Next Class: Project Meetings

- We'll share a schedule with specific time allocated for each team
- We will give feedback on proposals
- Opportunity for you to ask questions and discuss

Figures: Vaswani et al. Attention is All You Need. NeurIPS 2017.
Geva et al. Transformer Feed-Forward Layers Are Key-Value Memories. EMNLP 2021.

Carnegie Mellon University
Language Technologies Institute

Architecture-specific Tricks II

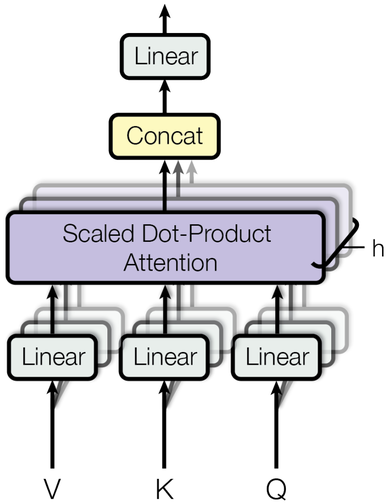
Transformers (mostly)

Yonatan Bisk & Emma Strubell

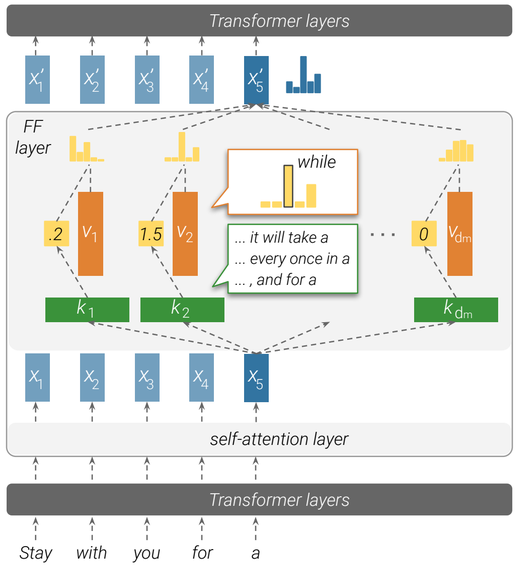
Why are transformers inefficient?

Transformers

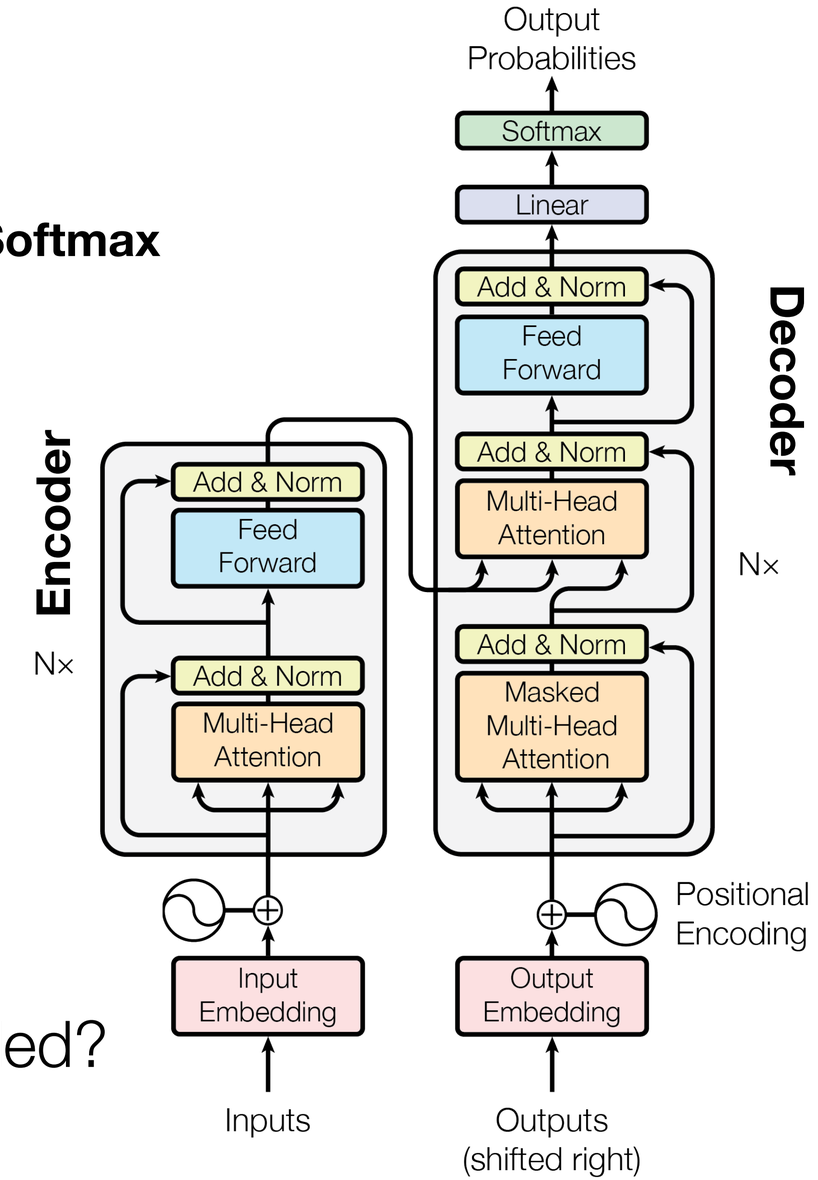
Multi-head self-attention



Feed-forward



Softmax

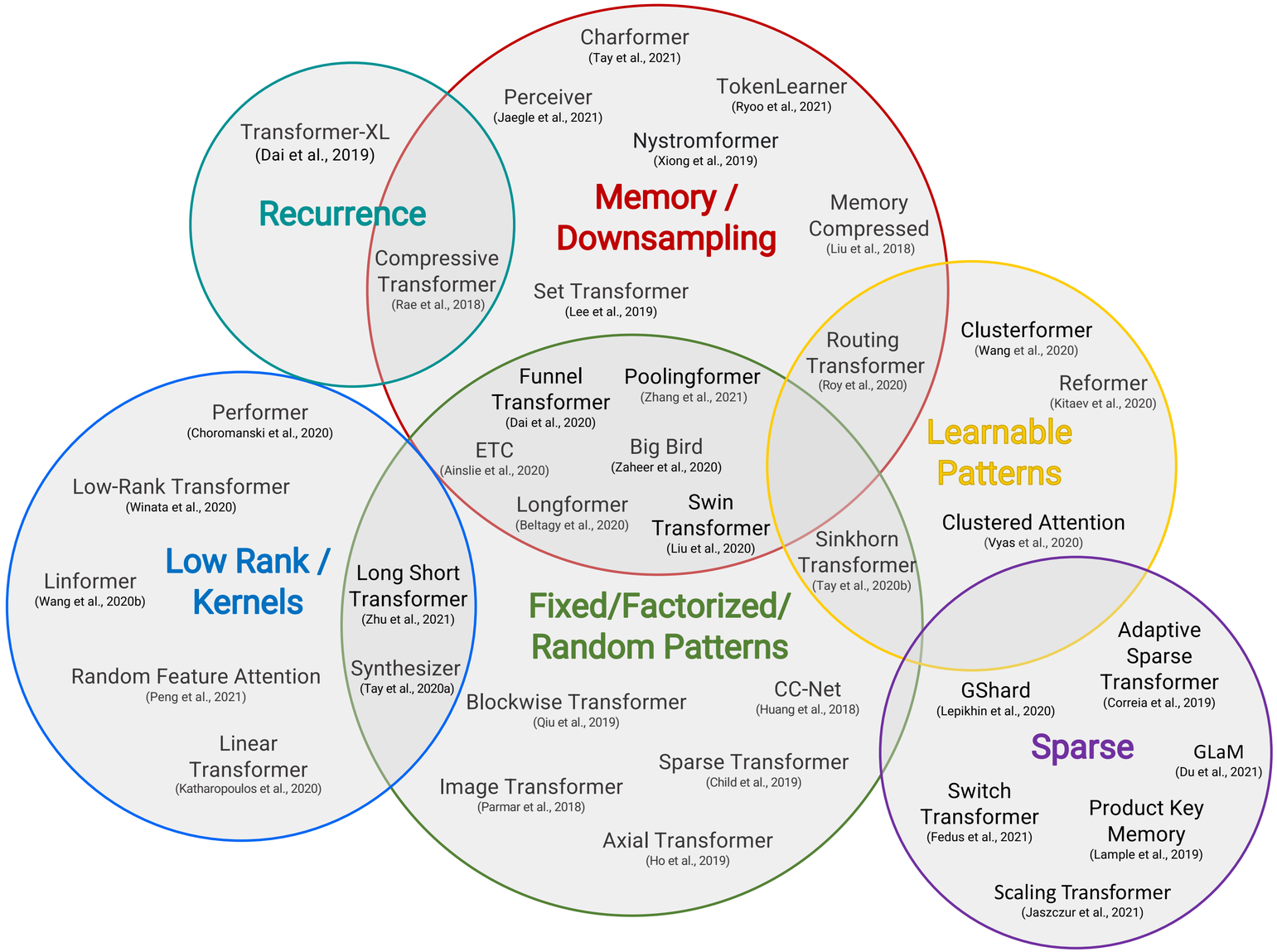


- How many representations at each layer?
- What interactions between those representations are needed?
- Do we need all layers all the time?

► **Check out The Annotated Transformer (Sasha Rush, Vincent Nguyen, Guillaume Klein) for a great tutorial!**

Figures: Vaswani et al. Attention is All You Need. NeurIPS 2017.
Geva et al. Transformer Feed-Forward Layers Are Key-Value Memories. EMNLP 2021.

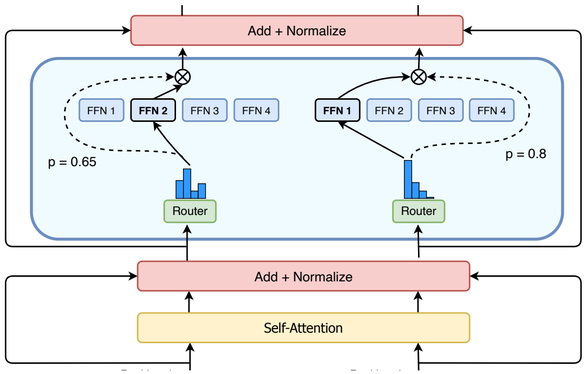
Efficient Transformers

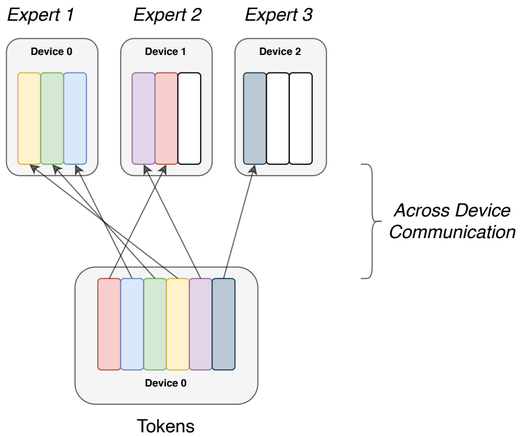


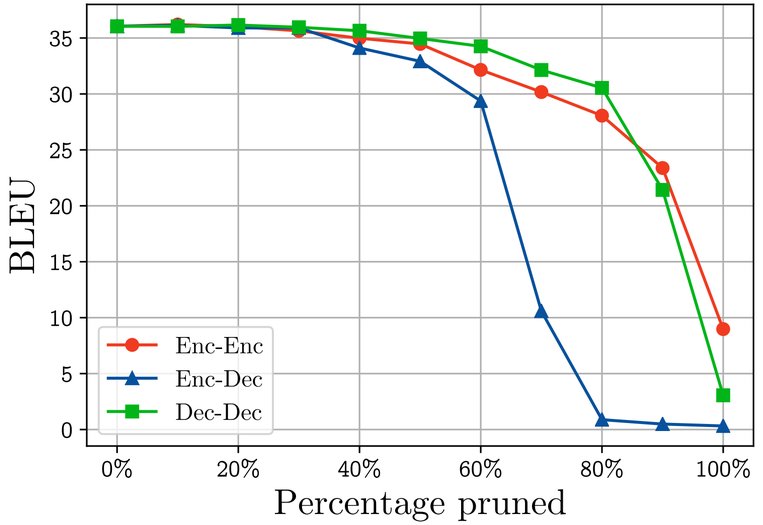
Y. Tay, M. Dehghani, D. Bahri, D. Metzler. Efficient Transformers: A Survey. arxiv 2009.06732, 2022.

Sparsity & pruning

- **Structured pruning:** Are Sixteen Heads Really Better than One? (Michel et al. 2020)
- **Adaptively sparse transformers:** Replace softmax with α-entmax to learn per-head, sparsity patterns (Correia et al. 2019).
- **Switch Transformer:** Mixture-of-experts (MoE) selects a subset of feed-forward parameters for each token (Fedus et al. 2022).







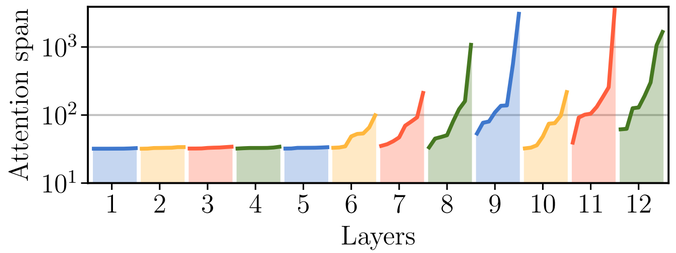
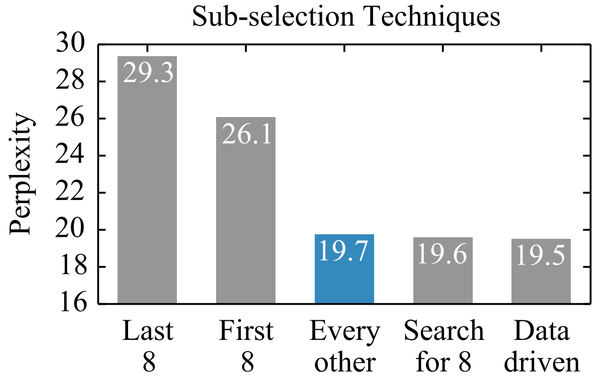


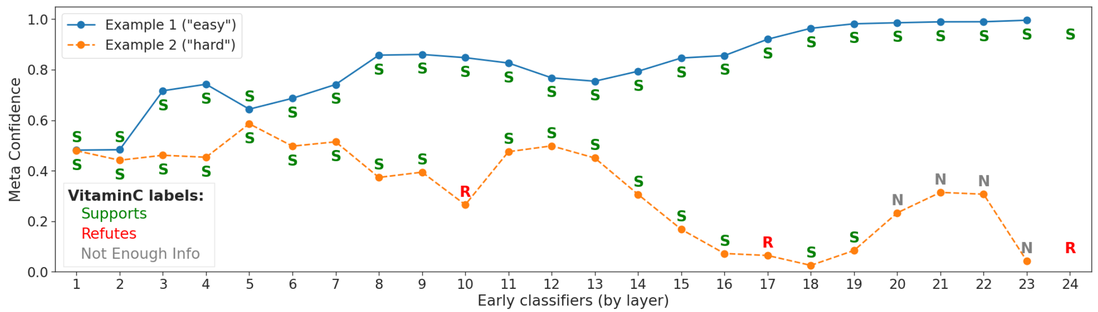
Figure 4: Adaptive spans (in log-scale) of every attention heads in a 12-layer model with span limit $S = 4096$. Few attention heads require long attention spans.

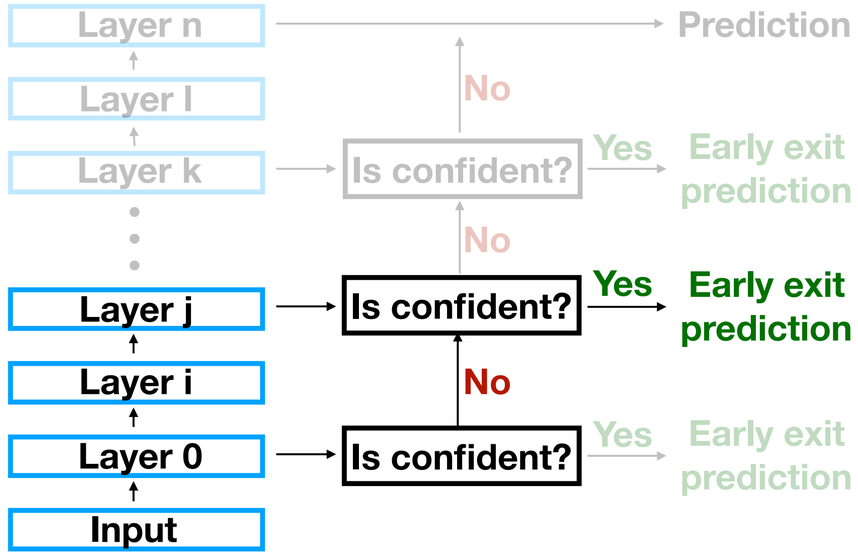
Removing entire layers



- **LayerDrop** (Fan et al. 2019): During training, randomly drop out entire layers. At test time, use a (fixed) subset of layers.

- **Early exit:** after every *k* computations, assess model confidence, predict.
  - **Universal Transformers** (Deghani et al. 2019): Token-wise early-exit.
  - **Sledgehammer** (Schwartz et al. 2020): Calibrated classifier confidence.
  - **Patience-based Early Exit** (PABEE; Zhou et al. 2020): Consistent predictions across *t* layers.
  - **Confident Adaptive Transformers** (CAT; Schuster et al. 2021): Meta consistency classifier.

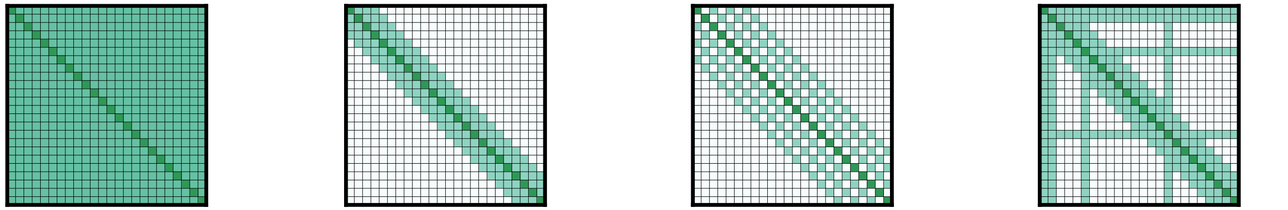




Fixed patterns, learnable patterns

Key idea: Avoid computing the full *N* x *N* self-attention matrix.

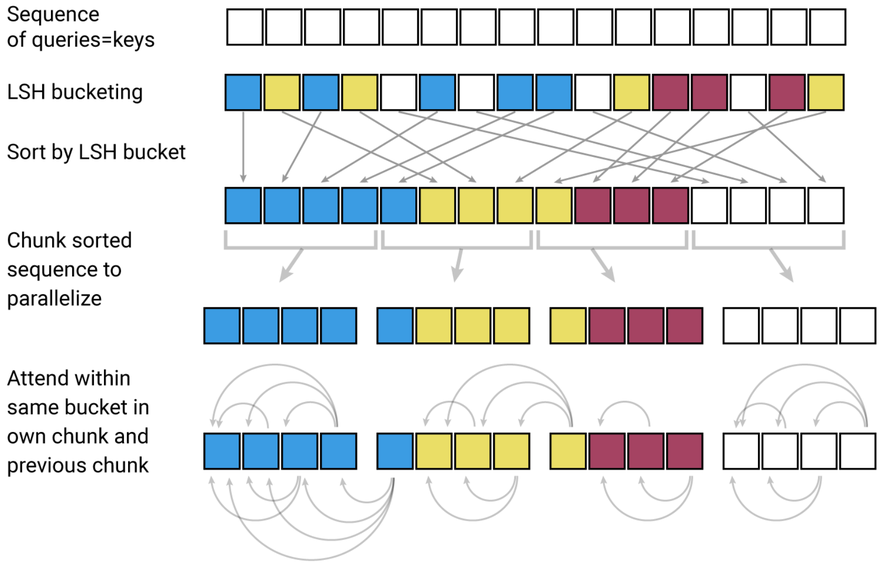
- **Fixed attention patterns:** Approximate full attention by combining fixed patterns.
  e.g. Sparse Transformer (Child et al. 2019),
  Longformer (Beltagy et al. 2020), Swin Transformer (Liu et al. 2021).



(a) Full n^2 attention (b) Sliding window attention (c) Dilated sliding window (d) Global+sliding window

Figure 2: Comparing the full self-attention pattern and the configuration of attention patterns in our Longformer.

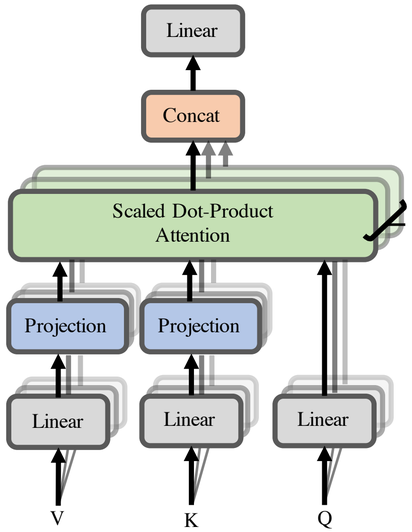
- Alternatively, **learned sparse attention patterns**
  e.g. Reformer (Kitaev et al. 2020), Sinkhorn Transformer (Tay et al. 2020).

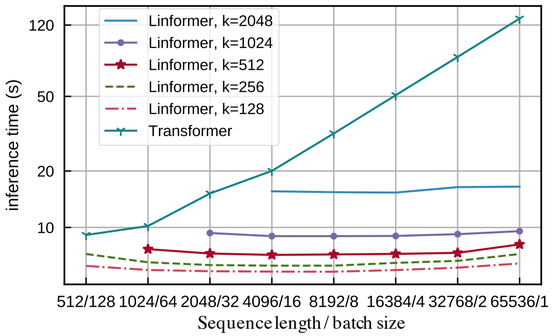


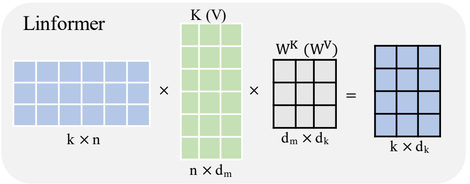
Low-rank / kernel approx. of self-attention

Key idea: Avoid computing the full *N* x *N* self-attention matrix.

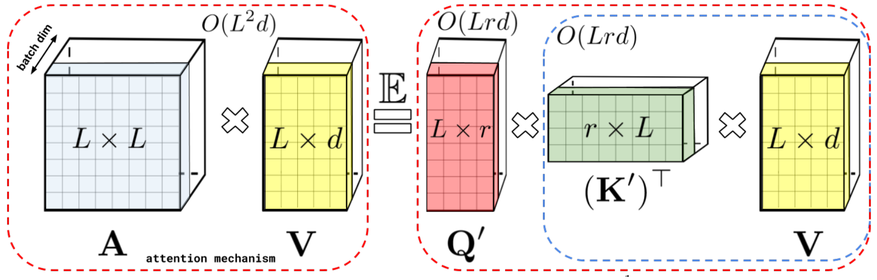
- **Low-rank approximation:** Assume the self-attention matrix is low-rank and factorize.
  e.g. Linformer (Wang et al. 2020).

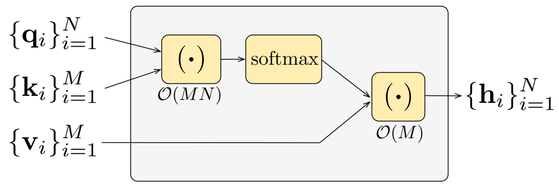




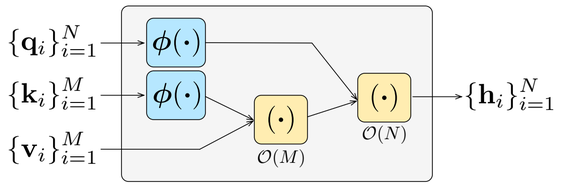


- **Kernelization:** Mathematical reformulation/ reparameterization of self-attention.
  e.g. LightConv/DynamicConv (Wu et al. 2019), Performer (Choromanski et al. 2020), FNet (Lee-Thorp et al, 2021), Random Feature Attention (Peng et al 2021)





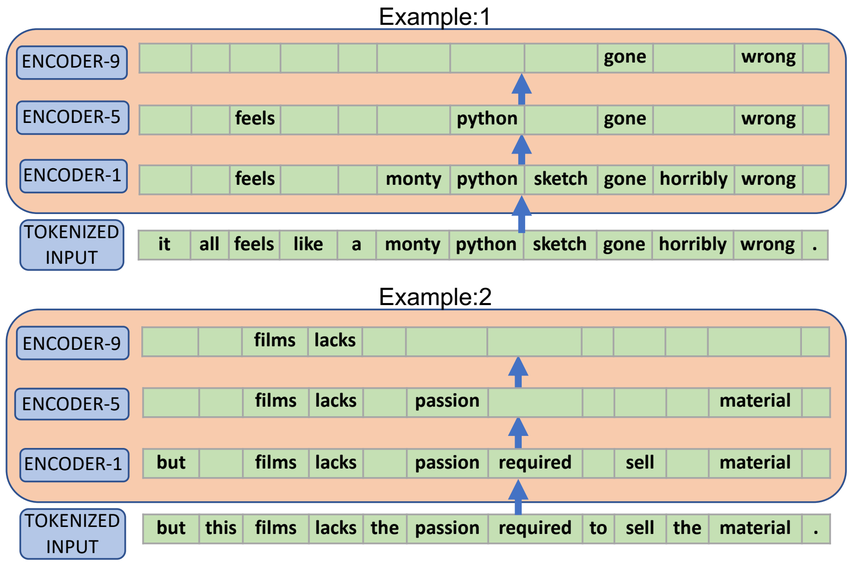
(a) Softmax attention.



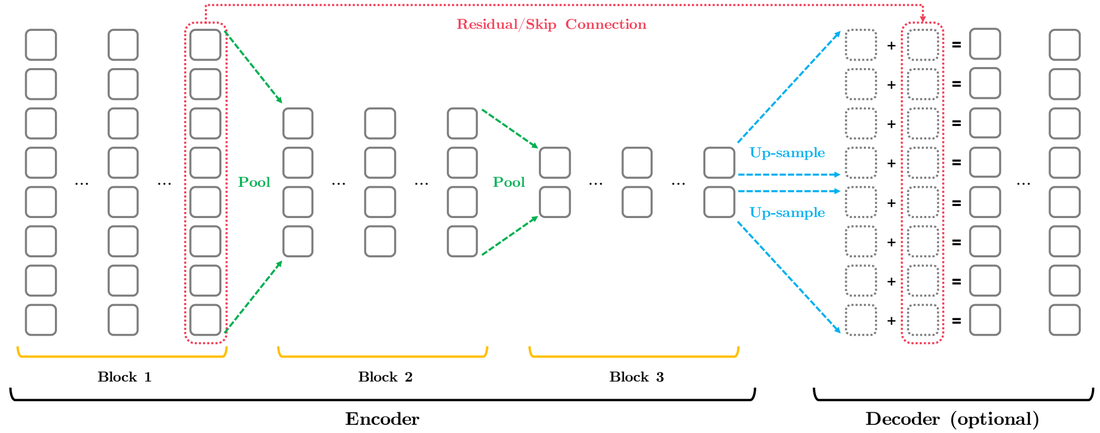
(b) Random feature attention.

Downsampling intermediate representations

- **Key idea:** Don't need a representation of every token (patch, …) at every layer.
- How to choose which representations to remove?



PoWER-BERT (Goyal et al. 2020): Extract only the most important representations at each layer (in practice: using attention scores).



Funnel-Transformer (Dai et al. 2020): Pool tokens into shared representations (in practice: merge pairs of adjacent tokens).

Do transformer modifications transfer across implementations and applications?

- **No**, based on experiments & perhaps this contributes to modifications not being adopted

How to devise models that will transfer?

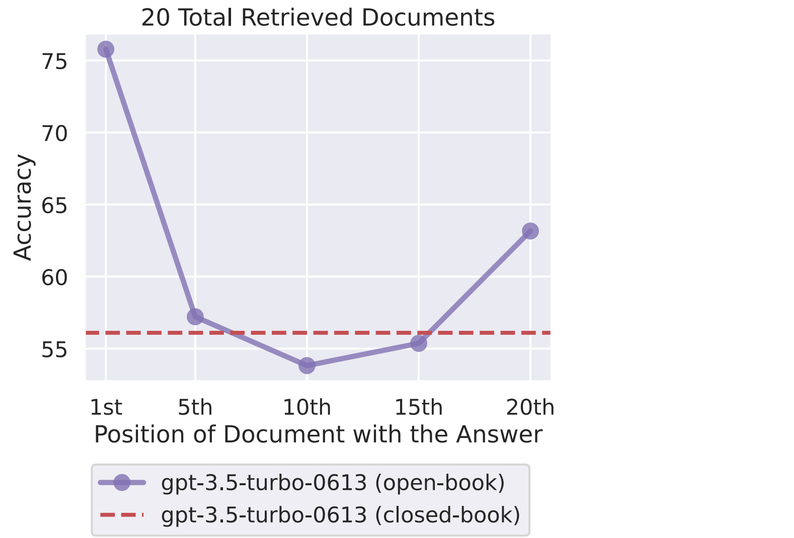
- Run experiments using different frameworks.
- Apply to a variety of downstream applications.
- Control hyperparameters to the extent possible, and/or measure robustness to hyperparameters.
- Report mean and standard deviation across trials.

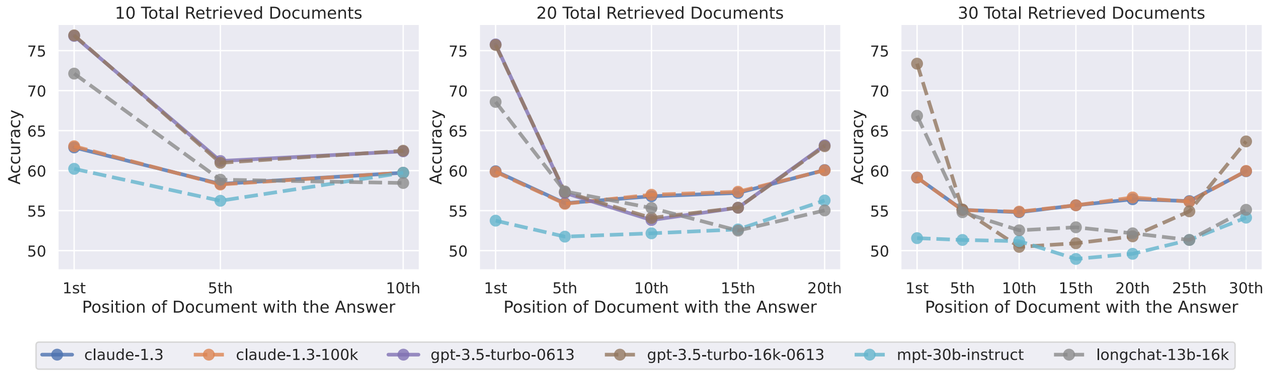
Model	Params	Ops	Step/s	Early loss	Final loss	SGLUE	XSum	WebQ	WMT EnDe
Vanilla Transformer	$223M$	$11.1T$	3.50	2.182 ± 0.005	1.838	71.66	17.78	23.02	26.62
GeLU	$223M$	$11.1T$	3.58	2.179 ± 0.003	1.838	**75.79**	**17.86**	**25.13**	26.47
Swish	$223M$	$11.1T$	3.62	2.186 ± 0.003	1.847	**73.77**	17.74	**24.34**	**26.75**
ELU	$223M$	$11.1T$	3.56	2.270 ± 0.007	1.932	67.83	16.73	23.02	26.08
GLU	$223M$	$11.1T$	3.59	2.174 ± 0.003	**1.814**	**74.20**	**17.42**	24.34	**27.12**
GeGLU	$223M$	$11.1T$	3.55	2.130 ± 0.006	**1.792**	**75.96**	**18.27**	**24.87**	**26.87**
ReGLU	$223M$	$11.1T$	3.57	2.145 ± 0.004	**1.803**	**76.17**	**18.36**	**24.87**	**27.02**
SeLU	$223M$	$11.1T$	3.55	2.315 ± 0.004	1.948	68.76	16.76	22.75	25.99
SwiGLU	$223M$	$11.1T$	3.53	2.127 ± 0.003	**1.789**	**76.00**	**18.20**	**24.34**	**27.02**
LiGLU	$223M$	$11.1T$	3.59	2.149 ± 0.005	**1.798**	**75.34**	**17.97**	**24.34**	26.53
Sigmoid	$223M$	$11.1T$	3.63	2.291 ± 0.019	1.867	**74.31**	17.51	23.02	26.30
Softplus	$223M$	$11.1T$	3.47	2.207 ± 0.011	1.850	**72.45**	17.65	**24.34**	**26.89**
RMS Norm	$223M$	$11.1T$	3.68	2.167 ± 0.008	**1.821**	**75.45**	**17.94**	**24.07**	**27.14**
Rezero	$223M$	$11.1T$	3.51	2.262 ± 0.003	1.939	61.69	15.64	20.90	26.37
Rezero + LayerNorm	$223M$	$11.1T$	3.26	2.223 ± 0.006	1.858	70.42	17.58	23.02	26.29
Rezero + RMS Norm	$223M$	$11.1T$	3.34	2.221 ± 0.009	1.875	70.33	17.32	23.02	26.19
Fixup	$223M$	$11.1T$	2.95	2.382 ± 0.012	2.067	58.56	14.42	23.02	26.31
24 layers, $d_{\text{ff}} = 1536$, $H = 6$	$224M$	$11.1T$	3.33	2.200 ± 0.007	1.843	**74.89**	17.75	**25.13**	**26.89**
18 layers, $d_{\text{ff}} = 2048$, $H = 8$	$223M$	$11.1T$	3.38	2.185 ± 0.005	**1.831**	**76.45**	16.83	**24.34**	**27.10**
8 layers, $d_{\text{ff}} = 4608$, $H = 18$	$223M$	$11.1T$	3.69	2.190 ± 0.005	1.847	**74.58**	17.69	**23.28**	**26.85**
6 layers, $d_{\text{ff}} = 6144$, $H = 24$	$223M$	$11.1T$	3.70	2.201 ± 0.010	1.857	**73.55**	17.59	**24.60**	**26.66**
Block sharing	$65M$	$11.1T$	3.91	2.497 ± 0.037	2.164	64.50	14.53	21.96	25.48
+ Factorized embeddings	$45M$	$9.4T$	4.21	2.631 ± 0.305	2.183	60.84	14.00	19.84	25.27
+ Factorized & shared embeddings	$20M$	$9.1T$	4.37	2.907 ± 0.313	2.385	53.95	11.37	19.84	25.19
Encoder only block sharing	$170M$	$11.1T$	3.68	2.298 ± 0.023	1.929	69.60	16.23	23.02	26.23
Decoder only block sharing	$144M$	$11.1T$	3.70	2.352 ± 0.029	2.082	67.93	16.13	**23.81**	26.08
Factorized Embedding	$227M$	$9.4T$	3.80	2.208 ± 0.006	1.855	70.41	15.92	22.75	26.50
Factorized & shared embeddings	$202M$	$9.1T$	3.92	2.320 ± 0.010	1.952	68.69	16.33	22.22	26.44
Tied encoder/decoder input embeddings	$248M$	$11.1T$	3.55	2.192 ± 0.002	1.840	**71.70**	17.72	**24.34**	26.49
Tied decoder input and output embeddings	$248M$	$11.1T$	3.57	2.187 ± 0.007	**1.827**	**74.86**	17.74	**24.87**	**26.67**
Untied embeddings	$273M$	$11.1T$	3.53	2.195 ± 0.005	**1.834**	**72.99**	17.58	**23.28**	26.48
Adaptive input embeddings	$204M$	$9.2T$	3.55	2.250 ± 0.002	1.899	66.57	16.21	**24.07**	**26.66**
Adaptive softmax	$204M$	$9.2T$	3.60	2.364 ± 0.005	1.982	**72.91**	16.67	21.16	25.56
Adaptive softmax without projection	$223M$	$10.8T$	3.43	2.229 ± 0.009	1.914	**71.82**	17.10	23.02	25.72
Mixture of softmaxes	$232M$	$16.3T$	2.24	2.227 ± 0.017	**1.821**	**76.77**	17.62	22.75	**26.82**
Transparent attention	$223M$	$11.1T$	3.33	2.181 ± 0.014	1.874	54.31	10.40	21.16	**26.80**
Dynamic convolution	$257M$	$11.8T$	2.65	2.403 ± 0.009	2.047	58.30	12.67	21.16	17.03
Lightweight convolution	$224M$	$10.4T$	4.07	2.370 ± 0.010	1.989	63.07	14.86	23.02	24.73
Evolved Transformer	$217M$	$9.9T$	3.09	2.220 ± 0.003	1.863	**73.67**	10.76	**24.07**	26.58
Synthesizer (dense)	$224M$	$11.4T$	3.47	2.334 ± 0.021	1.962	61.03	14.27	16.14	**26.63**
Synthesizer (dense plus)	$243M$	$12.6T$	3.22	2.191 ± 0.010	1.840	**73.98**	16.96	**23.81**	**26.71**
Synthesizer (dense plus alpha)	$243M$	$12.6T$	3.01	2.180 ± 0.007	**1.828**	**74.25**	17.02	**23.28**	26.61
Synthesizer (factorized)	$207M$	$10.1T$	3.94	2.341 ± 0.017	1.968	62.78	15.39	**23.55**	26.42
Synthesizer (random)	$254M$	$10.1T$	4.08	2.326 ± 0.012	2.009	54.27	10.35	19.56	26.44
Synthesizer (random plus)	$292M$	$12.0T$	3.63	2.189 ± 0.004	1.842	**73.32**	17.04	**24.87**	26.43
Synthesizer (random plus alpha)	$292M$	$12.0T$	3.42	2.186 ± 0.007	**1.828**	**75.24**	17.08	**24.08**	26.39
Universal Transformer	$84M$	$40.0T$	0.88	2.406 ± 0.036	2.053	70.13	14.09	19.05	23.91
Mixture of experts	$648M$	$11.7T$	3.20	2.148 ± 0.006	**1.785**	**74.55**	**18.13**	**24.08**	**26.94**
Switch Transformer	$1100M$	$11.7T$	3.18	2.135 ± 0.007	**1.758**	**75.38**	**18.02**	**26.19**	**26.81**
Funnel Transformer	$223M$	$1.9T$	4.30	2.288 ± 0.008	1.918	67.34	16.26	22.75	23.20
Weighted Transformer	$280M$	$71.0T$	0.59	2.378 ± 0.021	1.989	69.04	16.98	23.02	26.30
Product key memory	$421M$	$386.6T$	0.25	2.155 ± 0.003	**1.798**	**75.16**	17.04	**23.55**	**26.73**

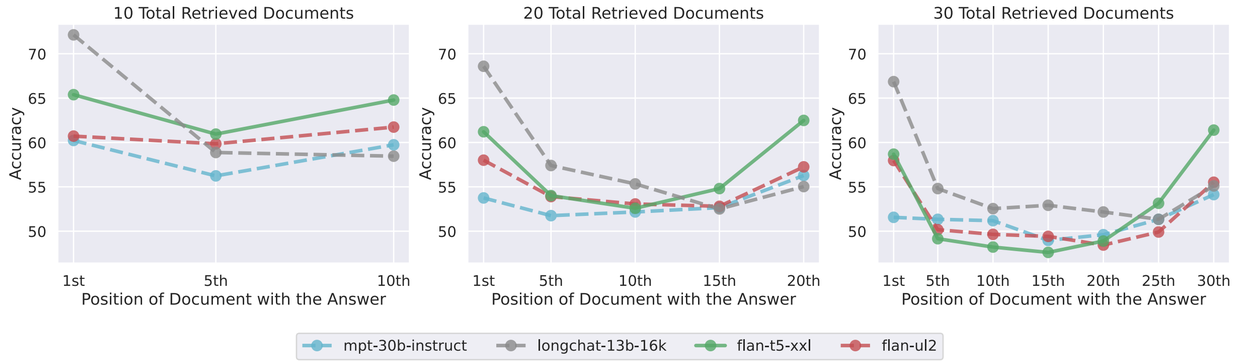
S. Narang, H. W. Chung, Y. Tay, W. Fedus, T. Fevry, M. Matena, K. Malkan, N. Fiedel, N. Shazeer, Z. Lan, Y. Zhou, W. Li, N. Ding, J. Marcus, A. Roberts, C. Raffel. Do Transformer Modifications Transfer Across Implementations and Applications? . EMNLP 2021.

Do Transformer LLMs process long contexts?

- **No.** Info is “lost in the middle.”
  - Encoder-decoder better than decoder-only, for context lengths seen during training.
  - Trend persists before & after instruction tuning.



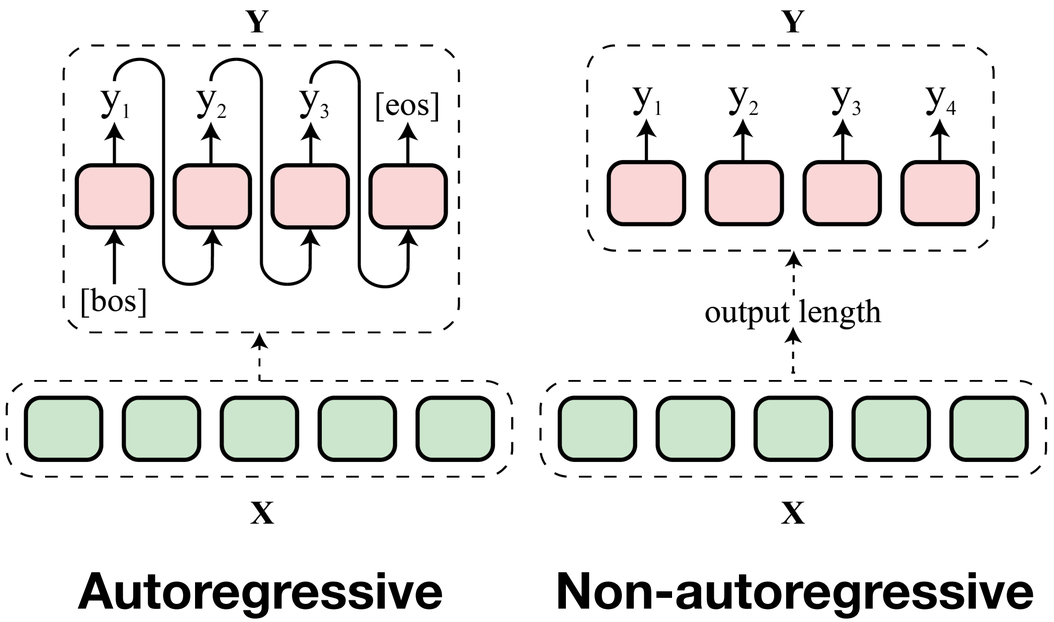




Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, Percy Liang.
Lost in the Middle: How Language Models Use Long Contexts. arXiv 2023.

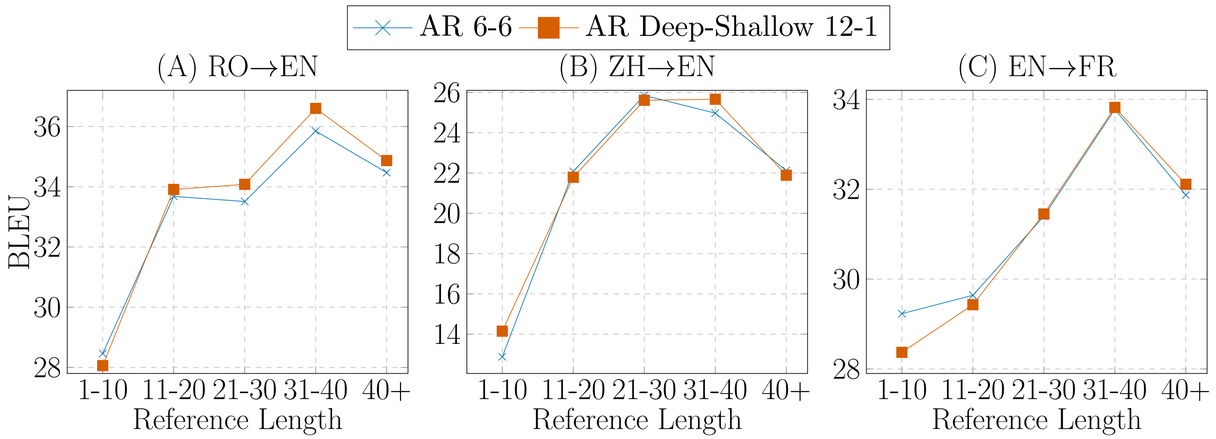
Encoder/decoders and non-autoregressive MT

- Autoregressive decoding is slow on accelerators: not trivially parallelizable.



	By Layer			Full Model		
	Enc.	AR Dec.	NAR Dec.	AR E-D	AR E-1	NAR E-D
Total Operations	$\mathcal{O}(N^2)$	$\mathcal{O}(N^2)$	$\mathcal{O}(TN^2)$	$\mathcal{O}(EN^2 + DN^2)$	$\mathcal{O}(EN^2 + 1 \cdot N^2)$	$\mathcal{O}(EN^2 + DTN^2)$
Time Complex.	$\mathcal{O}(N)$	$\mathcal{O}(N^2)$	$\mathcal{O}(TN)$	$\mathcal{O}(EN + DN^2)$	$\mathcal{O}(EN + N^2)$	$\mathcal{O}(EN + DTN)$

Table 1: Analysis of transformers. Time complex. indicates time complexity when full parallelization is assumed. N: source/target length; E: encoder depth; D: decoder depth; T: # NAR iterations.



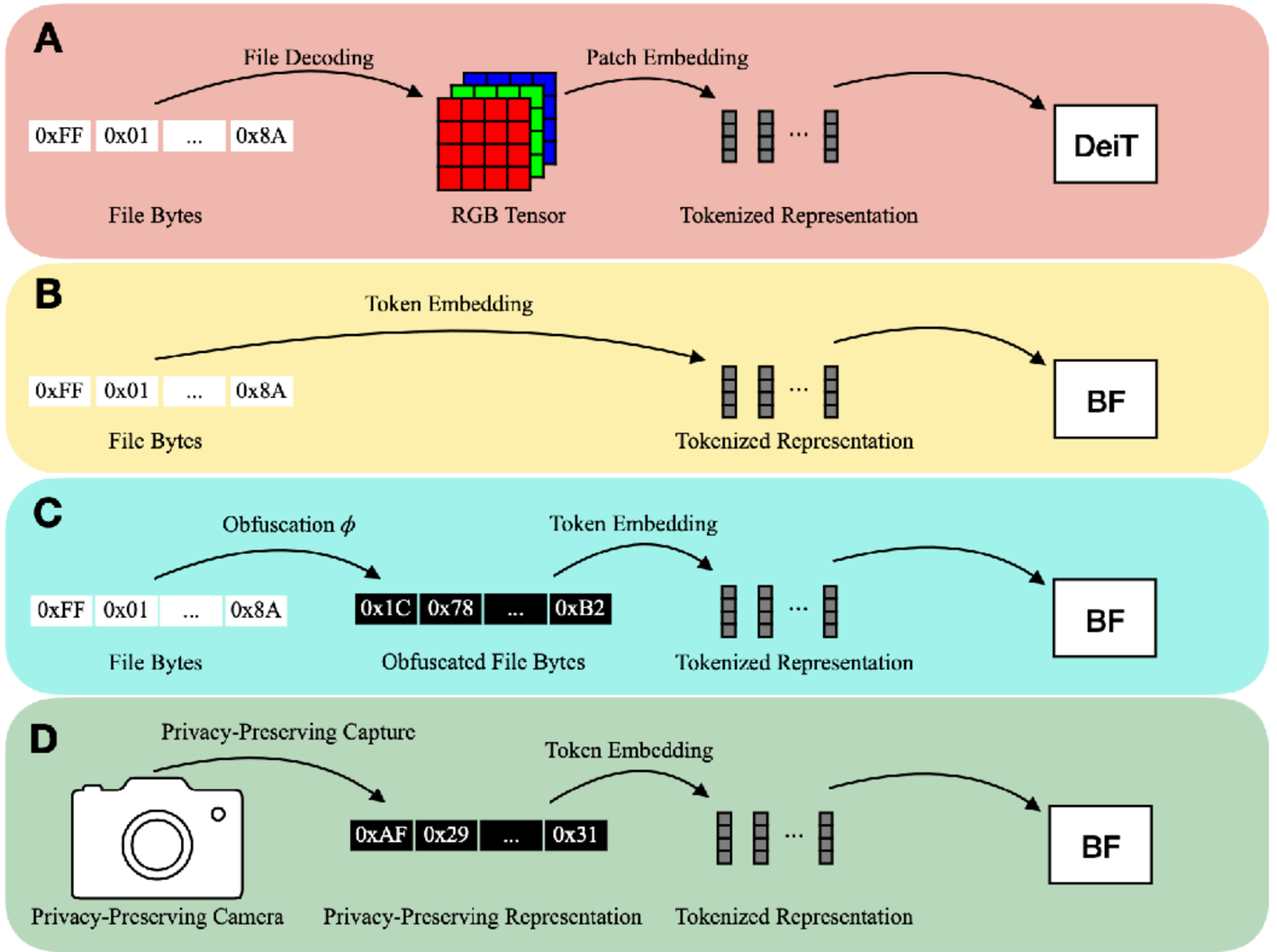
Y. Su, D. Cai, Y. Wang, D. Vandyke, S. Baker P. Li, N. Collier. Non-Autoregressive Text Generation with Pre-trained Language Models. EACL 2021.

Kasai et al. Deep Encoder, Shallow Decoder: Reevaluating Non-autoregressive Machine Translation. ICLR 2021.

Bytes Are All You Need

What's going to go wrong?

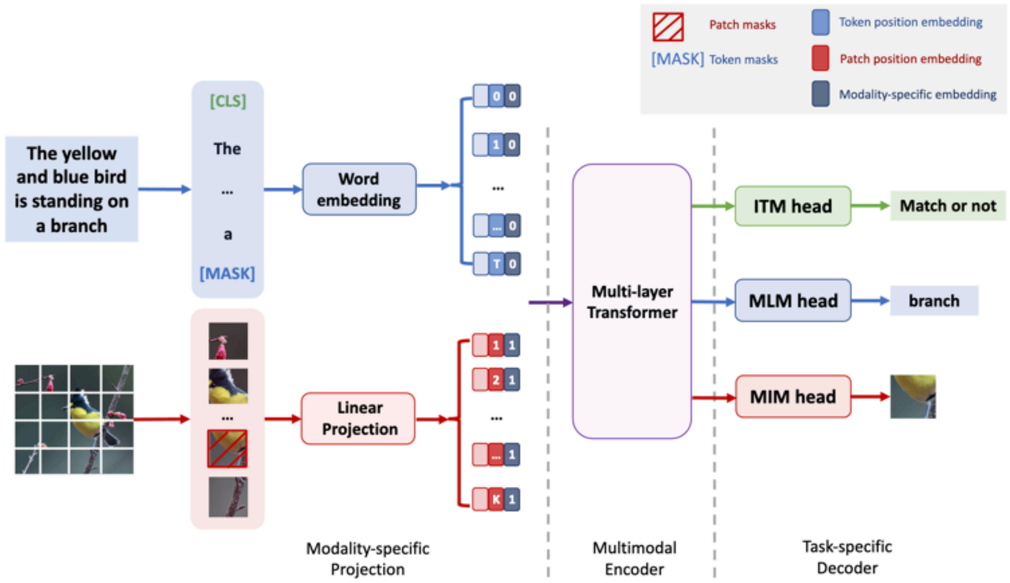
What's going to go right?



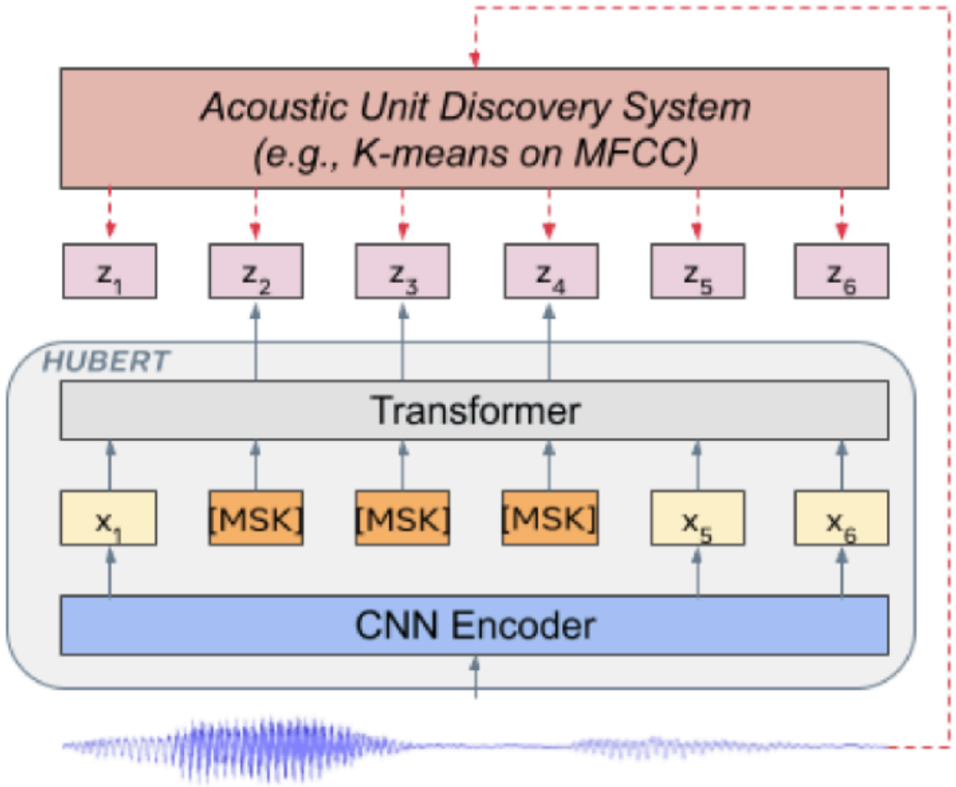
https://arxiv.org/abs/2306.00238

Sequence length and modalities

- How many tokens for an image?
- Audio sequence?



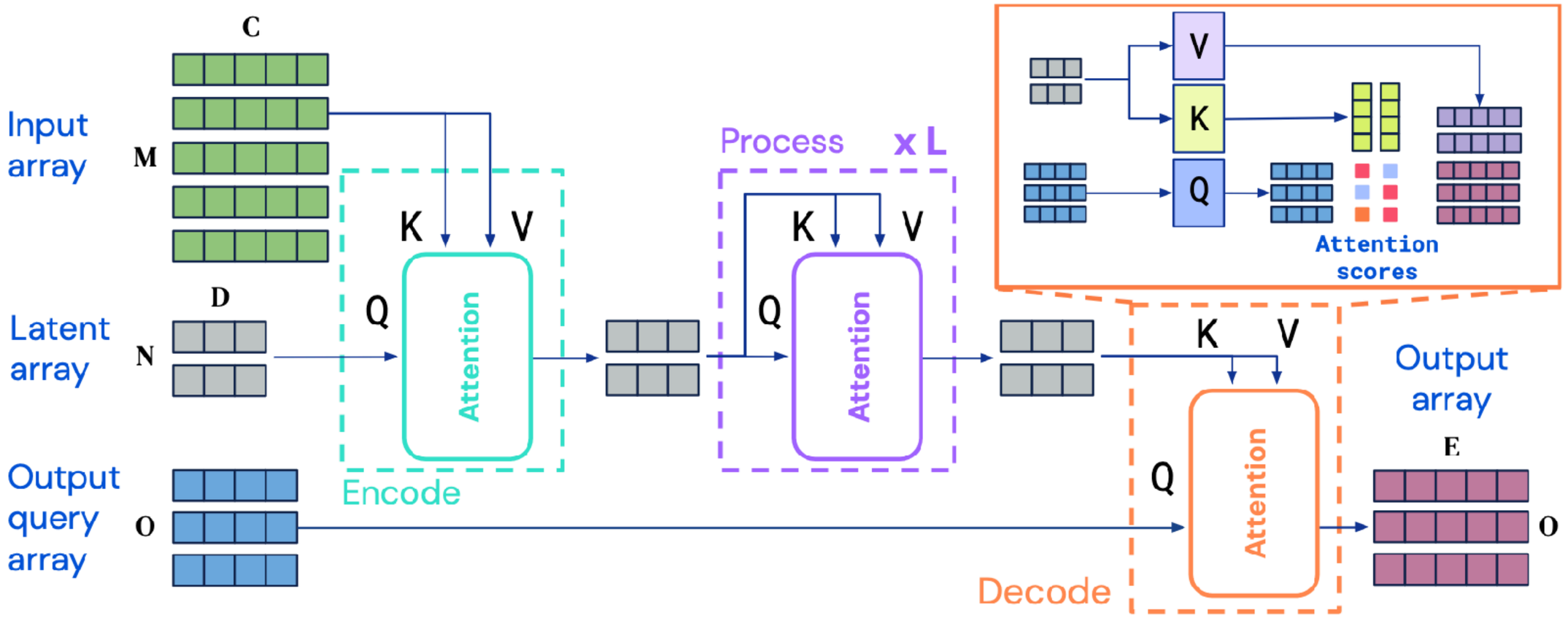
https://arxiv.org/abs/2205.09256



https://arxiv.org/abs/2106.07447

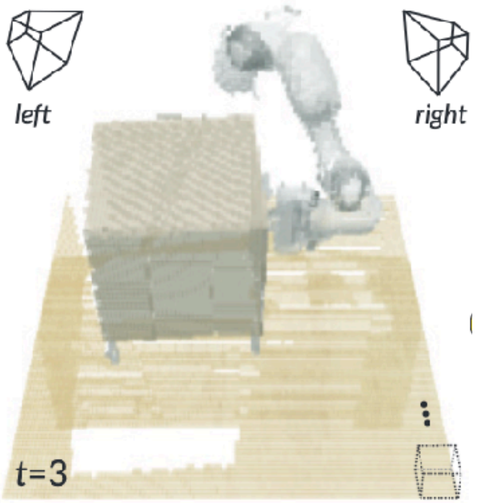
Perceiver IO

Choose your latent and output



https://arxiv.org/abs/2107.14795

Example Domain



"open the middle drawer"