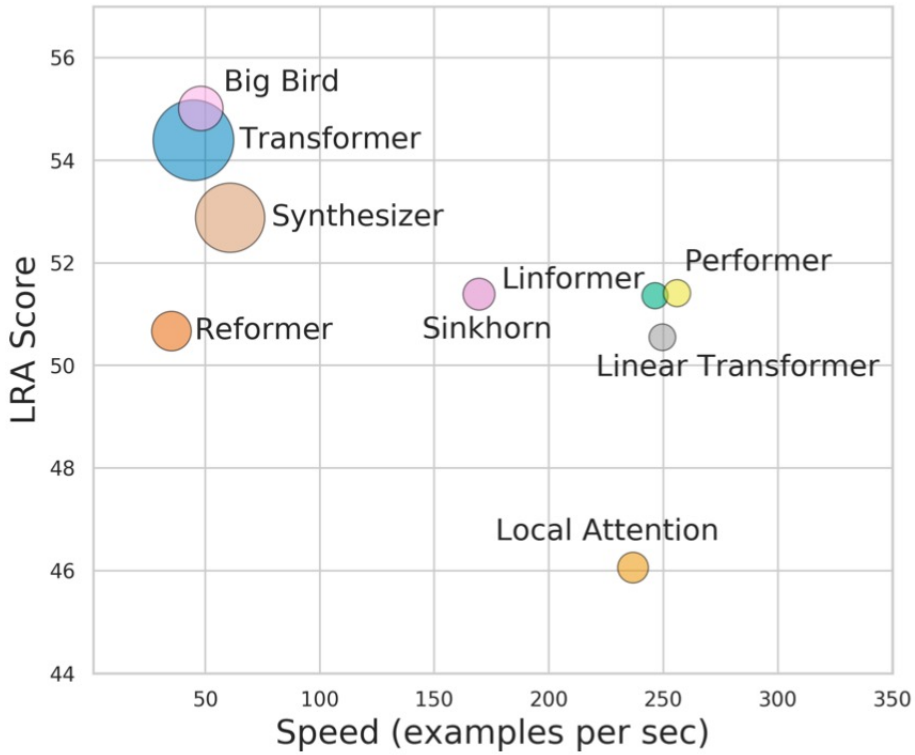


Transformers: Beyond NLP



https://mp.weixin.qq.com/s/OgTQ3O_6lvOG07U-tjpTDA

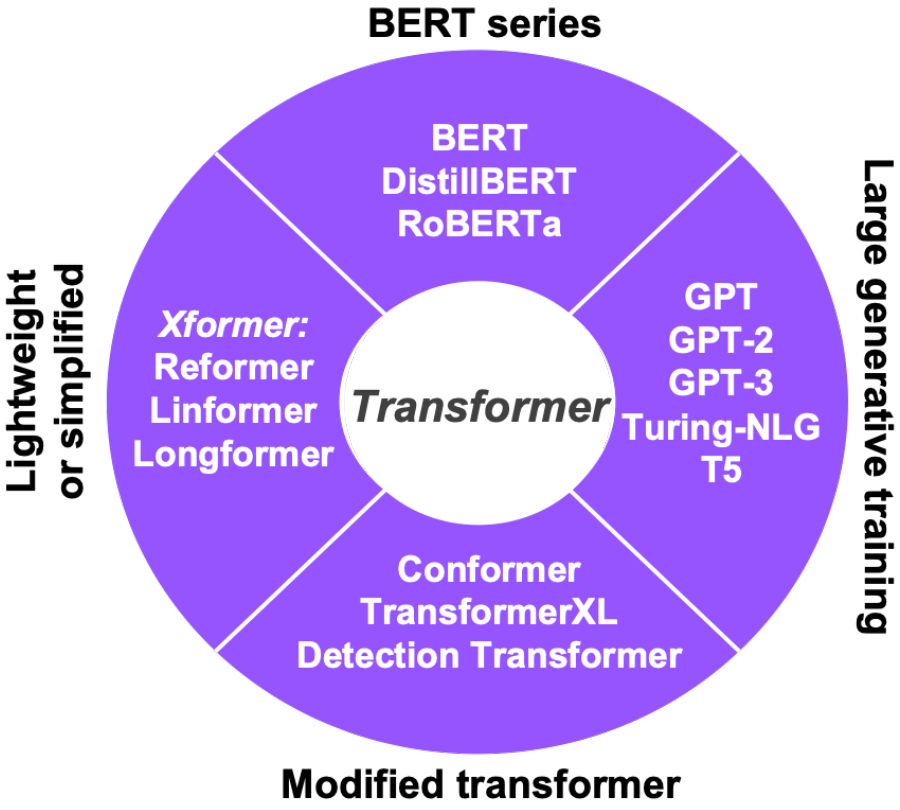
Transformer Family



Performance (y axis), speed (x axis), and memory footprint (size of the circles) of different models

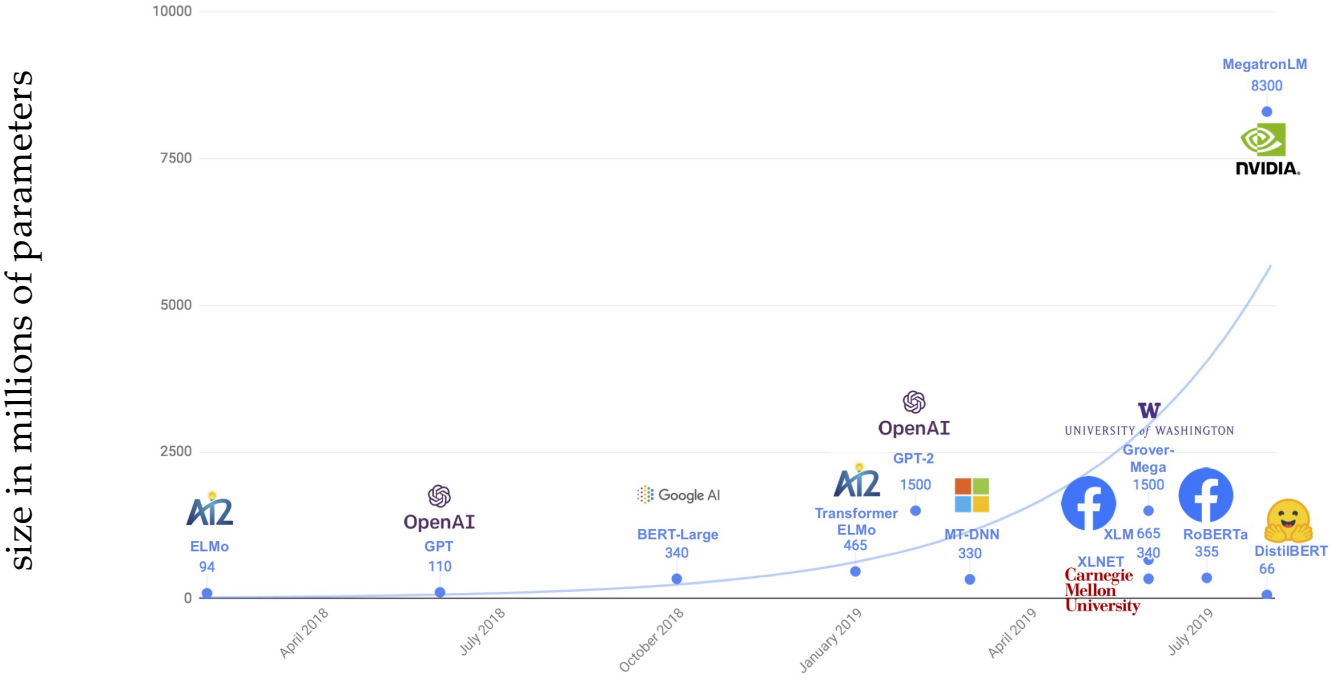
Tay et a. Long Range Arena : A Benchmark for Efficient Transformers, 2020

Transformer Family



🏎 Smaller, faster, cheaper, lighter: DistilBERT (a distilled version of BERT): deal-breaking

Oct 2019



Knowledge distillation (sometimes also referred to as *teacher-student learning*) is a **compression technique in which a small model is trained to reproduce the behavior of a larger model**

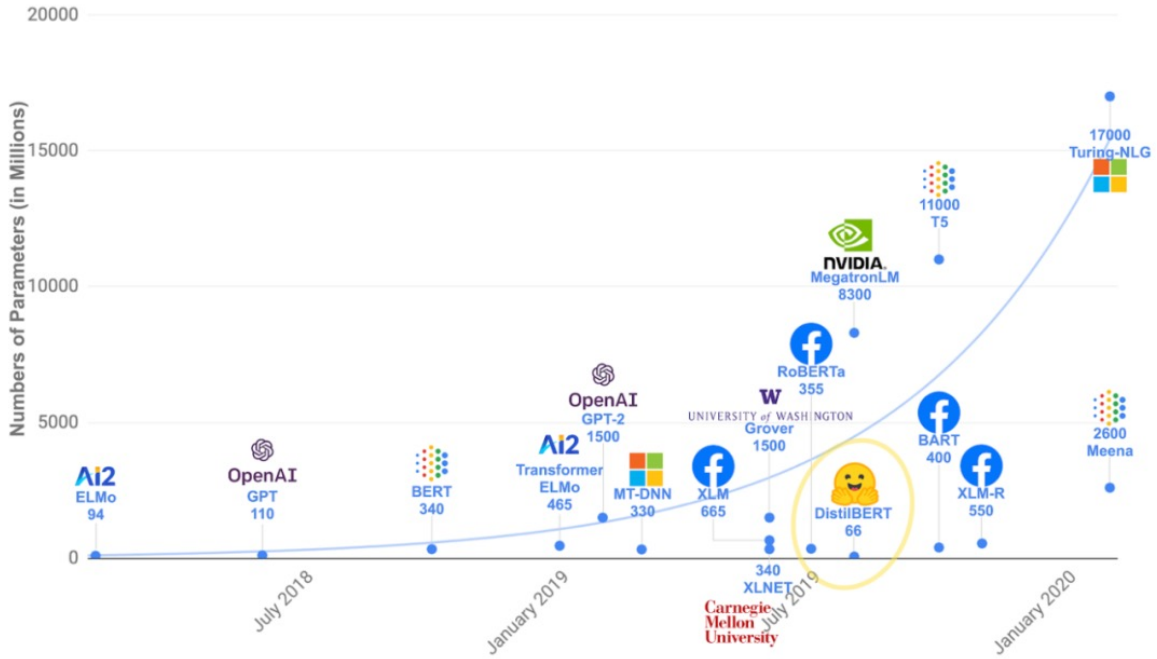
DistilBERT by HuggingFace showed that it is possible to reduce the size of a BERT model by 40% while retaining 97% of its language understanding capabilities and being 60% faster.

Smaller, faster, cheaper, lighter: DistilBERT (a distilled version of BERT): deal-breaking

Microsoft

Feb 2020

Turing-NLG
17B

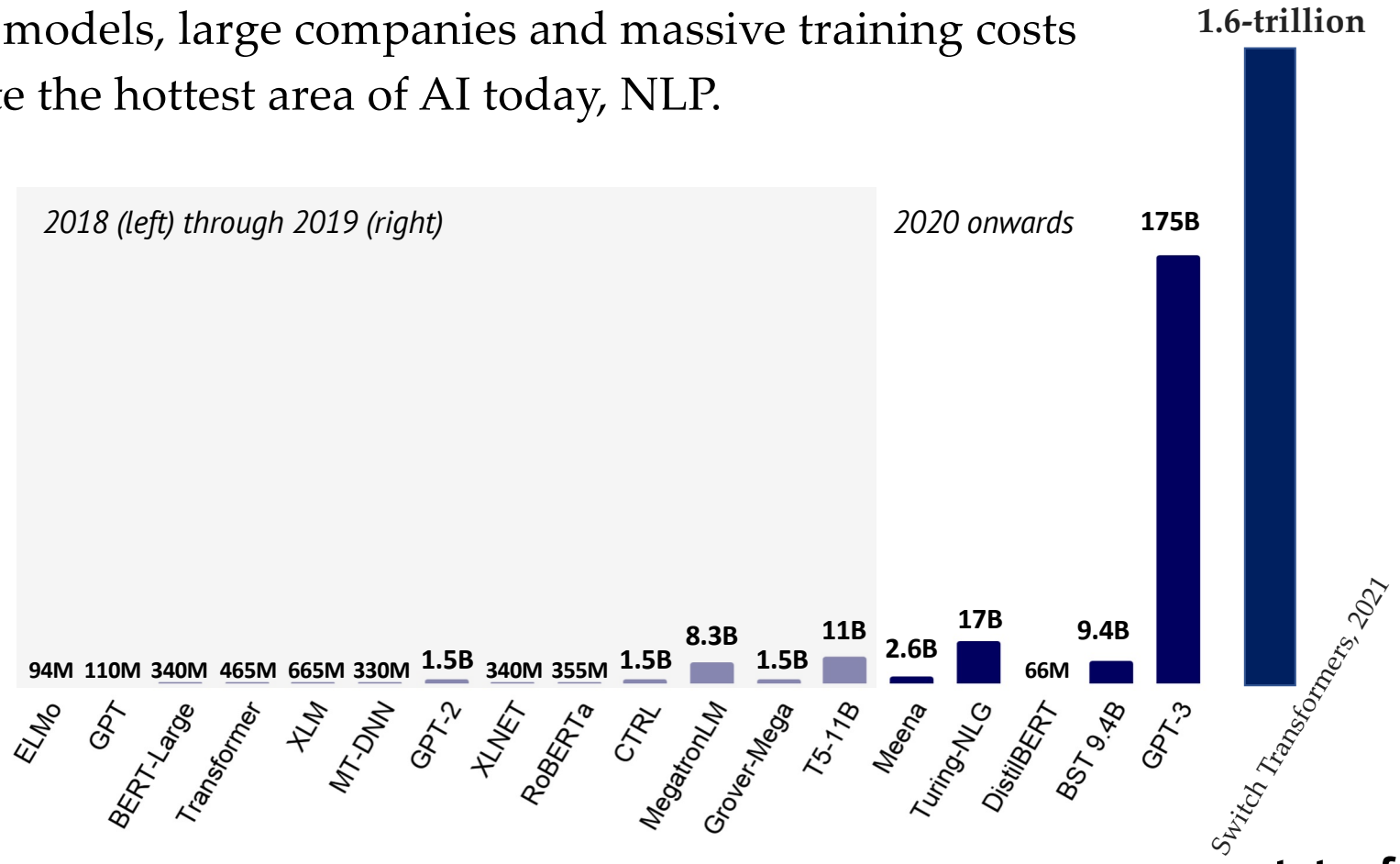


Knowledge distillation (sometimes also referred to as *teacher-student learning*) is a **compression technique in which a small model is trained to reproduce the behavior of a larger model**

DistilBERT by HuggingFace showed that it is possible to reduce the size of a BERT model by 40% while retaining 97% of its language understanding capabilities and being 60% faster.

Language models: Welcome to the Billion Parameter club

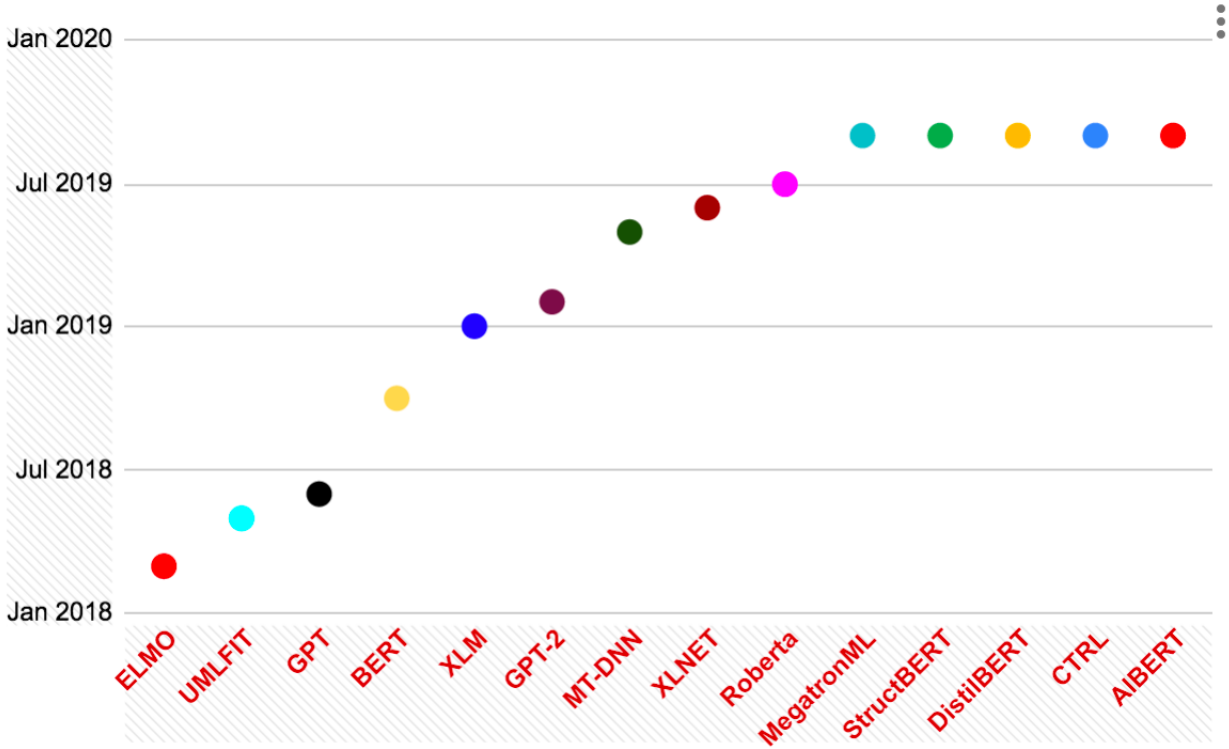
Huge models, large companies and massive training costs dominate the hottest area of AI today, NLP.



Note: The number of parameters indicates how many different coefficients the algorithm optimizes during the training process.

stateof.ai 2020

From BERT to ALBERT: Pre-trained Language Models

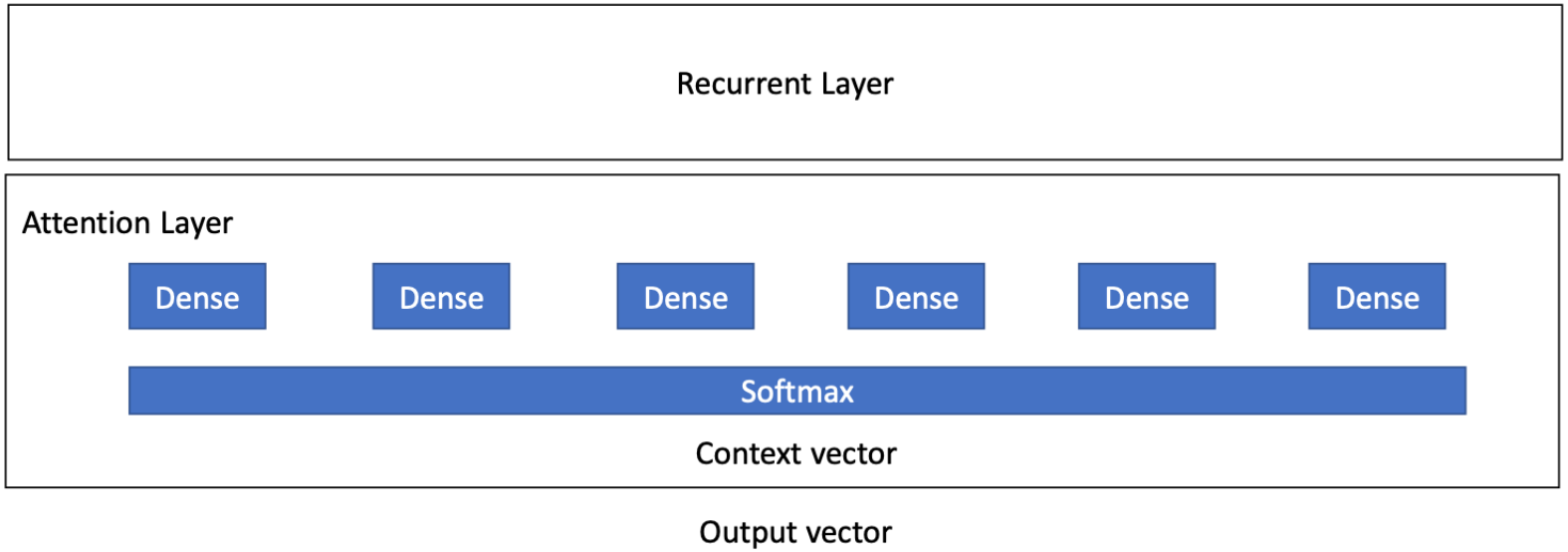


DistilBERT by HuggingFace showed that it is possible to reduce the size of a BERT model by 40% while retaining 97% of its language understanding capabilities and being 60% faster.

Generating Music

For fun, profit, and science

Generating Music



- Represent music digitally through MIDI file
- Here, each "word" has a pitch and duration
- Generate a context vector, but pull in information from very far back in the sequence

Generating Polyphonic Music with MuseGAN