WaveletGPT: Wavelets Meet Large Language Models

Anonymous authors
Paper under double-blind review

Abstract

Large Language Models (LLMs) have ushered in a new wave of artificial intelligence advancements impacting every scientific field and discipline. They are trained on a simple objective: to predict the next token given the previous context. We live in a world where most of the data around us, e.g., text, audio, and music, has a multi-scale structure associated with them. This paper infuses LLMs with traditional signal processing ideas, namely wavelets, during pre-training to take advantage of the structure. Without adding **any extra parameters** to a GPT-style LLM architecture, we achieve the same pre-training performance almost twice as fast for LLMs in text, raw audio, and symbolic music by imposing a structure on intermediate embeddings. When trained for the same number of training steps, we achieve significant gains in performance, which is comparable to pre-training a much larger neural architecture. Our architecture allows every next token prediction to have access to intermediate embeddings at different temporal resolution in every Transformer decoder layer. This work will hopefully pave the way for incorporating multi-rate signal processing ideas into traditional large language model pre-training. Further, we showcase pushing model performance by improving internal structure as opposed to just going after scale. [1]

1 Introduction and Related Work

Large Language Models (LLMs) have ushered in a super-renaissance of AI models and are touching every scientific and engineering discipline. At the heart of this revolution has been the Transformer architecture Vaswani et al. (2017), which was initially proposed for machine translation in natural language processing. Transformer architecture became the backbone of GPT (Generative Pretrained Transformer) language models Brown et (2020) first proposed by Open-AI, which has revolutionised the field. Modern LLMs are still trained on a straightforward objective: To predict the next token given the previous context, preserving the causality assumption. The exact recipe has been show to work not only for language but also for robotics Brohan et al. (2022; 2023), protein sequences Madani et al. (2020), raw audio waveformsVerma & Chafe (2021), acoustic and music tokens Huang et al. (2018); Verma & Smith (2020); Borsos et al. (2023), videos Yan et al. (2021) to name a few. This simple recipe of tokenization/creating an embedding and feeding it to transformers also has given rise to architectures in non-causal setups such as BERTDevlin et al. (2018), Vision Transformers Dosovitskiy et. al (2020), Audio Transformers Verma & Berger (2021) and Video Transformers Selva et al. (2023). The recent surge in multi-modal large language models similar to that proposed by Google with its Gemini family Team et al. (2023) would pave the way for another wave of applications in the future. With increased performance by scale, some of the models like GPT-3 are Brown et (2020) reaching hundreds of billions of parameters to that of Google's Switch Transformer has even reached trillion parameters Fedus et al. (2022). This has led to recent concerns that AI research is slowly moving out of academics and is getting confined to industry researchers as per the recent Washington Post article Nix (2024)

The theme for this work is to push the capabilities of the models to get capabilities of a much bigger architecture or achieve the same performance with faster convergence. Researchers have proposed several techniques to boost the performance of smaller architectures using larger models. From a few that are listed now, however our work is different that we propose to improve the performance during pre-training. One of the most popular ones is knowledge distillation, where a larger model is used to guide a smaller architecture

---

[1]* This work was carried out while XXXX was with the XXXX XXXX in the XXXX XXXX XXXX at XXXX XXXX

in terms of the number of parameters. Gu et al. (2024) used KL divergence-based criteria to improve the capabilities of generating text (next token prediction) from the teacher model's feedback. This still uses a powerful model to improve its performance rather than improve the smaller architecture trained from scratch. A line of work also proposed hierarchical transformers via upsampling-downsampling operation Nawrot et al. (2021) similar to an hour-glass U-Net architecture in computer vision Long et al. (2015). As compared with the Transformer baseline, given the same amount of computation, it can yield the same results as Transformers more efficiently. Our work has similarities and stark differences to Clockwork-RNN Koutnik et al. (2014). First proposed for improving long context modeling in RNNs, it splits the hidden neurons of an RNN into different modules, and each module, having different parameters, updates their states at different (clock) rates. Thus, at each time step, only a few modules(weights) are activated and updated during forward/backward pass. This allows the network to learn dependencies through processing and retaining long-term information at different rates from the high-speed and low-speed modules. Our architecture only tinkers with the intermediate embeddings with straightforward tweaks and does not introduce complex separate learning modules or update weights at different rates. Model pruning Sun et al. (2023), on the other hand removes weights based upon its saliency in impacting the performance to achieve the same performance of the large architecture, like LLAMA Touvron et al. (2023) with fewer compute flops, during inference. Again, the goal is to start with a trained large model at the outset rather than trying to achieve the same pretraining performance from scratch. We also do not discuss quantization-based algorithms Dettmers et al. (2024), as they are also focused on improving inference times/flops, or to fine-tune an pre-existing architecture.

The other line of work, similar to our work, is to tinker with the intermediate embeddings. Tamkin et. al (2020) proposed hand-tuned filters on the Discrete Cosine Transform Ahmed et al. (1974) of the latent space for different tasks like named entity recognition and topic modeling for non-causal architectures like BERT Devlin et al. (2018). However, they take a discrete cosine transform over the entire context length and thus cannot be adapted for applications such as language modeling, which predicts the next token in a given context. There have been similar papers on applying ideas from signal processing like methods to BERT like non-causal architectures and we will discuss two of them here FNet and WavSPA that are relevant for our current paper, both again proposed for BERT like architectures. Both of the papers present variants of improving attention blocks which is different than our work which is on causal decoder only architectures such as GPT. FNetLee-Thorp et al. (2022) removes the costly attention mechanism and replaces it with 2-D FFT block. This operation is however non-causal as it looks into future tokens for computing 2-D FFT for modifying the current tokens. WavSpA Zhuang et al. (2024) on the other hand, computes the attention block in the wavelet space. The premise is since wavelet transform is a multi-resolution transform capturing long term dependencies at multiple timne scales, the input sequences is transformed into wavelet space, attention mechanism is carried out, and then reconstructed back. However one of the major drawback is the operation is non-causal, i.e. to compute the wavelet transform one needs to look at the entire sequence length for capturing variations from coarsest to finest scales (as can be seen in Figure 1 of Zhuang et al. (2024)). Thus such modifications cannot be adapted to GPT like decoder only architectures. As we will see in our work, we modify only the intermediate embeddings leaving the rest of the architecture same as is in a causal manner. Our work is also inspired from neuroscience, which gives evidence that the human brain learns multi-scale representations for language at multiple time scales Caucheteux et al. (2023) as opposed fixed resolution representations. As we will see in our work, our paper proposes explicitly to impose multi-scale representation during pre-training onto every intermediate decoder embeddings.

The contribution of the paper is as follows:

- We propose, to the best of our knowledge, the first instance of incorporating wavelets into large language models. We propose the addition of multi-scale filters onto each of the intermediate embeddings of Transformer decoder layers using Haar wavelet. Our architecture allows every next token prediction to have accesss to multi-scale representations for the intermediate embeddings in every Transformer decoder layer as opposed to fixed resolution representations.

- We show that with the addition of no extra parameter, we can substantially speed the pretraining of a transformer-based LLM in the range of 40-60%. This finding is substantial given how ubiquitous the Transformer Decoder-based architectures are across various modalities. We also show that with

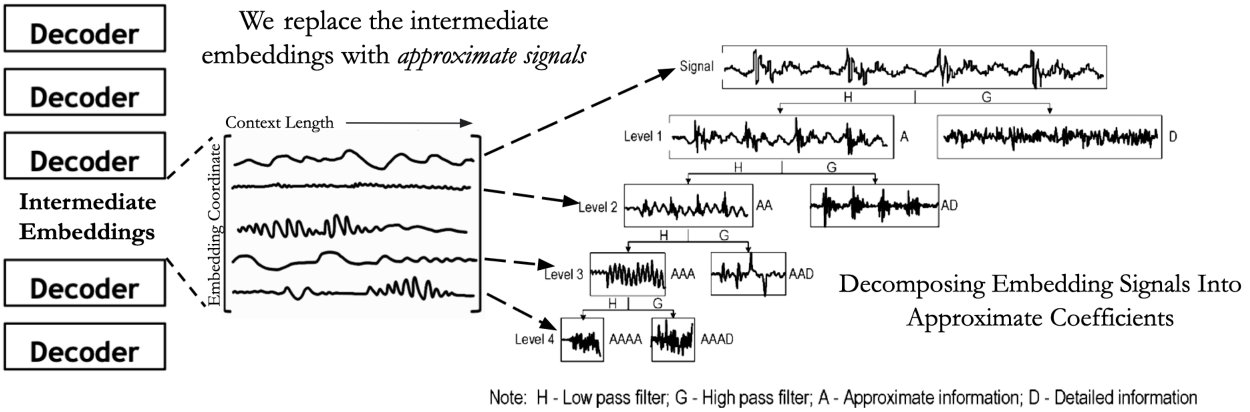
Decoder
Decoder
Decoder
Intermediate Embeddings
Decoder
Decoder
We replace the intermediate embeddings with *approximate signals*
Context Length
Embedding Coordinate
Signal
H
G
Level 1
A
D
Level 2
AA
AD
Level 3
AAA
AAD
Level 4
AAAA
AAAD
Decomposing Embedding Signals Into Approximate Coefficients
Note: H - Low pass filter; G - High pass filter; A - Approximate information; D - Detailed information

Figure 1: We find signals in every intermediate embeddings between decoder blocks of GPT. For each of these signals of length equal to the context length of GPT, we compute a simple 1-D discrete haar wavelet transform at different level to approximate the signal at different resolutions to mimick multi-scale structure that exists in real word for text, raw audio and symbolic music. The figure on right is from Gao & Yan (2006), which gives a more detailed account of non-stationary signal processing for 1-D signals. We go on the leftmost route of approximate coefficients allowing us to capture embeddings at different resolutions.

the same number of training steps, the model gives a substantial non-trivial performance boost, akin to adding several layers or parameters.

- We show that adding a wavelet-based operation gives a performance boost in three different modalities for the pretraining tasks regarding validation loss. These three modalities are text (text-8), raw audio (YoutubeMix), and symbolic music (MAESTRO). This shows that our method is generic enough for structured datasets.
- We also explore that by making these kernels learnable, which adds only a tiny fraction of the parameters, as compared to the primary model, we get an even further increase in the performance of our model, which allows the model to learn multi-scale filters on the intermediate embeddings from scratch.

The flow of the paper is as follows: Section 1 describes the introduction and related work to our proposed method, followed by the three datasets we used in Section 2. This is followed by a description of our methodology, wavelets, and how to connect large language model pretraining with introduction to wavelets in Section 3. For the experimental setup in Section 4, we show the performance of our system for various modalities, with scale and by making the multi-scale kernels learnable. Section 5 describes our future work, followed by Acknowledgment.

2 Dataset

We utilize three open-source datasets to showcase the strength of our proposed method. In addition, we choose them from three different domains: natural language, symbolic music, and raw audio waveform. For text, we choose text-8 Mikolov et al. (2012). We choose this over other datasets as i)it is a popular and widely cited character-level language modeling dataset for text and ii) in order to use a simple vocabulary (space + 26 lower case characters) to detach the effects of various tokenizers from our results, at least in one of the modalities. It has 100M characters with the split used in training, validation, and testing as given by Al-Rfou et al. (2019). We report the results for two more modalities other than text: raw waveform and symbolic music. For raw audio, the goal is again to predict the following sample given a context of samples. We use the YouTube-Mix-8 dataset, which has been used for long-context modeling Goel et al. (2022); Verma (2022).

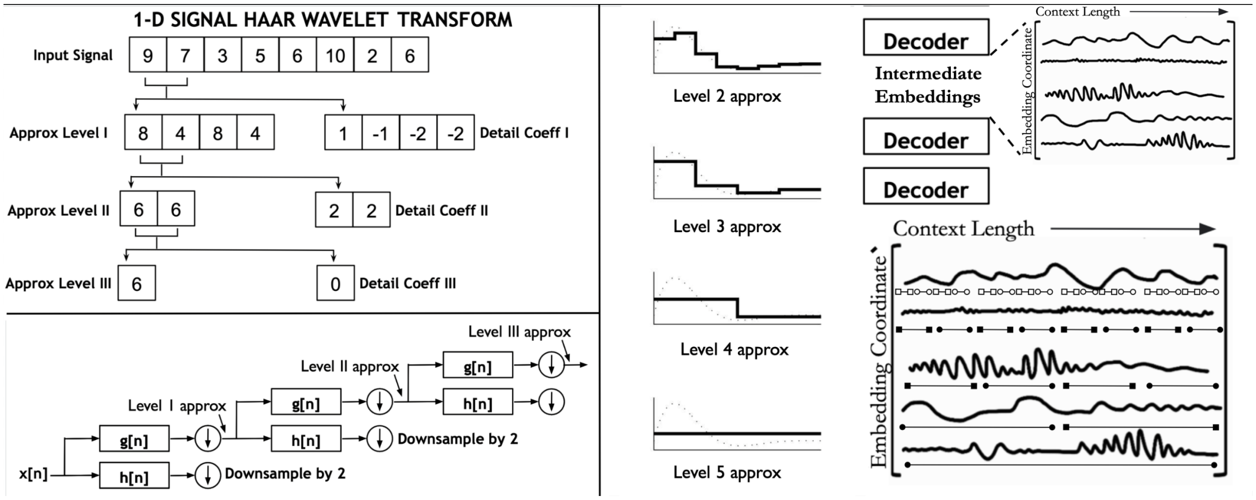

Figure 2: (Bottom left): A tree structure depicting a 3-level filter bank that can take signal and give us different resolutions of signal. In this case we only focus on the approximate coefficients, by passing it through an impulse response corresponding to the wavelet of choice and recursively down-sampling it. (Top left) How to compute the approximate and the detailed coefficients at various levels through a toy example. We recursively take first order average/differences followed by downsampling till we get ony a single scalar representative of the entire input signal. (Right) For a 32-length signal, we depict how different level of approximate coefficients of discrete haar wavelet transform capture the signal from the coarsest to fine details. The figure on (centre) and (top-left) redrawn from tutorial on wavelet transform from Flores-Mangas (2014). (right) How our architecture computes embeddings moving at different rate via the wavelet approximation where for some of the embedding dimension the information moves at the coarsest rate similar to level 5 approximation whereas for other dimensions, it follows finer resolution similar to a level 2 approximation. Notice how during token prediction all of this multiscale knowledge is present in all decoder layers.

Here, since we use 8-bit signals as we did in prior works, our vocabulary size is 256, with a sampling rate 16KHz. We use a third dataset, MAESTRO Hawthorne et al. (2018), which contains over 1000 MIDI files of popular classical music pieces. We use Google's tokenizer Huang et al. (2018), which can convert MIDI tracks into discrete tokens onto which we train our LLM with a vocabulary size of 388. An important point to note is the goal in all three modalities is not to chase the state of the art performance, as *this paper was written in an academic setting with very few computational resources available.* The goal is to rather shrink down GPT-like architecture and compare pre-training performance to the shrunk down version with/without adding multi-scale structure to the embeddings, **without adding a single extra learnable parameter.**

3 Methodology

This section will describe the approach to incorporating wavelets into Transformer based Large Language models while retaining the causality assumption. The ideas described here are generic and can be easily extrapolated to setups without using a Transformer architecture.

3.1 Incorporating Wavelets into Intermediate Embeddings

For any signal, we would compute one of the versions of discrete wavelet transform as we will describe and incorporate that back into the signal. Let us assume that $x_{(i)}^{l}$ is the output of the l^{th} decoder layer and represents the activation along the i^{th} coordinate. This activation signal will have a dimension equal to the context length of the transformer-based GPT model. In our case, we denote the context length as L. So now, if in the original GPT architecture, there were $N+1$ layers, with the embedding dimension as E, we would get $N.E$ signals of length L from all of the intermediate embeddings between two decoder blocks. E in our case goes from $[0-128)$ dimensions.

3.2 Introduction to Wavelets

A wavelet is a signal that typically has zero mean and a non-zero norm. A wavelet transform was first designed to overcome the shortcomings of a traditional Fourier-based representation. Given any signal $x[n]$, a discrete wavelet transform is akin to passing the signal through filters with different resolutions as can be seen in Figure 2. In its simplest form, throughout this paper, we will use Haar wavelet, which is simply a family of square-shaped functions. The family is obtained from a mother wavelet via scaling and shift operations. Given a mother wavelet function ψ, we come up with the child wavelets as

$$\psi_{j,k}[n] = \frac{1}{\sqrt{2^j}}\psi\left(\frac{n-k2^j}{2^j}\right)$$

where k is the amount of shift, and j is the scaling factor. For the case of the Haar wavelet, this would amount to the kernel being a simple averaging and difference functions, at different scales that we will exploit in our work. The advantages of operating a wavelet function in our work are clear from the definition: It allows us to have multi-scale operations on any signal and gives several perspectives of a signal at different resolutions as seen in Figure 1.

We now define discrete wavelet transform. Simply, it can pass any signal through a series of filters and downsampling operations. This operation as seen in Figure 2, should immediately strike resemblence to a convolutional neural net like Resnet He et. al (2016), which consists of learned convolutional filters analogous to $h[n]$ and $g[n]$, and downsampling operation like max-pooling. In traditional state of the art convolutional architecture, we typically go the route of following one branch of the Figure 2, i.e. we take output of filters, and downsample and do it recursively. We will in this paper will also do something similar. This was also one of the reasons wavelets were incredibly popular in early 90s and 2000s for image understanding, as one can see parallels to that of convolutional architectures Huang & Aviyente (2008); Kingsbury & Magarey (1998)

Let us assume that we choose a family of wavelets (Haar wavelet in our case); then it would be akin to passing the signal through a low-pass and a high-pass filter corresponding to the kernels in that family of wavelet transforms $g[n]$ and $h[n]$. In case of Haar wavelet transform, it is simply taking averaging and difference operation i.e. the impulse response of $g[n]$ and $h[n]$ are [1/2,1/2] and [1/2,-1/2] respectively. Let us look at the Figure 2 for a more detailed explanation of a discrete wavelet transform.

Let $x[n]$ be any 1-D length signal L. In order to get level 1 coefficients, we pass it through two filters with impulse response $g[n]$ and $h[n]$ followed by a downsampling operation. Thus the approximation coefficients y_{approx} and y_{detail} are simply the output of an LTI system followed by downsampling (by 2 here) defined as,

$$y_{\text{approx}}[n] = \sum_{k=-\infty}^{\infty} x[k]g[2n-k]$$
$$y_{\text{detail}}[n] = \sum_{k=-\infty}^{\infty} x[k]h[2n-k]$$

Now, in order to get multi-scale representations of the original signal, the same operation that is described above for $x[n]$ is carried out recursively now for y_a (approx) to get level 2 wavelet coefficients y_a^2 and y_d^2 (detail) and so on. Typically, the collection of signals describing approximate coefficients y_a and y_d along with their decomposition namely, $y_a, y_d, y_a^2, y_a^3, y_a^4$... are used for further processing for various application. We note that y_a^2, y_a^3, y_a^4 will have smaller lengths by a factor of 2, 4, 8, and so on. For Haar wavelet transform, we can recursively go down the route of approximate coefficients, and average the adjacent two samples. For retaining the causality assumption as we will see soon, we can simply take the average of the current and the past sample. We can see that higher order approximate coefficients capture averages at much larger context length if we keep going the route of only the approximate coefficients as can be seen in Figure 2. We can go to maximum depth till we are only left with a single value, a scalar, that is representative of the entire signal which is mean over the length of the signal. Haar wavelet transform computes averages and differences of the signal to get multi-resolution representation of the signal capturing low and high frequencies of the signal at different resolutions. This can be seen in Figure 2, where the same signal on the right is captured at the coarsest representation and then finer detail representations using Haar wavelets. We do this on the intermediate embeddings thereby allowing every next token prediction to have access to such representations.

3.3 Connecting wavelets and LLM embeddings

Often, in many signal processing applications, the first-order detail coefficients and all of the approximate coefficients are used to understand the contents of the signals at various levels. We also intend to carry out the same operation but now on signals we get from intermediate Transformer embeddings. However, we do not take any detailed coefficients and only look into the approximate ones. This was our premise: that real-world data around us is structured. For text, the structure at different levels ranges from letters, words, sentences, paragraphs, topic models, etc. In the case of symbolic music, it can be thought of as musical notes to motifs to pieces and so on. Since we chose Haar wavelet for the rest of this work, this can be approximated as a simple averaging operation as described in the previous section. If we keep going down the path of the approximate coefficients, we will eventually have only a single scalar, which is the average of the whole signal for the case of the Haar wavelet. In order to get the same sequence length from the approximation coefficients as the original signal, there can be several ways, with up-sampling the signal back to the original length being one of them. As part of nomenclature, we call the signal that is approximated at a particular level with the same length as the *"approximate signal" at that level to discern it from the approximate coefficients which are smaller in length.* In Figure 2 (R), in order to get the signal approximation at various levels, which is equal to the original input signal $x[n]$, the wavelet kernel being averaging operation, we take the approximate coefficients and multiply it with the kernel at that level. ([1,1], [1,1,1,1], ... and so on). This can be reflected from the piece-wise constant function as seen in Figure 2. The reconstructed signal $x_{recon}[n]$, which is one way of getting the *approximate signal* is computed from its wavelet coefficients, at a particular level j as c_j as,

$$x^{j}_{recon}[n] = \sum_{k} c_k \cdot \psi_{j,k}[n]$$

In order to simplify computing the *approximate signal*, in a differentiable manner within Transformer architecture, we opt for a simple variant to the equation described above. In the case of the Haar wavelet, being a simple averaging operation, we take moving average of the input signal with varying kernel length. We keep on increasing the kernel's length for averaging until it becomes that of the context length (when a single scalar approximates the whole signal). The kernel length decides which level of approximation of signal we are interested in. Since LLMs operate on causality assumption for any input signal and a given kernel length, if needed, we get the modified value of the signal at a location by computing the moving average of the prior samples of the input signal within the kernel length. We zero-pad the signal to the left to account for the cases and token dimensions when the length of the signal is less than that of the kernel.

The discrete Haar wavelet transform at different levels give multiple versions of the same signals. This might create more copies of the same signal and mess up the structure and the dimensions of the intermediate Transformer embeddings. In order to avoid this issue, we create different resolutions for different approximation to the signals. The resolution at which we look at the signal is now parameterized by the coordinate in the model dimension, which will be explained in more detail in the next section.

3.4 Wavelet Coefficients by Embedding Dimension Coordinates

One option would be to take each of the signals $x^{l}_{(i)}$ in each of the co-ordinate of every decoder layer, and compute each of their *approximate signals* in Level I, II, III, IV and so on. This would have exploded the number of signals that we have. For example, for an context length of 512, we would need nine more copies of with a resolution of 512, 256, 128, 64, 32, 16, 8, 4, 2 describing level I to IX coefficients of the original signal. This would tremendously increase the complexity of our architecture, in our case a GPT, and would have required significant architectural changes to incorporate an increase in information via multiple additional resolution signals. To mitigate this, we come up with a novel solution as follows:

Algorithm 1 Wavelet-GPT

```
  $E$: Model or Embedding Dimension
  $L$: Context Length
  $N+1$: Number of Decoder Layers
  for layer $l = 1, 2, \dots, N$ do
      $\mathbf{x}^l \leftarrow$ Output of Transformer $l^{th}$ Decoder Block                //Dimension $E$ x $L$
      $\mathbf{xn}^l \leftarrow$ Modified Transformer Embedding Replacing $\mathbf{x}^l$
      $\mathbf{xn}^l_{(i)} \leftarrow \mathbf{x}^l_{(i)}$   For Embedding dimension   $i > E/2$

      $\mathbf{f(i)} \leftarrow 2^F$where    //Finding kernel length a function of embedding coordinate as nearest power of 2
              $F = int(L_k * (i - E/2)/(E/2 - 1))$   $L_k = \lfloor \log_2(L) \rfloor + 1$   i $< E/2$

      $\mathbf{xn^l_{(i)}(k)} \leftarrow \frac{\mathbf{1}}{\mathbf{f(i)}} \sum_{\mathbf{m=k-f(i)}}^{\mathbf{k}} \mathbf{x^l_{(i)}(m)}$   i $>= E/2$            //For Non-learnable fixed Haar wavelet
      $\mathbf{xn^l_{(i)}(k)} \leftarrow \sum_{\mathbf{m=0}}^{\mathbf{f(i)-1}} \mathbf{h(m)} \cdot \mathbf{x^l_{(i)}(k-m)}$   i $>= E/2$   //For learnable multi-resolution wavelet kernel $h$
  end for
```

We do not compute all levels of *approximate signal* for each of the intermediate embedding dimension signals across tokens. We parameterize the level to be computed for the *approximate signal* by the index of the embedding dimension itself. Another important thing is that we want to steer the embeddings only a little into the inductive biases we impose. Transformers have been wildly successful, without incorporating any inductive biases. We ideally want the best of both of the worlds where we nudge intermediate GPT embeddings in only half of the dimensions. For this, we retain half of the intermediate embedding signals along the coordinate dimension at the exact resolution i.e. with no change. For the embedding coordinates from 64 to 128 ($E/2$ to E) when the model dimension is 128, we do not do any processing or manipulations. For the other half, we do some processing parameterized by their index i. Mathematically, if $x^l_{(i)}$ is an intermediate embedding after the l^{th} decoder layer along the i^{th} coordinate dimension, then for half of the coordinate dimensions of the modified new signal $xn^l_{(i)}$ will remain same as that of $x^l_{(i)}$ for i from $E/2$ to E. For the second half, we impose our structure with using **approximate signal** at a particular level.

This is mainly because Transformers are quite expressive, and we want to avoid too much tinkering with what they learn. For the other half, $xn^l_{(i)}$ is the modified latent space that is obtained from $x^l_{(i)}$ by first getting the wavelet coefficient level corresponding to that of the coordinate i. We use a simple mapping function f to take the coordinate dimension i as an argument. In our case, f takes in the argument from i, ranging from 0 to $E/2$ (0-64), and returns the kernel size corresponding to the approximation coefficient level between I and IX. We use a simple linear function that slowly increases the index from I to IX between 0 to $E/2$. So when i is 0, $f(i)$ maps to level I approximation kernel 2, and when i is $E/2$ (64 in our case), $f(i)$ maps to level IX approximation kernel of length 512, (or for generic case, the level that would corresponding to the coarsest representation i.e. a single scalar). Now, let us find out how we compute the modified new signal $xn^l_{(i)}$ that replaces the original intermediate Transformer embeddings $x^l_{(i)}$. $f(i)$ denotes the kernel size for the coordinate i. Now, the modified signal is as follows and explained in Algorithm 1 as :

$$xn^l_{(i)} = x^l_{(i)} \text{ for } i > E/2$$

$$xn^l_{(i)}(k) = \frac{1}{f(i)} \sum_{m=k-f(i)}^{k} x^l_{(i)}(m).$$

For the cases where $k - f(i) < 0$, we zero-pad the signal to make this signal valid for the average to be computed. In other simple words, for the case of the Haar wavelet, the modified signal is nothing but a causal moving average filter that computes the average of values of the embedding signal along i^{th} coordinate with a kernel size as a function $f(i)$. As described in the equation above, this operation is a simple algebraic operation that does not add any parameters to the architecture. This retains the causality assumption critical in LLM and prevents any leakage from happening to the future tokens from any of the embedding dimensions.

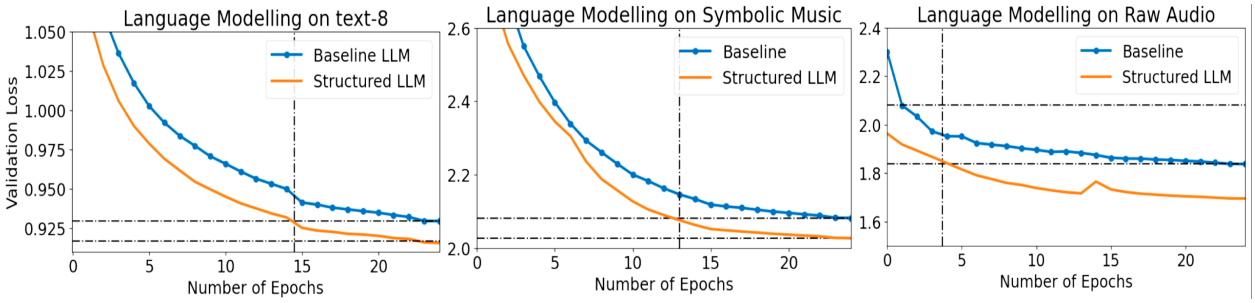

Figure 3: We report results for three modalities namely natural language, symbolic music and raw audio. We see that we achieve much faster convergence almost twice as fast. When trained for the same number of epochs, we see that we achieve a substantial improvement in the pre-training performace which is equivalent to having a much larger architecture. The black vertical line denotes the epoch at which our architecture achieves the same performance as that of our baseline architecture.

3.5 Imposing Structure: Toy Example

As we can see from the Figure 4, we have shown a toy example to depict how we impose a structure onto decoder Transformer embeddings. In Figure 4 (left), on top, eight variations along the token dimension are present, with onset (largest values or sudden bursts) at token index numbers 32, 64, and so on and decreasing to zero in the next token and then again increasing to the largest value linearly till the next interval. As motivated in the introduction before, datasets around us have an inherent structure present in them. In order to capture this structure, we impose a structure onto intermediate Transformer embeddings in every layer. For the toy example, we can see from Figure 4 (left), no bottom, that we retain the embeddings at the exact same resolution for half of the embedding dimensions (split by white line). For the other half of the embedding dimension, across the context length, we slowly increase the kernel length across which we compute the average in a causal manner. We reach the last embedding dimension, which moves the slowest and takes the average across the token dimension with the kernel size equal to the context length (zero-padding the signal if necessary). This creates highways that allow some coordinates of the embeddings to move at different rates, with coordinates from $E/2$ to E being at the same rate as what the Transformer decides and coordinates from 0 to $E/2$ linearly changing from moving at the same rate to being the slowest. Allowing the embeddings to move at different speeds in every intermediate decoder layer, from the lowest possible speed to the original speeds, and to allow attention mechanism to make use of multi-scale features moving at difference rate at every layer and every token is a very powerful idea as we will see in the next section.

4 Experiments

In this section, we explain how we incorporated the idea of infusing wavelets into large language model pre-training. We trained all of the models from scratch, which required substantial computing. However, the main aim of these experiments is to show how the performance of the models across three modalities improves with/without doing intermediate modifications on embeddings. Since we do not add any parameters when we modify intermediate emebddings with wavelet transform, we can compare the two models both in terms of the performance boost that the new architecture achieves and the convergence speedups.

4.1 Baseline

All models, similar to the GPT-2 architecture, consist of a stack of Transformer decoder layers. Since each requires pre-training the models from scratch, we choose the following setup. Every modality, namely text, symbolic music, and raw waveform, has the same architecture topology with a context length of 512. We choose the number of decoder blocks to be 10, with 128 as the embedding dimension, the feed-forward dimension to be 512, and the number of heads to be 8. We opt for a two-layer MLP inside the Transformer block instead of a single layer, with both the layers sharing the same number of neurons, i.e., 512, that of the feed-forward dimension. The final output layer of the Transformer decoder is then followed by a dense layer of 2048 neurons, followed by a dense layer of the same size as the vocabulary. This vocabulary size varies in the three modalities. For text8, it is 27, which is the number of characters plus an added extra token for space. For the raw waveform, we use an 8-bit resolution waveform at 16kHz, which is similar to the reported in Goel et al. (2022); Verma (2022), thus yielding 256 as a vocab size. For symbolic music, we utilize Google's tokenizer Huang et al. (2018) to convert MIDI data to discrete tokens yielding 388 sized vocabulary. The baseline models in all three of them was simply a stack of Transformer decoder blocks without tinkering any embeddings. For the proposed architecture we explained in the previous section, retain half of the embedding co-ordinates without any tweaks. For other half we impose a multi-scale structure parameterized by the coordinate in the embedding dimension for all of the intermediate layers. We do not add any single parameter in this setup, and compare the performance with this tweak for all three modalities. We do this because we want to showcase the powerfulness of our algorithm for a rich variation of modalities for LLM pre-training. We do not compare against powerful, larger architectures going after scale, as this paper required pre-training from scratch. Rather, we take shrunk down version of GPT-2 architecture, viable in *academia with limited resources* and compare with/without adding wavelets to the architecture in terms of pre-training performance.

4.2 Training Details

All models were trained from scratch in the Tensorflow framework Abadi et al. (2016) for 25 epochs. We used a mirrored strategy for multi-GPU training. The learning rate schedule was chosen to be 3e-4 to start with reduced till 1e-5, whenever loss started plateauing. The number of training points available in all three models was roughly 1M, yielding the total number of tokens to be 1/2 billion. These were randomly cropped from the dataset of choice. Apart from setting a default dropout rate of 0.1 in MLP and attention layers, no other regularization was carried out. The performance metric chosen to compare is only the negative log-liklihood loss, as this method improves the core architecture of the Transformer based GPT, and helps in the objective that we want to achieve: predict the next token. Since we are operating on intermediate embeddings, we believe that our work can hopefully generalize to setups where there is structured data available similar to text, raw audio and symbolic music, where one can go from fine-grained structure to a coarse structure. We can see from Figure 4 on how for a toy example we can impose a multi-scale structure that allows attention mechanism to not only learn dependencies across various embeddings but also injects into these embedding coordinates some information that can capture coarse and fine-grained structure.

4.3 Performance on modalities

In this section, with the added modifications, we compare the performance of our baseline architecture across three modalities, namely text, symbolic music, and audio waveform with/without the addition of wavelet-based intermediate operation. We see that we have achieved a substantial increase in performance in all three modalities when we trained for the same amount of training steps. To give an analogy for natural language, an increase of 0.04 is akin to going from a 16-layer architecture to a 64-layer model on a text-8 dataset papers-with code (2024). As we can see from the Figure 3, we see that we achieve almost twice as fast convergence as compared to the original architecture in terms of training steps. This is particularly important as we can see that the GPT-like architecture can indeed take advantage of the structure that we imposed on half of the embedding dimensions. This speedup, i.e. the number of epochs/steps taken to achieve the same performance when the loss starts to plateau is even smaller for raw audio. One of the reasons this can be attribute to is the fact that audio signal remain quasi-stationary for smaller time-scales i.e. 20ms-30ms for harmonic sounds. For a sampling rate of 16KHz, a context length of 512 would correspond to 32ms, which may be one of the reason that some of the coordinates nail down the contents of the context, in a fewer coordinates onto which we impose structure. The convergence happens much much faster for raw waveform LLM setup as compared to being almost twice as fast in text-8 and symbolic music. We also compare the absolute clock run times for our modifications in both learnable and non-learnable setups. We report the time take to complete one epoch relative to that of our baseline architecture. We see that the time taken to run the architecture has also similar times as that of the baseline, as the only operation carried out was simple

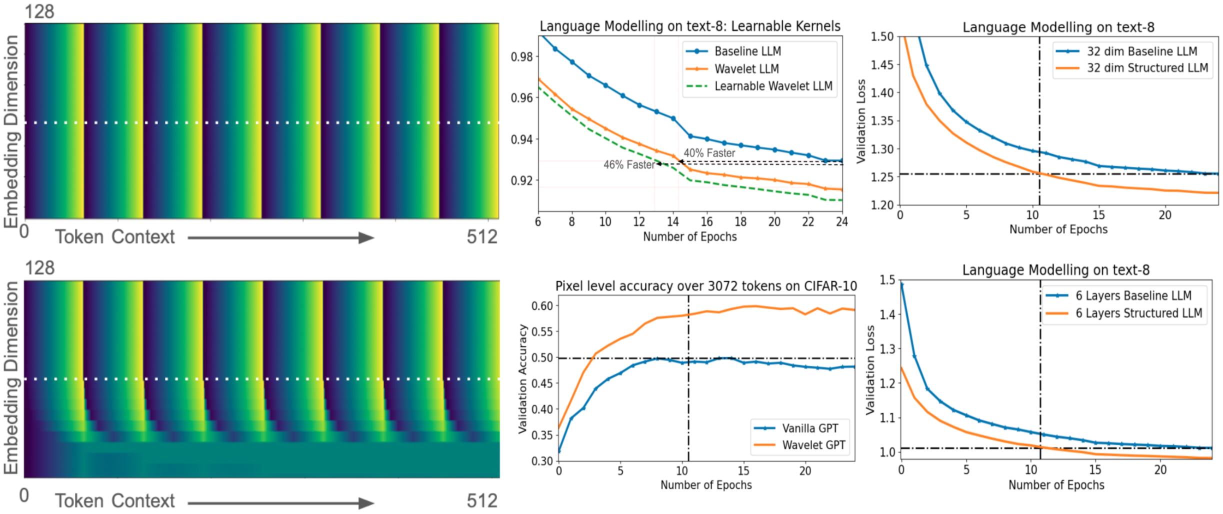

Figure 4: (Left) A toy example where variations of embeddings shown via heat-map are moving along token dimension and how our processing imposes a multi-rate structure where different embedding dimensions are moving at different rates while retaining the causality assumption. This allows intermediate latent space to allow to learn information moving at different rates at every token. Notice how the pattern disperses from the dimension 64 to 0. (Right) Validation loss of LLM pretraining on text-8 with addition of the structure with making the kernels learnable. We can see that we achieve the same performance almost twice as fast. When trained for the same number of epochs, we get performance boost akin to adding additional decoder layers. We also show how our architecture behaves on text-8 for smaller model dimension of 32 retaining the convergence speeds as 128-dim model and shallow depth version with 6 layers and 128 model dimension. We also see for Long Range Arena image benchmark, we get a boost of 10% with addition of no extra parameters.

averaging for the case of Haar wavelet or learning a single filter convolutional kernel with variable context lengths over various embedding dimensions. This makes our work exciting as for a Transformer GPT like setup, as it reduces training times by substantial margins. It will be interesting to see if this behaviour holds for model architectures like LLAMA, training which is far beyond the resources available to the authors.

Table 1: Comparison of the negative-log likelihood (NLL) scores (log base e) for our architecture with three modalities with/without adding wavelet-based hierarchical structure, and for learnable wavelet transform.

Modality	Baseline	Proposed	Same Performance Epoch	SpeedUp	Relative GPU Hours
Text	0.93	0.915	14.5 epochs	42%	1.013
Raw Audio	1.84	1.7	3.7 epochs	85%	1.042
Symbolic Music	2.08	2.02	13 epochs	48%	1.059
Text (Learnable)	0.93	0.91	12.9 epochs	46%	1.094

4.4 Effect of Depth And Model Dimension

Here we explore two variants of our architecture – what would happen if we reduce the model dimension from 128 to 32 and reduce the number of layers. We carry out all the experiments for text-8. We can see that for the variant where we reduce the model dimension to 32 for a 10-layer Transformer decoder architecture with 8 heads, the model still retains faster convergence as seen in Figure 4 and achieves the performance without doing the modification (as seen as Baseline) in around 10 epochs. For the second experiment, we retain the exact architecture as proposed in our experiments reported in Table 1. However now, we only have 6 Transformer Decoder layers, keeping rest of the parameters the same (feed forward dimension four times that of the model dimension, 8 attention heads), to see the effect of depth. We see that the model continues to hold and again achieves the performance of the model trained for about 25 epochs almost twice as fast. Both of these experiments, are shown in Figure 4.

4.5 Making multi-scale kernels learnable

As described in the previous section, we can see that with the addition of no parameters onto Transformer decoder layers, by imposing a multi-scale structure, we can make pre-training significantly faster. In this experiment, we allow each of the kernels to be learnable. In the previous section, we defined the shape of the kernel as a Haar wavelet. We looked at the approximate coefficients of intermediate layer activations across all of the layers, with different resolutions occurring at different embedding dimensions. Now, in this experiment, we allow each kernel to be learnable. So now, instead of a Haar wavelet operation, we allow each kernel to be learnable for getting the *approximate signal* for various resolutions. Before, we were taking average in order to compute the *approximate signal* at a particular embedding dimension which is a convolutional with kernel of length L equal to $(1/L, 1/L, 1/L, 1/L...)$. In this experiment, we make the L length kernel learnable from scratch, which is another way of computing the *approximate signal.* This simple operation for our base models only allows 0.02M (20k) extra parameters to the Transformer decoder architecture. Unlike the previous setup, which did not add any extra parameter, this further improves our performance from 40% to 46% faster convergence to achieve similar performance, as seen from the Figure 4. This was carried out on the text-8 dataset. All results are reported on cross-entropy loss computed to the base e. This further validates our method and showcases further improvements and strength of our work.

5 Long Range Arena Benchmarks

We adapt our architecture to benchmark long-range arena (LRA) tasks Tay et al. (2021). It consists of various datasets that allow models to handle long-range prediction over sequence tasks over diverse domains, pushing the ability of Transformer architecture and other variants. We use three modalities: text, images, and mathematical expressions to test the model's ability to understand similarity, structure, and reasoning over extended contexts. We only use transformer-based architecture as reported recently in Liu et al. (2024). The other variants are state space architectures and hybrid models. For text, we carry out text classification on IMDb review dataset Maas et al. (2011) on byte-level data with a context length of 2048 as input. The goal here is binary classification, which determines whether a movie has a positive or a negative review. For images, we use classification on CIFAR-10 as part of the image modality of LRA benchmarks. It is a pixel-level classification of the image that takes in as an input a sequence of pixels with values ranging from 0-255 with a length of 3072 and the output being one of the ten categories as the output. Finally, we benchmark on Long ListOps. It tests the capability of the architecture to understand hierarchically structured data in an extended context setup. As described in the LRA paper Tay et al. (2021), "The dataset is comprised of sequences with a hierarchical structure and operators `MAX`, `MEAN`, `MEDIAN` and `SUM_MOD` that are enclosed by delimiters (brackets). An example (much shorter) sequence is as follows:

INPUT: `[MAX 4 3 [MIN 2 3] 1 0 [MEDIAN 1 5 8 9, 2]]` **OUTPUT:** `5`

In our task, we use a version of ListOps of sequence lengths of up to `2K` to test the ability to reason hierarchically while handling long contexts. In the above example, the model needs to access all tokens and model the logical structure of the inputs to make a prediction. The task is a ten-way classification task and is considerably challenging." We use the setup provided by Khalitov et al. (2022) to extract the data and be uniform with other benchmarks. We experiment with almost the same architecture for all three modalities and only change the embedding matrix to account for different tokenizers and output categories. For a baseline, we use a 6-layer **Transformer decoder** only architecture, that is, **causal**, with a model dimension of 32 and a feed-forward dimension four times that of the embedding dimension. We take the last token of the sequence as an embedding that is extracted for classification, thus a 32-dimensional vector, which is then followed by a dense layer of 2048 neurons and a dense layer equal to the number of categories. The input is passed through an embedding layer that converts discrete tokens into a 32-dimensional vector. The input vocabulary of text, image, and list-ops is 256, 256, and 16, respectively. The context length is 2048, 3072, and 1999 tokens, respectively. The output categories are 2, 10, and 10, respectively. For our modified architecture,

Table 2: Performance of predicting outcomes of list operations in the LRA (Tay et al. (2020b)) as reported in Liu et al. (2024). Bold indicates the best-performing model and underlines the second best. We use a baseline architecture for all three benchmarks, as reported in section 5, followed by modifying the intermediate embeddings with no parameter gain whatsoever by imposing a hierarchical structure. We do not report non-transformer or hybrid architectures.

Transformer Based Attention Models	**ListOps**	**Text**	**Image**
Transformer (Vaswani et al. (2017))	36.37	64.27	42.44
Local Attention (Tay et al. (2020b))	15.82	63.98	41.46
Linear Trans. (Katharopoulos et al. (2020))	16.13	<u>65.90</u>	42.34
Linformer (Wang et al. (2020))	35.70	53.94	38.56
Sparse Transformer (Child et al. (2019))	17.07	63.58	44.24
Performer (Choromanski et al. (2021))	18.01	65.40	42.77
Sinkhorn Transformer (Tay et al. (2020a))	33.67	61.20	41.23
Longformer (Beltagy et al. (2020))	35.63	64.02	40.83
BigBird (Zaheer et al. (2020))	36.05	64.02	40.83
Luna-256 (Ma et al. (2021))	37.25	65.78	47.86
Reformer (Kitaev et al. (2020))	37.27	56.10	38.07
FNET (Lee-Thorp et al. (2022) (Non-Causal))	37.27	56.10	38.07
WavSPA – AdaWavSpA Transformer - (Non-Causal) (Zhuang et al. (2024))	<u>55.40</u>	**81.60**	<u>55.58</u>
Ours (GPT Baseline With Classification Head)	41.65	65.32	49.81
Ours (WaveletGPT With Classification Head)	**57.5**	<u>66.38</u>	**59.81**

similar to how we described earlier, we introduce our waveletGPT module between every decoder layer. We retain half of the embedding dimensions as is. For the other half, we use non-learnable kernels, increasing the kernel size from 2,4,8 to 512 linearly for dimensions from 16 to 32, retaining the causality assumption. This introduces highways that hierarchically process the data at every embedding and every Transformer decoder layer without adding any parameter similar to what we carried out for LLM. As we can see from Table 2, we achieve non-trivial gains in all three modalities where even small gains are worth reporting. We outperform non-causal based methods, e.g., Zhuang et al. (2024) significantly with almost 2% on ListOps and 4.5% with a much smaller architecture, shallower model(ours 32 dimensions, six layers vs. 128 dimensions, eight layers). Compared to a non-causal architecture FNet, we significantly outperformed all three LRA tasks, with 20% points on ListOps and Image and 10% on text. One of the biggest jumps is seen in the ListOps task, which requires modeling a hierarchical, tree-like structure of the math operations, which our model is best suited as motivated earlier. To the best of our knowledge (Liu et al. (2024)), this achieves the best Performance of simple attention-based Transformer architecture on Long-Range arena tasks.

6 Conclusion and Future Work

We showcase the powerful incorporation of a core signal processing idea, namely wavelets, into large language model pre-training. By imposing a multi-scale structure onto every intermediate embeddings, we see that we can achieve convergence 40-60% faster convergence, as compared to the same baseline architecture, with the addition of no extra parameter. We achieve a substantial performance boost if we train for the same number of steps. Our method generalizes across three modalities: raw text, symbolic music, and raw audio, giving similar performance speed ups. Several exciting directions can be explored in future work, including incorporating more advanced ideas from wavelets and multi-resolution signal processing onto large language models. It will be interesting to see how model behaves for different variants of multi-scale structure.

References

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. {TensorFlow}: A system for {Large-Scale} machine learning. In *12th USENIX symposium on operating systems design and implementation (OSDI 16)*, pp. 265–283, 2016.

Nasir Ahmed, T_ Natarajan, and Kamisetty R Rao. Discrete cosine transform. *IEEE transactions on Computers*, 100(1):90–93, 1974.

Rami Al-Rfou, Dokook Choe, Noah Constant, Mandy Guo, and Llion Jones. Character-level language modeling with deeper self-attention. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pp. 3159–3166, 2019.

Iz Beltagy, Matthew E. Peters, and Arman Cohan. Longformer: The long-document transformer. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 6150–6160, 2020. URL `https://www.aclweb.org/anthology/2020.emnlp-main.519/`.

Zalán Borsos, Raphaël Marinier, Damien Vincent, Eugene Kharitonov, Olivier Pietquin, Matt Sharifi, Dominik Roblek, Olivier Teboul, David Grangier, Marco Tagliasacchi, et al. Audiolm: a language modeling approach to audio generation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 2023.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*, 2023.

T. Brown et, al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

Charlotte Caucheteux, Alexandre Gramfort, and Jean-Rémi King. Evidence of a predictive coding hierarchy in the human brain listening to speech. *Nature human behaviour*, 7(3):430–441, 2023.

Rewon Child, Erich Elsen, David Kim, and Geoffrey Hinton. Sparse transformer. In *Proceedings of the 33rd Conference on Neural Information Processing Systems (NeurIPS 2019)*, 2019. URL `https://arxiv.org/abs/1904.10509`.

Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, David Belanger, Lucy Colwell, and Adrian Weller. Rethinking attention with performers. In *Proceedings of the 9th International Conference on Learning Representations (ICLR)*, 2021. URL `https://openreview.net/forum?id=Ua6zuk0WRH`.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms. *Advances in Neural Information Processing Systems*, 36, 2024.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

A. Dosovitskiy et. al. Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39, 2022.

Fernando Flores-Mangas. Discrete waveelet transform. *The Washington Post*, Spring 2014. URL `https://www.cs.toronto.edu/~mangas/teaching/320/slides/CSC320L11.pdf`.

Robert X Gao and Ruqiang Yan. Non-stationary signal processing for bearing health monitoring. *International journal of manufacturing research*, 1(1):18–40, 2006.

Karan Goel, Albert Gu, Chris Donahue, and Christopher Ré. It's raw! audio generation with state-space models. In *International Conference on Machine Learning*, pp. 7616–7633. PMLR, 2022.

Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. MiniLLM: Knowledge distillation of large language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL `https://openreview.net/forum?id=5h0qf7IBZZ`.

Curtis Hawthorne, Andriy Stasyuk, Adam Roberts, Ian Simon, Cheng-Zhi Anna Huang, Sander Dieleman, Erich Elsen, Jesse Engel, and Douglas Eck. Enabling factorized piano music modeling and generation with the maestro dataset. *arXiv preprint arXiv:1810.12247*, 2018.

K. He et. al. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770, 2016.

Cheng-Zhi Anna Huang, Ashish Vaswani, Jakob Uszkoreit, Noam Shazeer, Ian Simon, Curtis Hawthorne, Andrew M Dai, Matthew D Hoffman, Monica Dinculescu, and Douglas Eck. Music transformer. *arXiv preprint arXiv:1809.04281*, 2018.

Ke Huang and Selin Aviyente. Wavelet feature selection for image classification. *IEEE Transactions on Image Processing*, 17(9):1709–1720, 2008.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, pp. 5156–5165. PMLR, 2020. URL `https://arxiv.org/abs/2006.16236`.

Ruslan Khalitov, Tong Yu, Lei Cheng, and Zhirong Yang. Sparse factorization of square matrices with application to neural attention modeling. *Neural Networks*, 152:160–168, 2022.

Nick Kingsbury and Julian Magarey. Wavelet transforms in image processing, 1998.

Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. In *Proceedings of the 8th International Conference on Learning Representations (ICLR)*, 2020. URL `https://openreview.net/forum?id=rkgNKkHtvB`.

Jan Koutnik, Klaus Greff, Faustino Gomez, and Juergen Schmidhuber. A clockwork rnn. In *International conference on machine learning*, pp. 1863–1871. PMLR, 2014.

James Lee-Thorp, Joshua Ainslie, Ilya Eckstein, and Santiago Ontanon. FNet: Mixing tokens with Fourier transforms. In Marine Carpuat, Marie-Catherine de Marneffe, and Ivan Vladimir Meza Ruiz (eds.), *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 4296–4313, Seattle, United States, July 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.naacl-main.319. URL `https://aclanthology.org/2022.naacl-main.319`.

Zicheng Liu, Siyuan Li, Li Wang, Zedong Wang, Yunfan Liu, and Stan Z Li. Short-long convolutions help hardware-efficient linear attention to focus on long sequences. *arXiv preprint arXiv:2406.08128*, 2024.

Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3431–3440, 2015.

Xuezhe Ma, Xiang Kong, Sinong Wang, Chunting Zhou, Jonathan May, Hao Ma, and Luke Zettlemoyer. Luna: Linear unified nested attention. In *Advances in Neural Information Processing Systems 34 (NeurIPS 2021)*, pp. 1235–1246, 2021. URL `https://proceedings.neurips.cc/paper/2021/hash/14319d9cfc6123106878dc20b94fbaf3-Abstract.html`.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pp. 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/P11-1015`.

Ali Madani, Bryan McCann, Nikhil Naik, Nitish Shirish Keskar, Namrata Anand, Raphael R Eguchi, Po-Ssu Huang, and Richard Socher. Progen: Language modeling for protein generation. *arXiv preprint arXiv:2004.03497*, 2020.

Tomáš Mikolov, Ilya Sutskever, Anoop Deoras, Hai-Son Le, Stefan Kombrink, and Jan Cernocky. Subword language modeling with neural networks. *preprint (http://www. fit. vutbr. cz/imikolov/rnnlm/char. pdf)*, 8 (67), 2012.

Piotr Nawrot, Szymon Tworkowski, Michał Tyrolski, Łukasz Kaiser, Yuhuai Wu, Christian Szegedy, and Henryk Michalewski. Hierarchical transformers are more efficient language models. *arXiv preprint arXiv:2110.13711*, 2021.

Naomi Nix. Silicon valley is pricing academics out of ai research. *The Washington Post*, March 2024. URL `https://www.washingtonpost.com/technology/2024/03/10/big-tech-companies-ai-research/`.

papers-with code. Language modelling on text8. March 2024. URL `https://paperswithcode.com/sota/language-modelling-on-text8`.

Javier Selva, Anders S Johansen, Sergio Escalera, Kamal Nasrollahi, Thomas B Moeslund, and Albert Clapés. Video transformers: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023.

Mingjie Sun, Zhuang Liu, Anna Bair, and J Zico Kolter. A simple and effective pruning approach for large language models. *arXiv preprint arXiv:2306.11695*, 2023.

A. Tamkin et. al. Language through a prism: A spectral approach for multiscale language representations. *Advances in Neural Information Processing Systems*, 33, 2020.

Yi Tay, Donald Metzler, Xin Zhao, and Shuaiqiang Zheng. Sinkhorn transformer: Generating long-form text via randomized greedy sorting. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, pp. 9408–9419, 2020a. URL `http://proceedings.mlr.press/v119/tay20a.html`.

Yi Tay, Mostafa Dehghani, Samira Abnar, Yikang Shen, Dara Bahri, Philip Pham, Jinfeng Rao, Liu Yang, Sebastian Ruder, and Donald Metzler. Long range arena : A benchmark for efficient transformers. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=qVyeW-grC2k`.

Zhilin Tay, Mostafa Dehghani, Ashish Vaswani, Noam Shazeer, and Jakob Uszkoreit. Local attention. In *Proceedings of the International Conference on Learning Representations*, 2020b.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pp. 5998–6008, 2017.

P. Verma and C. Chafe. A generative model for raw audio using transformer architectures. *arXiv preprint arXiv:2106.16036*, 2021.

Prateek Verma. Goodbye wavenet–a language model for raw audio with context of 1/2 million samples. *arXiv preprint arXiv:2206.08297*, 2022.

Prateek Verma and Jonathan Berger. Audio transformers: Transformer architectures for large scale audio understanding. *arXiv preprint arXiv:2105.00335*, 2021.

Prateek Verma and Julius Smith. A framework for contrastive and generative learning of audio representations. *arXiv preprint arXiv:2010.11459*, 2020.

Sinong Wang, Belinda Z. Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*, 2020.

Wilson Yan, Yunzhi Zhang, Pieter Abbeel, and Aravind Srinivas. Videogpt: Video generation using vq-vae and transformers. *arXiv preprint arXiv:2104.10157*, 2021.

Manzil Zaheer, Guru Guruganesh, Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. Big bird: Transformers for longer sequences. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 17283–17297, 2020. URL `https://proceedings.neurips.cc/paper/2020/hash/c8512d142a2d849725f31a9a7a361ab9-Abstract.html`.

Yufan Zhuang, Zihan Wang, Fangbo Tao, and Jingbo Shang. Wavspa: Wavelet space attention for boosting transformers' long sequence learning ability. In *Proceedings of UniReps: the First Workshop on Unifying Representations in Neural Models*, pp. 27–46. PMLR, 2024.