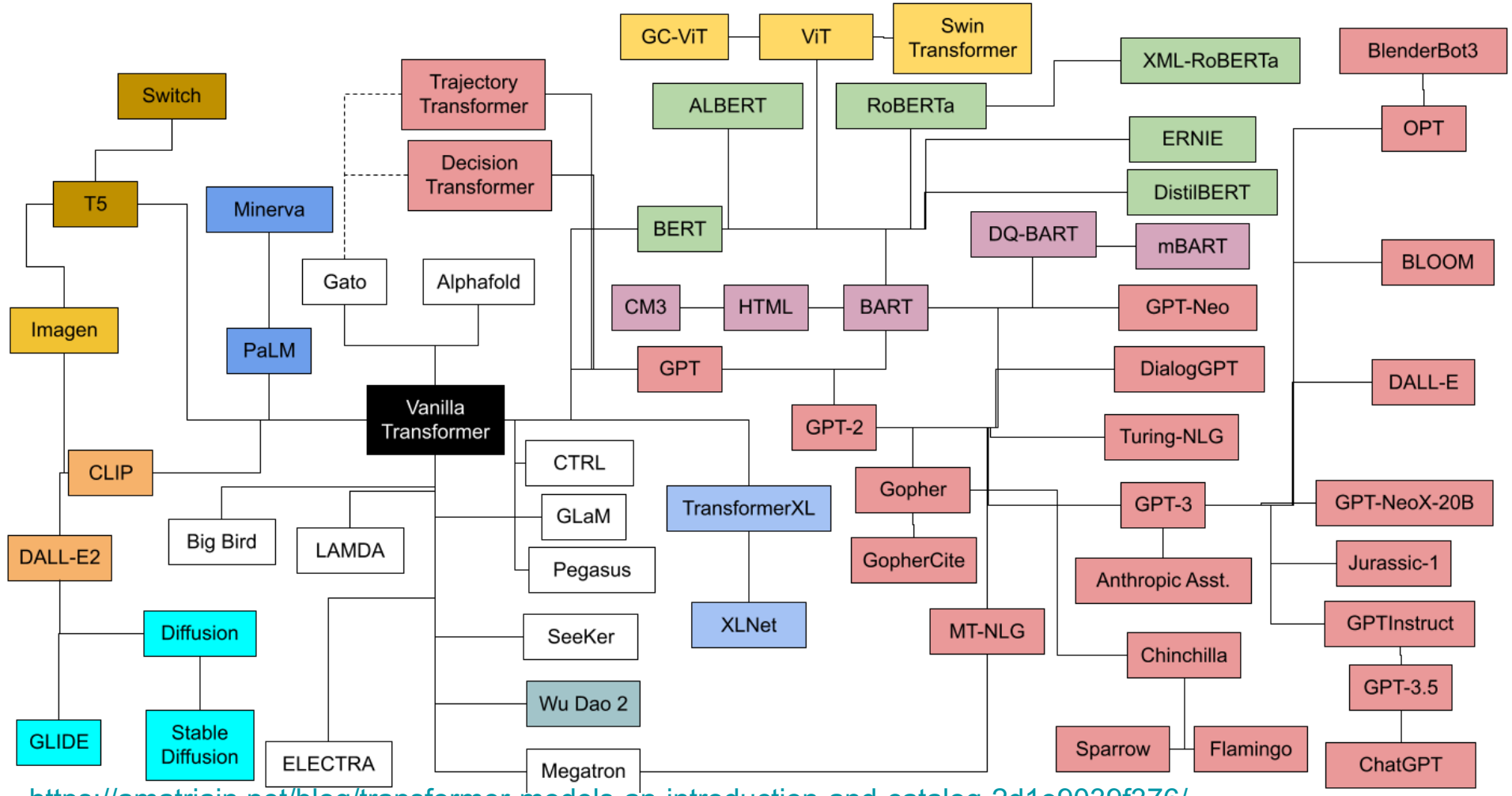
The BERT’s family

Longer, bigger, smaller, smarter



https://amatriain.net/blog/transformer-models-an-introduction-and-catalog-2d1e9039f376/

Pre-trained transformer types (by training method)

Encoder only (autoencoder)

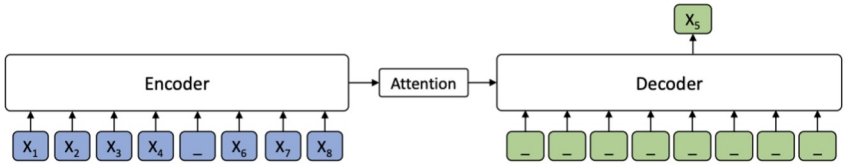
- BERT, ALBERT, RoBERTa
- Seq classification, token classification
- Masked words and predict

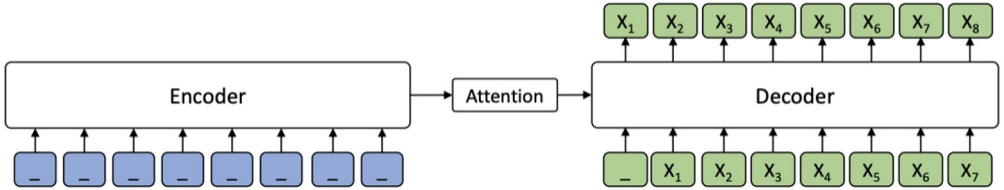
Encoder-decoder (seq2seq)

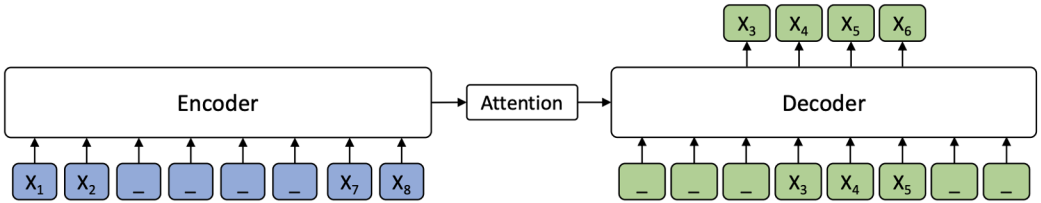
- MASS, BART, T5
- Machine translation, text summary
- Mask phrases

Decoder only (autoregressive)

- GPT, CTRL
- Text generation
- Predict next word



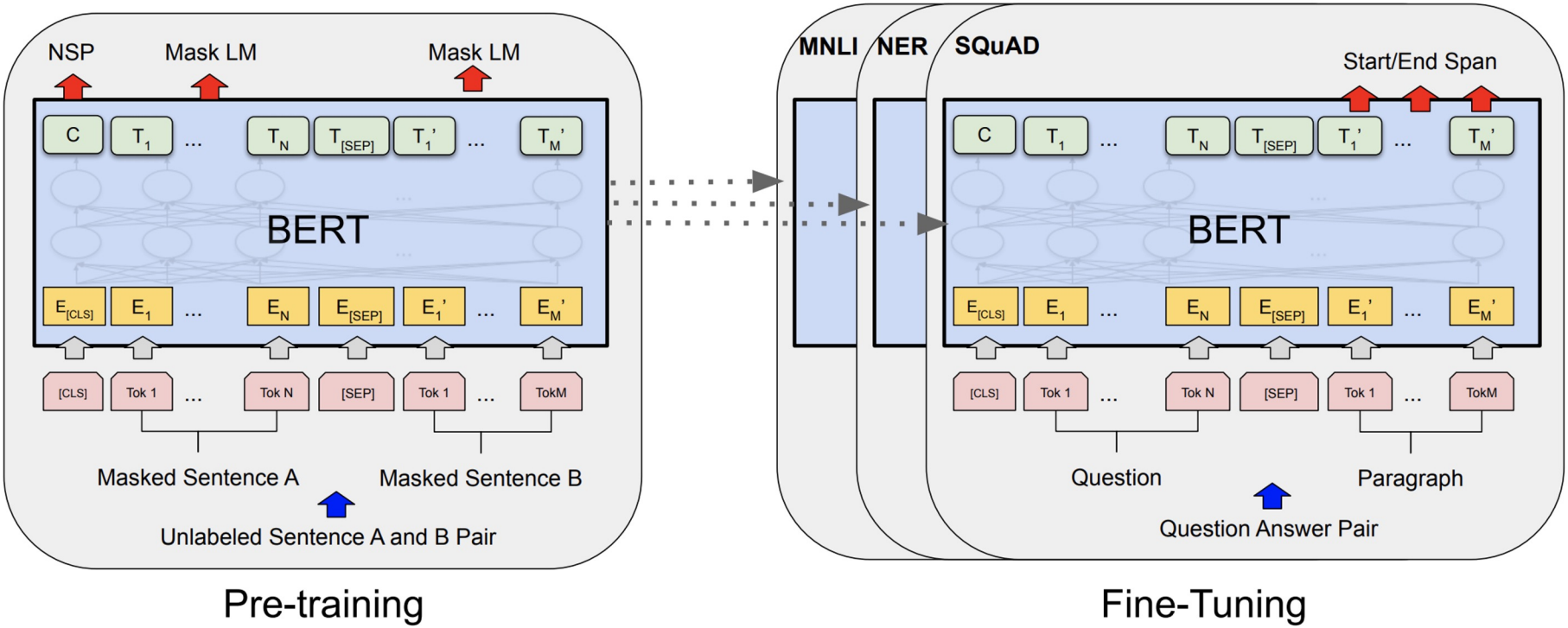




https://huggingface.co/transformers/summary.html https://arxiv.org/pdf/1905.02450.pdf

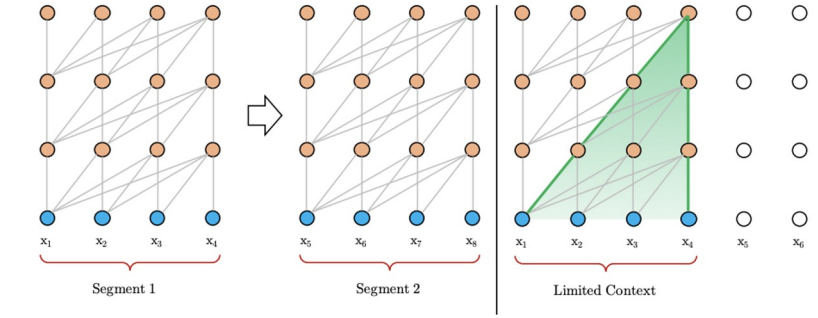
BERT

Full transformer
Masked LM to learn bi-directional LM
Next sentence prediction to learn discourse

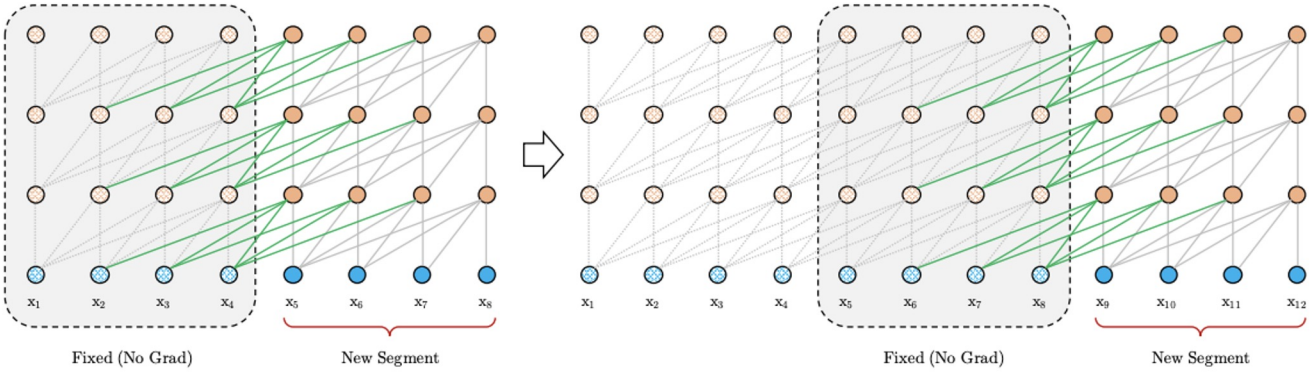


https://arxiv.org/abs/1810.04805

XL-transformer



Chunking limits the length of context



(a) Training phase.

Let next chunk attention has access to previous chunk

Used in XL-Net and others

Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context

Roberta (Robustly optimized BERT approach)

A trick and tuning study

Dynamic masking > static

Next sentence prediction is not optimal

Larger batch + higher learning rate

Masking	SQuAD 2.0	MNLI-m	SST-2
reference	76.3	84.3	92.8
Our reimplementation:			
static	78.3	84.3	92.5
dynamic	78.7	84.0	92.9

bsz	steps	lr	ppl	MNLI-m	SST-2
256	1M	1e-4	3.99	84.7	92.7
2K	125K	7e-4	**3.68**	**85.2**	**92.9**
8K	31K	1e-3	3.77	84.6	92.8

RoBERTa: A Robustly Optimized BERT Pretraining Approach
Used in WangchanBERTa
Highly recommended to go watch https://www.youtube.com/watch?v=kXPMLX0vfYU
for more info on WangchanBERTa

Megatron-LM

Distributed training

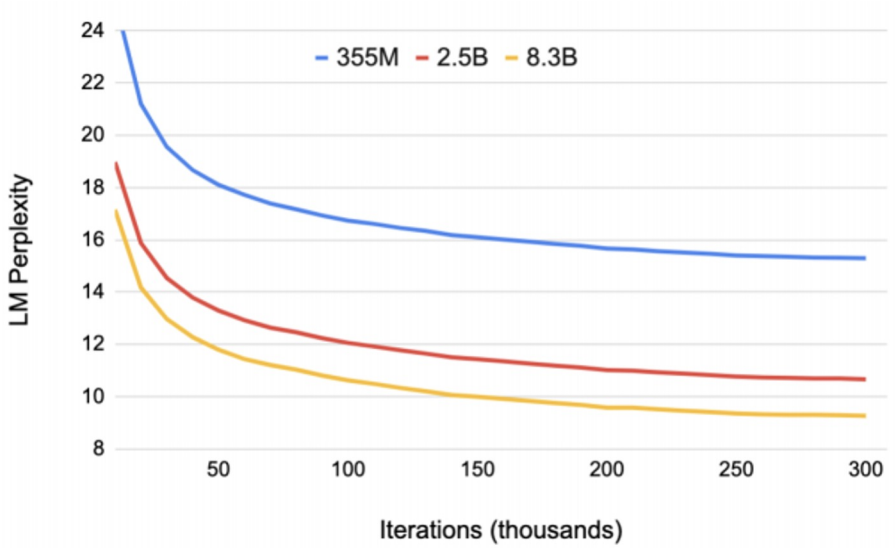
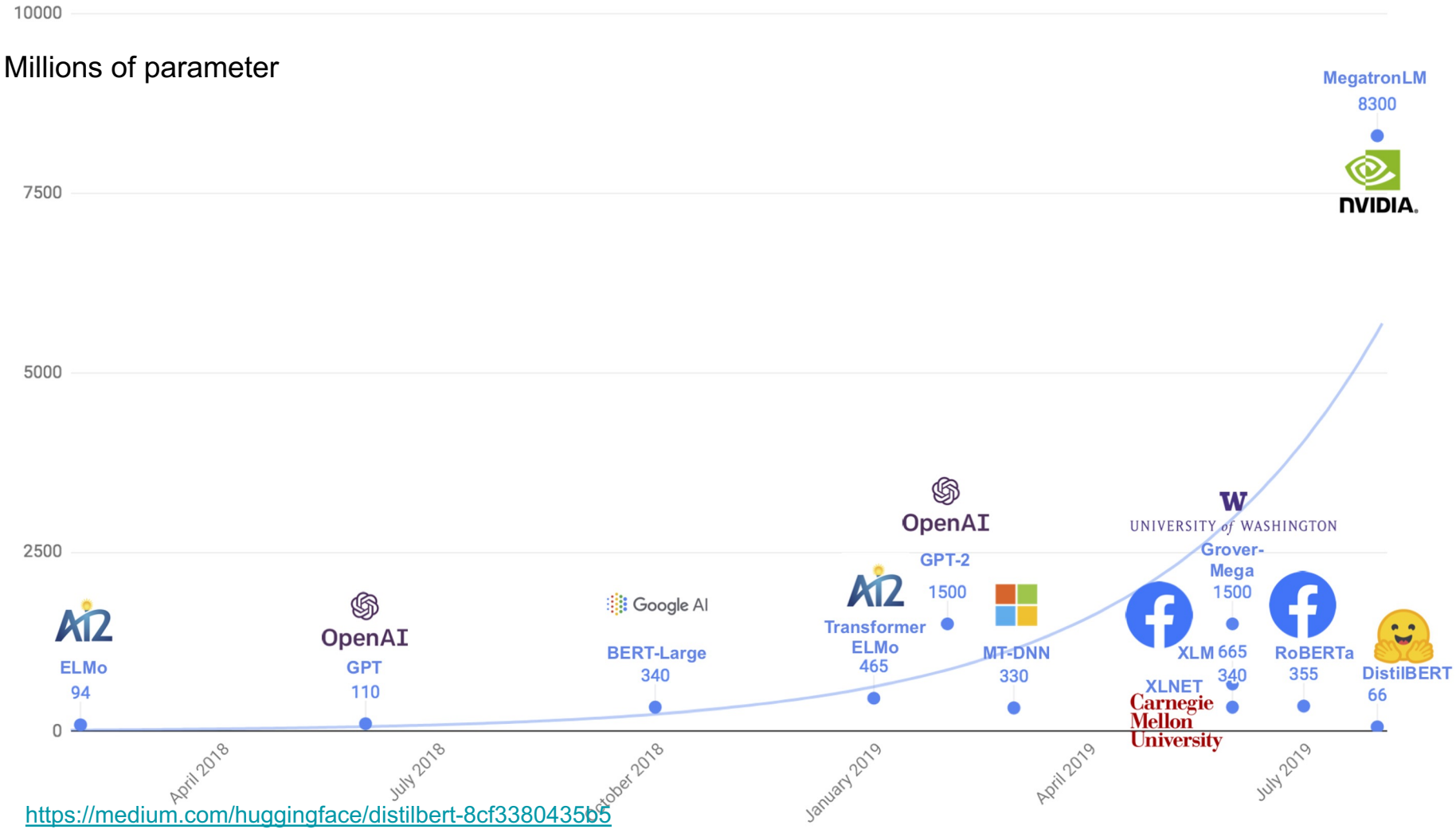


Figure 7. Validation set perplexity. All language models are trained for 300k iterations. Larger language models converge noticeably faster and converge to lower validation perplexities than their smaller counterparts.

https://arxiv.org/abs/1909.08053



https://medium.com/huggingface/distilbert-8cf3380435b5

Scaling law in language models

More data more params more compute leads to better models

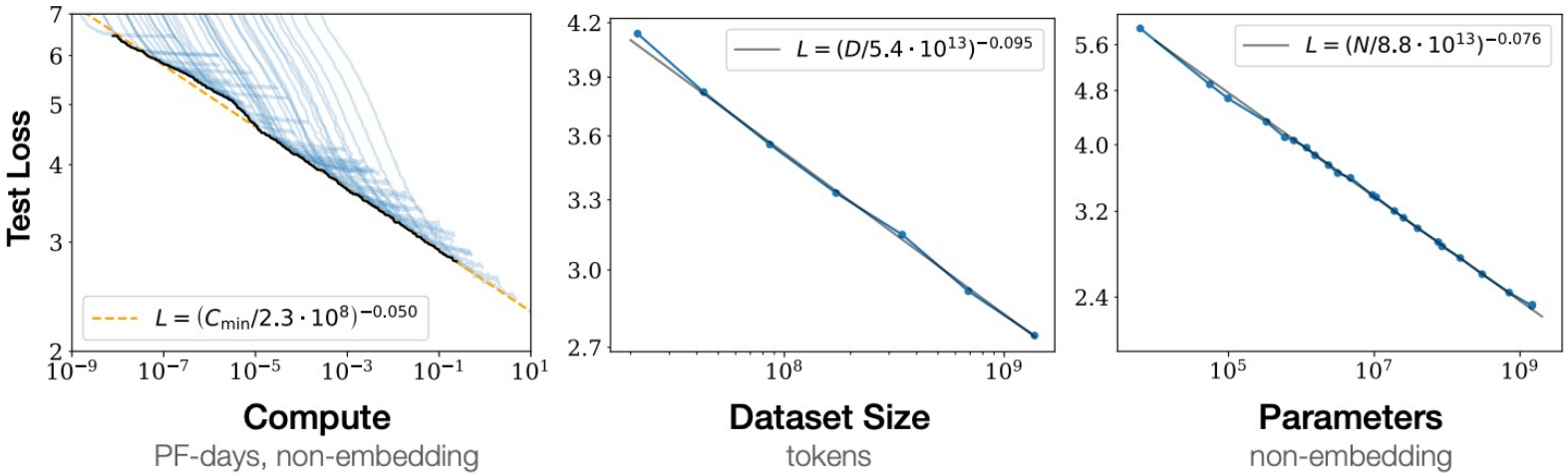


Figure 1 Language modeling performance improves smoothly as we increase the model size, datasetset size, and amount of compute[2] used for training. For optimal performance all three factors must be scaled up in tandem. Empirical performance has a power-law relationship with each individual factor when not bottlenecked by the other two.

Scaling Laws for Neural Language Models https://arxiv.org/pdf/2001.08361.pdf

Scaling law in language models

Training usually stops early for efficiency reasons

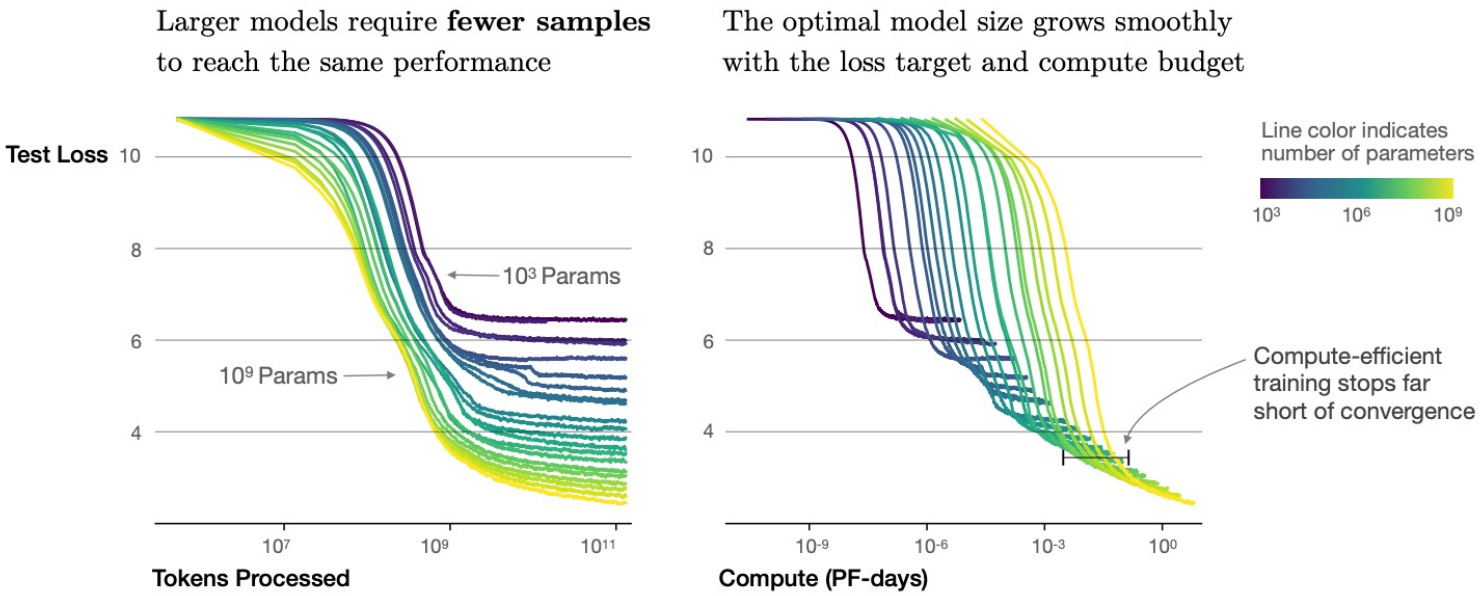


Figure 2 We show a series of language model training runs, with models ranging in size from 10^3 to 10^9 parameters (excluding embeddings).

Scaling Laws for Neural Language Models https://arxiv.org/pdf/2001.08361.pdf

Scaling law in language models

Longer training can be beneficial

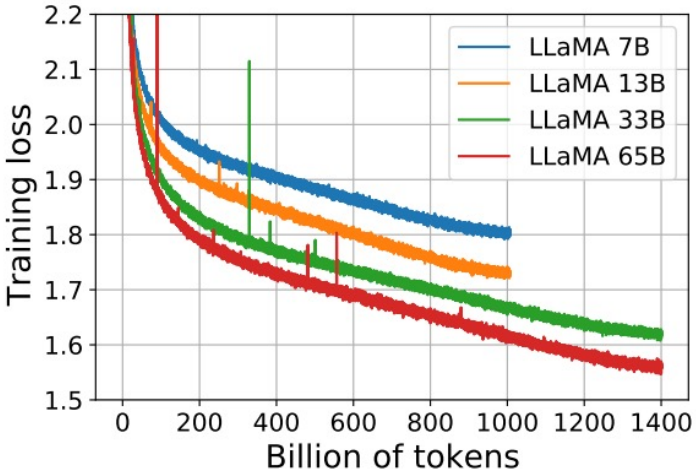
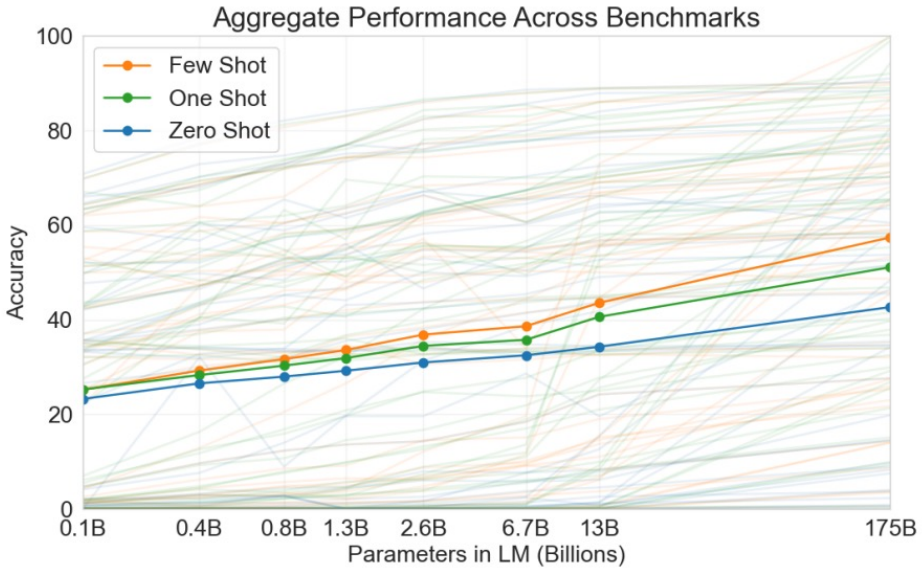


Figure 1: **Training loss over train tokens for the 7B, 13B, 33B, and 65 models.** LLaMA-33B and LLaMA-65B were trained on 1.4T tokens. The smaller models were trained on 1.0T tokens. All models are trained with a batch size of 4M tokens.

LLaMA: Open and Efficient Foundation Language Models https://arxiv.org/abs/2302.13971

Scaling law in language models



Loss (perplexity) is generally a good proxy for improvements on downstream tasks

Figure 1.3: Aggregate performance for all 42 accuracy-denominated benchmarks While zero-shot performance improves steadily with model size, few-shot performance increases more rapidly, demonstrating that larger models are more proficient at in-context learning. See Figure 3.8 for a more detailed analysis on SuperGLUE, a standard NLP benchmark suite.

Language Models are Few-Shot Learners https://arxiv.org/pdf/2005.14165.pdf

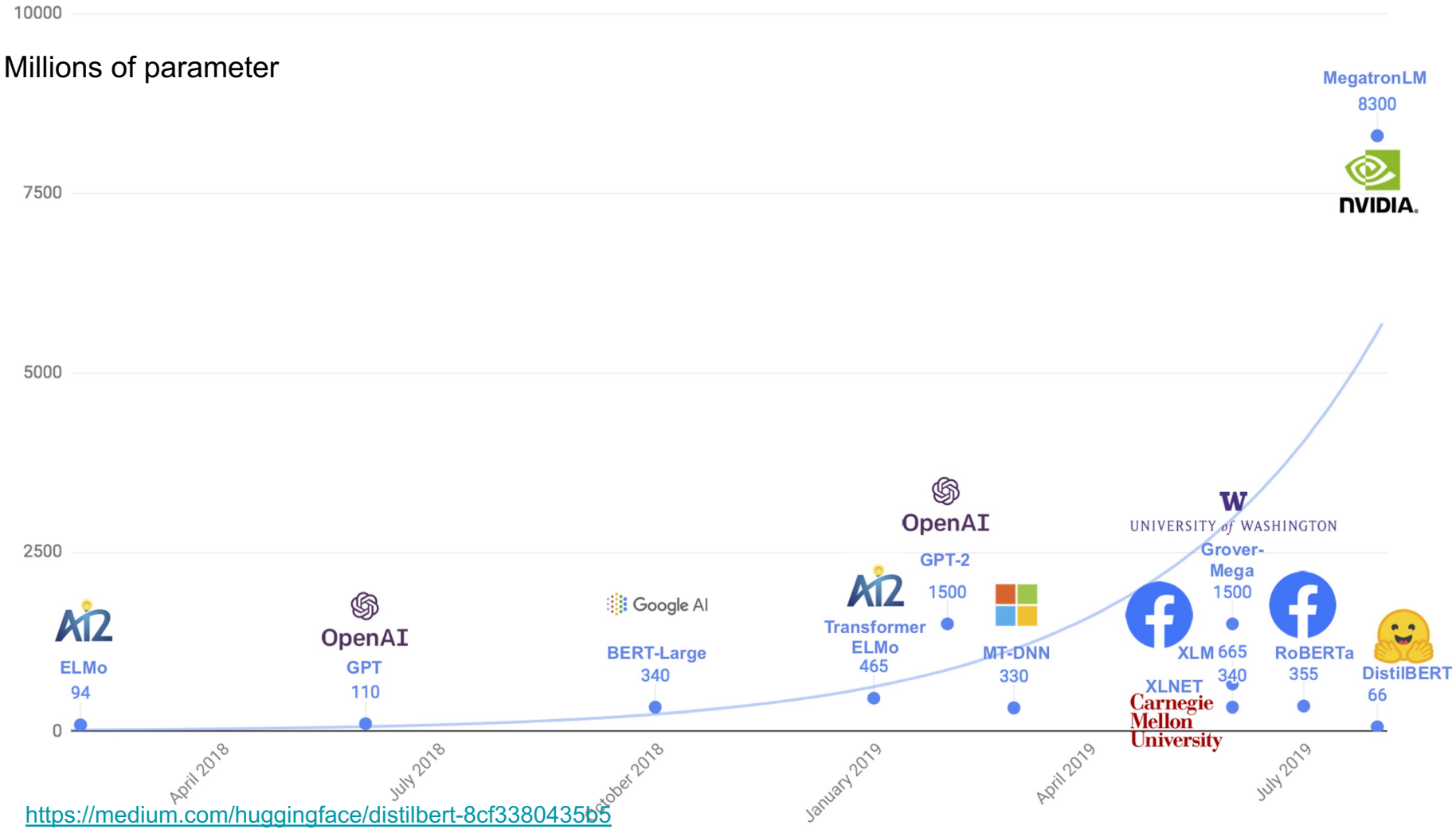
Scaling law in language models

Besides data size, data quality matters

	Dataset Size	Pile (val) (BPB)	Pile (test) (BPB)	WikiText (PPL)	LAMBADA (PPL)	LAMBADA (ACC)
The Pile	825 GiB	**0.9281**	**0.9433**	**5.59**	12.78	**50.1**
CC-100 (en)	300 GiB	1.3143	1.3293	8.27	**11.78**	49.7
Raw CC	45927 GiB†	1.1180	1.1275	11.75	19.84	43.8

Table 3: Size-controlled evaluation results. Each dataset is deduplicated against all evaluation metrics and subsampled to approximately 40GB to control for the effects of dataset size. For LAMBADA, we use the variant of the data introduced in Radford et al. (2019) and only evaluate the perplexity on the final token rather than the final word. For WikiText, we report the perplexity per GPT-2 token. † indicates that the size is an estimate.

The Pile: An 800GB Dataset of Diverse Text for Language Modeling https://arxiv.org/pdf/2101.00027.pdf



https://medium.com/huggingface/distilbert-8cf3380435b5

Distill bert

Knowledge distillation to get smaller models

Reduce the # of transformer layers by half. Use tricks in Roberta.

Use KL-divergence between teacher and student model

“Cheaper training”
eight 16GB V100 GPUs for approximately three and a half days

	Nb of parameters (millions)	Inference Time (s)
GLUE BASELINE (ELMo + BiLSTMs)	180	895
BERT base	110	668
DistilBERT	66	410

	Macro Score	CoLA	MNLI	MNLI-MM	MRPC		QNLI	QQP		RTE	SST-2	STS-B		WNLI
		mcc	acc	acc	acc	f1	acc	acc	f1	acc	acc	pearson	spearmanr	acc
GLUE BASELINE (ELMo + BiLSTMs)	**68.7**	44.1	68.6 (avg)		70.8	82.3	71.1	88.0	84.3	53.4	91.5	70.3	70.5	56.3
BERT base	**78.0**	55.8	83.7	84.1	86.3	90.5	91.1	90.9	87.7	68.6	92.1	89.0	88.6	43.7
DistilBERT	**75.2**	42.5	81.6	81.1	82.4	88.3	85.5	90.6	87.7	60.0	92.7	84.5	85.0	55.6

https://github.com/huggingface/transformers https://medium.com/huggingface/distilbert-8cf3380435b5

ALBERT

Want higher hidden units without growing the model. Factorized embedding matrix

VxE -> VxE+ExH

Share attention layer parameters across layers. More stable training as a side effect.

NSP is easy compared to LM tasks (multi-task imbalance)
Next sentence prediction (random sentence) -> Sentence order prediction (swapped sentence or not)

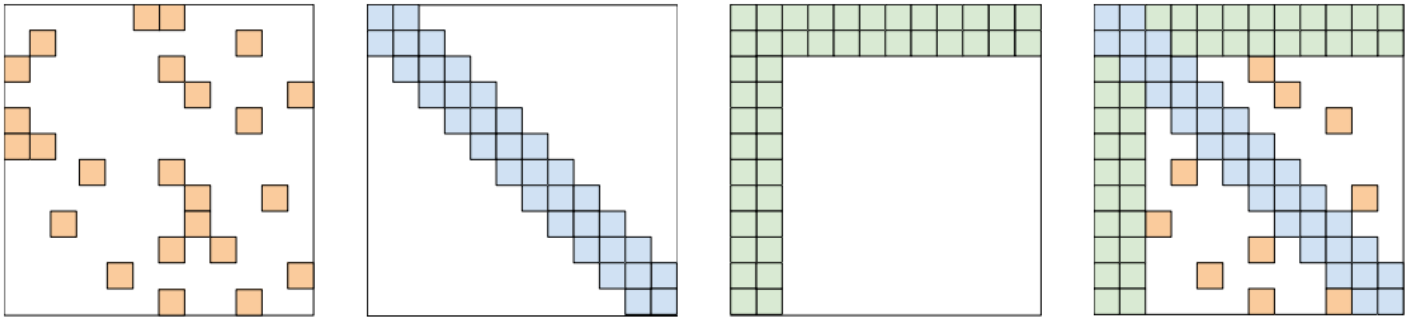
Model		Parameters	Layers	Hidden	Embedding	Parameter-sharing	Avg	Speedup
BERT	base	108M	12	768	768	False	82.1	17.7x
	large	334M	24	1024	1024	False	85.1	3.8x
	xlarge	1270M	24	2048	2048	False	76.7	1.0
ALBERT	base	12M	12	768	128	True	80.1	21.1x
	large	18M	24	1024	128	True	82.4	6.5x
	xlarge	59M	24	2048	128	True	85.5	2.4x
	xxlarge	233M	12	4096	128	True	**88.7**	1.2x

Some experiments show dropout hurt performance

https://arxiv.org/abs/1909.11942

Sparse attentions

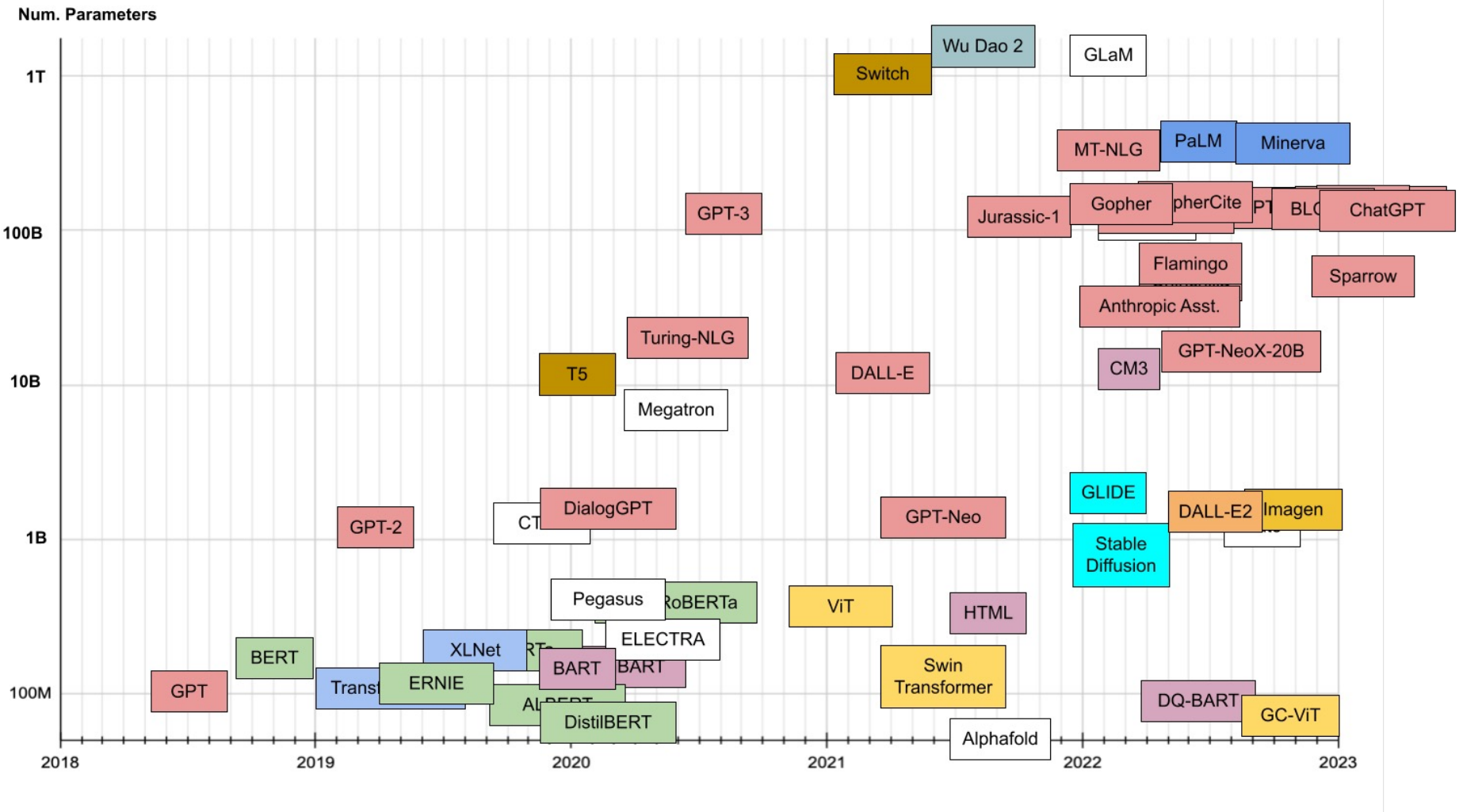
- Instead of computing the full attention matrix, people have found that many parts can be dropped



(a) Random attention (b) Window attention (c) Global Attention (d) BIGBIRD

Figure 1: Building blocks of the attention mechanism used in BIGBIRD. White color indicates absence of attention. (a) random attention with $r = 2$, (b) sliding window attention with $w = 3$ (c) global attention with $g = 2$. (d) the combined BIGBIRD model.

Big Bird: Transformers for Longer Sequences https://arxiv.org/pdf/2007.14062.pdf



https://amatriain.net/blog/transformer-models-an-introduction-and-catalog-2d1e9039f376/

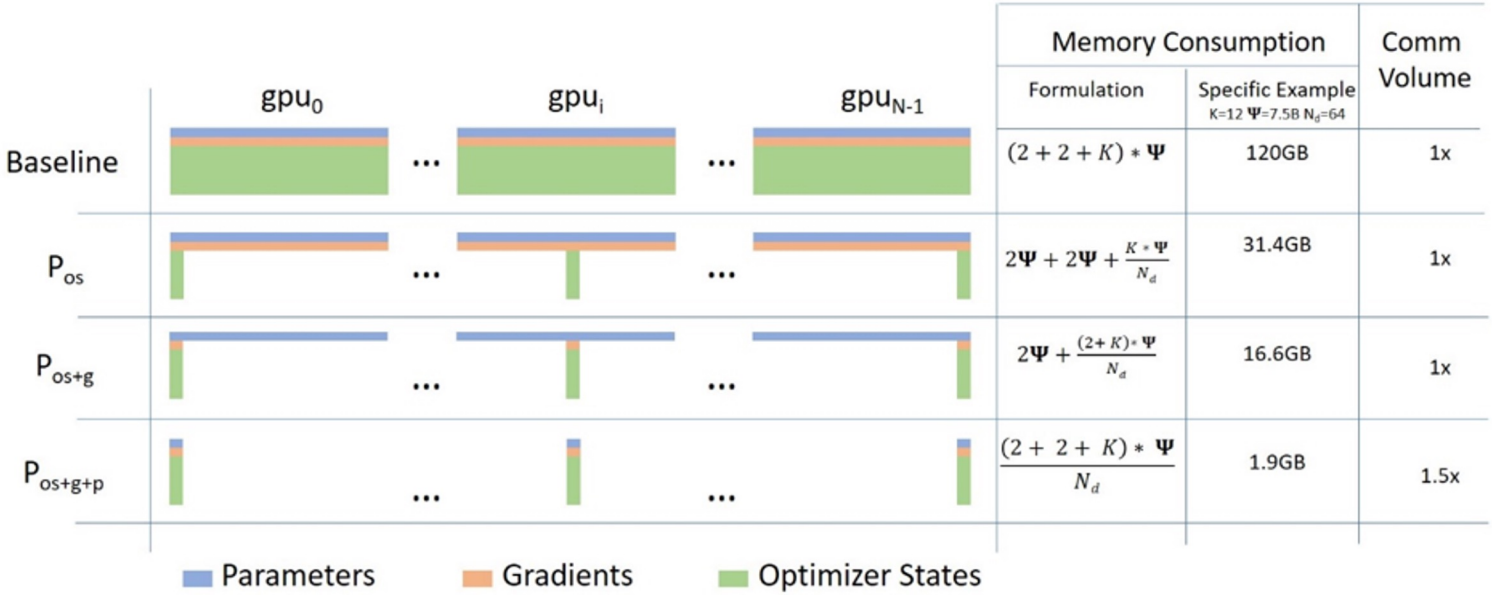
Infra and hardware improvements

- Need special support for very large models
- Some key improvements
    - Better distributed systems parallelization: zero & deepspeed
    - Better Tuning: uTransfer, Lion
    - Better low-level code: Flashattention

Zero & DeepSpeed

Better memory management across multiple GPUs to reduce communication overhead

Clever management of training parameters across GPUs and machines



	Memory Consumption		Comm Volume
	Formulation	Specific Example K=12 Ψ=7.5B N_d=64	
Baseline	$(2+2+K)*\Psi$	120GB	1x
P_{os}	$2\Psi+2\Psi+\frac{K*\Psi}{N_d}$	31.4GB	1x
P_{os+g}	$2\Psi+\frac{(2+K)*\Psi}{N_d}$	16.6GB	1x
P_{os+g+p}	$\frac{(2+2+K)*\Psi}{N_d}$	1.9GB	1.5x

https://www.microsoft.com/en-us/research/blog/zero-deepspeed-new-system-optimizations-enable-training-models-with-over-100-billion-parameters/

Side note on large batch size training

Don't Decay the Learning Rate, Increase the Batch Size

Samuel L. Smith, Pieter-Jan Kindermans, Chris Ying, Quoc V. Le

It is common practice to decay the learning rate. Here we show one can usually obtain the same learning curve on both training and test sets by instead increasing the batch size during training. This procedure is successful for stochastic gradient descent (SGD), SGD with momentum, Nesterov momentum, and Adam. It reaches equivalent test accuracies after the same number of training epochs, but with fewer parameter updates, leading to greater parallelism and shorter training times. We can further reduce the number of parameter updates by increasing the learning rate ϵ and scaling the batch size $B \propto \epsilon$. Finally, one can increase the momentum coefficient m and scale $B \propto 1/(1-m)$, although this tends to slightly reduce the test accuracy. Crucially, our techniques allow us to repurpose existing training schedules for large batch training with no hyper-parameter tuning. We train ResNet-50 on ImageNet to 76.1% validation accuracy in under 30 minutes.

Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour
Don't Decay the Learning Rate, Increase the Batch Size

uTransfer

Hyper tune on smaller model

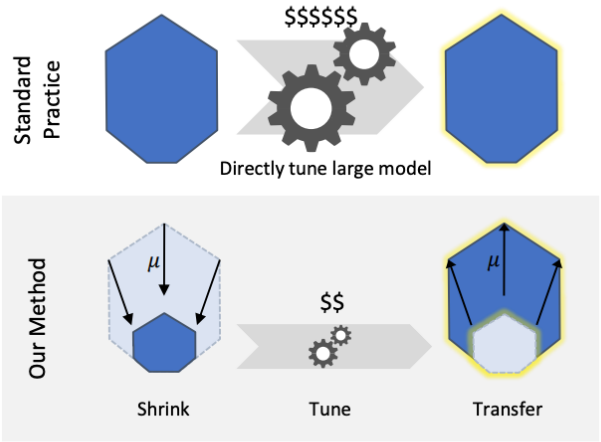


Figure 2: Illustration of μTransfer

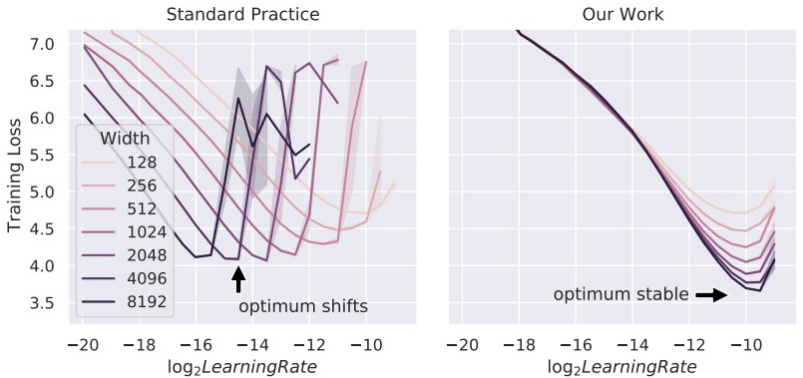


Figure 1: Training loss against learning rate on Transformers of varying d_{model} trained with Adam. Conventionally and in contrast with our technique, different widths do not share the same optimal hyperparameter; wider networks do not always perform better than narrower ones; in fact they underperform the same-width networks in our technique even after tuning learning rate. See Sections 3 and 4 for experimental setup.

https://openreview.net/forum?id=Bx6qKuBM2AD

Lion - EvoLved Sign Momentum

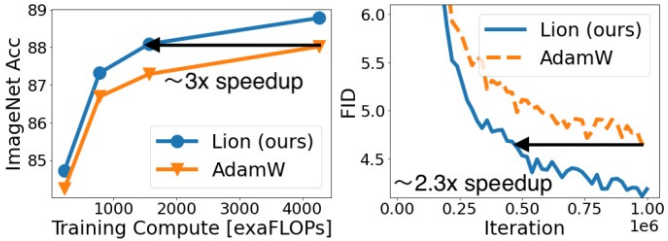
https://github.com/lucidrains/lion-pytorch
https://arxiv.org/abs/2302.06675

- Evolutionary-based search for new optimizer by DeepMind

Table 1: Accuracy of BASIC-L (Pham et al., 2021) on ImageNet and several robustness benchmarks. We apply Lion to both vision tower pre-training and vision-language contrastive training stages. The previous SOTA results on *zero-shot* and *fine-tuning* ImageNet accuracy are 86.3% and 91.0% (Yu et al., 2022).

Optimizer	Zero-shot						Fine-tune
	ImageNet	V2	A	R	Sketch	ObjectNet	ImageNet
Adafactor	85.7	80.6	85.6	95.7	76.1	82.3	90.9
Lion	**88.3**	**81.2**	**86.4**	**96.8**	**77.2**	**82.9**	**91.1**

Figure 1: **Left**: ImageNet fine-tuning accuracy vs. pre-training cost of ViT models on JFT-300M. **Right**: FID of the diffusion model on 256^2 image generation. We use DDPM for 1K steps w/o guidance to decode image. As a reference, the FID of ADM is 10.94 (Dhariwal and Nichol, 2021).

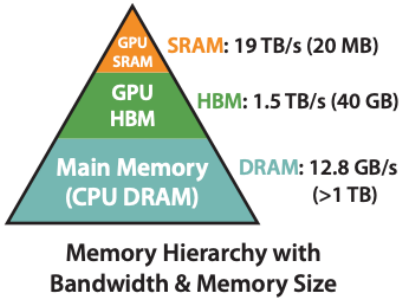


Program 1: Discovered optimizer Lion. $\beta_1 = 0.9$ and $\beta_2 = 0.99$ by default are derived from Program 4. It only tracks momentum and uses the sign operation to compute the update. The two gray lines compute the standard decoupled weight decay, where λ is the strength.

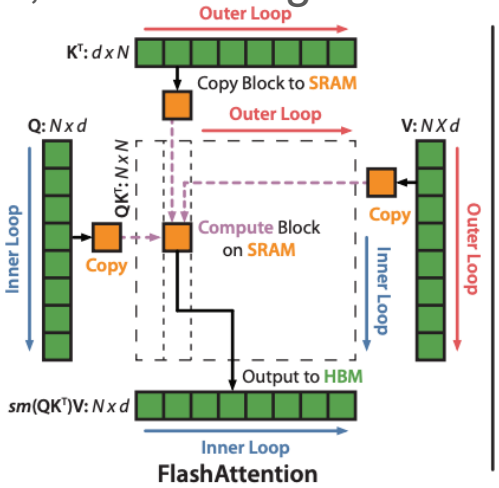
```
def train(weight, gradient, momentum, lr):
  update = interp(gradient, momentum, β1)
  update = sign(update)
  momentum = interp(gradient, momentum, β2)
  weight_decay = weight * λ
  update = update + weight_decay
  update = update * lr
  return update, momentum
```

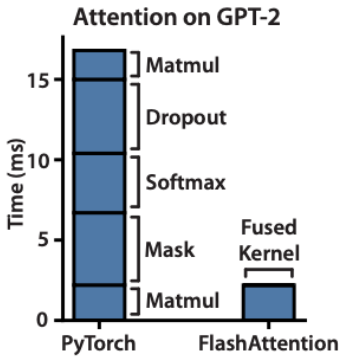
FlashAttention

- Low-level code optimization by taking advantage of the memory heirachy of the GPU – exact attention calculation
- Higher FLOP but lower memory transfer = W in speed (~3x)
- Use less memory by using recompute rather than saving in memory (linear memory growth, ~20x savings at 4k sequence length)



Memory Hierarchy with Bandwidth & Memory Size





https://arxiv.org/abs/2205.14135
https://github.com/HazyResearch/flash-attention

FlashAttention

Integrated into machine learning frameworks

- Pytorch: integrated into core Pytorch in nn.Transformer. Pytorch 2, Triton
- Huggingface's transformers library. On-going, blogpost coming soon.
- Microsoft's DeepSpeed: FlashAttention is integrated into DeepSpeed's inference engine.
- Nvidia's Megatron-LM. This library is a popular framework on training large transformer language models at scale.
- MosaicML Composer library. Composer is a library for efficient neural network training.
- EleutherAI's GPT-NeoX. This is a research library for training large language transformer models at scale based on NVIDIA's Megatron-LM and Microsoft's DeepSpeed.

FlashAttention currently supports:

1. Turing, Ampere, Ada, or Hopper GPUs (e.g., H100, A100, RTX 3090, T4, RTX 2080).
2. fp16 and bf16 (bf16 requires Ampere, Ada, or Hopper GPUs).
3. Head dimensions that are multiples of 8, up to 128 (e.g., 8, 16, 24, ..., 128). Head dim > 64 backward requires A100 or H100.

https://arxiv.org/abs/2205.14135
https://github.com/HazyResearch/flash-attention

Longer, bigger, smaller, smarter

XLNet
Roberta
Megatron
Distill Bert
Albert
BigBird

Tricks to make better transformers

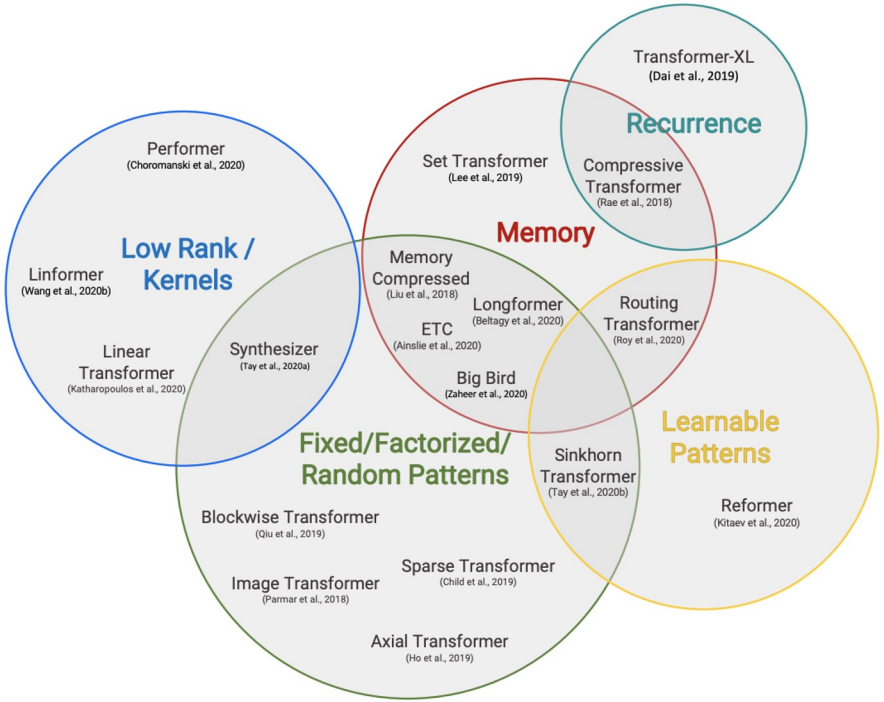
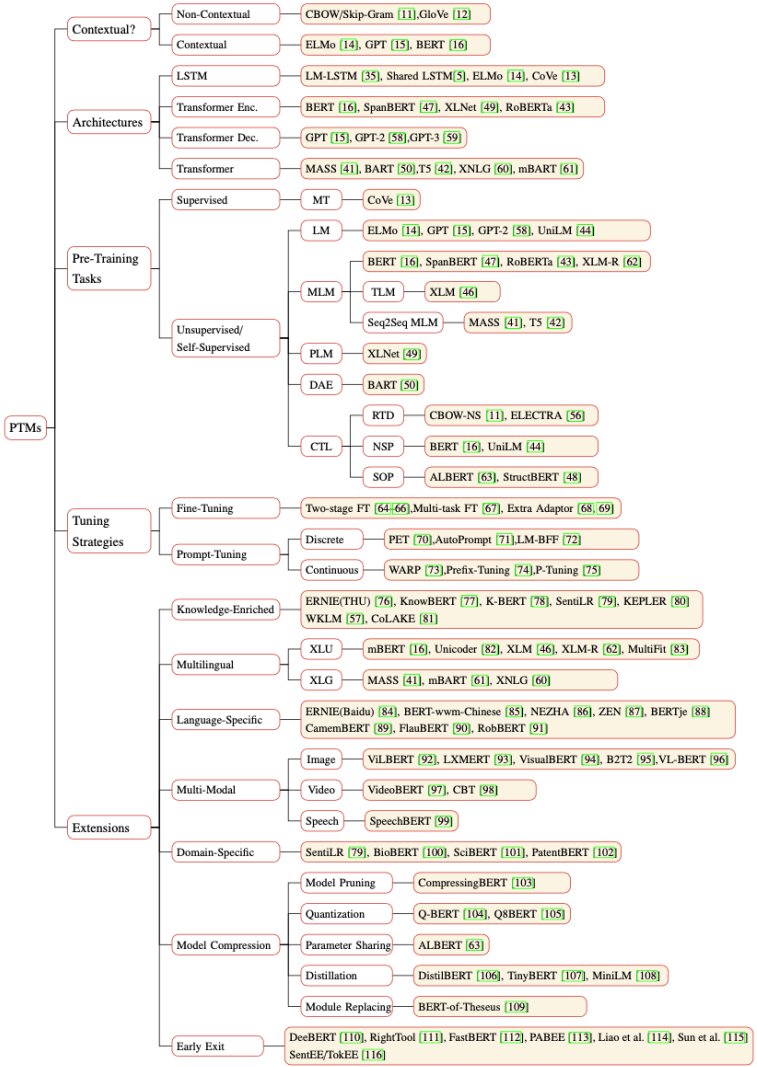


Figure 2: Taxonomy of Efficient Transformer Architectures.

Efficient Transformers: A Survey, 2020 https://arxiv.org/pdf/2009.06732.pdf



Pre-trained Models for Natural Language Processing: A Survey
https://arxiv.org/pdf/2003.08271.pdf

Homework

Working with WangchanBERTa

- A Roberta model
- Different pretrained models with different domains/tokenizations

Schedule

27-Mar-2023	BERT++ MT & QA Project Announcement + Paper Announcement	HW8 due, HW9 & HW10 out
3-Apr-2023	Recent Research in NLP	
10-Apr-2023	Paper Presentation & Progress Report	HW9 due
17-Apr-2023	Songkran Holiday	HW10 due
24-Apr-2023	NLP Application (Guest)	Guest Report
1-May-2023	Project Presentation	

HOW TO READ A SCIENTIFIC ARTICLE

2 Paper types

- Review article/tutorial
  - Give insights about the field
  - Useful for learning about a new field
  - Read multiple to avoid the author’s bias
  - Title usually has “review” or “tutorial”
- Primary research article
  - More details on the experiments and results

Parts of an article

- Abstract
- Introduction
- Methods
- Results and discussion
- Conclusion
- Reference

Things to look for before reading an article

- Publication date
- Author names
  - Previous and newer publications
- Keywords
- Acknowledgements and funding sources

CHARFORMER: FAST CHARACTER TRANSFORMERS VIA GRADIENT-BASED SUBWORD TOKENIZATION

Yi Tay*, Vinh Q. Tran*, Sebastian Ruder†, Jai Gupta, Hyung Won Chung, Dara Bahri
Zhen Qin, Simon Baumgartner, Cong Yu, Donald Metzler
Google Research and DeepMind†
yitay@google.com, vqtran@google.com

https://arxiv.org/pdf/2106.12672.pdf

Getting the big picture

- Read the abstract
- Read the introduction
  - What is the research question?
  - What is the method?
  - What had been done? How is it different from other work?
- Look at figures and results

Tip: keep track of terms you don’t understand

First reading

- Reread the introduction
- Skim methods
- Read results and discussion
  - Does the figures make sense now?
- Write on the article!

Understanding the article

- Reread the article (until you get what you want)
- Check references for parts you don’t understand
- Reread the abstract
  - Does your understanding match the abstract?
- Note down important points. This might come in handy when you write you paper/thesis!

Evaluating the article

- Does the method make sense?
  - What are the limitations that the authors mention?
  - Are there other limitations?
  - Can it be used in other situations?
- Are the experiments legitimate?
  - The sample size is big enough? How big?
  - What kind of dataset is used? Hard enough to current standards? Baselines?
  - The evaluation criterion is sound?
- Have these results been reproduced?
  - Look for articles that cite this paper

ML paper checklist

- What is being done?
- How is it being done?
  - How is it different from previous work
- What is the dataset?
  - Nature of dataset. How many training/testing samples? How many classes/vocab size? Etc.
- Evaluation
  - What are the baselines?
- Practicality
  - Prone to parameter tuning?
  - Computing resource / Runtime (training and testing)
  - Other assumptions? Supervised, unsupervised, etc.

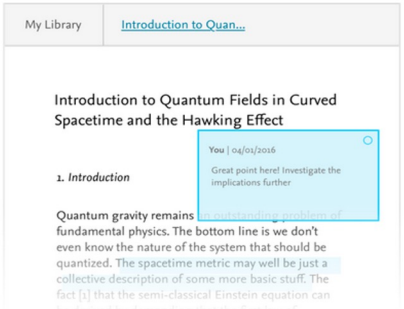
Useful tools

- https://scholar.google.com
  - For finding other articles by the same authors or paper that cites the article

- https://www.mendeley.com/
  - Reference manager

Annotate as you read

Easily add your thoughts on documents in your own library, even from mobile devices. For ease of collaboration, you can also share documents with groups of colleagues and annotate them together.



- Many famous papers has web presence
  - reddit, twitter, or openreview
  - video explaining the paper

Useful resources

Most famous NLP papers have blogs explaining them nowadays

Huggingface usually implements a paper within a couple weeks

https://lilianweng.github.io/lil-log/

Yannic's youtube channel
https://www.youtube.com/channel/UCZHmQk67mSJgfCCTn7xBfew

Paperwithcode's twitter
https://twitter.com/paperswithcode

Paper presentation

Group of 3-5 people

Pick from the list of papers we provide

8 minute presentation + 2 minute QA

Should cover the ML paper checklist

	BERT++ MT & QA	HW8 due, HW9 &
27-Mar-2023	Project Announcement + Paper Announcement	HW10 out
3-Apr-2023	Recent Research in NLP	
10-Apr-2023	Paper Presentation & Progress Report	HW9 due
17-Apr-2023	Songkran Holiday	HW10 due
24-Apr-2023	NLP Application (Guest)	Guest Report
1-May-2023	Project Presentation	

Project

Group of 3-5 people

Anything text/NLP related

Must has some application component (cannot be purely basic NLP task)

	BERT++ MT & QA	HW8 due, HW9 &
27-Mar-2023	Project Announcement + Paper Announcement	HW10 out
3-Apr-2023	Recent Research in NLP	
10-Apr-2023	Paper Presentation & Progress Report	HW9 due
17-Apr-2023	Songkran Holiday	HW10 due
24-Apr-2023	NLP Application (Guest)	Guest Report
1-May-2023	Project Presentation	