

AUDIO TRANSFORMERS

Prateek Verma and Jonathan Berger

Stanford University
450 Jane Stanford Way, Stanford CA, 94305,
prateekv, brg@stanford.edu

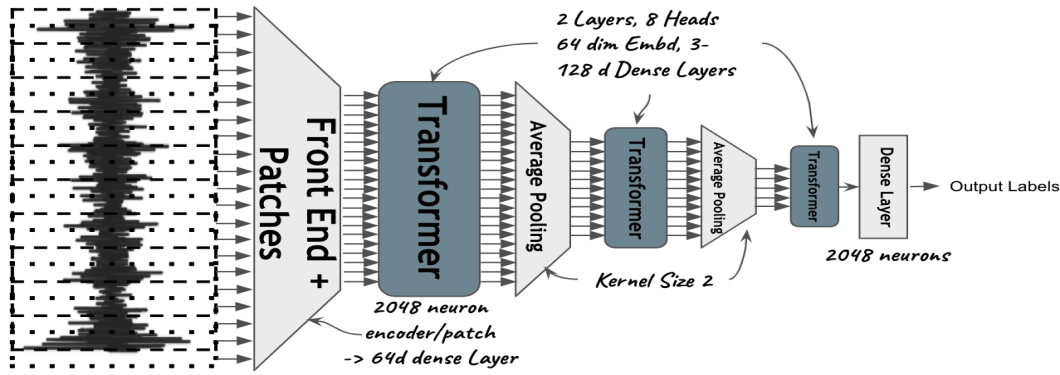


Figure 1: An overview of the proposed Audio Transformer architecture using front end fully connected encoder with Transformer layers and pooling layers. It takes 1s of input, and divides it into patches of size 25ms, followed by learning a front end, to feed it to Transformer.

ABSTRACT

Over the past two decades, CNN architectures have produced compelling models of sound perception and cognition, learning hierarchical organizations of features. Analogous to successes in computer vision, audio feature classification can be optimized for a particular task of interest, over a wide variety of datasets and labels. In fact similar architectures designed for image understanding have proven effective for acoustic scene analysis. Here we propose applying Transformer based architectures without convolutional layers to raw audio signals. On a standard dataset of Free Sound 50K, comprising of 200 categories, our model outperforms convolutional models to produce state of the art results. This is significant as unlike in natural language processing and computer vision, we do not perform unsupervised pre-training for outperforming convolutional architectures. On the same training set, with respect mean average precision benchmarks, we show a significant improvement. We further improve the performance of Transformer architectures by using techniques such as pooling inspired from convolutional network designed in the past few years. In addition, we also show how multi-rate signal processing ideas inspired from wavelets, can be applied to the Transformer embeddings to improve the results. We also show how our models learns a non-linear, non-constant bandwidth filter-bank, which shows an adaptable time frequency front end representation for the task of audio understanding, different from other tasks e.g. pitch estimation.¹

¹*Although not the first instance of an acoustic scene understanding model without convolutions, this is, to our knowledge, the first end-to-end one. At the same time as ViT, [1] showed how in a two-step process, one can achieve the same.

Index Terms— Transformers, audio understanding, wavelets

1. INTRODUCTION AND RELATED WORK

Acoustic scene analysis is a classical signal processing and machine learning problem whose goal is to predict the contents of an input signal within a brief duration, typically one second. In addition to modeling perception, computer simulation of hearing combined with models of other sensory systems will help bridge the gap between humans and computers. For the past decade, CNNs have become a de-facto architecture in learning mappings from fixed dimensional inputs to fixed dimensional outputs [2, 3]. CNN architectures inspired from vision [2], adapted for acoustic scene understanding, achieve similar performance gains for audio also.

The core backbone of this work is Transformer architecture which recently have recently produced state of the art results in a variety of domains, including protein sequences [4], text [5, 6], symbolic music [7], video [8, 9] and image understanding [10, 11]. By learning transformers on the latent representations, and conditioning a wavenet generator, they were able to achieve compelling results in music generation [12] and style transfer, which was impossible without the guidance of meta-data and convolutional architectures [13]. They have also been used in learning latent audio representations such as [14, 15] for solving pseudo-tasks such as infilling to learn time-dependent representations [1, 6]. As opposed to learning latent representations, the reduced time-scales of Transformers can advantageously model input representations. A major drawback of convolutional architecture is the fixed filter across the entire input. Furthermore, Transformers take advantage of attention

mechanism with the output at a location dependent upon the input at some other location.

The core idea of this work is to replace traditional convolutional based architectures [3], combined convolutional and Transformer architectures [16, 17], and recurrent architectures [18, 19] with a purely Transformer based architecture. Our work is distinct from the method proposed in [1] which was not an end-to-end approach and which required a two step approach (specifically, learning a dictionary of latent codes, and using the discrete latent codes as an input to transformer architectures). Similar approaches were successfully used in areas such as speech recognition [16] to mimic BERT [6]. All these state of the art performances were possible due to the architectures' ability to model long term dependency inputs and the attention mechanism present in them enabling focus only on the part of the input that is important [20].

The organization of the paper is as follows: Section I introduces the problem and the literature survey followed by the dataset we used to benchmark the results in section II. The next section details the methodology followed by results and discussion in Section IV. We conclude the paper in Section V followed by our thoughts of future work and references.

2. DATASET

We train and evaluate our architectures with FSD50K [21], an open dataset of over 51k audio files comprising over 100 hours of manually labeled audio using 200 classes drawn from the AudioSet [3] ontology. FSD50K is freely available under the creative commons license, contains many more high quality audio annotations, and twice number of training examples in the balanced set-up than AudioSet. We used the already provided training and the validation splits to tune the model and tested them on the evaluation setup provided. In total there are about 51,197 clips available ranging from 0.3-30s. We downsample all the clips to 16kHz sampling rate using [22]. We follow the same setup for reporting the results as done in [21]. All the training was carried on 1s audio chunks with the labels inherited for all the chunks in clips greater than 1s. For samples less than 1s the audio clip is repeated to fill 1s, resulting in a single training example for that clip. On an average, the duration per clip is 7.6s, with 1.22 average labels per clip, uploaded by 7225 user ids, thus encompassing a diverse range of sources, acoustic environments, microphones, and locations, to name a few.

3. METHODOLOGY

3.1. Baseline Transformer Architectures

This section describes the Transformer architecture as described in [20] that we used to train the system as shown in Figure 1. A detailed explanation is given in [23], but for the sake of clarity and completeness we describe it here. As a black-box, which we would describe in more detail in this section, it takes as an input a sequence of a fixed length T , and produces the same length but with a chosen dimension, which we call E , which denotes the size of the latent space. More specifically, it maps a sequence $\mathbf{x} = (x_1, x_2, \dots, x_T)$ to a sequence of same length T , namely $\mathbf{z} : (z_1, z_2, \dots, z_T)$, where each of the dimensions of (z_1, z_2, \dots, z_T) is the chosen hyperparameter E , which in our case is 64, the size of the embedding. For the sake of brevity, we would explain only one Transformer Encoder, and for a model with layers L , each of the stack is superimposed on the other one.

Each Transformer module consists of a attention block and a feed-forward block. The output of each of them is passed through a layer norm and a residual layer. So after both the attention block and the feed-forward block, if the input to a sub-block (attention F_a or feed-forward F_{ff} block) is a sequence x_b , instead of passing the output directly to the next module/sub-block, we pass along the block layer norm and the residual output x_{bo} as $x_{bo} = \text{LayerNorm}(x_b + F_{a/ff}(x_b))$ This follows the notion that layer-norm/skip connections help in better convergence/improved performance. We now describe each of the two sub-blocks that are part of the transformer block namely, i) multi-headed causal attention ii) feed-forward architecture

3.1.1. Multi-Headed Causal Attention

A multi-headed causal attention function can be described as a weighting function that decides how to get the output of each step. It learns a probabilistic score of how important each of the embedding, is while predicting the output. A multi-headed attention consists of first learning a probabilistic score. It is then multiplied with each of the inputs to determine how important each of the input is for prediction of the embedding for a position pos belonging to $1, 2, 3, \dots, T$. We use scaled-dot product attention as the type of attention mechanism. A query, key and a value vector is learned for each of the position for each of the inputs. This is done by implicitly learning matrices, W_Q , W_K and W_V to produce a query vector q , key vector k and value vector v for each of the inputs for a single attention head. We take the dot product of query and key vectors, the result is multiplied by a normalization factor (the inverse of square root of size of the vector as done in [20]), before taking a soft-max across all the inputs. Each of the value vector is multiplied by this score to get the output of the attention module. Mathematically, for a query matrix Q , key matrix K , and a value matrix V , it is defined as, $\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})$. We can also learn multiple such attention maps for h attention heads, defined as, $\text{MultiHeadAttention}(Q, K, V) = \text{Concat}(h_1, h_2, \dots, h_h)W_o$, where each of the attention heads h_i is defined as

$$\text{Attention}(Q_i, K_i, V_i) = \text{softmax}(\frac{Q_i K_i^T}{\sqrt{d_k}})$$

and W_o is a matrix learned during training.

3.1.2. Feed Forward Architecture & Positional Information

We weigh the saliency of each of the input signal via multi-headed attention for passing at a position pos . Each of the input at position pos is passed through a feed-forward architecture. We have the output of the feed-forward layers x_{bo} for an input x_b , for the dimension of feed-forward layers d_{ff} , in case of 2-layer network is, $FF(x_b) = \max(0, x_b W_1 + b_1) W_2 + b_2$. We apply this function identically at each of the inputs. As described in [20], to each of the inputs, positional encoding are added. As the input is passed on as a list, the model does not take into account the relative position, and thus the positional encoding are needed. For any position pos for the dimension i of the latent space, we use sinusoidal function, i.e. to each position pos and embedding dimension i in E , we add,

$$PE_{pos, 2i/2i+1} = \sin/\cos(pos/10000^{(2i/E)})$$

This adds positional information for each point in time, of input with dimension E , before passing thorough self-attention layers.

3.2. Adapting Transformer Architecture for raw waveforms

We adapt Transformer architectures using ideas from traditional signal processing. Since the Transformer has $\mathcal{O}(n^2)$ complexity w.r.t memory and computation requirements, we choose to follow the traditional route of windowing the signal. For all the experiments, as discussed before we work with 1s of audio input sampled at 16kHz yielding 16,000 samples.

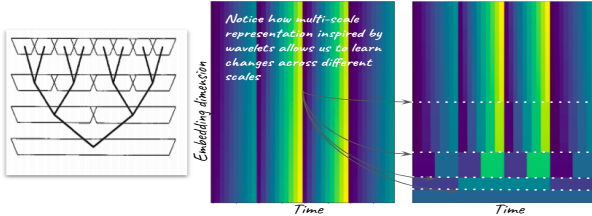


Figure 2: Core idea of wavelets utilizing multi-scale learning on (left) from [24], and using them to create a layer that operates on intermediate Transformer embeddings at various scales. We show a demo signal and we retain half of them, and modify the other half using variable sized windows.

The window length is chosen to be 25ms, choosing a non-overlapping rectangular window. The rectangular window provides the network an optimal windowing function which, as we will see in a few of the learned filters, adapts itself to a shape of hanning/hamming window. We fix the front end to be a dense layer of size 2048 neurons followed by another layer of size 64, primarily to adapt to the size of the embedding layer of Transformer. A single dense layer of size 2048 successfully learned a filter-back to learn a neural-time frequency representation as shown in [25]. This design was chosen as it produced state of the art results for an equally difficult problem of pitch estimation in polyphonic audio [25], with feed forward layers. Since Transformer layers consist only of attention + feed-forward blocks, we achieve an end-to-end architecture that does not have any convolutional operators. This yields a front end representation of 40 time steps each of size 64 dimensions, (64 being a hyper-parameter). We choose 6 layers of Transformer module with the size of latent code being 64 for each of the layers, and 8 attention head, with 128 dim 3-layer dense layer to convert to the desired feature space. For comparing with a smaller model, we choose 3-layers of Transformers with similar setup. The last layer of Transformer is reduced to a smaller dimension using average pooling across time. The output of the last dense layer of dimension 200, chosen same as number of output labels.

3.3. Transformer Architectures Inspired From CNNs: Pooling

We explored further performance enhancements to the baseline Transformer architecture proposed in the previous section. For this we draw inspiration of convolutional architectures used for the past decade to understand images [26] and audio [3]. The traditional models e.g. Resnet-50 [2], consists of using a combination of convolutional layers followed by pooling. The use of pooling layers has two advantages. It reduces the number of computations by reducing the size of inputs in the higher layers. More importantly it allows the higher-layer neurons to have much broader receptive field sizes, and allows the network to learn hierarchically ordered features, which is important for a particular problem. Pooled Trans-

formers outperform the baseline Transformer architecture, while retaining the same number of parameters. In our experiments average pooling performed significantly better than max-pooling, as it retains more information on the signal. As described in Figure 1, we use pooling across time after every two layers of Transformers, with stride 1, to reduce the dimensionality of the input by a factor of 2, and shows significant performance gain as compared to the original Transformer architecture without pooling.

3.4. Learning multi-scale embeddings

In this adaptation, we draw inspiration from wavelet decomposition and success of pooling layers. We explored if we can decompose the intermediate embeddings out of the Transformer, at multiple scale similar to idea of wavelet decomposition. In order to achieve it, we fix up our kernel to be average operation across *all* windows chosen at a particular level. Notice that we choose different window sizes at different dimensions of embedding along the time axis. The manner of implementation is again a design choice and there are several interesting ideas possible in future, including the choice of kernel. We draw inspiration from the work carried out in [24], as seen in Figure 2. We adapt the window size, in factors of 1,2,4,8 and so on, following a geometric progression. The value is assigned to *all* of the elements as opposed to reducing the size, as done in pooling thus retaining the same size. This operation is fully differentiable, and can be trained in end-to-end architectures. This is different than work carried out on spectral filtering [27], as we choose to operate firstly with variable window size as opposed to fixed windows, and secondly do not take explicit hand-crafted bands of filters. Additionally, we choose to model the space of embeddings-time hierarchically with only a few large windows, and large number of smaller window, most of them being 1 to retain the embeddings at their original scale. This retains the original transformer embeddings, with half of the embeddings unchanged, and tinkers with the other half. This combination has been at the core of wavelet transforms.

4. RESULTS & DISCUSSION

For all of the architectures, we only tuned learning rate to be consistent with the results shown in [21]. All of the Transformers have 6 layers (3 for small transformers) with 64 dim embeddings, and 3-Layer 128 neuron feed forward layers, and 8 attention head. The front end consists of 1024/2048 dimensional layer followed by a 64 dimensional dense layer for small and large transformers. We compared the same Transformer architectures with that of using i) pooling layers ii) multi-scale filters. We observed that even the smallest of the Transformer architectures outperform traditional convolutional architectures. This is quite significant, unlike problems in vision [10], where the margin was not as significant. Another observation is also that the performance keeps improving with more depth. All the models were trained using Tensorflow framework [28], with Huber Loss as the error criteria between the predictions and the ground truth, using Adam optimizer [29]. We see that the multi-scale approach outperforms the same transformer architecture without the intermediate layers, which is encouraging. However, is not able to beat the pooling layers. This may perhaps be due to ability of Transformers to better model smaller latent embeddings across time.

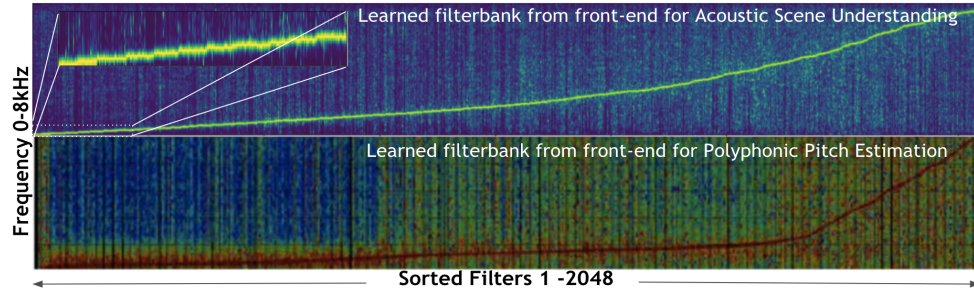


Figure 3: Sorted filters, learned by the front end, learns a problem specific non linear, non constant bandwidth filter-bank. This is shown by comparing it to that learned by the same front end for polyphonic pitch estimation as shown in [25].

Table 1: Comparison of various proposed architecture as shown in the table below for mean average precision (mAP) metric. We see how even baseline Transformer architectures without using any convolutional layers can outperform widely used CNN architectures for acoustic scene understanding by significant margins. [21]

Neural Model Architecture	mAP	# Param
CRNN [21]	0.417	0.96M
VGG-like [21]	0.434	0.27M
ResNet-18 [21]	0.373	11.3M
DenseNet-121 [21]	0.425	12.5M
Small Transformer	0.469	0.9M
Large 6- Layer Transformer	0.525	2.3M
Large Transformer with multi-scale filters	0.541	2.3M
Large 6- Layer Transformer with Pooling	0.537	2.3M

4.1. What the front end learns

We follow a strategy similar to that described in [25] to understand what the filters learn. We deploy the same front end in our work which is again a feed-forward layer consisting of 2048 neuron, followed by a 64-dim dense layer. This is similar to the analogy of getting a mel-like representation which is learnable end-to-end. After training, we take the first layer and sort the filter according to the peaks of their Fourier representation. We see that it manages to learn a non-linear, non-constant bandwidth filterbank as seen in Figure 3. We also see that with using the same front end for two different applications, namely for pitch estimation and acoustic scene understanding, the shape and the resolution of the learned filterbank is different. In addition, we can also see a step-wise pattern, which shows multiple filters assigned to the same frequency bin to account for the phase variations of the input signals. Figure 4 depicts a few chosen filters for the sake of discussion here. We observe a variety of ideas that can be interpreted from signal processing perspective, and also to take into account the characteristics of the input signal i.e. frequency, timbre, and energy. We can see, in center-top row, that a filter learns a pure sinusoidal basis of a certain frequency. Furthermore, it also manages to learn a windowing function that closely resembles hanning/hamming window. The filters in the left column present at the top-bottom are characteristic of an onset detector, which can, respectively, be a slow/rapid onset. Further, the filter present in the second row, third column shows a slowly moving signal, which may be latching onto the overall energy envelop of a signal for certain characteristic sounds. These correlations to traditional signal processing ideas present in these

filters are compelling.

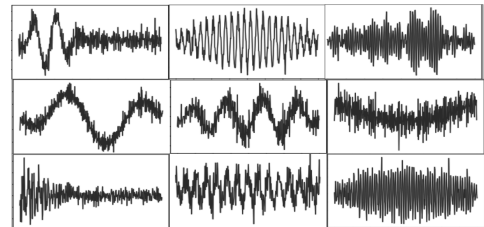


Figure 4: Filters learned from the first layer of front end show strong correlations to signal processing, particularly learning sinusoidal signals, onset detectors, energy envelops, and windowing functions

5. CONCLUSION & FUTURE WORK

We have shown here how a Transformer architecture without the use of convolutional filters can be adapted for large scale audio understanding. The work is promising, outperforming other convolutional architectures by a significant margin. We show our model can learn a time frequency front end that is adaptable to the particular problem of interest, in this case, large scale audio understanding. There are several possible research directions ahead. With the advancements in Transformer architectures such as switch transformers [30], and sparse transformers [31], these results would further improve. Additionally, with the success of unsupervised representation learning architectures for audio [1], it will be interesting to do large scale pre-training for making robust audio representations. It will be also be useful to explore a wider search over hyperparameters to increase the reported precision scores, which most likely will go up even further.

6. REFERENCES

- [1] P. Verma and J. Smith, “A framework for contrastive and generative learning of audio representations,” *arXiv preprint arXiv:2010.11459*, 2020.
- [2] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

- [3] J. F. Gemmeke, D. P. Ellis, D. Freedman, A. Jansen, W. Lawrence, R. C. Moore, M. Plakal, and M. Ritter, "Audio set: An ontology and human-labeled dataset for audio events," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2017, pp. 776–780.
- [4] A. Madani, B. McCann, N. Naik, N. S. Keskar, N. Anand, R. R. Eguchi, P.-S. Huang, and R. Socher, "Progen: Language modeling for protein generation," *arXiv preprint arXiv:2004.03497*, 2020.
- [5] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, *et al.*, "Language models are few-shot learners," *arXiv preprint arXiv:2005.14165*, 2020.
- [6] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.
- [7] C.-Z. A. Huang, A. Vaswani, J. Uszkoreit, N. Shazeer, I. Simon, C. Hawthorne, A. M. Dai, M. D. Hoffman, M. Dinulescu, and D. Eck, "Music transformer," *arXiv preprint arXiv:1809.04281*, 2018.
- [8] C. Sun, A. Myers, C. Vondrick, K. Murphy, and C. Schmid, "Videobert: A joint model for video and language representation learning," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 7464–7473.
- [9] R. Girdhar, J. Carreira, C. Doersch, and A. Zisserman, "Video action transformer network," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 244–253.
- [10] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.
- [11] N. Parmar, A. Vaswani, J. Uszkoreit, L. Kaiser, N. Shazeer, A. Ku, and D. Tran, "Image transformer," in *International Conference on Machine Learning*. PMLR, 2018, pp. 4055–4064.
- [12] P. Dhariwal, H. Jun, C. Payne, J. W. Kim, A. Radford, and I. Sutskever, "Jukebox: A generative model for music," *arXiv preprint arXiv:2005.00341*, 2020.
- [13] P. Verma and J. O. Smith, "Neural style transfer for audio spectrograms," *arXiv preprint arXiv:1801.01589*, 2018.
- [14] P. Verma, C. Chafe, and J. Berger, "Neuralogram: A deep neural network based representation for audio signals," *arXiv preprint arXiv:1904.05073*, 2019.
- [15] A. v. d. Oord, O. Vinyals, and K. Kavukcuoglu, "Neural discrete representation learning," *arXiv preprint arXiv:1711.00937*, 2017.
- [16] A. Baevski, H. Zhou, A. Mohamed, and M. Auli, "wav2vec 2.0: A framework for self-supervised learning of speech representations," *arXiv preprint arXiv:2006.11477*, 2020.
- [17] A. Baevski, S. Schneider, and M. Auli, "vq-wav2vec: Self-supervised learning of discrete speech representations," *arXiv preprint arXiv:1910.05453*, 2019.
- [18] A. Haque, M. Guo, and P. Verma, "Conditional end-to-end audio transforms," *arXiv preprint arXiv:1804.00047*, 2018.
- [19] A. Haque, M. Guo, P. Verma, and L. Fei-Fei, "Audio-linguistic embeddings for spoken sentences," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 7355–7359.
- [20] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *arXiv preprint arXiv:1706.03762*, 2017.
- [21] E. Fonseca, X. Favory, J. Pons, F. Font, and X. Serra, "Fsd50k: an open dataset of human-labeled sound events," *arXiv preprint arXiv:2010.00475*, 2020.
- [22] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, *et al.*, "Scipy 1.0: fundamental algorithms for scientific computing in python," *Nature methods*, vol. 17, no. 3, pp. 261–272, 2020.
- [23] G. Klein, Y. Kim, Y. Deng, J. Senellart, and A. M. Rush, "Opennmt: Open-source toolkit for neural machine translation," in *Proc. ACL*, 2017. [Online]. Available: <https://doi.org/10.18653/v1/P17-4012>
- [24] J. Berger, R. R. Coifman, and M. J. Goldberg, "Removing noise from music using local trigonometric bases and wavelet packets," *Journal of the Audio Engineering Society*, vol. 42, no. 10, pp. 808–818, 1994.
- [25] P. Verma and R. W. Schafer, "Frequency estimation from waveforms using multi-layered neural networks," in *INTER-SPEECH*, 2016, pp. 2165–2169.
- [26] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.
- [27] A. Tamkin, D. Jurafsky, and N. Goodman, "Language through a prism: A spectral approach for multiscale language representations," *Advances in Neural Information Processing Systems*, vol. 33, 2020.
- [28] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, *et al.*, "Tensorflow: A system for large-scale machine learning," in *12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16)*, 2016, pp. 265–283.
- [29] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.
- [30] W. Fedus, B. Zoph, and N. Shazeer, "Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity," Jan 2021. [Online]. Available: <https://arxiv.org/abs/2101.03961>
- [31] R. Child, S. Gray, A. Radford, and I. Sutskever, "Generating long sequences with sparse transformers," *arXiv preprint arXiv:1904.10509*, 2019.