Pretrained Transformers As Universal Computation Engines

Kevin Lu, Aditya Grover, Pieter Abbeel, Igor Mordatch

Shengze Wang, Eli Zachary

Motivation

To generalize pretrained transformers to different modalities and tasks with minimal pretraining

Related Work

Multimodal Transformers:

CLIP, ViLBERT, DALL-E, etc.

Transfer Learning:

ViT (vision), T5 (language), UDSMProt (protein sequences), English to Python, Between Languages

Latent structure is important for transferring between modalities

Pretraining and Finetuning on a Different Task:

Adapter Networks, FiLM, self-modulation, prompt learning, etc.

Method

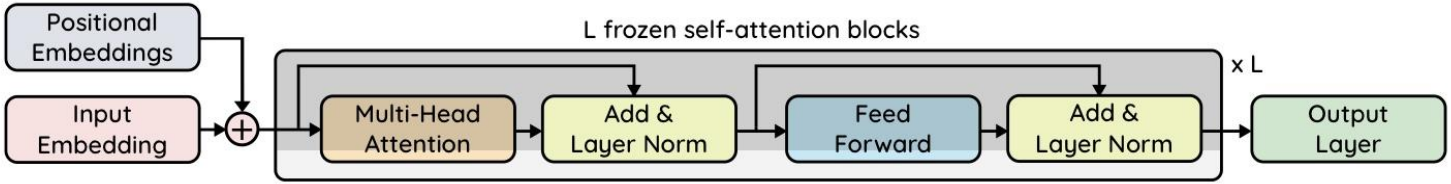
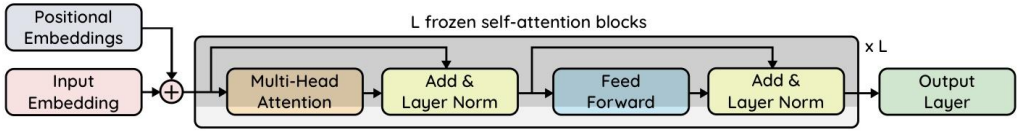


Figure 2: Frozen Pretrained Transformer (FPT). The self-attention & feedforward layers are frozen.

Given a Frozen Pretrained Transformer (FPT), learn to transform the input to the transformer representation and transform the output to the target task with minimal effort

What To Learn?



Input linear layer: to learn to query the FPT from a new domain

Positional Embeddings: universal, but still beneficial to finetune

Layer Norm: to adapt to downstream task in a new domain

Output Linear Layer: to transform the output to a new domain

Target Tasks for Evaluation

Bit Memory: memorize bit strings. 6 bit strings (len 1000) -> 120 tokens of dim 50. Mask with p=0.5

Bit XOR: predict bitwise XOR for string pairs. 2 bit string (len 5) -> 10 tokens of dim 1.

ListOps: parse long math expressions. 512 tokens of dim 15. E.g. “Max 4 3 [Min 2 3] 1 0” -> “4”

MNIST: classify 32x32 gray handwriting image. 64 tokens of dim 16 (i.e 4x4 patches).

CIFAR: 64 tokens of dim 16.

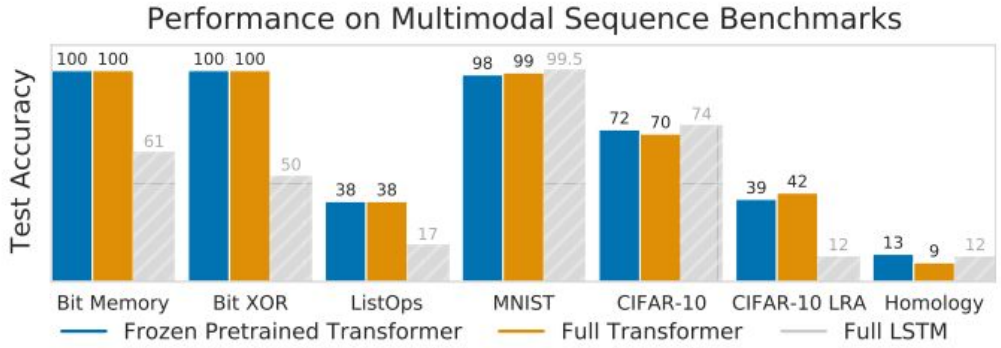
CIFAR-LRA: CIFAR but long range reasoning with minimal inductive bias. 1 token of dim 1024.

Remote Homology Detection: predict protein fold. 1195 possible labels. Consider only sequences of len < 1024. Up to 1024 tokens of dim 25.

Experimental Results

All models unless stated otherwise have 12 layers and 768 hidden dimensions

Can pretrained language models transfer to different modalities?



- FPT performs comparably to or better than transformers or LSTMs fully pretrained on each task
- Recovers exact algorithm on bit tasks
- Fully trained transformers on ListOps, CIFAR-10, and CIFAR-10 LRA used 3 layers

What is the importance of the pre-training modality?

Model	Bit Memory	XOR	ListOps	MNIST	C10	C10 LRA	Homology
FPT	100%	100%	38.4%	98.0%	68.2%	38.6%	12.7%
Random	75.8%	100%	34.3%	91.7%	61.7%	36.1%	9.3%
Bit	100%	100%	35.4%	97.8%	62.6%	36.7%	7.8%
ViT	100%	100%	37.4%	97.8%	72.5%	43.0%	7.5%

- FPT compared to GPT-2 architecture with random parameters, transformer trained on Bit Memory, and ViT
- FPT outperforms all other models, except ViT on vision tasks
- Language appears the be the best modality to use as a "base"

How important is the transformer architecture compared to LSTM architecture?

Model	Bit Memory	XOR	ListOps	MNIST	CIFAR-10	C10 LRA	Homology
Trans.	75.8%	100%	34.3%	91.7%	61.7%	36.1%	9.3%
LSTM	50.9%	50.0%	16.8%	70.9%	34.4%	10.4%	6.6%
LSTM*	75.0%	50.0%	16.7%	92.5%	43.5%	10.6%	8.6%

- LSTM is unmodified and has 3 layers, LSTM* has some elements of Transformer architecture and 12 layers
- All are randomly initialized
- Self-attention architecture accounts for much but not all of Transformers' accuracy

How important is the transformer architecture compared to LSTM architecture?

Layers	ListOps	MNIST	CIFAR-10	C10 LRA
12	16.2%	11.7%	10.8%	10.4%
3	16.8%	70.9%	34.4%	10.4%

Model	ListOps	MNIST	CIFAR-10	C10 LRA
12-Layer LSTM	16.2%	11.7%	10.8%	10.4%
+ Residual Connections	16.8%	70.9%	34.4%	10.4%
+ Positional Embeddings	16.7%	92.5%	43.5%	10.6%
Random Transformer	34.3%	91.7%	61.7%	36.1%

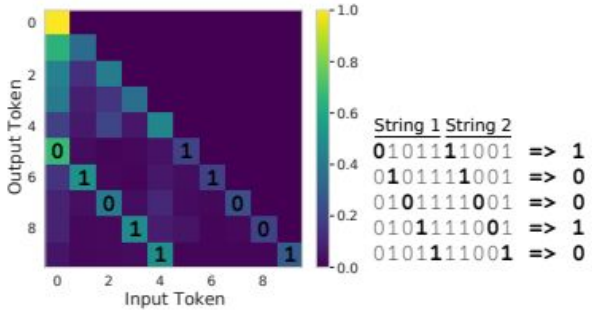
- Repeated unmodified LSTM layers can result in data loss
- Residual connections and positional embeddings account for some but not all of Transformers' qualities

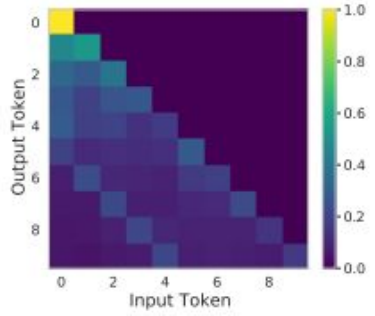
Does language pre-training improve compute efficiency over random initialization?

Model	Memory	XOR	ListOps	MNIST	C10	C10 LRA	Homology
FPT	1×10^4	5×10^2	2×10^3	5×10^3	4×10^5	3×10^5	1×10^5
Random	4×10^4	2×10^4	6×10^3	2×10^4	4×10^5	6×10^5	1×10^5
Speedup	$4\times$	$40\times$	$3\times$	$4\times$	$1\times$	$2\times$	$1\times$

- FPT has far greater gains in efficiency over a randomly initialized model than it does in accuracy
- FPT is also much faster than Bit Memory pretrained model, even on bit tasks(!)

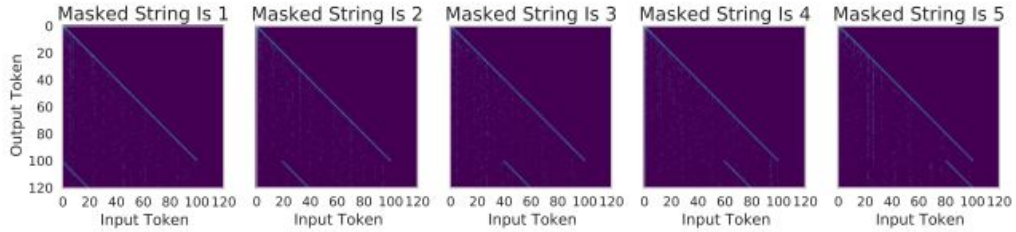
Do the frozen attention layers attend to modality-specific tokens?





- FPT's attention layers show the model discerns a noticeable pattern for solving XOR
- A randomly initialized transformer also shows this behavior, but not as strongly

Do the frozen attention layers attend to modality-specific tokens?



- Same diagonal pattern exists in Bit Memory tasks on FPT
- Here FPT attends to the correct string by finding similarity to the inputs, not solely using position
- Interpretable attention layer patterns were not found on the other tasks

Does freezing the transformer prevent overfitting or underfitting?

Model	# Layers	Test Accuracy	Train Accuracy
FPT (GPT-2)	12	38.6%	38.5%
Vanilla Transformer	3	42%	70%
Linformer	3	39%	97%

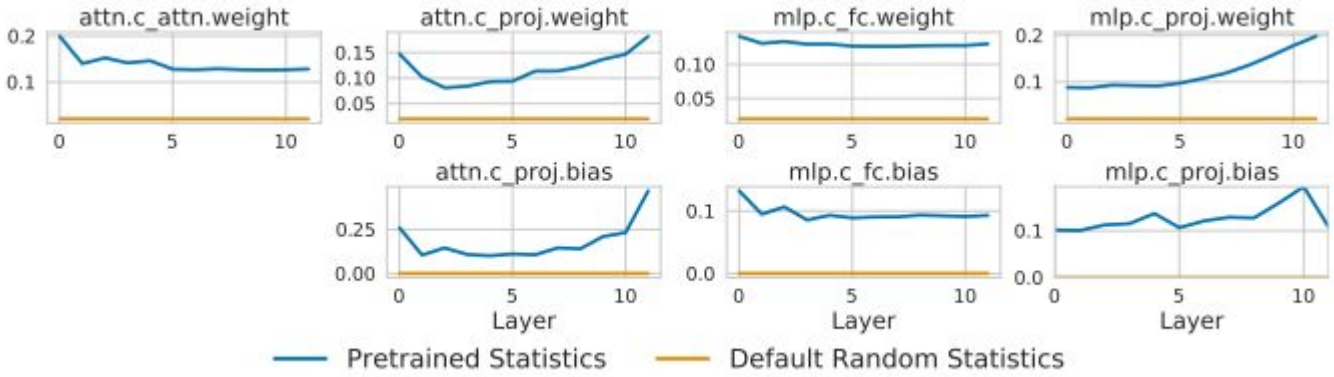
- FPT has essentially no difference between test accuracy and training accuracy, and it barely tends to underfit
- Other tested methods greatly overfit
- FPT is likely good at scaling

Does performance scale with model size?

Model Size	# Layers	Total Params	Trained Params	FPT	Random
Small (Base)	12	117M	106K	68.2%	61.7%
Medium	24	345M	190K	69.8%	64.0%
Large	36	774M	300K	72.1%	65.7%

- Short answer: Yes
- Layers, total parameters, and trained parameters all improve accuracy as they are added
- Contrast with models such as LSTM

Can performance be attributed simply to better statistics for initialization?



Can performance be attributed simply to better statistics for initialization?

Initialization	Memory	XOR	ListOps	MNIST	C10	C10 LRA	Homology
Pretrained	100%	100%	38.4%	98.0%	68.2%	38.6%	12.7%
Statistics Only	100%	100%	37.4%	97.2%	56.5%	33.1%	11.0%
Default	75.8%	100%	34.3%	91.7%	61.7%	36.1%	9.3%

- Statistics are necessary but not sufficient to reach FPT's improvements over a randomly initialized transformer

Can we train a transformer by only finetuning the output layer?

Task	Speedup	Output Only	FPT	Full Transformer
ListOps	$500-2000\times$	32.8%	38.4%	38%
CIFAR-10 LRA	$500-2000\times$	24.7%	38.6%	42%

- Finetuning only the output layer is *much* faster, but leads to a significant drop in performance

What is the role of model depth in token mixing?

With finetuning layernorm

Number of Layers	Pretrained	Random
1	17%	17%
2	36%	16%
6	38%	35%

Without finetuning layernorm

Number of Layers	Pretrained	Random
1	12%	-
3	18%	-
6	33%	-
12	33%	17%
24	-	17%

- FPT mixes its tokens at a significantly lower depth than a random transformer with layernorm finetuned
- Without finetuning layernorm, a random transformer never mixes tokens to the extent of FPT

Finetune More Parameters?

Model	Memory	XOR	ListOps	MNIST	C10	C10 LRA	Homology
FPT	100%	100%	38.4%	98.0%	68.2%	38.6%	12.7%
+ Feedforward	100%	100%	36.0%	98.3%	76.6%	38.2%	13.1%
+ Attention	100%	100%	36.8%	89.0%†	47.7%†	23.0%	10.9%
+ Both	100%	100%	35.8%	93.1%†	32.9%	21.0%	10.5%

Table 13: Additionally finetuning either the feedforward layers, attention layers, or both. We do not use a per-layer learning scheme/etc. †training diverged, number reported before divergence.

Task	Base (FPT)	+ Finetuning All FF Layers	+ Finetuning Last Attn Layer
CIFAR-10	68.2%	76.6%	80.0%

Which Parameters are Important?

Task	output only	+ layernorm	+ input	+ positions
Bit Memory	76%	94%	100%	100%
Bit XOR	56%	98%	98%	100%
ListOps	15%	36%	36%	38%
MNIST	23%	96%	98%	98%
CIFAR-10	25%	54%	60%	68%
CIFAR-10 LRA	17%	39%	39%	39%
Homology	2%	9%	10%	13%

Table 15: Ablation by successively adding certain parameters to the list of finetuned parameters for pretrained frozen transformers.

Initialization	Frozen Layer Norm	Finetuned Layer Norm
Pretrained	61.5%	68.2%
Random	55.0%	61.7%

Table 16: Test accuracy on CIFAR-10 when only finetuning the input and output layer parameters.

Across Architectures?

Task	GPT-2 (FPT Default)	BERT	T5	Longformer
ListOps	38.4%	38.3%	15.4%	17.0%
CIFAR-10	68.2%	68.8%	64.7%	66.8%

Table 17: Test accuracy for frozen pretrained transformer variants (base model sizes).

Conclusions & Takeaway

1. FPT transformer to different modalities well with minimal efforts.
2. Better to pretrain in the same modality as target task.
3. MSA better than LSTM, especially for long sequences.
4. FPT converges faster.
5. FPT underfits the data. Increasing model capacity helps.
6. Best to jointly finetune: input, positional embedding, layer norm, feedforward layer, last attn layer, and output layer. (Everything except most self-attention)

Questions

1. Why does finetuning the self-attention hurt performance (even diverges)?
2. Are these experiments enough to arrive at the conclusions?
3. What does the input linear layer do?