Pretrained Transformers as Universal Computation Engines

By Priyanka Sachan
priyanka_1901CS43@iitp.ac.in

Introduction

- **TRANSFORMERS**
  - Have shown broad success in deep learning serving as the backbone of large models.
  - Utilize self attention layers to extract features across tokens of a sequence.

- **COMMON TRAINING PATTERNS**
  - Training large models on unsupervised or weekly supervised objectives.
  - After that, finetuning or evaluating zero-shot generalization on a downstream task.
  - However, the downstream tasks that have been studied are generally restricted to the same modality as the original training set.

Introduction

- Exploring the generalization capabilities of a transformer in transferring from one modality to another.

- **HYPOTHESIS**
  - Transformers, can be pretrained on a data-rich modality (i.e. where data is plentiful, such as a natural language corpus) and identify feature representations that are useful for arbitrary data sequences, enabling effective downstream transfer to different modalities without expensive finetuning of the self-attention layers.
  - In particular, pretrained language models (LMs) are capable of in terms of generalizing to other modalities with sequential structure, including numerical computation, image classification, and protein fold prediction.

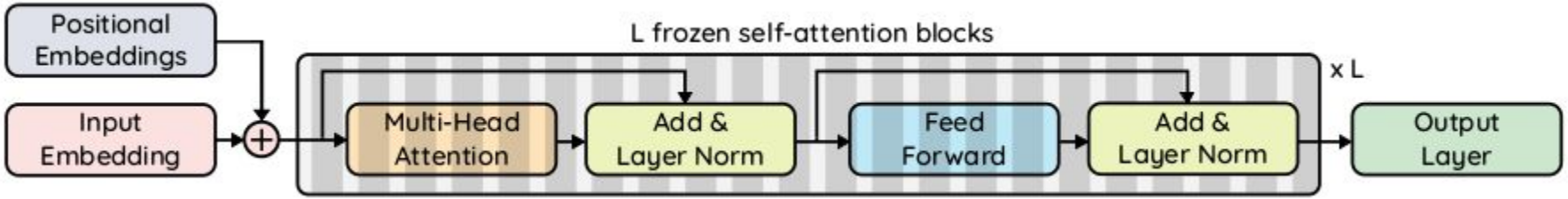
Evaluation Tasks

- Bit memory
- Bit XOR: x0 ^ x1 = y
- ListOps: [MAX 4 3 [MIN 2 3] 1 0]
- MNIST: The tokens given to the model are 4 x 4 image patches.(total 64 tokens)
- CIFAR-10: Same with MNIST
- CIFAR-10 LRA: 1 x 1 image patches (total 1024 tokens with dimension 1)
- Remote homology detection: predicting protein fold problem. (1024 tokens of dimension 25)

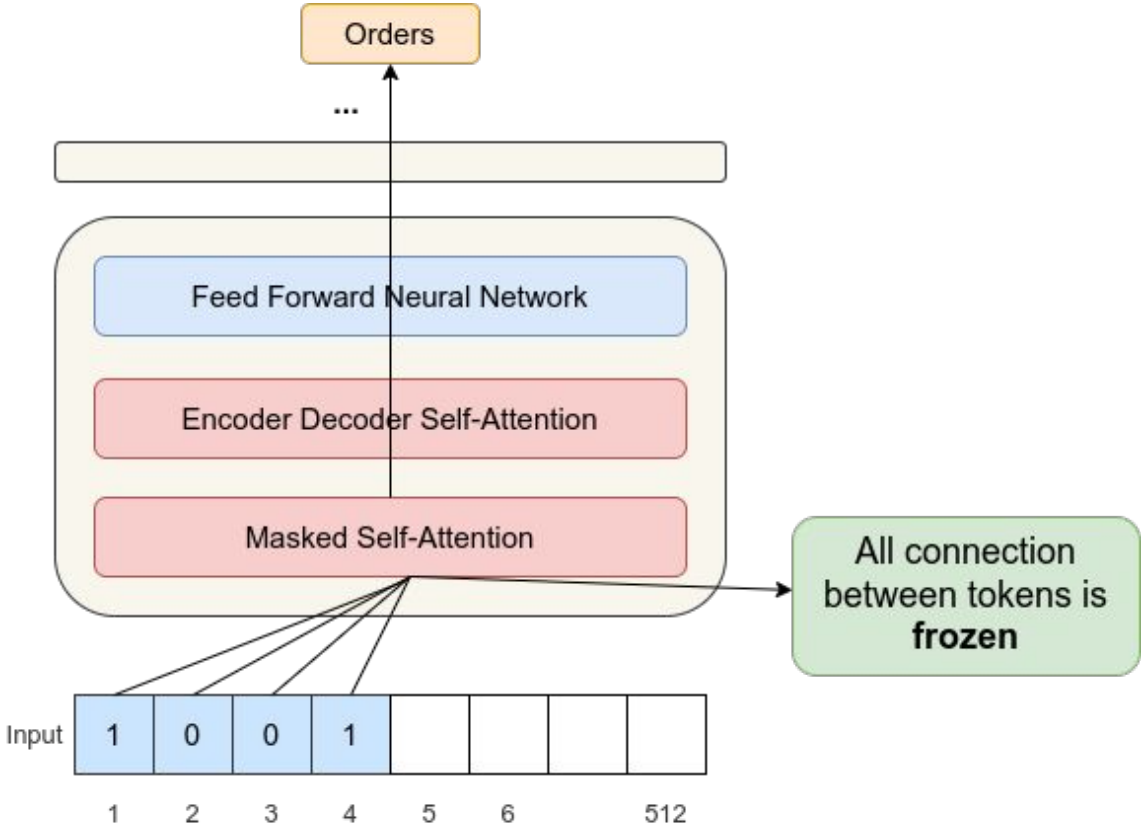
Architecture

Taking a transformer model pretrained on natural language data, GPT-2, and finetuning only the linear input and output layers, as well as the positional embeddings and layer norm parameters. Both self-attention and feedforward layers of the residual blocks are frozen.

This model is called a Frozen Pretrained Transformer(FPT).

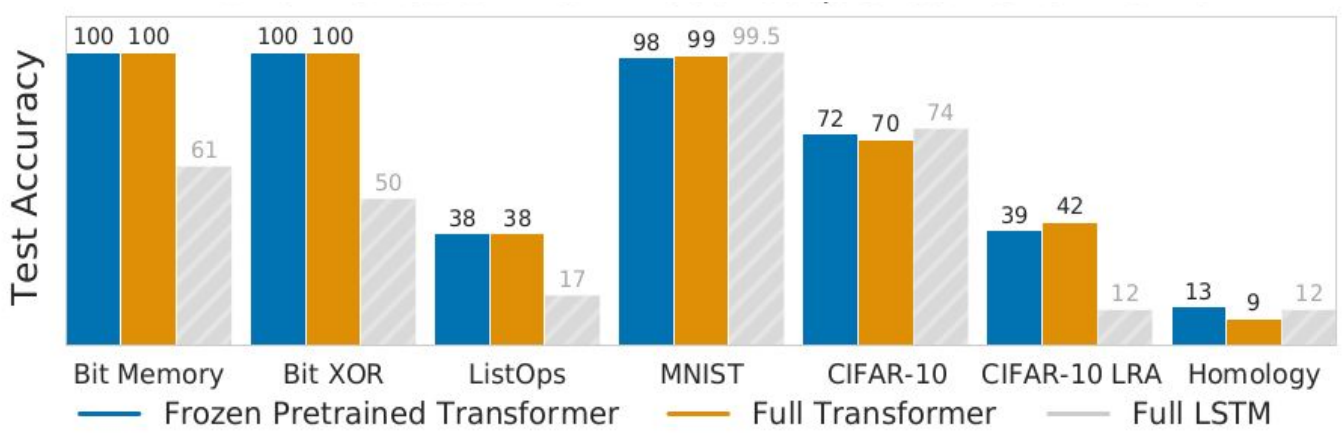


Frozen self-attention for arbitrary inputs



Performance on Multimodal Sequence Benchmarks

- FPT displays comparable performance to training the entire transformer or LSTM models, despite finetuning only 0.1% of the total number of parameters of the transformer model and none of the self-attention parameters.
- Because it is difficult to fully train a 12-layer transformer on small datasets, for CIFAR-10, the authors report the full transformer results for a 3-layer model.



Performance on Multimodal Sequence Benchmarks

- FPT displays comparable performance to training the entire transformer or LSTM models, despite finetuning only 0.1% of the total parameters of the pretrained transformer model and none of the self-attention parameters.
- Because it is difficult to fully train a 12-layer transformer on small datasets, for CIFAR-10, the authors report the full transformer results for a 2-layer model.

Frozen connections learned by language are as good as trained downstream

Empirical Evaluation

Can pretrained language models transfer to different modalities?

- Across all seven tasks considered, FPT achieves comparable, if not marginally better performance than fully training a transformer.
- These results support the idea that these models are learning representations and performing computation that is agnostic to the modality.
- Both transformer variants significantly outperform LSTMs on some tasks, particularly ListOps and CIFAR-10 LRA, which have long sequence lengths.

Model	Bit Memory	XOR	ListOps	MNIST	CIFAR-10	C10 LRA	Homology
FPT	100%	100%	38.4%	98.0%	72.1%	38.6%	12.7%
Full	100%	100%	38%	99.1%	70.3%	42%	9%
LSTM	60.9%	50.1%	17.1%	99.5%	73.6%	11.7%	12%

Conclusion

- The self-attention layers learned by a language model may have properties amenable to efficient universal computation.
- Worth investigating the use of other data-rich modalities (e.g., vision) or a hybrid of multiple domains being used to provide the necessary substrate for pretraining a universal computational engine.

Thanks!