Summer semester 2023

344.063 KV Special Topic:

Natural Language Processing with Deep Learning

Large Language Models

Navid Rekab-saz

navid.rekabsaz@jku.at

JOHANNES KEPLER UNIVERSITY LINZ

Institute of Computational Perception

Agenda

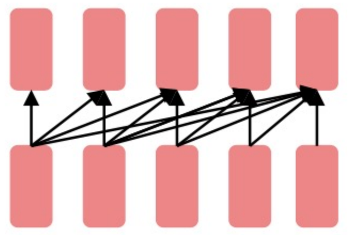
- LLM categories
- A tour around!
- Compression and overparameterization

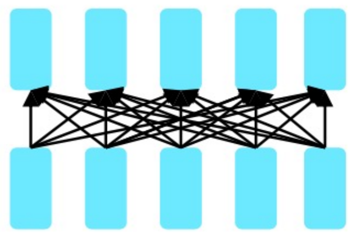
Agenda

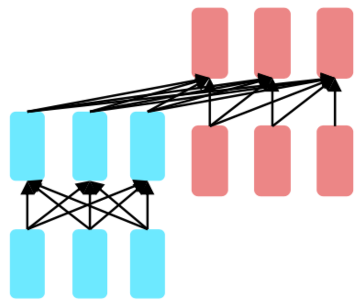
- **LLM categories**
- A tour around!
- Compression and overparameterization

Three types of large-scale pretrained LMs

- Decoders
  - "Normal" LM objective: predict the next token conditioned on the previous tokens (unidirectional)
  - Training and inference is auto-regressive (one after each other)
  - Particularly suited for generating text
- Encoders
  - Input is encoded into contextualized embeddings
  - Some variations (like BERT) also provide sequence embedding and pair-sequence embedding
  - Training is bidirectional – model sees whole the sequence (past and future)
- Encoder-Decoders
  - The encoder encodes whole the input (bidirectional)
  - The decoder generates the output in auto-regressive fashion

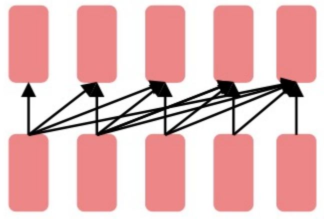




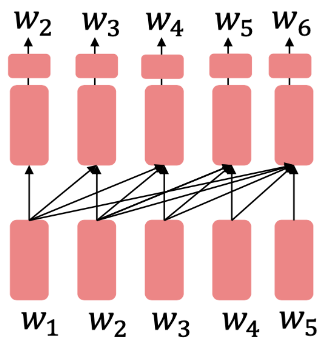


Decoders LM

- The architecture can be composed of (multi-layers of) RNNs or Transformers

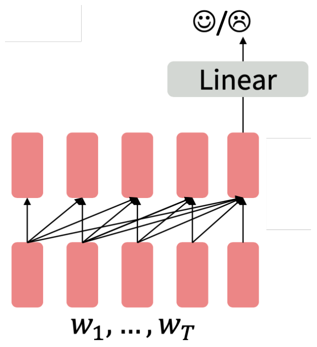


- Training
  - "normal" LM objective: predict the probability distribution of the next word and optimize the network based on the actual word



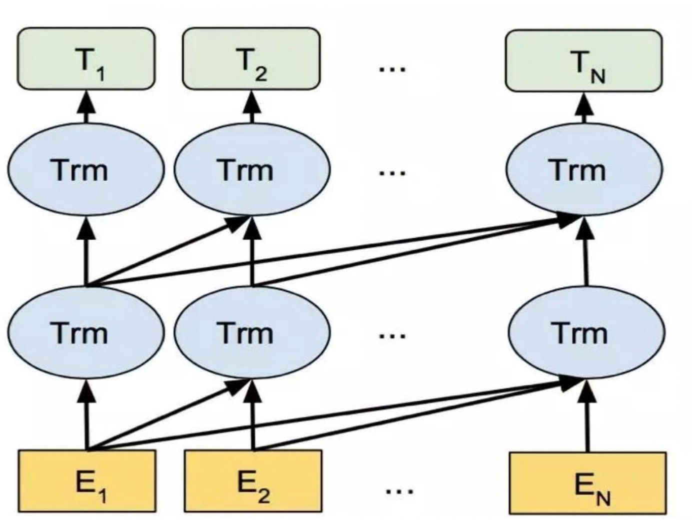
- Text generation
  - Next word is generated by sampling from the predicted probability distribution

- Downstream tasks
  - Fine-tuning and prediction in downstream task is commonly done using the last output embedding



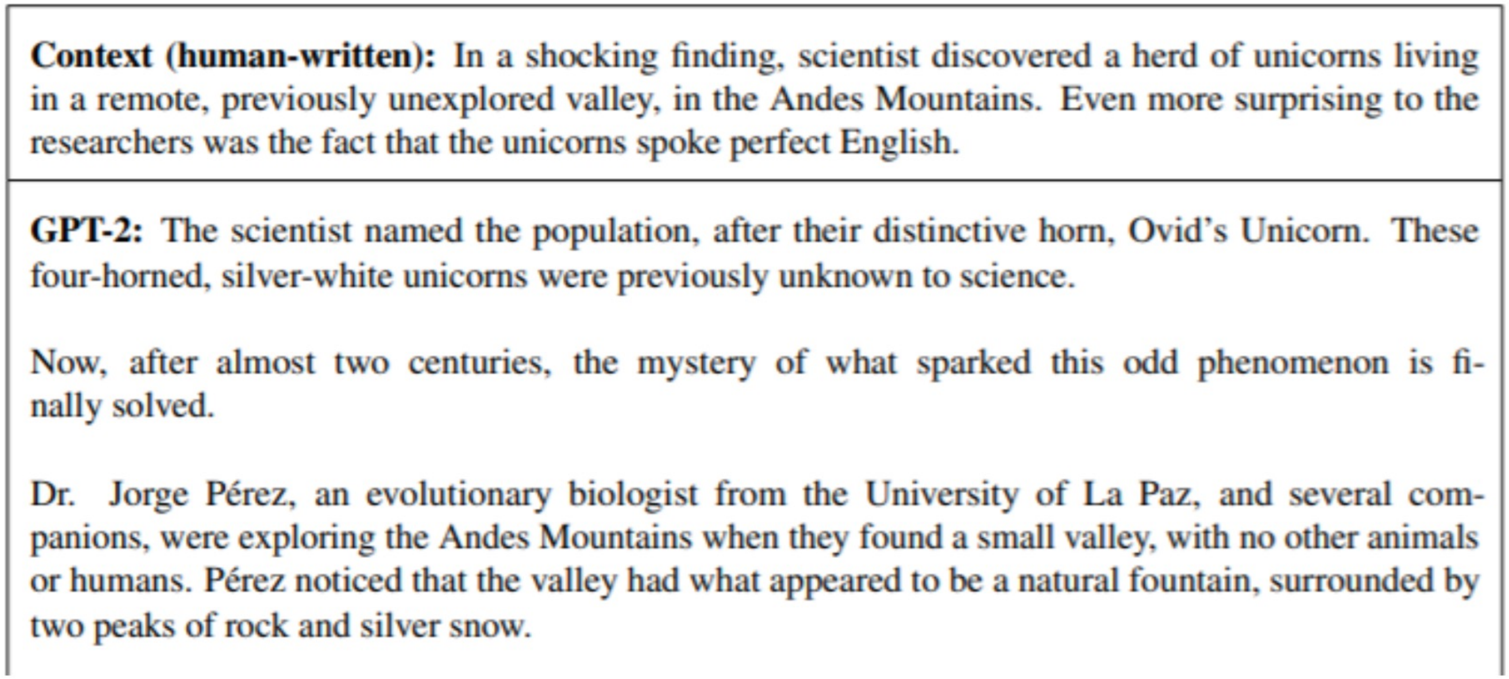
GPT: Generative Pretrained Transformer

- 12 layers of Transformer decoder
- Tokenization using Byte-pair encoding
- Trained on BooksCorpus
- GPT is followed by much larger models: GPT-2 and GPT-3



Text Generation

- GPT-2 and later GPT-3 show very “convincing” results on text generation
  - Try here: https://transformer.huggingface.co



Context (human-written): In a shocking finding, scientist discovered a herd of unicorns living in a remote, previously unexplored valley, in the Andes Mountains. Even more surprising to the researchers was the fact that the unicorns spoke perfect English.

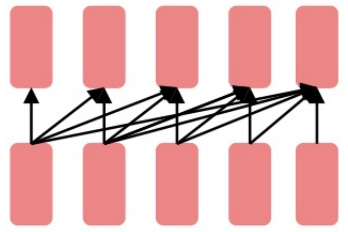
GPT-2: The scientist named the population, after their distinctive horn, Ovid’s Unicorn. These four-horned, silver-white unicorns were previously unknown to science.

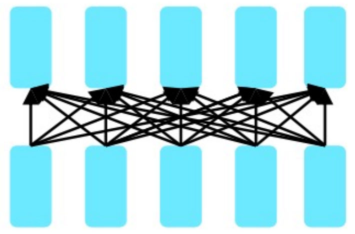
Now, after almost two centuries, the mystery of what sparked this odd phenomenon is finally solved.

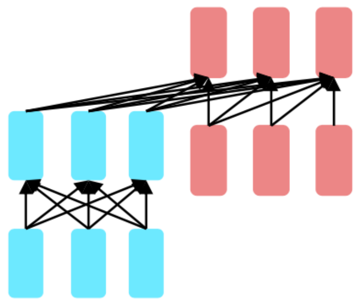
Dr. Jorge Pérez, an evolutionary biologist from the University of La Paz, and several companions, were exploring the Andes Mountains when they found a small valley, with no other animals or humans. Pérez noticed that the valley had what appeared to be a natural fountain, surrounded by two peaks of rock and silver snow.

Three types of large-scale pretrained LMs

- Decoders
  - "Normal" LM objective: predict the next token conditioned on the previous tokens (unidirectional)
  - Training and inference is auto-regressive (one after each other)
  - Particularly suited for generating text
- Encoders
  - Input is encoded into contextualized embeddings
  - Some variations (like BERT) also provide sequence embedding and pair-sequence embedding
  - Training is bidirectional – model sees whole the sequence (past and future)
- Encoder-Decoders
  - The encoder encodes whole the input (bidirectional)
  - The decoder generates the output in auto-regressive fashion

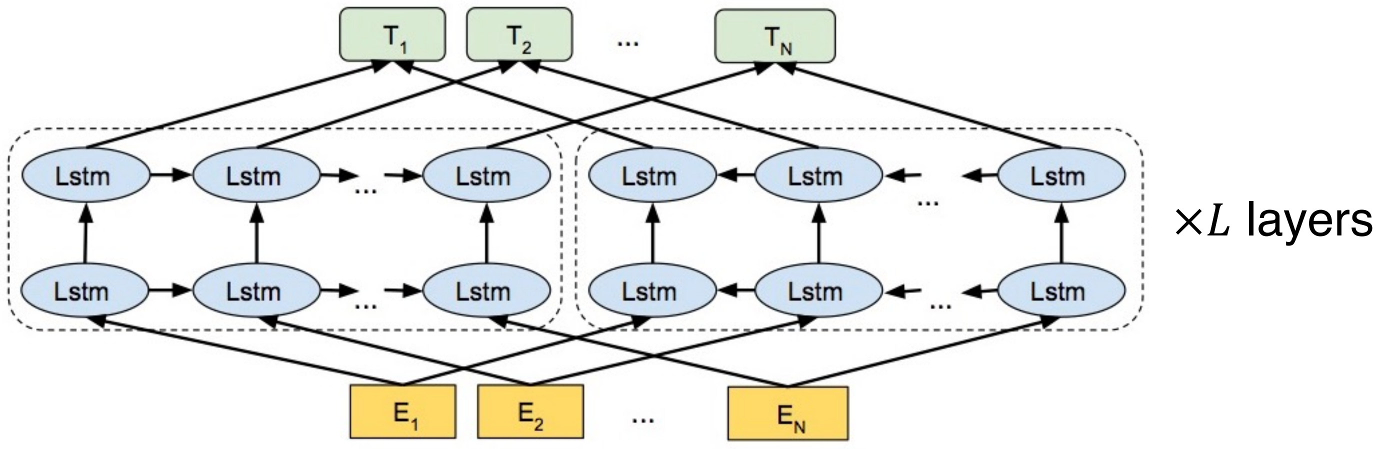






ELMo: Embeddings from Language Models

- ELMo's architecture: Multi-layer Bidirectional LSTM
- The model outputs the contextualized embeddings of the words of the input sequence
- Trained by predicting the next word, done in both directions
  - One time from left to right and one time from right to left
- Input word embeddings are created using character-based CNN
- Time complexity of both training and inference is a factor of sequence length
  - Due to the "step-by-step" nature of RNNs



Masked Language Model (MLM)

- Limitation of “normal” language modeling objective: it is constrained to using only the left (or right) context
  - Though language understanding requireds the full context!

- Masked Language Model masks out $k\%$ of the input sequence and predicts the masked words in output

<u>**Example**</u>

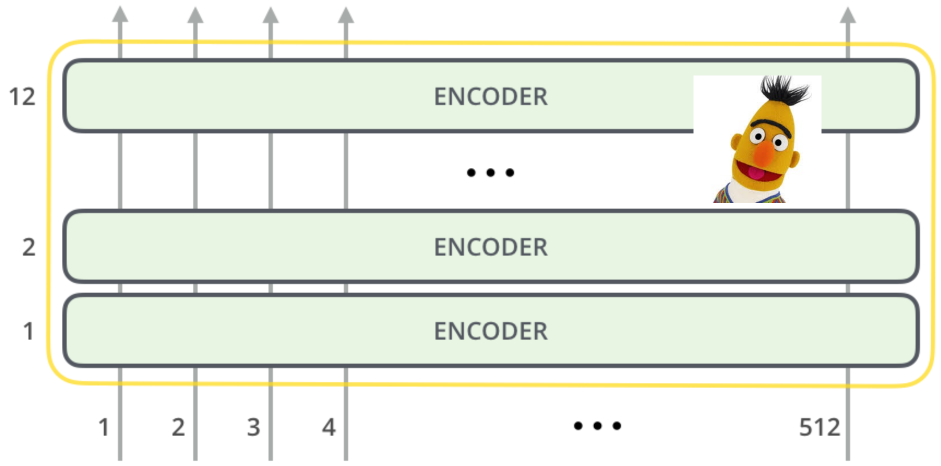
sequence: Jim made spaghetti for his girlfriend and he was very proud!

Input: Jim made [MASK] for his girlfriend and [MASK] was very proud!

$$\begin{tikzcd}[column sep=huge]
\text{[MASK]} \arrow[d] & \text{[MASK]} \arrow[d] \\
\text{spaghetti} & \text{he}
\end{tikzcd}$$

predict: spaghetti he

BERT : Bidirectional Encoder Representation from Transformers



- BERT is a pre-trained language model, composed of multi-layers of Transformer encoder
- BERT …
  - provides <u>contextualized word embeddings</u>
  - uses <u>WordPiece</u> for tokenization
  - uses sentence (sequence) pair encoding which provides …
    - <u>sequence embedding</u>
    - an embedding for <u>relation estimation</u> between <u>two sequences</u>

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2019, June). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proc. of NAACL-HLT (2019)*

Training

- Trained using MLM on Wikipedia + BookCorpus

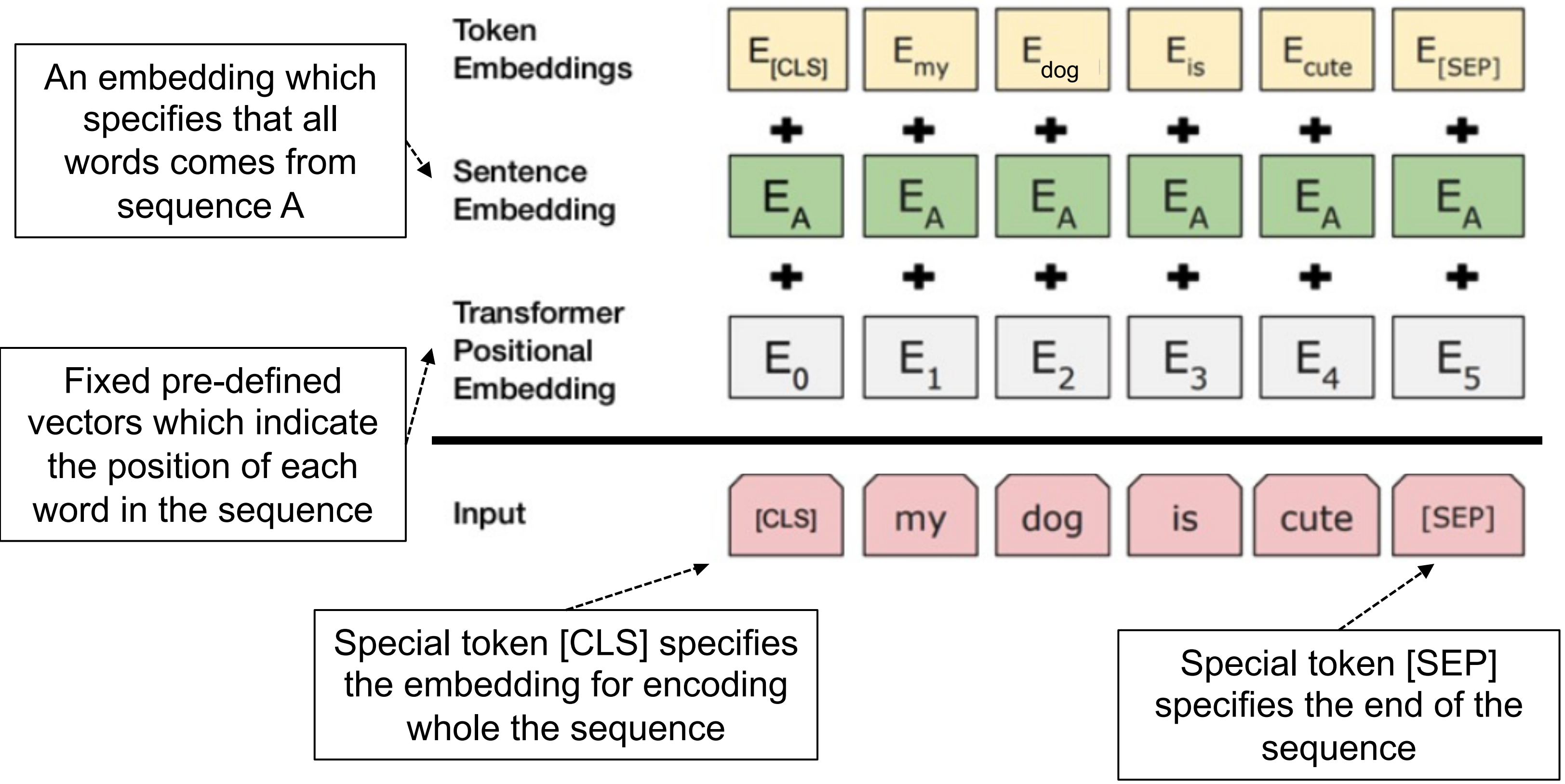
- Dictionary size is ~30K tokens (due to WordPiece subword tokenization)
- Specs of some provided pre-trained models:
  - BERT-Tiny: 2-layer, 128-hidden, 2-head, ~4M parameters*
  - BERT-Mini: 4-layer, 256-hidden, 4-head, ~11M parameters*
  - BERT-Base: 12-layer, 768-hidden, 12-head, ~110M parameters*
  - BERT-Large: 24-layer, 1024-hidden, 16-head, ~340M parameters*

- Some resources:
  - https://github.com/google-research/bert
  - Library to have BERT models in PyTorch: https://huggingface.co/transformers/

* For comparison, a (static) word embedding like word2vec with vocabulary size 200K and vector size 768 has 153M parameters

Input to BERT – one sequence

- The input embeddings to BERT are in fact the sum of three types of embeddings


An embedding which specifies that all words comes from sequence A
Fixed pre-defined vectors which indicate the position of each word in the sequence
Token Embeddings
Sentence Embedding
Transformer Positional Embedding
Input
$E_{[CLS]}$ E_{my} E_{dog} E_{is} E_{cute} $E_{[SEP]}$
E_A E_A E_A E_A E_A E_A
E_0 E_1 E_2 E_3 E_4 E_5
[CLS] my dog is cute [SEP]
Special token [CLS] specifies the embedding for encoding whole the sequence
Special token [SEP] specifies the end of the sequence

Sentence (Sequence) pair encoding

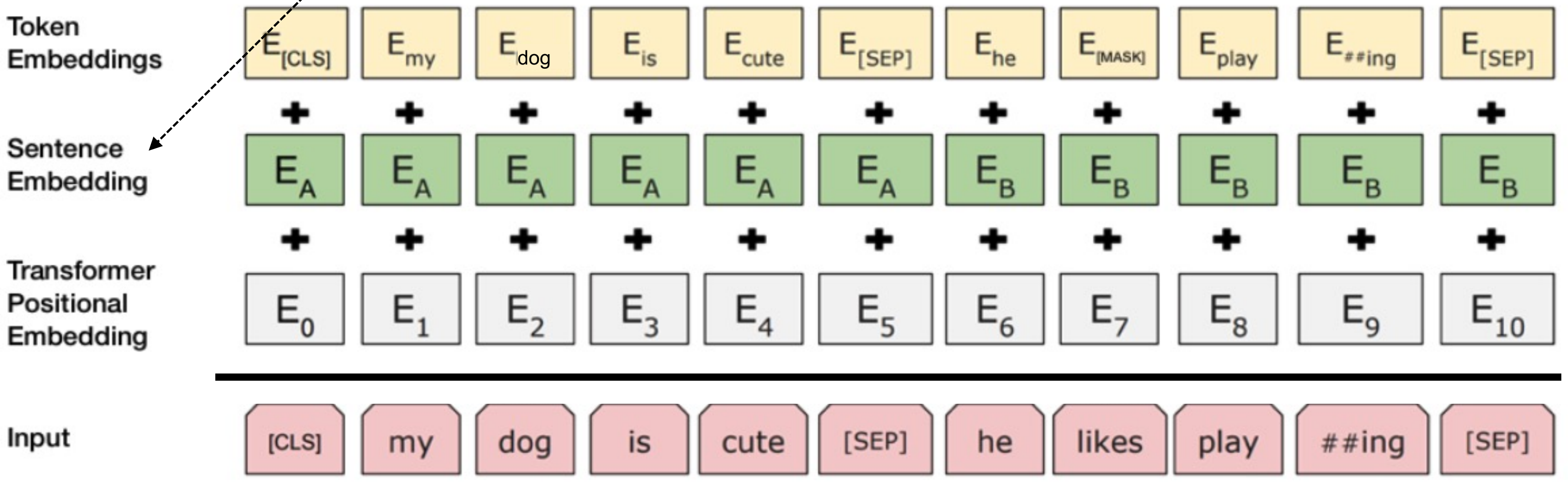
- Many NLP tasks need to calculate the relation between two sequences
  - E.g., question answering, information retrieval, natural language inference, paraphrasing, etc.

- During training, BERT also learns the relationships between two sequences using an additional binary classifier objective
  - The binary classifier take the output embedding of [CLS]
  - It predicts whether Sequence B is the actual sequence that proceeds Sequence A or a random sentence
  - This classifier is jointly optimized with the MLM objective

- If one sequence is given, the output of [CLS] is sequence embedding
- If two sequences are given, the output of [CLS] is the feature vector of the relation between the sequences

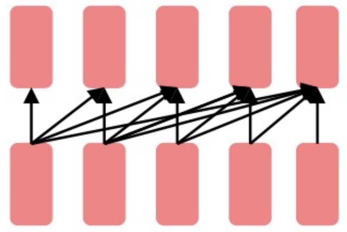
Input to BERT – two sequences

Sentence embeddings make a distinction between the embeddings of sequence A and sequence B

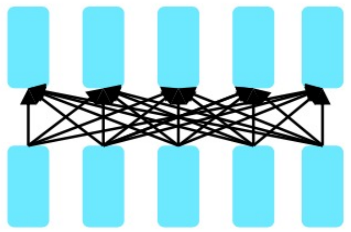

Token Embeddings
$E_{[CLS]}$ E_{my} E_{dog} E_{is} E_{cute} $E_{[SEP]}$ E_{he} $E_{[MASK]}$ E_{play} $E_{\#\#ing}$ $E_{[SEP]}$
Sentence Embedding
E_A E_A E_A E_A E_A E_A E_B E_B E_B E_B E_B
Transformer Positional Embedding
E_0 E_1 E_2 E_3 E_4 E_5 E_6 E_7 E_8 E_9 E_{10}
Input
[CLS] my dog is cute [SEP] he likes play ##ing [SEP]

Three types of large-scale pretrained LMs

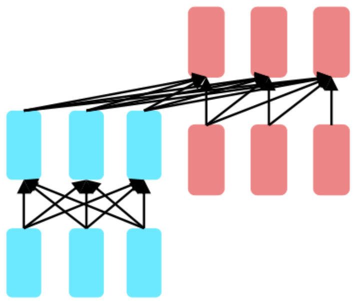
- Decoders
  - "Normal" LM objective: predict the next token conditioned on the previous tokens (unidirectional)
  - Training and inference is auto-regressive (one after each other)
  - Particularly suited for generating text



- Encoders
  - Input is encoded into contextualized embeddings
  - Some variations (like BERT) also provide sequence embedding and pair-sequence embedding
  - Training is bidirectional – model sees whole the sequence (past and future)

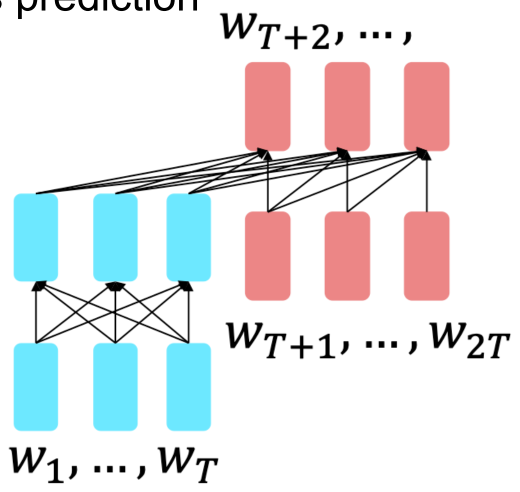


- Encoder-Decoders
  - The encoder encodes whole the input (bidirectional)
  - The decoder generates the output in auto-regressive fashion



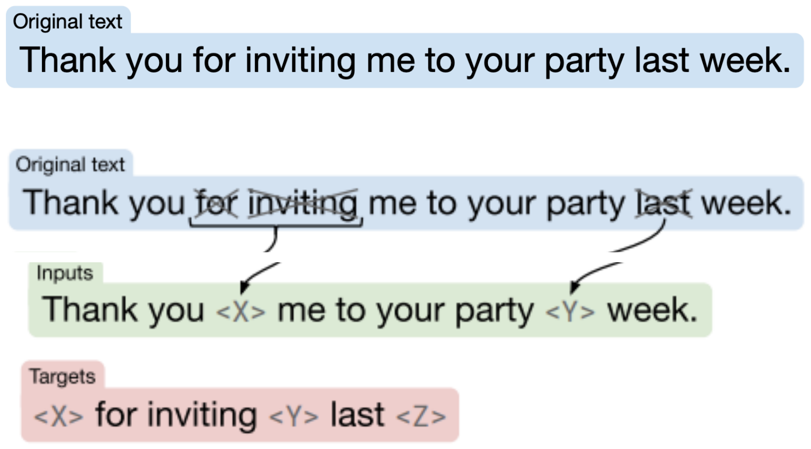
Training encoder-decoders

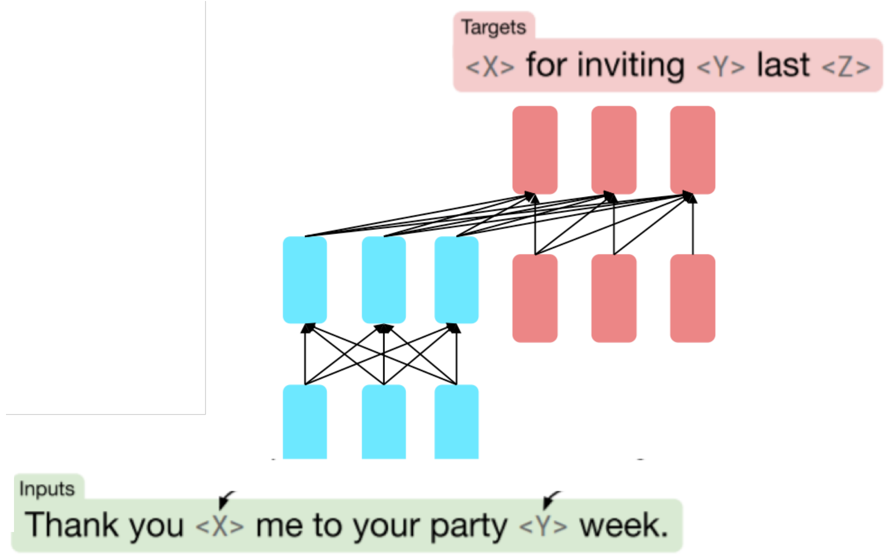
- How should we train an encoder-decoder LM?
- First approach – Using two consecutive sequences and next word prediction of the second sequence:
  - Take two sequences that follow each other in the corpus
    - Like $w_1, \dots, w_T$ and $w_{T+1}, \dots, w_{2T}$
  - Pass the first sequence to the encoder
  - Apply "Normal" LM objective at decoder:
    - Generate the words of the second subsequence at decoder's output one after each other
    - Optimize whole the model based on decoder's prediction
- This approach resembles decoders LM
  - Though here the decoder has also access to the information coming from the encoder (a larger context)



T5: Text-to-Text Transfer Transformer

- Second approach – span corruption
  - For a given sequence, randomly select a few spans with different lengths
  - Replace the selected spans with unique placeholders.
  - Pass the edited sequence to the encoder
  - Generate the removed spans in the decoder
  - Optimize whole the model based on decoder's prediction


Original text
Thank you for inviting me to your party last week.
Original text
Thank you ~~for inviting~~ me to your party ~~last~~ week.
Inputs
Thank you <X> me to your party <Y> week.
Targets
<X> for inviting <Y> last <Z>

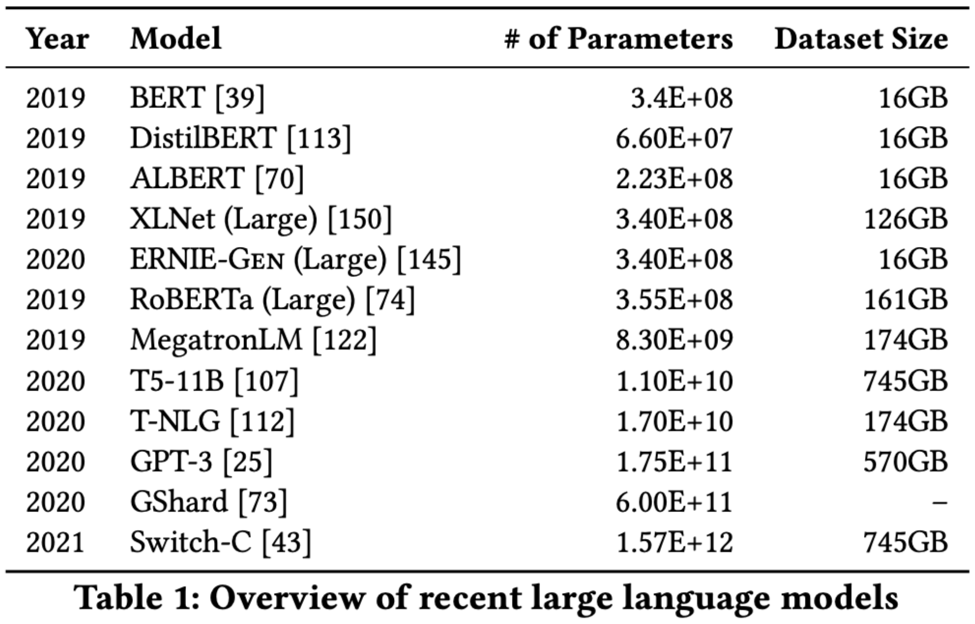

Targets
<X> for inviting <Y> last <Z>
Inputs
Thank you <X> me to your party <Y> week.

Raffel, Colin, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer." *Journal of Machine Learning Research* 21 (2020)

Agenda

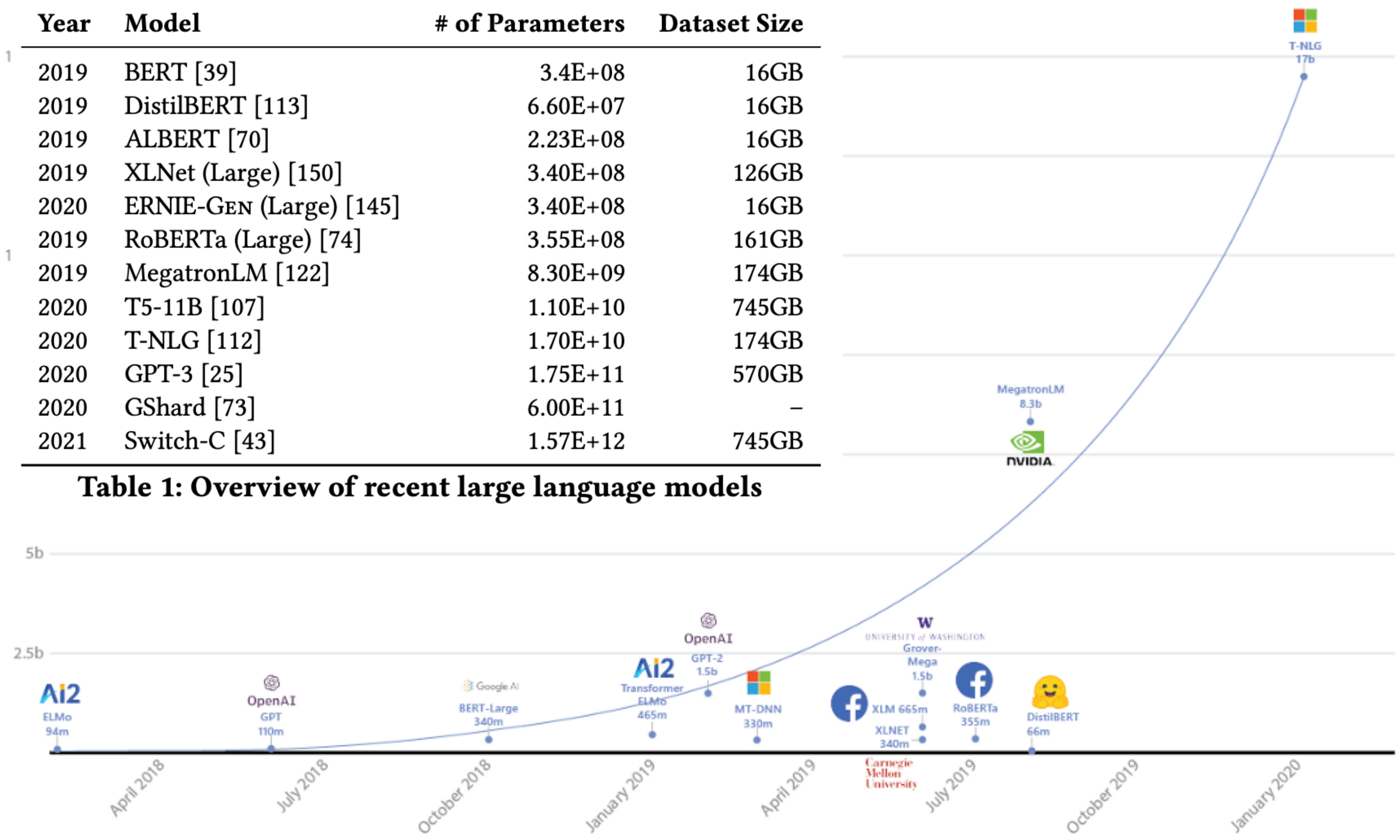
- LLM categories
- **A tour around!**
- Compression and overparameterization

Trend!

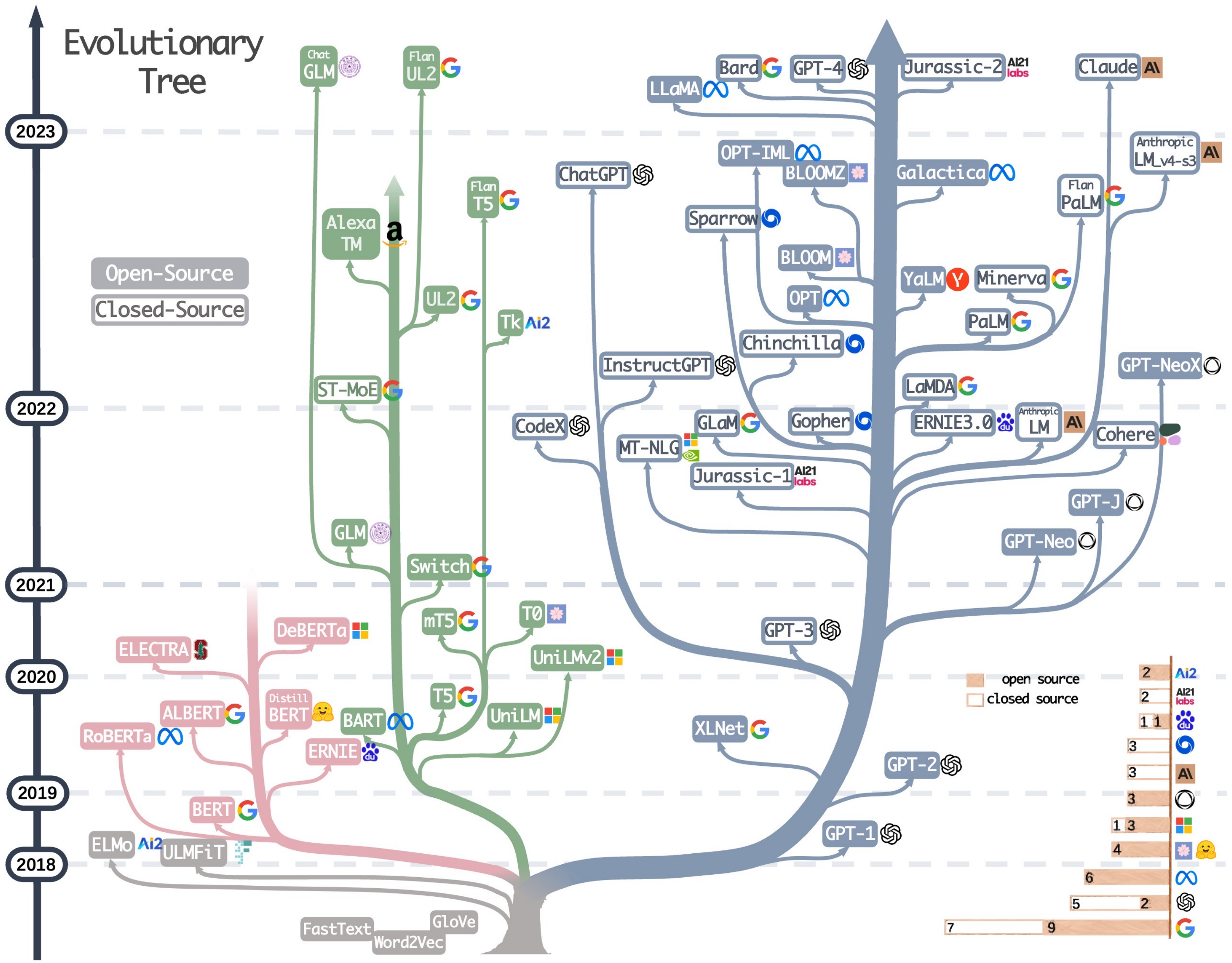


Year	Model	# of Parameters	Dataset Size
2019	BERT [39]	3.4E+08	16GB
2019	DistilBERT [113]	6.60E+07	16GB
2019	ALBERT [70]	2.23E+08	16GB
2019	XLNet (Large) [150]	3.40E+08	126GB
2020	ERNIE-Gen (Large) [145]	3.40E+08	16GB
2019	RoBERTa (Large) [74]	3.55E+08	161GB
2019	MegatronLM [122]	8.30E+09	174GB
2020	T5-11B [107]	1.10E+10	745GB
2020	T-NLG [112]	1.70E+10	174GB
2020	GPT-3 [25]	1.75E+11	570GB
2020	GShard [73]	6.00E+11	–
2021	Switch-C [43]	1.57E+12	745GB

Table 1: Overview of recent large language models

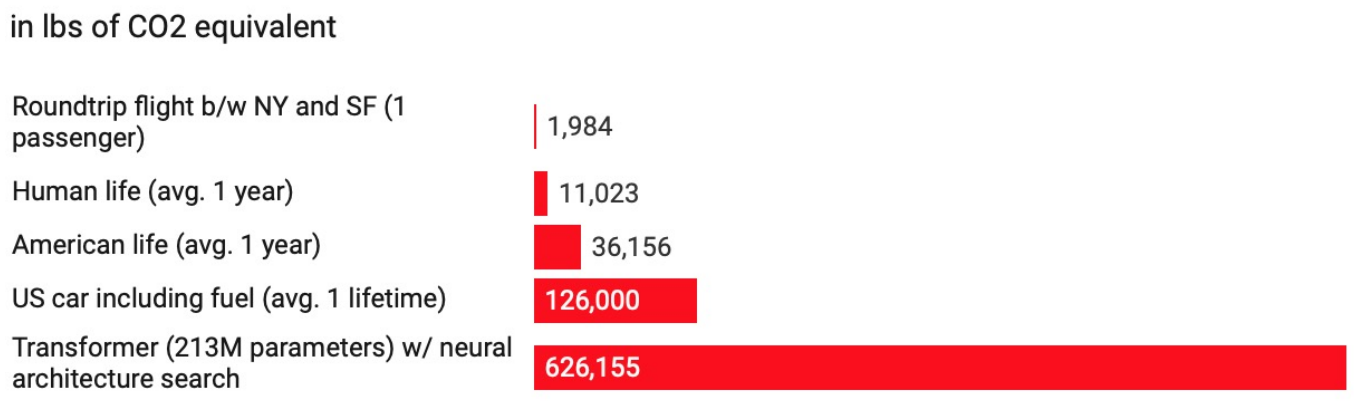


Source: https://www.groundai.com/project/distilbert-a-distilled-version-of-bert-smaller-faster-cheaper-and-lighter/1
Bender, E. M., Gebru, T., McMillan-Major, A., & Shmitchell, S. (2021, March). On the Dangers of Stochastic Parrots: Can Language Models Be Too Big?🦜. In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*



https://github.com/Mooler0410/LLMsPracticalGuide

Carbon footprint of NLP



Model	Hardware	Power (W)	Hours	kWh·PUE	CO_2e	Cloud compute cost
$\text{Transformer}_{base}$	P100x8	1415.78	12	27	26	\$41–\$140
Transformer_{big}	P100x8	1515.43	84	201	192	\$289–\$981
ELMo	P100x3	517.66	336	275	262	\$433–\$1472
BERT_{base}	V100x64	12,041.51	79	1507	1438	\$3751–\$12,571
BERT_{base}	TPUv2x16	—	96	—	—	\$2074–\$6912
NAS	P100x8	1515.43	274,120	656,347	626,155	\$942,973–\$3,201,722
NAS	TPUv2x1	—	32,623	—	—	\$44,055–\$146,848
GPT-2	TPUv3x32	—	168	—	—	\$12,902–\$43,008

Strubell, E., Ganesh, A., & McCallum, A.. Energy and Policy Considerations for Deep Learning in NLP. In *Proceedings of ACL* (2019).
Source: https://www.technologyreview.com/2019/06/06/239031/training-a-single-ai-model-can-emit-as-much-carbon-as-five-cars-in-their-lifetimes/

GPT-4

Taken from: https://drive.google.com/file/d/1DbyuDKEv0ajkYFlwNgqhRt2RdrK6joz9/view

- Multimodal - input: text and images, output: text
- Long inputs - accepts up to 32k tokens, allowing it to process long documents.
- Evaluated on human benchmarks - “exhibits human-level performance on various professional and academic

Limitations

- Closed - OpenAI didn’t disclose any details about the model or the data
- Training data contamination
- Dangerously over-hyped - Microsoft says it exhibits “sparks of AGI”
- Same old, same old - it still hallucinates, plagiarizes, is still biased, and still lacks knowledge of events that occurred after September 2021

Bard

Taken from: https://drive.google.com/file/d/1DbyuDKEv0ajkYFlwNgqhRt2RdrK6joz9/view

- Google search enhanced with the conversational AI LaMDA.
- Summarizes results from multiple documents to answer complex searches
- Can answer questions about recent events because it retrieves information from the web.
- Uses a lightweight version of LaMDA that can run faster.

Limitations

- It makes up stuff with even more confidence. The demo already included a factual error, which cost Google $100 billions in shares.

Source: https://drive.google.com/file/d/1DbyuDKEv0ajkYFlwNgqhRt2RdrK6joz9/view

ChatGPT

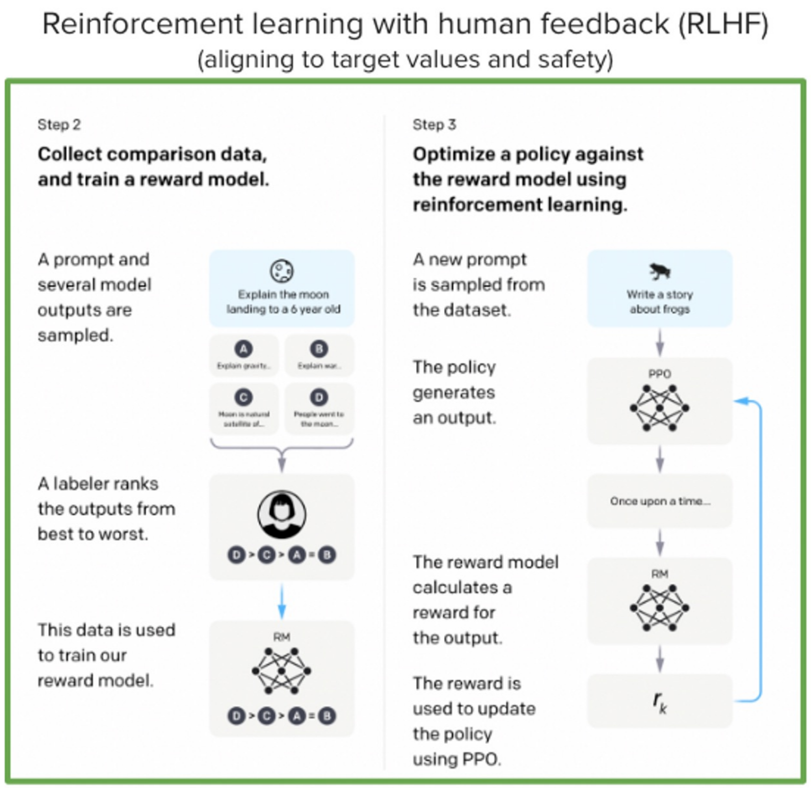
Taken from: https://drive.google.com/file/d/1DbyuDKEv0ajkYFlwNgqhRt2RdrK6joz9/view

- General-purpose chatbot that can answer questions about various topics.
- Answer factoid questions, generates creative text like poems, code, and more.
- Answers sophisticated questions with high accuracy.
- Programmed to decline answering some questions, including requests to generate offensive content.

Limitations

- Sometimes writes plausible-sounding and confident-looking incorrect answers.
- Sensitive to the phrasing of the prompt and may give inconsistent answers.
- It’s bad at numbers.
- Doesn’t really “understand” language the way humans do (and certainly not sentient!)
- Can’t answer questions about recent events because it was trained on data up to 2021.
- Better than previous models in reasoning, but still achieves about 65% accuracy across benchmarks.
- It is especially bad at inductive, spatial, mathematical, and multi-hop reasoning.
- Despite best efforts from OpenAI, offensive language filters can be bypassed with simple tricks.

Reinforcement learning with human feedback (RLHF)


Reinforcement learning with human feedback (RLHF)
(aligning to target values and safety)

Step 2

Collect comparison data, and train a reward model.

A prompt and several model outputs are sampled.

A labeler ranks the outputs from best to worst.

This data is used to train our reward model.

Step 3

Optimize a policy against the reward model using reinforcement learning.

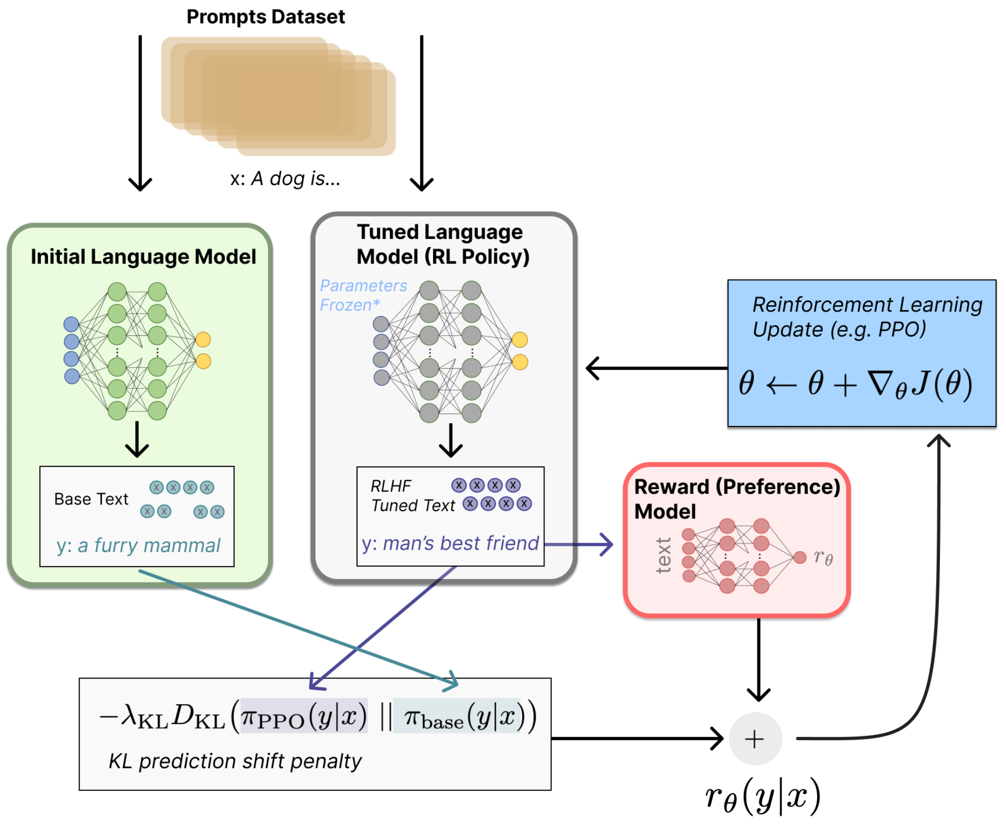
A new prompt is sampled from the dataset.

The policy generates an output.

The reward model calculates a reward for the output.

The reward is used to update the policy using PPO.

Explain the moon landing to a 6 year old
A
B
C
D
D > C > A = B
RM
D > C > A = B
Write a story about frogs
PPO
Once upon a time...
RM
r_k


Prompts Dataset
x: A dog is...
Initial Language Model
Base Text
y: a furry mammal
Tuned Language Model (RL Policy)
Parameters Frozen*
RLHF Tuned Text
y: man's best friend
Reinforcement Learning Update (e.g. PPO)
$\theta \leftarrow \theta + \nabla_\theta J(\theta)$
Reward (Preference) Model
text
r_θ
$-\lambda_{\text{KL}} D_{\text{KL}}(\pi_{\text{PPO}}(y|x) \,||\, \pi_{\text{base}}(y|x))$
KL prediction shift penalty
+
$r_\theta(y|x)$

Source: Ouyang, Long, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang et al. "Training language models to follow instructions with human feedback." *Advances in Neural Information Processing Systems* 35 (2022): 27730-27744.
A dataset: https://huggingface.co/datasets/Anthropic/hh-rlhf/viewer/Anthropic--hh-rlhf/train?row=1
https://huggingface.co/blog/rlhf

Sparks of Artificial General Intelligence: Early experiments with GPT-4

Sébastien Bubeck Varun Chandrasekaran Ronen Eldan Johannes Gehrke
Eric Horvitz Ece Kamar Peter Lee Yin Tat Lee Yuanzhi Li Scott Lundberg
Harsha Nori Hamid Palangi Marco Tulio Ribeiro Yi Zhang

Microsoft Research

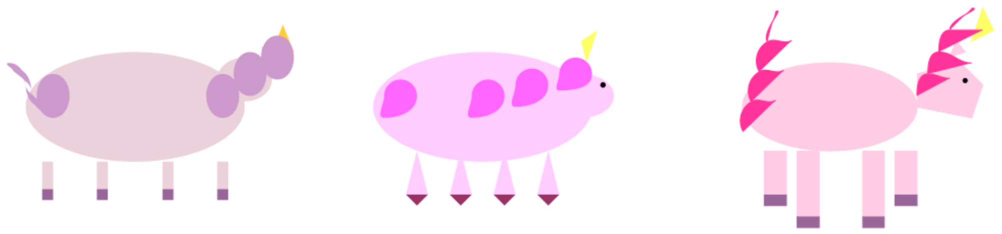


Figure 1.3: We queried GPT-4 three times, at roughly equal time intervals over the span of a month while the system was being refined, with the prompt "Draw a unicorn in TikZ". We can see a clear evolution in the sophistication of GPT-4's drawings.

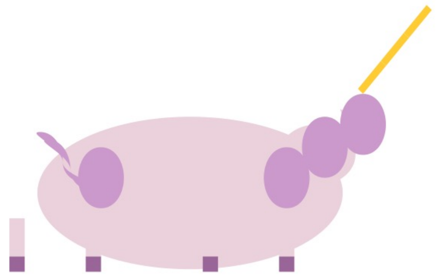


Figure 1.4: We gave to GPT-4 a transformed version of the TikZ code it produced for Figure 1.1, with the part drawing the horn removed. We asked for code to add back the horn, and display the result. This demonstrates that GPT-4 can "see" despite being a pure language model (we emphasize again that the version we test with is **not** multimodal).

GPT-4: Understanding beliefs

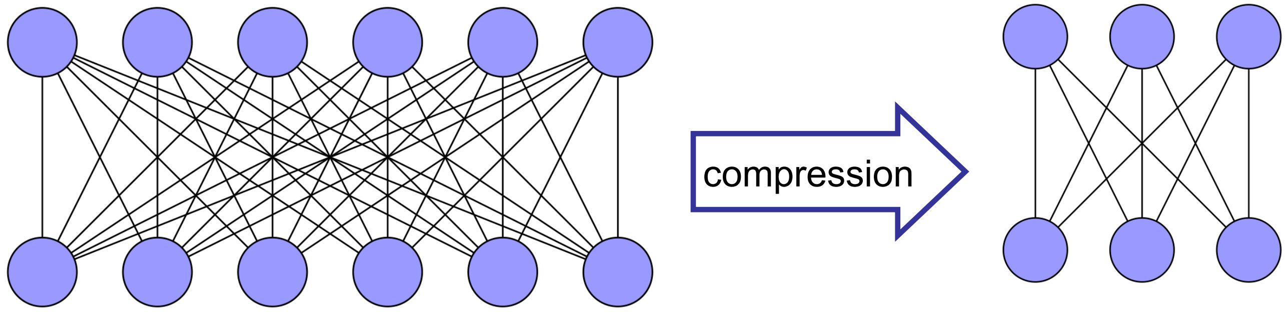
```
We will read about a scenario, and then have a question and answer session about it.
--
Scenario:
Alice and Bob have a shared Dropbox folder.
Alice puts a file called 'photo.png' inside /shared_folder/photos.
Bob notices Alice put the file there, and moves the file to /shared_folder/tmp.
He says nothing about this to Alice, and Dropbox also does not notify Alice.
--<|endofprompt|>
Q: After the call, Alice wants to open 'photo.png'. In which folder will she look for it?
Alice will most likely look for 'photo.png' in /shared_folder/photos, the folder where she orig
inally put it. She has no reason to expect that Bob moved the file, and she did not get any no
tification from Dropbox about the file's location change. Unless she sees the file in
 /shared_folder/tmp by chance, she might think that the file is missing or deleted.
```

Agenda

- LLM categories
- A tour around!
- **Compression and overparameterization**

Model compression

- Model compression methods reduce the size of a model
  - applied as a post-processing, but also during training
- A compressed model
  - Efficient in practice:
    - faster inference time
    - better suited to low-resource settings (i.e. on mobile phones)
  - Less energy consumption

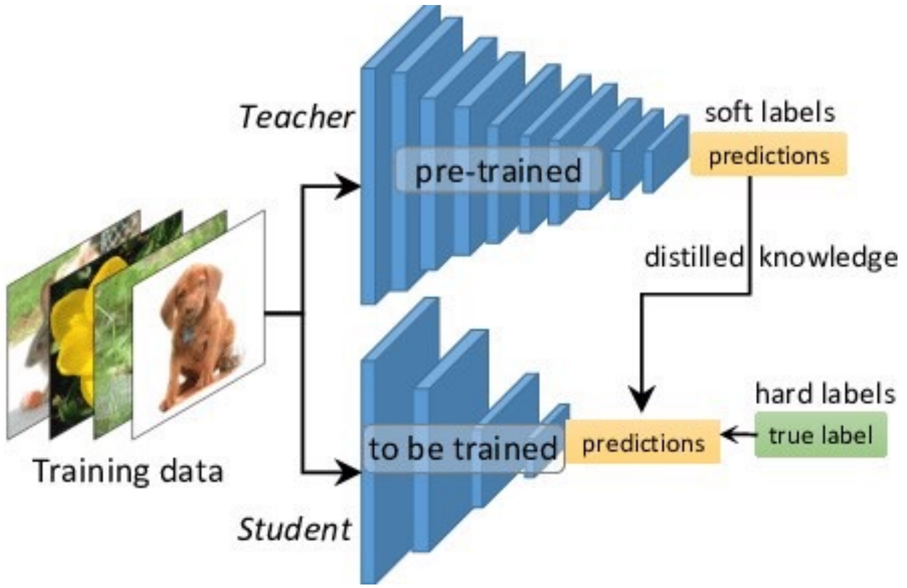


Model compression methods

- Knowledge distillation
  - A smaller model (student) is trained to reproduce the *behavior* of a larger model (teacher)
  - Student *mimics* teachers output or internal representations

Example: DistilBERT

- Distillation loss is defined according to the prediction probabilities of a pre-trained BERT
- Reduce the size to 40% while retaining 97% of performance on GLUE tasks



Sanh, V., Debut, L., Chaumond, J., & Wolf, T. (2019). DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.
Figure source: https://towardsdatascience.com/knowledge-distillation-simplified-dd4973dbc764

Model compression methods

- Quantization
  - Quantization methods decrease the numerical precision of model parameters
    - For instance, by turning the 32-bit float parameters of a pre-trained model to 8-bit integers

Model compression methods

- Pruning
  - To reduce the extent of a network by removing the superfluous and unnecessary *neurons*, *nodes*, *heads*, *etc.*
  - Pruning can be done after training or during training

A common (post-processing) procedure

1. Train the model
2. Remove the unnecessary units, selected based on their
   - magnitudes, gradients, activations, etc.
3. Fine-tune the pruned network
4. Repeat the last two steps iteratively

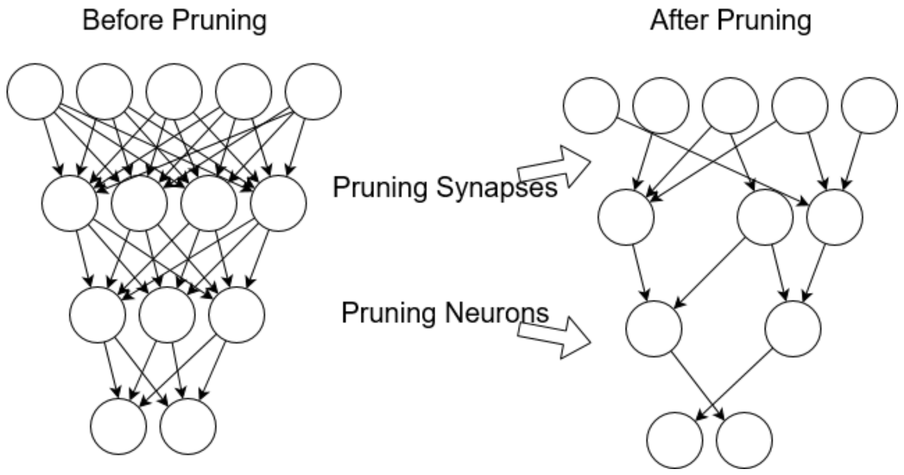


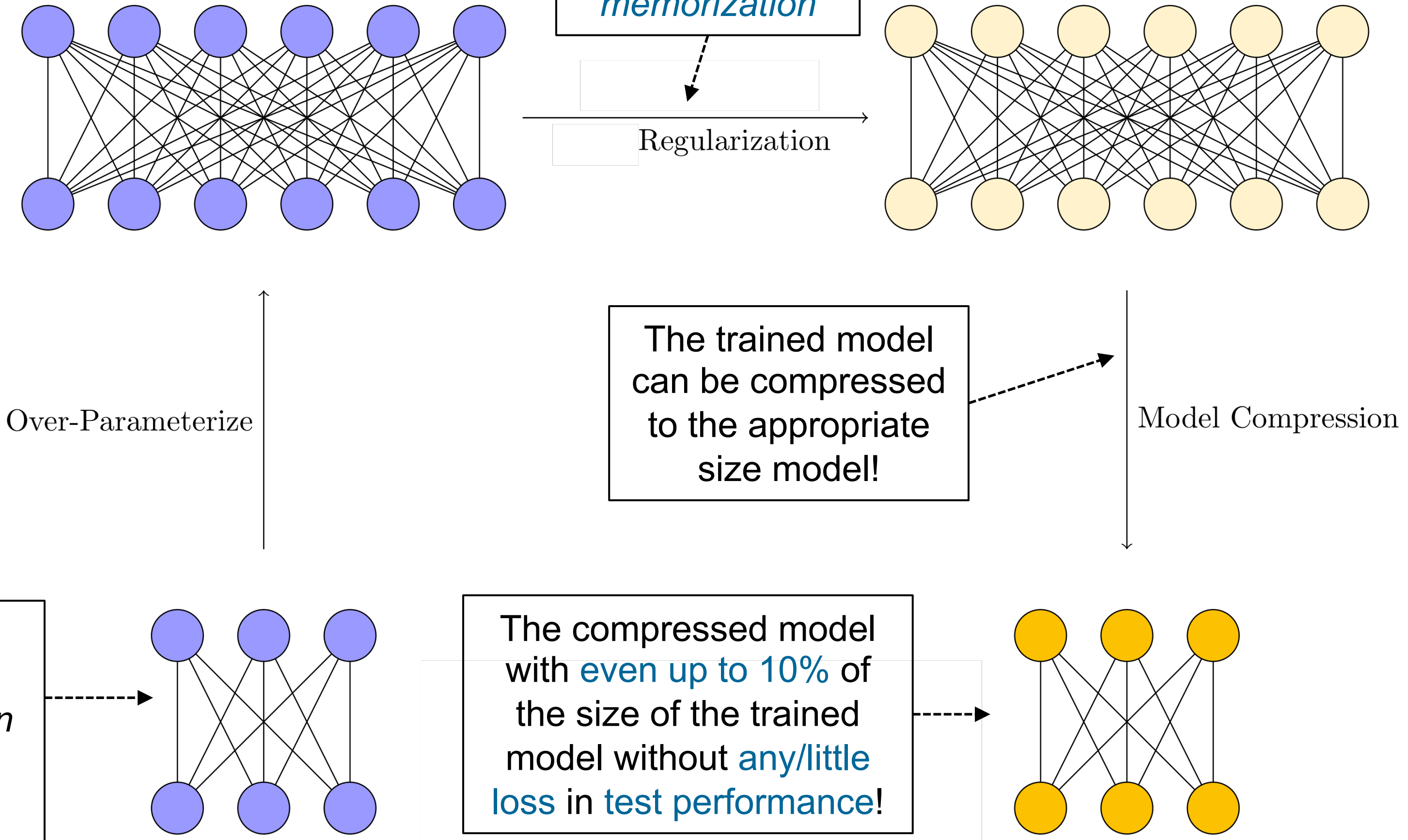
Figure source: https://software.intel.com/content/www/us/en/develop/articles/compression-and-acceleration-of-high-dimensional-neural-networks.html

Over-parameterization – a paradox!

Create a model with much higher capacity

Apply extreme regularization to avoid *data memorization*

Trained model with "simple" optimization (i.e. SGD)



The trained model can be compressed to the appropriate size model!

A model with appropriate capacity that *in principle* can solve the problem

The compressed model with even up to 10% of the size of the trained model without any/little loss in test performance!

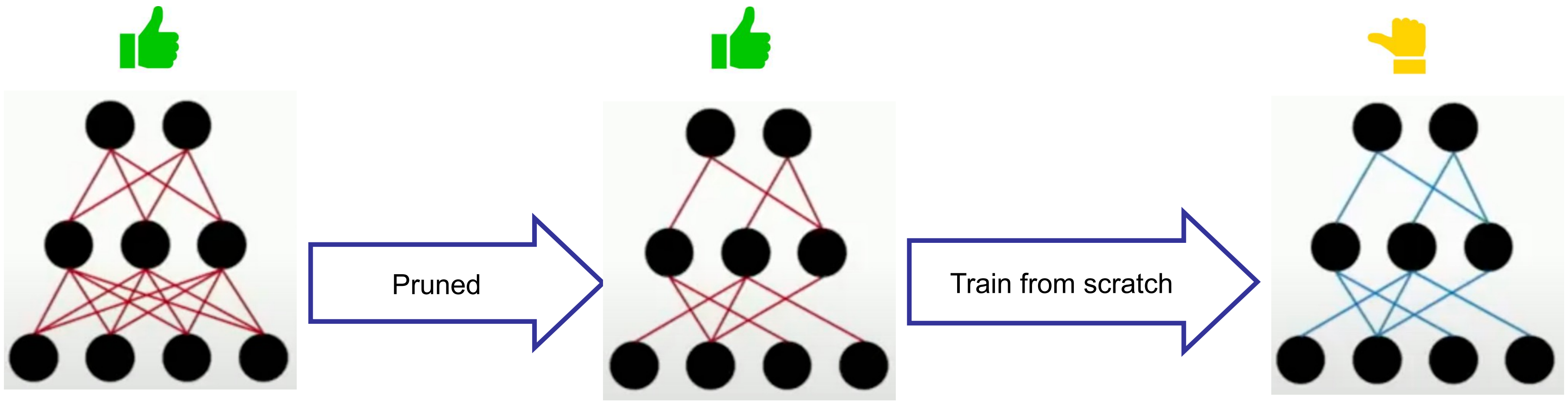
Over-parameterization is in paradox to classical ML doctrine of *"Thou shalt start simple, and then go complex."*

Figure adopted from here (a nice read): http://mitchgordon.me/machine/learning/2020/01/13/do-we-really-need-model-compression.html

Can we then start from small compressed models?!

Consider compression with pruning…

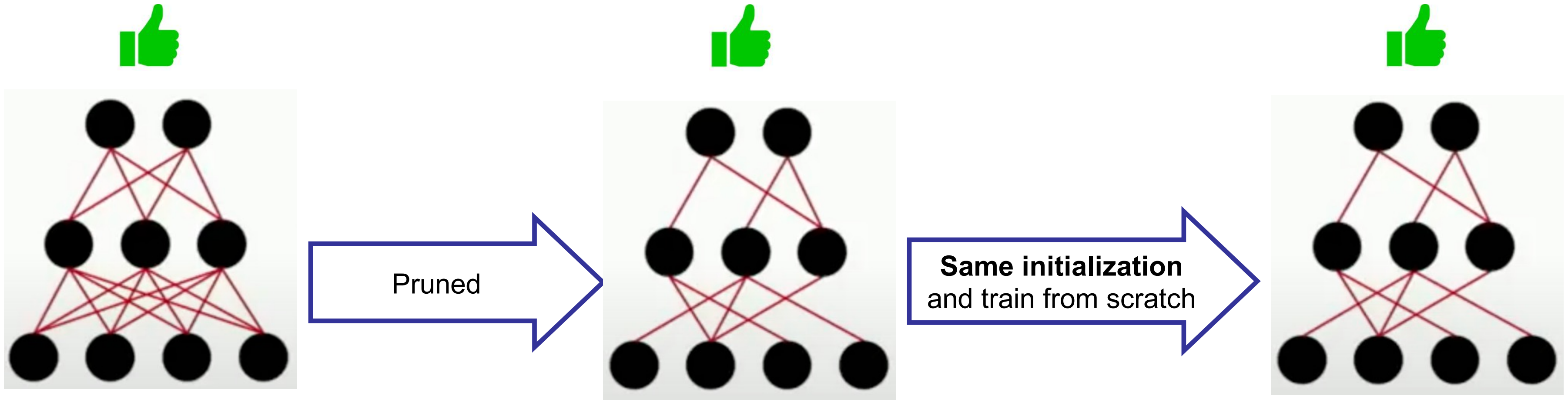
- What if we take the pruned model, and re-train it from scratch (with new initializations)?
  - It does not reach the same performance as the compressed model



Can we then start from small compressed models?!

Consider compression with pruning…

- What if we take the pruned model, and re-train it from scratch (with new initializations)?
    - It does not reach the same performance as the compressed model
- However, if we take the structure of the pruned model, and use the same initialization as the original model …
    - We achieve the same or even better results!



Lottery ticket hypothesis

- The Lottery Ticket Hypothesis (rephrased):

*dense, trainable networks contain sparse trainable subnetworks (i.e., winning tickets) that are equally capable**

** When trained in isolation, they reach test prediction comparable to the original network in a similar number of iterations.*

- Though, we don't know (yet) beforehand how to find these subnetworks
  - What are their structures?
  - What are their initializations?
- But if we do … we can achieve the same results by training much smaller networks