Pretrained Transformers as Universal Computation Engines

Kevin Lu,[1,2] **Aditya Grover,**[2,3] **Pieter Abbeel,**[1] **Igor Mordatch**[4]

[1] UC Berkeley, [2] Facebook AI Research, [3] UCLA, [4] Google Brain
kzl@fb.com

Abstract

We investigate the capability of a transformer pretrained on natural language to generalize to other modalities with minimal finetuning – in particular, without finetuning of the self-attention and feedforward layers of the residual blocks. We consider such a model, which we call a Frozen Pretrained Transformer (FPT), and study finetuning it on a variety of sequence classification tasks spanning numerical computation, vision, and protein fold prediction. In contrast to prior works which investigate finetuning on the same modality as the pretraining dataset, we show that pretraining on natural language can improve performance and compute efficiency on non-language downstream tasks. Additionally, we perform an analysis of the architecture, comparing the performance of a random initialized transformer to a random LSTM. Combining the two insights, we find language-pretrained transformers can obtain strong performance on a variety of non-language tasks.

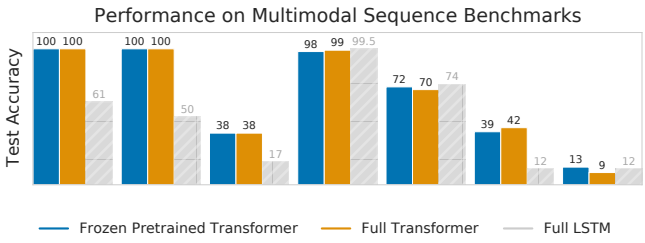

Figure 1: A *frozen* language-pretrained transformer (FPT) – without finetuning the self-attention and feedforward layers – can achieve strong performance compared to a transformer fully trained from scratch on a downstream *modality* on literature benchmarks (Tay et al. 2020; Rao et al. 2019). We show results on diverse classification tasks (see Section 2.1): numerical computation (Bit Memory/XOR, ListOps), image classification (MNIST, CIFAR-10, LRA), and protein fold prediction (Homology). We also show results for a fully-trained from-scratch LSTM as a baseline. Our code is available at: github.com/kzl/universal-computation

Copyright © 2022, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

1 Introduction

The transformer architecture (Vaswani et al. 2017) has shown broad successes in deep learning, serving as the backbone of large models for tasks such as modeling natural language (Brown et al. 2020), images (Dosovitskiy et al. 2020), proteins (Jumper et al. 2021), and multimodal tasks comprising of both images and text (Lu et al. 2019; Radford et al. 2021). Inspired by these successes, we seek to explore the generalization capabilities of a transformer in transferring from one modality to another.

Classical approaches to sequence processing used recurrent neural network (RNN) approaches (Rumelhart, Hinton, and Williams 1985; Hochreiter and Schmidhuber 1997). In contrast, transformers utilize self-attention layers to extract features across tokens of a sequence, such as words (Vaswani et al. 2017) or image patches (Dosovitskiy et al. 2020). Furthermore, it has become common practice to train large models on unsupervised objectives before 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: for example, train GPT (Radford et al. 2018) on a large language corpus, and finetune on a small task-specific dataset. Our goal in this work is to investigate finetuning on modalities distinct from the training modality.

We hypothesize that transformers – namely the self-attention layers – can be pretrained on a data-rich modality (i.e. where data is plentiful, such as a language corpus) and identify feature representations that are useful for *arbitrary* data sequences, enabling downstream transfer to different modalities. In particular, we seek to investigate what pretrained language models (LMs) are capable of in terms of generalizing to other modalities with sequential structure.

To investigate this hypothesis, we take a transformer model pretrained on natural language data, GPT-2 (Radford et al. 2019), and finetune only the linear input and output layers, as well as the positional embeddings and layer norm parameters. These decisions are made to highlight the parameters already in the language model, and not for performance purposes. We call this model a Frozen Pretrained Transformer (FPT). On a range of tasks across a variety of modalities – including numerical computation, image classification, and protein fold prediction – FPT displays comparable performance to training the entire transformer from

scratch, matching reported benchmarks for these tasks (Figure 1). Additionally, we find FPT models also converge faster during training. Our results suggest the self-attention layers learned by a language model may perform computation that is universal across modalities. Through a series of experiments, we investigate what contributes to cross-modal transfer by isolating sub-components of these models.

2 Methodology

2.1 Tasks

We evaluate on a diverse set of classification tasks representative of different modalities. In particular, we are interested in if language models are inherently capable of *universal computation*, by which we mean the ability to learn representations for predictive learning across diverse modalities.

Bit Memory. Similar to the task proposed by (Miconi, Stanley, and Clune 2018), we consider a bit memory task where the model is shown 5 bitstrings each of length 1000. Afterwards, the model is shown a masked version of one of the bitstrings, where each bit is masked with probability 0.5, and the model is tasked with producing the original bitstring. The bitstrings are broken up into sequences of length 50, so that the models are fed 120 tokens of dimension 50.

Bit XOR. Similar to the bit memory task, the model is shown 2 bitstrings of length 5, where the model must predict the element-wise XOR. The bitstrings are shown 1 bit at a time, so the models are fed 10 tokens of dimension 1.

ListOps. Taken from (Nangia and Bowman 2018; Tay et al. 2020), the model is shown a sequence of list operations (ex. `[ MAX 4 3 [ MIN 2 3 ] 1 0 ]`) and tasked with predicting the resulting output digit (ex. `4`). This task evaluates the ability of a model to parse mathematical expressions and evaluate over a long context. The model is shown 1 token at a time, so the models are fed 512 tokens of dimension 15.

MNIST. On MNIST, the model must classify a handwritten digit from a 32×32 black-and-white image. The tokens given to the model are 4×4 image patches, so the models are fed 64 tokens of dimension 16.

CIFAR-10. On CIFAR-10 (Krizhevsky et al. 2009), the model will be given 4×4 image patches, so the models are fed 64 tokens of dimension 16.

CIFAR-10 LRA. This is a modified version of the above task taken from the Long Range Arena (LRA) benchmark where the images are converted to grayscale and flattened with a token length of 1 (Tay et al. 2020). As a result, the input sequence consists of 1024 tokens of dimension 1. This task is more challenging than vanilla CIFAR-10 as the models must learn patterns over a significantly longer sequence length and have minimal spatial inductive bias.

Remote homology detection. In this task, we are interested in predicting the fold for a protein, represented as an amino acid sequence. We use the datasets provided by TAPE (Rao et al. 2019; Fox, Brenner, and Chandonia 2013; Hou, Adhikari, and Cheng 2018), where the train/test split is generated by holding out certain evolutionary groups. Note that we do not pretrain on Pfam (El-Gebali et al. 2019), which is common in other works. There are 20 common and 5 uncommon amino acids (25 different types of inputs), and there are 1195 possible labels to predict. We only consider sequences of length less than 1024 for simplicity. The models are thus fed up to 1024 tokens of dimension 25.

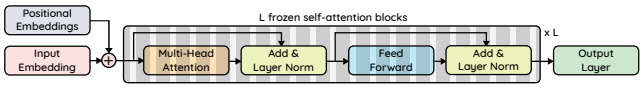

Figure 2: Frozen Pretrained Transformer (FPT). The self-attention & feedforward layers are frozen.

2.2 Architecture

The architecture we use is summarized in Figure 2. We seek to highlight the internal computation already present in the language model, and so freeze the main components of the model. Thus we consider finetuning the following parameters of a pretrained GPT-2 model (Radford et al. 2019):

- **Output layer:** it is crucial to finetune the output layer since we transfer to a completely new task. We use the simplest instantiation of an output network – a single linear layer applied to the last output token output by the transformer – in order to highlight that almost all the computation is being performed by the frozen transformer.
- **Input layer:** it is important to reinitialize a new input layer since we are reading in a new modality; in essence, we are learning how to query the transformer. This contrasts with prior unsupervised embedding evaluation techniques, such as linear probing – due to the change in modality, we should train the input layer as well, and evaluate if the frozen intermediate transformer model performs effective computation. Again, we use a linear layer to minimize the amount of computation outside the transformer.
- **Layer norm parameters:** as common in other finetuning works (Rebuffi, Bilen, and Vedaldi 2017; Houlsby et al. 2019), we also finetune the layer norm affine parameters, which adapt to the downstream task statistics.
- **Positional embeddings:** we generally also see a small benefit to finetuning the positional embeddings, which are similar to the input layer parameters.

Given the cheap linear scaling of these parameters, the parameter counts of large transformer models are dominated by the self-attention and feedforward layers, which grow quadratically with the sequence length and layer width. For the base CIFAR-10 model with 124M parameters, these come out to approximately 0.086% of the network. We further ablate the importance of each parameter in Section 3.11.

Note that, crucially, all communication between tokens in the model are frozen. The data in each datapoint is chunked into discrete tokens (bits, image patches, amino acids, etc.), and can only reference each other via the frozen attention connections, which are not trained; additionally, neither the output nor the input layers are connected to multiple tokens. Our key investigation is to analyze the computation that is already inherent in the language model, and hence we do a minimal amount of computation that is learned on the downstream 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%

Table 1: Investigation of pretraining modality. 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.

3 Empirical Evaluations

In this section, we review the results demonstrating transfer from language to other modalities, and seek to better understand why this occurs and what enables this transfer. All model sizes are the base model size (12 layers, 768 hidden dimension), unless stated otherwise; due to the number of experiments, we use one seed for each reported accuracy.

3.1 Can Pretrained Language Models Transfer to Different Modalities?

We investigate if the self-attention and feedforward layers – the main body – of a pretrained transformer can be applied to a classification problem in a different modality without finetuning. To do this, we apply our base procedure as described above, where the input embedding layer, output readout layer, and layer norm parameters are finetuned.

We minimally tune the FPT models, using the standard pretrained GPT-2 and default PyTorch learning rate of 10^{-3}. We compare to fully training a transformer from scratch, without pretraining, using the same learning rate and batch size; we swept over layer sizes of 3 or 12, as some of the fully trained models benefited from smaller size due to optimization challenges. Additionally, we compare to state-of-the-art from literature when available (full transformer on ListOps, CIFAR-10 LRA, and Remote Homology; LSTM on Remote Homology), as these works performed more in-depth sweeps on hyperparameters.

Our results are shown in Figure 1. We find that across all seven tasks considered, FPT achieves comparable performance to the fully trained transformer benchmark results. We believe these results support the idea that these models are learning representations and performing computation that is agnostic to the modality. We also note that both transformer variants significantly outperform LSTMs on some tasks, particularly ListOps and CIFAR-10 LRA, which have long sequence lengths of 512 and 1024, respectively.

Furthermore, unlike some other works utilizing transformers for vision, we use minimal spatial bias to emphasize the universal sequential aspect of the problem – for instance, we do not interleave self-attention and convolution layers. Note that we also do not use 2D positional embeddings (or other domain-specific techniques), hence providing very weak inductive prior to the model. Our reasoning for these decisions is to evaluate the ability of transformers to work on arbitrary sequential tasks.

3.2 What is the Importance of the Pretraining Modality?

We now compare pretraining on language to other pretraining methods for base model sizes:

- **Random initialization (Random):** initialization of the frozen transformer parameters randomly using the default initialization choices for GPT-2, i.e. without pretraining.
- **Bit memory pretraining (Bit):** pretraining from scratch on the Bit Memory task and then freezing the parameters before transferring. This allows the transformer to gain supervision working with arbitrary bit strings and performing memory/denoising on independent inputs.
- **Image pretraining (ViT):** Vision Transformer (Dosovitskiy et al. 2020) pretrained on ImageNet-21k (Deng et al. 2009). Note ViT does not have an autoregressive mask like GPT-2; however, we found FPT with a BERT (Devlin et al. 2018) backbone performs very similarly to GPT-2, so this distinction is not very important.

These experiments highlight the significance of pretraining – as opposed to simply the transformer architecture – and compare language to other methods of supervision. Our results are shown in Table 1. Although the random transformers can achieve surprisingly strong accuracies, there is a considerable gap to using natural language pretraining, such as in MNIST, where random transformers achieve similar performance to a linear classifier on top of raw features (92%). Thus we believe that while the transformer architecture might be naturally conducive to these evaluations, the attention mechanisms used to transfer may be nontrivial and not fully specified by the architecture. We also find that, in addition to performance benefits, language pretraining improves convergence compared to the randomly initialized transformer (see Section 3.4).

Pretraining on bit memory improves performance compared to the random models, but still lags behind training on natural language data. Furthermore, measured by gradient steps, all models converge faster than the randomly initialized transformers (more details in Section 3.4), indicating that all modes of pretraining improve upon random initialization even without considering accuracy.

Additionally, while freezing a vision transformer yields better improvements on CIFAR-10, pretraining on images is not uniformly better; e.g., ViT is worse on protein classification. One hypothesis is that protein sequences are structured like language, in terms of discrete units of information with a "grammar" (Saar et al. 2021), so transfer from language to proteins may be more natural.

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%

Table 2: Test accuracy of frozen randomly initialized transformers vs frozen randomly initialized LSTM models. Frozen LSTMs perform very poorly. LSTM* adds additional architecture improvements to match the transformers (see "+ Pos" in Table 4).

3.3 How Important is the Transformer vs LSTM Architecture?

In Section 3.2 we found the transformer architecture can already be fairly effective in this regime, even with only random parameters. In this section, we consider using a random LSTM architecture instead of the transformer, allowing us to consider the raw effect of architecture and ablating pretraining. Like FPT, we finetune the input, output, and layernorm parameters for the LSTMs.

Our results are shown in Table 2. "LSTM" refers to a 3-layer "standard" LSTM with a hidden dimension of 768, matching standard implementations of LSTMs, without residual connections or positional embeddings. We include this comparison to represent the traditional method more faithfully, but add these additional architectural components below. This matches the width of the FPT models, but not the depth or total parameter count (note that LSTMs also do not have positional embeddings). In the same style of FPT and GPT-2, we do not use a bidirectional LSTM. We find that the self-attention architecture already serves as an effective inductive bias for universal computation, improving significantly over the recurrent LSTM model and comprising most of the improvement in test accuracy from random LSTM to FPT.

To better isolate the contribution of attention, we add additional features to standard LSTMs. We compare the 3-layer "standard" LSTM to a 12-layer "standard" LSTM, matching the depth of the FPT networks. Under these model choices, we report the performance of a frozen random 3-layer vs 12-layer LSTM in Table 3. Naively, the 12-layer model is much worse than the 3-layer model, hinting that there is some loss of information by repeated LSTM layers.

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

Table 3: Test accuracy of frozen randomly initialized "standard" LSTMs with a hidden dimension of 768. The 12-layer LSTM achieves only near-trivial performance.

We also experiment with ablating other architectural improvements included with the transformer architecture in Table 4. Once residual connections (He et al. 2016) are added, the 12-layer LSTM makes up a lot of the performance drops, hinting that residual connections could make up for loss of information from the LSTM layers which otherwise linearly combine the features. We also add positional embeddings, which finishes bridging the gap between standard LSTM implementations and the transformer. Even with these additional benefits, the LSTM still performs worse. Note that the final 12-layer LSTM has about the same number of trainable parameters as the transformer.

Model	ListOps	MNIST	CIFAR-10	C10 LRA
LSTM	16.2%	11.7%	10.8%	10.4%
+ Skip	16.8%	70.9%	34.4%	10.4%
+ Pos	16.7%	92.5%	43.5%	10.6%
Trans.	34.3%	91.7%	61.7%	36.1%

Table 4: Test accuracy of 12-layer frozen randomly initialized LSTMs with architecture modifications to match transformers: residual connections (Skip) and positional embeddings (Pos). LSTM is worse in all settings.

3.4 Does Language Pretraining Improve Compute Efficiency Over Random Initialization?

We investigate compute efficiency by considering the number of gradient steps to converge for FPT vs random transformer models, shown in Table 5. We generally find FPT converges faster, which indicates language pretraining can yield compute benefits for non-language tasks. While random transformer models achieve decent test accuracies, in particular when compared to random LSTMs, there is still a considerable gap in the compute efficiency compared to using pretraining. Note that bit memory pretraining introduced in Section 3.2 generally falls between the two models, and notably is $6\times$ slower than FPT on Bit XOR, which is significantly better than random.

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

Table 5: Approximate gradient steps until convergence for pretrained (FPT) vs randomly initialized (Random) models. Language pretraining converges faster in terms of gradient steps (also translates to wall-clock time).

3.5 Do the Frozen Attention Layers Attend to Modality-Specific Tokens?

We investigate if FPT attends to semantically meaningful patterns in the data. We plot the attention weights (i.e. the values of the softmax of query-key dot product) from the first layer. We show the results in Figures 3 and 4 for the bit tasks. Note GPT-2 is autoregressive, so the upper right corner of the attention mask is zeroed out. On these tasks, FPT yields an interpretable attention pattern despite not training the self-attention layers themselves. We did not find easily interpretable patterns on the other tasks.

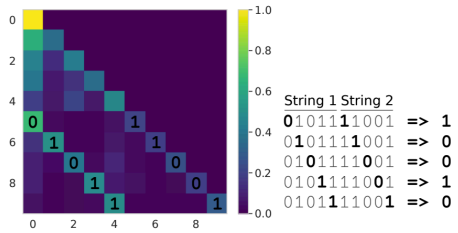

Figure 3: On Bit XOR, the model must produce the element-wise XOR of two bitstrings presented sequentially (bits 0-4 are the first bitstring, 5-9 are the second). FPT attends positionally to the two bits that are XOR'ed by the output token.

Figure 4: On Bit Memory, the model must return one of five strings (inputs 0-99) given a masked version of one of the strings (inputs 100-119). Each token is 50 bits. FPT learns to attend to the correct string based on finding similarity to the inputs, not relying solely on position as in Bit XOR.

We also include the attention map for Bit XOR using a randomly initialized transformer (which also solves the task) in Figure 5. This model also learns to exploit the diagonal pattern, although the strength is a little weaker. This indicates that while the random transformer still learns to solve the task, it learns a less semantically interpretable/strong attention pattern.

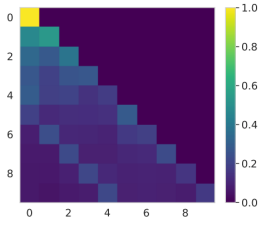

Figure 5: A transformer with frozen randomly initialized self-attention layers also learns to correlate the two diagonal elements on Bit XOR, although the magnitude of the diagonals is lower.

3.6 Does Performance Scale With Model Size?

We evaluate the efficacy of adding more parameters to these models on CIFAR-10. Most of the additional parameters are in the transformer layers and are trained during the natural language pretraining phase. Our results for pretrained and random models are in Table 6. Unlike fully training a transformer, which exhibits more overfitting and divergence during training with larger models, increasing model size stably increases the capacity of the models. This result indicates our observations and results are likely to scale as we move towards larger models and higher-data regimes.

Model Size	Trained Params	FPT	Random
Small (Base)	106K	68.2%	61.7%
Medium	190K	69.8%	64.0%
Large	300K	72.1%	65.7%

Table 6: CIFAR-10 test accuracy of larger FPT models.

3.7 Can Performance Be Attributed Simply to Better Statistics for Initialization?

In this section, we ablate taking the layer-wise mean and standard deviation from the pretrained model and using it to initialize a random transformer, in order to ablate if a better initialization scheme via an "oracle" standard deviation can recover the performance of FPT. Note that the GPT-2 initialization scheme initializes parameters as Gaussian; traditionally, the standard deviation is 0.02 by default.

We show the results using this initialization scheme in Table 7 (note that all of the weights, biases, layer norm, and positional embeddings are initialized – both mean and variance – in this fashion). This yields better results on most tasks (more in Supplementary Material), but does poorly on CIFAR-10. As a result, we believe the benefits of language pretraining cannot be recovered with a simple better initialization scheme, although future work in transformer initialization could yield different results.

Init	Memory	ListOps	C10	C10 LRA
FPT	100%	38.4%	68.2%	38.6%
Stats	100%	37.4%	56.5%	33.1%
Default	75.8%	34.3%	61.7%	36.1%

Table 7: Test accuracy when initializing parameters with pretrained weights (FPT) vs randomly initializing according to the mean and variance of the pretrained transformer (Stats) vs defaul random initialization (Default).

3.8 Can We Train a Transformer by Only Finetuning the Output Layer?

We consider using FPT solely for naive feature extraction for linear classification, where we fix a randomly initialized input layer and freeze all parts of the model except for the output. Note that this resembles resevoir computing/echo state

networks (see Section 4 for discussion). The model evaluates on every example in the training set once, caches the features, and then we train a linear output layer. This enables subsequent epochs after the first to run extremely quickly, but does not easily handle dropout/data augmentations, and scales well in terms of number of epochs, but not in dataset size. Note that this is mathematically equivalent to linear classification. Our results are shown in Table 8. Although we find speedups extremely significant and they obtain nontrivial performance, performance significantly degrades and the models also exhibit overfitting (likely due to lack of regularization; unlike the training of FPT, dropout is not applied).

Task	Speedup	Output Only	FPT
ListOps	500×	32.8%	38.4%
CIFAR-10 LRA	500×	24.7%	38.6%

Table 8: Training only the output layer as a linear regression problem. Speedup refers to wall clock time per epoch.

3.9 How Does Model Depth Affect Token Mixing?

One interesting question is the importance of the depth of the transformer for generating represention which "mix" tokens: for instance, if there is only one layer and the parameters are random, it is unlikely for the tokens to be mixed well, whereas if there are many layers, there are many chances for the tokens to mix and form interesting representations useful for downstream tasks. We investigate this on ListOps by considering pretrained vs random models, where we only take the first X layers of the 12-layer pretrained model (i.e. for X=3, we use the first 3 layers of the pretrained GPT-2 model and perform classification from those hidden states). Additionally, to maximally highlight the importance of the pretrained layers, we randomly initialize the input layer, and do not train the input or positional parameters.

With finetuning layernorm. We first investigate this question with finetuning the layernorm parameters (i.e. we finetune only the output layer and the layernorm parameters). Results are shown in Table 9. Both models are unable to do well with only one layer, but the pretrained model performs significantly better than the random model at 2 layers, indicating that while the difference in performance at 12 layers is relatively small, there is a great benefit to using pretrained layers for when considering a small number of layers in that the tokens are "mixed" faster.

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

Table 9: Test accuracy on Listops while varying model depth and finetuning layernorm parameters. Pretrained layers "mix" tokens faster, performing better at low depths.

Without finetuning layernorm. We now investigate this question without finetuning the layernorm parameters, and only finetuning the output parameters, as in the reservoir computing setup in Section 3.8. Note this is equivalent to linear classification. This setting is the most challenging since all processing that is able to mix tokens is done by either random or pretrained parameters, and we are only able to train a linear layer on top of the output of the last token; as a result, the *only* token mixing that is done is performed entirely by the pretrained self-attention layers. Results are shown in Table 10. The random model is poor even for a large number of layers, while the pretrained model can still do reasonably well, even though it requires more layers than before.

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

Table 10: Test accuracy on Listops while varying model depth and only training output parameters. Even for a large number of layers, the random model performs poorly.

3.10 Can Training More Parameters Improve Performance?

Our focus in this work was primarily to investigate if and how efficient, general-purpose pretraining can transfer across modalities. However, for practical applications, it would naturally be better to choose a more specialized finetuning scheme or add more trainable parameters, to try and maximize performance.

On CIFAR-10, we experiment with additionally finetuning the last attention and all the feedforward layers, shown in Table 11. Generally these naive approaches can yield better performance, we are optimistic about the possibility of smarter universal training methods *improving* performance in future work, such as with adapters (Houlsby et al. 2019).

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

Table 11: Test accuracy on CIFAR-10 when finetuning additional parameters. In addition to FPT, if we finetune the feedforward layers and the last self-attention layer, we can achieve 80% accuracy.

3.11 Which Parameters of the Model Are Important To Finetune?

We now run ablations for only finetuning select parameters of the pretrained GPT-2 to see which parameters are most sensitive. Our results are in Table 12; full results are in the Supplementary Material. Similar to a study of random CNNs

by (Frankle, Schwab, and Morcos 2020), we generally find the layer norm parameters to be most important.

Task	output only	+ LN	+ input	+ pos
Memory	76%	94%	100%	100%
XOR	56%	98%	98%	100%
ListOps	15%	36%	36%	38%
MNIST	23%	96%	98%	98%
CIFAR-10	25%	54%	60%	68%
C10 LRA	17%	39%	39%	39%
Homology	2%	9%	10%	13%

Table 12: Ablation by successively adding certain parameters to the list of finetuned parameters.

4 Related Work and Discussion

Multimodal transformers. Transformers (Vaswani et al. 2017) were first used successfully for natural language processing (Radford et al. 2018; Devlin et al. 2018; Radford et al. 2019; Brown et al. 2020). In recent years, they have also been shown to be effective architectures for other modalities. Work specifically tackling multimodal tasks include (Kaiser et al. 2017), who showed a single model could learn a variety of multimodal tasks with an attention architecture. ViLBERT (Lu et al. 2019) and CLIP (Radford et al. 2021) jointly learn over images and text, embedding each modality with distinct transformers. Our work is most similar to DALL-E (Ramesh et al. 2021), which uses a single transformer to embed both the image and text modalities, which we consider to be generating a "universal latent space" that projects any type of input into a single latent space. Such a latent space would be useful for a model that could learn from many sources of supervision.

Transformers in transfer settings. There are also many works looking at transformers specifically in the context of in-modality transfer, such as ViT for vision (Dosovitskiy et al. 2020), T5 for language (Raffel et al. 2019), and UDSMProt for protein sequences (Strodthoff et al. 2020). (Hernandez et al. 2021) do a thorough investigation of transfer with language pretraining, notably showing transfer from English to Python, which they consider to be reasonably distanced from English; many works have also looked at transferring from one langauge to another (Artetxe, Ruder, and Yogatama 2019; Ponti et al. 2019). Similar to our work, (Papadimitriou and Jurafsky 2020) investigate transfer for LSTMs between modalities including code, different languages, and music, finding that pretraining on "non-linguistic data with latent structure" can transfer to language, finding grammatical structure in a modality to be important, although we generally investigate the other direction, explore more distanced modalities, and highlight the transformer architecture.

Global workspace theory. Linear probing is a common technique for evaluating the embeddings learned by an unsupervised model (Donahue, Krähenbühl, and Darrell 2016; Oord, Li, and Vinyals 2018; Chen et al. 2020), which is reasonable when you finetune on the same modality as the pretrained one. However, when finetuning on a different modality, as in our setting, we have to reframe this notion of generalizable embedding quality – instead of only finetuning the output layer, we also want to finetune the input layer, and instead evaluate the ability of the frozen intermediate model to perform generalizable *computation*. This is reminiscent of Global Workspace Theory (Baars 1993), which revolves around the notion that there is a "blackboard" that different parts of the brain send data to; we might consider the frozen language model as being a blackboard in this setting. Language might also be a natural choice of model for this blackboard, as there are hypotheses that language may serve as a good multipurpose high-level representation for cognitive behavior and conscious planning (Andreas, Klein, and Levine 2017; Goyal and Bengio 2020).

Reservoir computing. Similarly to the FPT setup and Global Workspace Theory, in reservoir computing (Tanaka et al. 2019) and echo state networks (Jaeger 2001; Jaeger and Haas 2004), a random recurrent network is frozen and only the output readout layer is trained. These models are very fast to train, using a similar setup as in Section 3.8, because the activations of the recurrent network can be cached and it is unnecessary to backpropagate over time. Somewhat differently to the FPT architecture, echo state networks are recurrent and thus feed back into themselves, which allows the outputs of the random frozen network to modulate future inputs. Unlike echo state networks, we also notably finetune the input and positional embeddings, which allow the inputs to the frozen network to adapt to a particular modality/for a query to the frozen network to be learned. Echo state networks are also similar to the perspective of self-attention applying a data-dependent filter to the inputs, as opposed to 1D convolutions, which are fixed filters regardless of inputs.

5 Conclusion

We proposed transferring a pretrained transformer language model for downstream tasks in non-language modalities. We showed empirically that these models could achieve performance competitive with transformers fully trained on the downstream task, relying solely on frozen parameters from the language model to perform the bulk of the computation. We believe this work can serve as the foundation for future work investigating transfer between modalities, and training regimes for multimodal universal models.

We note a limitation of our analysis is that we analyze specific models on a restricted set of tasks. More investigation can highlight if similar behavior occurs for other models on other tasks. As training regimes for these models evolve, performing similar experiments may yield different results, and we are excited for more research in this direction.

For high stakes applications in the real-world, there are potential concerns with transfer of harmful biases from one modality to one another using pretrained transformer models (Sheng et al. 2019; Bender et al. 2021). Mitigating these biases is an active area of research (Grover et al. 2019; Choi et al. 2020). Conversely, there are also potential upsides with FPT models being able to better exploit representative datasets from one or more modalities, which merit future investigation as well.

Acknowledgments

We would like to thank Luke Metz, Kimin Lee, Fangchen Liu, Roshan Rao, Aravind Srinivas, Nikita Kitaev, Daniel Freeman, Marc'Aurelio Ranzato, Jacob Andreas, and Ashish Vaswani for valuable feedback and discussions.

References

Andreas, J.; Klein, D.; and Levine, S. 2017. Learning with latent language. *arXiv preprint arXiv:1711.00482*.

Artetxe, M.; Ruder, S.; and Yogatama, D. 2019. On the cross-lingual transferability of monolingual representations. *arXiv preprint arXiv:1910.11856*.

Baars, B. J. 1993. *A cognitive theory of consciousness*. Cambridge University Press.

Bender, E. M.; Gebru, T.; McMillan-Major, A.; and Shmitchell, S. 2021. On the dangers of stochastic parrots: Can language models be too big. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency; Association for Computing Machinery: New York, NY, USA*.

Brown, T. B.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

Chen, T.; Kornblith, S.; Norouzi, M.; and Hinton, G. 2020. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, 1597–1607. PMLR.

Choi, K.; Grover, A.; Singh, T.; Shu, R.; and Ermon, S. 2020. Fair generative modeling via weak supervision. In *International Conference on Machine Learning*, 1887–1898. PMLR.

Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K.; and Fei-Fei, L. 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, 248–255. Ieee.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Donahue, J.; Krähenbühl, P.; and Darrell, T. 2016. Adversarial feature learning. *arXiv preprint arXiv:1605.09782*.

Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; et al. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.

El-Gebali, S.; Mistry, J.; Bateman, A.; Eddy, S. R.; Luciani, A.; Potter, S. C.; Qureshi, M.; Richardson, L. J.; Salazar, G. A.; Smart, A.; Sonnhammer, E. L. L.; Hirsh, L.; Paladin, L.; Piovesan, D.; Tosatto, S. C. E.; and Finn, R. D. 2019. The Pfam protein families database in 2019. *Nucleic Acids Research*, 47(D1): D427–D432.

Fox, N. K.; Brenner, S. E.; and Chandonia, J.-M. 2013. SCOPe: Structural Classification of Proteins—extended, integrating SCOP and ASTRAL data and classification of new structures. *Nucleic acids research*, 42(D1): D304–D309.

Frankle, J.; Schwab, D. J.; and Morcos, A. S. 2020. Training batchnorm and only batchnorm: On the expressive power of random features in cnns. *arXiv preprint arXiv:2003.00152*.

Goyal, A.; and Bengio, Y. 2020. Inductive Biases for Deep Learning of Higher-Level Cognition. *arXiv preprint arXiv:2011.15091*.

Grover, A.; Song, J.; Agarwal, A.; Tran, K.; Kapoor, A.; Horvitz, E.; and Ermon, S. 2019. Bias correction of learned generative models using likelihood-free importance weighting. *arXiv preprint arXiv:1906.09531*.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

Hernandez, D.; Kaplan, J.; Henighan, T.; and McCandlish, S. 2021. Scaling Laws for Transfer. *arXiv preprint arXiv:2102.01293*.

Hochreiter, S.; and Schmidhuber, J. 1997. Long short-term memory. *Neural computation*, 9(8): 1735–1780.

Hou, J.; Adhikari, B.; and Cheng, J. 2018. DeepSF: deep convolutional neural network for mapping protein sequences to folds. *Bioinformatics*, 34(8): 1295–1303.

Houlsby, N.; Giurgiu, A.; Jastrzebski, S.; Morrone, B.; De Laroussilhe, Q.; Gesmundo, A.; Attariyan, M.; and Gelly, S. 2019. Parameter-efficient transfer learning for NLP. In *International Conference on Machine Learning*, 2790–2799. PMLR.

Jaeger, H. 2001. The "echo state" approach to analysing and training recurrent neural networks-with an erratum note. *Bonn, Germany: German National Research Center for Information Technology GMD Technical Report*, 148(34): 13.

Jaeger, H.; and Haas, H. 2004. Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication. *science*, 304(5667): 78–80.

Jumper, J.; Evans, R.; Pritzel, A.; Green, T.; Figurnov, M.; Tunyasuvunakool, K.; Ronneberger, O.; Bates, R.; Žídek, A.; Bridgland, A.; Meyer, C.; Kohl, S. A. A.; Potapenko, A.; Ballard, A. J.; Cowie, A.; Romera-Paredes, B.; Nikolov, S.; Jain, R.; Adler, J.; Back, T.; Petersen, S.; Reiman, D.; Steinegger, M.; Pacholska, M.; Silver, D.; Vinyals, O.; Senior, A. W.; Kavukcuoglu, K.; Kohli, P.; and Hassabis, D. 2021. High Accuracy Protein Structure Prediction Using Deep Learning. *Nature*.

Kaiser, L.; Gomez, A. N.; Shazeer, N.; Vaswani, A.; Parmar, N.; Jones, L.; and Uszkoreit, J. 2017. One model to learn them all. *arXiv preprint arXiv:1706.05137*.

Krizhevsky, A.; et al. 2009. Learning multiple layers of features from tiny images. *University of Toronto*.

Lu, J.; Batra, D.; Parikh, D.; and Lee, S. 2019. Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. *arXiv preprint arXiv:1908.02265*.

Miconi, T.; Stanley, K.; and Clune, J. 2018. Differentiable plasticity: training plastic neural networks with backpropagation. In *International Conference on Machine Learning*, 3559–3568. PMLR.

Nangia, N.; and Bowman, S. R. 2018. Listops: A diagnostic dataset for latent tree learning. *arXiv preprint arXiv:1804.06028*.

Oord, A. v. d.; Li, Y.; and Vinyals, O. 2018. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*.

Papadimitriou, I.; and Jurafsky, D. 2020. Pretraining on non-linguistic structure as a tool for analyzing learning bias in language models. *arXiv preprint arXiv:2004.14601*.

Ponti, E. M.; Vulić, I.; Cotterell, R.; Reichart, R.; and Korhonen, A. 2019. Towards zero-shot language modeling. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2893–2903.

Radford, A.; Kim, J. W.; Hallacy, C.; Ramesh, A.; Goh, G.; Agarwal, S.; Sastry, G.; Askell, A.; Mishkin, P.; Clark, J.; et al. 2021. Learning transferable visual models from natural language supervision. *Image*, 2: T2.

Radford, A.; Narasimhan, K.; Salimans, T.; and Sutskever, I. 2018. Improving language understanding by generative pre-training. *OpenAI*.

Radford, A.; Wu, J.; Child, R.; Luan, D.; Amodei, D.; and Sutskever, I. 2019. Language models are unsupervised multitask learners. *OpenAI*.

Raffel, C.; Shazeer, N.; Roberts, A.; Lee, K.; Narang, S.; Matena, M.; Zhou, Y.; Li, W.; and Liu, P. J. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*.

Ramesh, A.; Pavolv, M.; Goh, G.; Gray, S.; Chen, M.; Child, R.; Misra, V.; Mishkin, P.; Krueger, G.; Agarwal, S.; and Sutskever, I. 2021. DALL·E: Creating Images from Text. *OpenAI*.

Rao, R.; Bhattacharya, N.; Thomas, N.; Duan, Y.; Chen, X.; Canny, J.; Abbeel, P.; and Song, Y. S. 2019. Evaluating Protein Transfer Learning with TAPE. In *Advances in Neural Information Processing Systems*.

Rebuffi, S.-A.; Bilen, H.; and Vedaldi, A. 2017. Learning multiple visual domains with residual adapters. *arXiv preprint arXiv:1705.08045*.

Rumelhart, D. E.; Hinton, G. E.; and Williams, R. J. 1985. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science.

Saar, K. L.; Morgunov, A. S.; Qi, R.; Arter, W. E.; Krainer, G.; Lee, A. A.; and Knowles, T. P. J. 2021. Learning the molecular grammar of protein condensates from sequence determinants and embeddings. *Proceedings of the National Academy of Sciences*, 118(15).

Sheng, E.; Chang, K.-W.; Natarajan, P.; and Peng, N. 2019. The woman worked as a babysitter: On biases in language generation. *arXiv preprint arXiv:1909.01326*.

Strodthoff, N.; Wagner, P.; Wenzel, M.; and Samek, W. 2020. UDSMProt: universal deep sequence models for protein classification. *Bioinformatics*, 36(8): 2401–2409.

Tanaka, G.; Yamane, T.; Héroux, J. B.; Nakane, R.; Kanazawa, N.; Takeda, S.; Numata, H.; Nakano, D.; and Hirose, A. 2019. Recent advances in physical reservoir computing: A review. *Neural Networks*, 115: 100–123.

Tay, Y.; Dehghani, M.; Abnar, S.; Shen, Y.; Bahri, D.; Pham, P.; Rao, J.; Yang, L.; Ruder, S.; and Metzler, D. 2020. Long Range Arena: A Benchmark for Efficient Transformers. *arXiv preprint arXiv:2011.04006*.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L.; and Polosukhin, I. 2017. Attention is all you need. *arXiv preprint arXiv:1706.03762*.