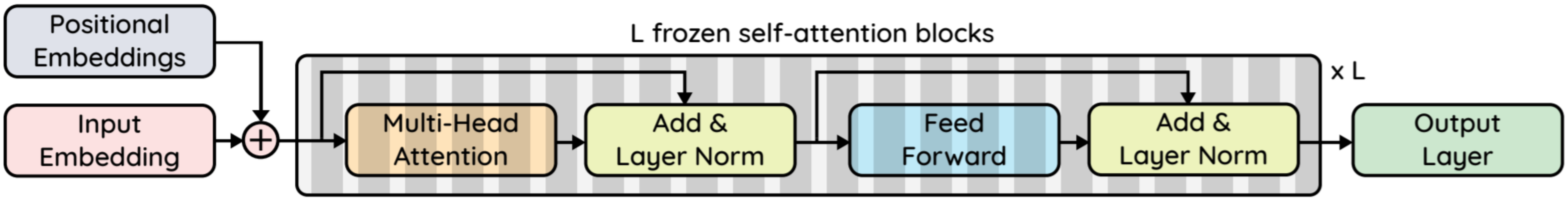
Pretrained transformers as universal computation engines

UC Berkeley + Facebook AI + Google Brain

Under review (NeurIPS 2021)

Transformer

- Transformer and pretrained language model
  - Machine translation
  - Question Answering
  - …
  - + Protein folding / interaction
  - + Computer vision tasks
  - …



Transformers are everywhere!

Research question

- Generalization capability of transformers
  - Same modality: Good
    - (e.g., language → language)
- How about transferring to other modalities?
  - (e.g., language → vision, protein folding, …)
  - This work investigated the generalization capability of "self-attention layers" in transformers

Downstream tasks

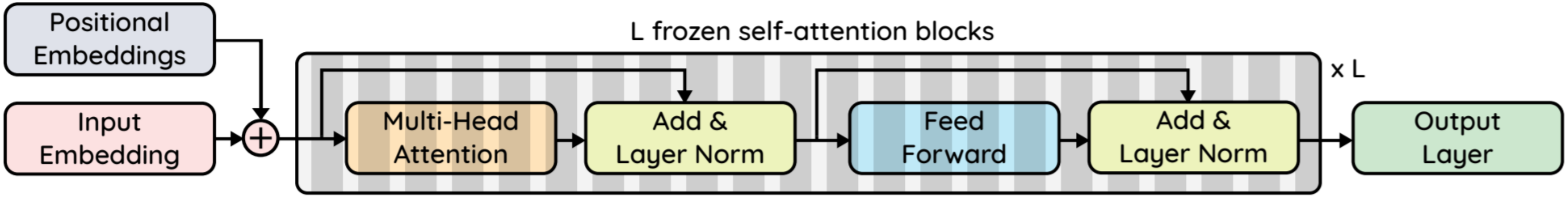
- Bit memory
  - Model should memorize five bitstrings each of length 1000
  - Input: Masked version of five bitstrings
  - Output: Recovered bitstrings
- Bit XOR
  - Input: Two bitstrings of length 5
  - Output: Element-wise XOR of the given bitstrings
- ListOps
  - Compute sequence of list operations
  - Input: [MAX 4 3 [MIN 2 3] 1 0]
  - Output: 4

Downstream tasks

- MNIST
  - 4x4 image patches → 49 tokens of dimension 16
- CIFAR-10
  - 4x4 image patches → 64 tokens of dimension 48
- CIFAR-10 LRA (long range arena)
  - Images are converted to grayscale and flattened
  - 1x1 image patches → 1024 tokens of dimension 1
- Remote homology detection
  - Protein folding prediction
  - Input: Amino acid sequence (dataset provided TAPE)

Experiment 1

- Pretrained GPT-2 with language dataset
  - # of dimension: 768 / # of layers: 12
  - For some datasets, different hyperparameters were used



- Finetuning
  - Input / output layer
  - Layer norm parameters
  - Positional embeddings
  - → 0.086% of entire parameters

Experiment 1 (Cont'd)

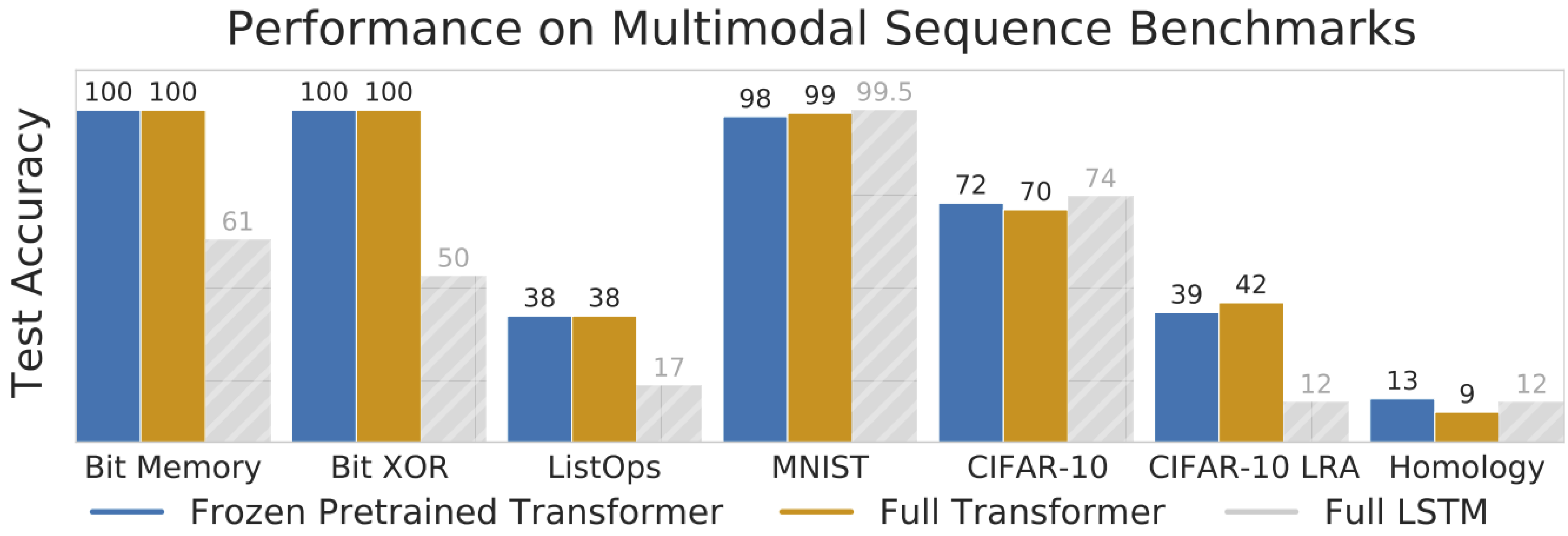


Figure 1: A *frozen* language-pretrained transformer (FPT) – without finetuning the self-attention and feedforward layers – can match the performance of a transformer fully trained on a downstream modality from scratch. We show results on diverse classification tasks (see Section 2.1): numerical computation (Bit Memory/XOR, ListOps), image classification (MNIST, CIFAR-10), and protein fold prediction (Homology). We also show results for a fully trained LSTM to provide a baseline.

Experiment 2

- Only language-pretraining works?
  - Authors compared other pretraining methods
    - Random initialization (Random) == no pretraining
    - Bit memory pretraining (Bit)
    - Image pretraining (ViT): ImageNet-21k classification

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%

Table 2: Test accuracy of language-pretrained (FPT) vs randomly initialized (Random) vs Bit Memory pretraining (Bit) vs pretrained Vision Transformer (ViT) models. The transformer is frozen.

Experiment 3

- In experiment 2, random initialization achieves surprisingly high accuracies, but there exists a considerable gap
  - How about random LSTM?

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%

Table 3: Test accuracy of randomly initialized transformers vs randomly initialized LSTM models. Note that unlike in Figure 1, the LSTM here is frozen. Frozen LSTMs perform very poorly.

- *"Self-attention architecture already serves as an effective inductive bias for universal computation"*

Experiment 4

- In experiment 2, random initialization achieves surprisingly high accuracies, but there exists a considerable gap
  - How about convergence?

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	$4\times$	$40\times$	$3\times$	$4\times$	$1\times$	$2\times$	$1\times$

Table 4: Approximate number of gradient steps until convergence for pretrained (FPT) vs randomly initialized (Random) models. Note that we use the same batch size and learning rate for both models.

Experiment 5

- Do self-attention layers work well?

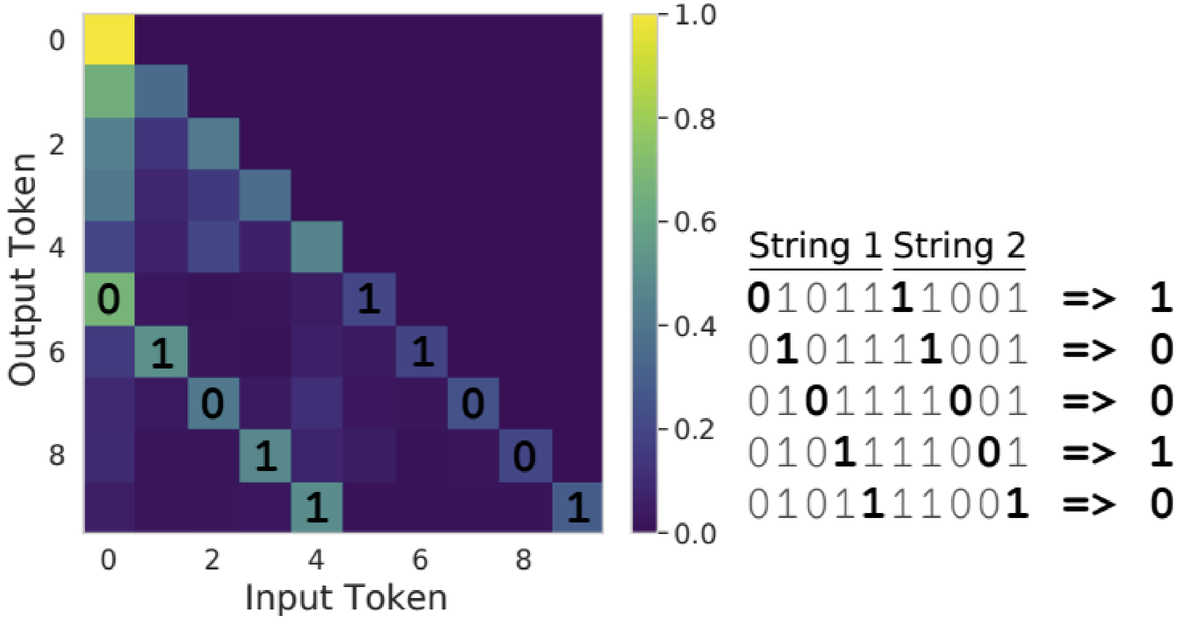


Figure 3: On Bit XOR, the model must produce the element-wise XOR of two bitstrings presented sequentially (inputs 0-4 are the first bitstring, inputs 5-9 are the second). Each token is one bit. FPT learns to attend positionally to the two bits that are XOR'ed by the output token.

Experiment 6

- Overfitting / Underfitting?

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

Table 5: Train vs test accuracies on CIFAR-10 LRA task.

Experiment 7

- 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%

Table 6: Test accuracy of larger frozen transformer models on CIFAR-10.

Experiment 8

- **Better initialization from pretrained transformer**

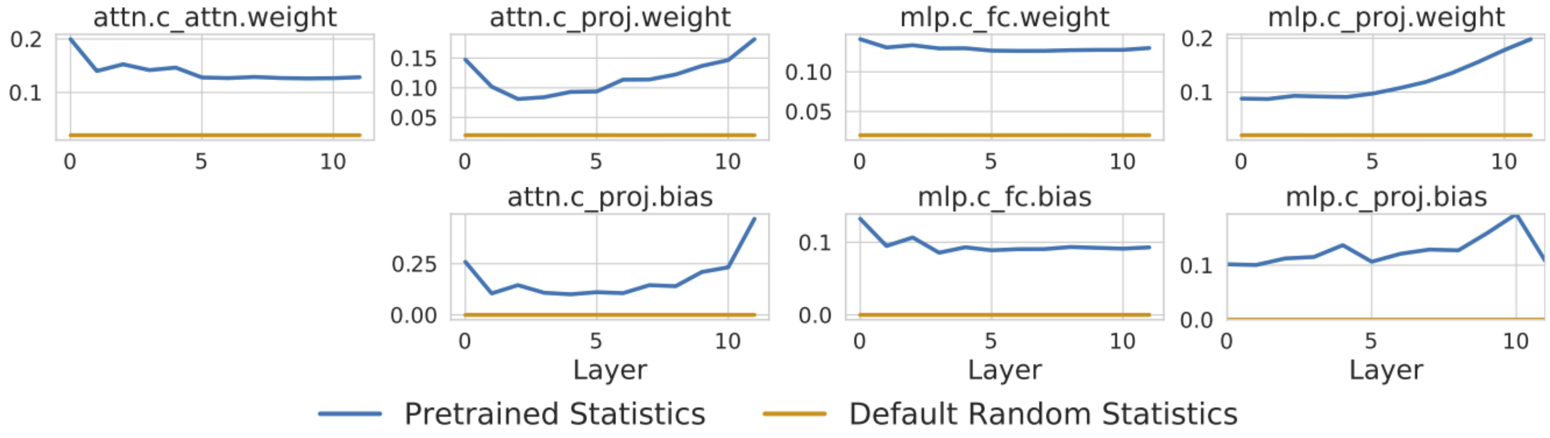


Figure 5: Standard deviation of the parameters by layer for the pretrained GPT-2 model versus default initialization hyperparameters (0.02 for weights and 0 for biases).

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%

Table 7: Test accuracy when initializing parameters with pretrained weights (i.e., FPT) vs randomly initializing parameters according to the mean and variance of the pretrained transformer (Statistics Only) vs random initialization with default parameters (Default).

Discussion