Wake Word Detection Using Transformers

11-685 Introduction to Deep Learning

Adeep Biswas, Daniel Hoskins, Mukunda Das, Sanjan Das
Language Technology Institute
Carnegie Mellon University
Pittsburgh, PA 15213
{ adeepb, dhoskins, mukundad, sanjand } @andrew.cmu.edu

August 8, 2022

Abstract

Transformers have achieved very promising results in the field of keyword spotting. Notably, the Keyword Transformer (KWT) of 2021 set the state of the art, delivering better accuracy and comparable latency when compared with earlier convolutional and attention approaches. In this work, we improve the KWT by modifying the activation functions and normalizations. These modifications reduce the latency of the model while maintaining comparable keyword classification accuracy.

1 Introduction

Voice-enabled assistants and smart devices have become commonplace among people these days. These small-footprint embedded edge devices require technology to detect a "keyword" or a "wake word" spoken by the user, named so since they "wake" up the device. Detection of the keyword triggers more complex tasks and processing that generally occurs on the cloud. Some popular examples of keywords are "Okay Google" and "Hey Siri". Keyword spotting models benefit from small model sizes and low detection latency in order to enable voice assistants properly. In this project, our goal is to explore improvement in the application of transformers for the purpose of wake word detection.

There are some key requirements for keyword spotting models since they need to be run on edge devices. The model needs to be small, have low latency, be highly energy efficient and also be able to learn from limited data (especially when the keyword set gets extended). Accuracy of detection is important because it helps edge devices to not be triggered with similar-sounding words or adversarial audio attacks. Keywords are generally very short in syllables and length and can be spoken with a range of modulations. Classification is a challenge since there will be significantly fewer features to extract.

The machine learning community has been leveraging the Transformer architecture across domains and improving on previous neural network architecture designs. Over the last two years, Transformers have been the model of choice for keyword spotting. Transformers, with their simple architecture, have been able to outperform more complex models that mix convolutional, recurrent, and attentive layers. Our work explores the enhancement of the Keyword Transformer. This architecture was inspired by the Vision Transformer. It utilizes self-attention on time-domain patches of mel spectrograms.

Our work looks at changing the design of the Keyword Transformer, in particular, its GELU activation functions and layer normalizations, in an attempt to improve the model's latency while minimizing the reduction in the model's testing accuracy.

2 Literature Review

Traditionally, probabilistic techniques such as Hidden Markov Models-based generative models have commonly been applied for keyword detection problems by using discriminative training methods [Wilpon et al., 1990]. One of the main advantages of employing HMM-based models is that they are computationally efficient, making them suitable for edge devices. However, the recent advancement in deep learning, along with the growing availability of computational resources, has allowed deep neural network-based architectures to outperform HMM-based models in terms of accuracy [Chen et al., 2014]. Researchers have tried combining the benefits of HMM model with the accuracy of neural networks to propose hybrid DNN-HMM models where the DNN computes the state probabilities, and the HMM decoder combines these probabilities using dynamic programming. Such hybrid models, when trained end to end rather than independently component-wise, have shown great improvement in performance [Ashish Shrivastava and Tuzel, 2021]. However, the state of the art results in terms of accuracy still belong largely to neural network-based approaches.

Among neural network approaches, the research trend has been to explore the application of convolutional neural networks, recurrent neural networks, as well as some hybrid neural networks for the purpose of keyword detection, with promising results in all approaches [Berg et al., 2021]. These neural network-based approaches typically require a pre-processing pipeline to extract mel-frequency cepstrum coefficients (MFCC) from the audio signals, and certain pre-training or data synthesis techniques have been shown to improve these model performances [Davis and Mermelstein, 1980].

In the Convolutional Neural Networks domain, DS-CNN introduced the idea of depthwise separability to significantly improve the performance of vanilla CNN-based architecture and achieve the state of the art keyword detection result of that time [Yundong Zhang and Chandra, 2017]. Over the past few years, several improvements have been suggested over this, such as using broadcasted residual learning where the network configures most of the residual functions as 1D temporal convolution in addition to the standard spatial convolutions to boost

model accuracy [Byeonggeun Kim and Sung, 2021] and using intensive noise data augmentation techniques to make the CNN architectures more robust [Majumdar and Ginsburg., 2020]. The current benchmark on the Google Speech Commands dataset is held by the TripletLoss-res15 model, which introduced the idea of using triplet loss-based efficient embeddings for keyword signal representation and recognition [Vygon and Mikhaylovskiy, 2021].

In the Recurrent Neural Network domain, Attention-RNN was the first significant model that beat the accuracy score of DS-CNN to demonstrate the importance of recurrence and attention in the keyword detection challenge [Douglas Coimbra de Andrade and Bernkopf., 2018]. Rybakov et al. [2020], 2020 added the concept of multi-headed attention to further improve the classification accuracy while also making the distinction between streaming and non-streaming neural network models for keyword detection. The results achieved by using RNNs have further propelled researchers to apply recurrence in CNN-based models as well as using temporal convolutions, as discussed above.

In the last few years, the application of transformers for keyword detection has become the growing direction of research with the advent of promising models such as Keyword Transformer-3, which has been able to achieve SOTA accuracy scores [Berg et al., 2021]. The main advantage introduced with KWT-3 is that it can be trained end to end without requiring any pre-training or augmented data while also being much less computation-intensive than other SOTA models, which mix convolutional, recurrent, and attention-based layers [Berg et al., 2021]. Deokjin Seo and Jung [2021] have also shown that transformer-based encoders can be used with transfer learning from larger encoders, pre-trained on massive speech data corpus, in cases of sparse data availability to achieve highly accurate results. In our work, we have taken the Keyword Transformer as our baseline model. To further explore the application of transformers for wake word detection, we aim to improve the Keyword Transformer's accuracy and efficiency.

3 Dataset

Our project uses Version 2 of the Google Speech Commands Dataset Warden [2018]. This dataset contains 24 target keywords that range from utterances of the digits zero to nine, as well as "Yes", "No", "Up", "Down", "Left", "Right", "On", "Off", "Stop", "Go", "Backward", "Forward", "Follow" and "Learn". In addition, there are two other classes for labeling non-trigger words (so that a model trained on this dataset can learn to ignore them) and background noise (such as that of a running tap, dish washing, etc.).

There are 105,829 clips in the dataset. Of these, 84,843 clips are used for training, 11,005 clips are used for testing, and the remaining 9,981 clips are used for validation. The distribution of datapoints across different class labels is between 3,800 - 4,005 for all labels except "Forward", "Backward", "Follow" and "Learn", which have around 1,500 points each. The background noise and non-trigger word clips are included during training, testing and validation time.

All audio clips are in the .wav format and the size of the entire dataset is around 3.3GB.

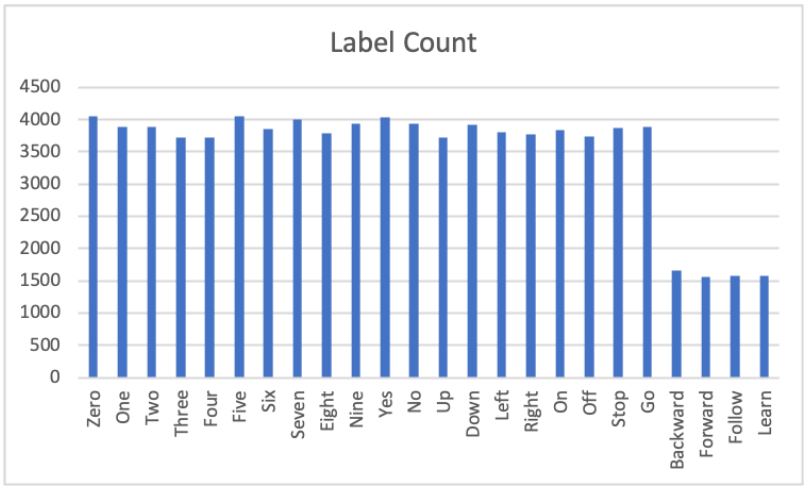

Figure 1: Class label distribution in the dataset, not including that of noise or non-trigger words

4 Baseline Experiment

4.1 Motivation

Several reasons make the Keyword Transformer a compelling baseline. First, it had achieved state of the art performance on multiple tasks involving the Google Speech Commands Dataset when it was released in 2021 and has since garnered demand for improvements. Second, the multi-head self attention transformer-based architecture is relatively simple compared to many state of the art deep learning applications. Many of the complex architectures, data augmentation strategies, and normalization techniques have, by and large, not yet been applied to keyword spotting. This provides multiple clear directions for exploration. Finally, the architecture can be implemented within a matter of days/weeks, allowing for significant exploratory progress to be made in a relatively short time. We discuss our future directions in depth in the subsequent sections.

4.2 Description of baseline

Our implementation of the baseline is built off the official repository for the Keyword Transformer at https://github.com/ARM-software/keyword-transformer/.

The model attempts to detect spoken instances of any of the following classes: silence, unknown words, or any one of a predefined set of keywords, K. K is the

set that includes all keywords of interest. The unknown words class includes all words not in K. This description is consistent with a typical keyword spotting paradigm.

4.2.1 Data Preprocessing

Both frequency domain and time domain representations are informative and important for success in speech recognition tasks. Thus, the input data type must capture both types of information. Furthermore, the Mel-scale captures human perceptual differences by mapping frequency changes to a linear representation of their perceived change in frequency - a desirable property for speech recognition tasks. Mel-spectrograms are chosen as the input data type because they capture time information as well as frequency information (frequency is captured on the Mel-scale).

The data from the Google Speech Commands dataset was preprocessed to generate Mel-spectrograms from the raw audio. As was done with Berg et al. [2021], we partition the mel spectrograms into patches (split at certain points on the time axis). This technique was inspired by the work with the Vision Transformer and it is done so that the attention mechanism can attend to different points in time (i.e., the patches are attended to). The final representation is a mel-scale spectrogram of 98 time windows with 40 mel-scale frequencies.

4.2.2 Model description

First, non-overlapping time-domain patches of the mel-scale spectrogram are projected into the frequency domain using a linear projection matrix, W_0. These features are then prepended with a "learned class token", X_{class} and appended with a positional embedding matrix $X_{\text{positional}}$. The result, denoted X_0, serves as the inputs to the encoder.

Similar to the class token used in the Bert model, the (learned) X_{class} token is ultimately input into the decoder and "transformed into a class prediction" [Dosovitskiy et al., 2021]. Thus, the class token must be capable of representing the "whole spectrogram" Berg et al. [2021]. The final transformer input is shown in 1.

$$X_0 = [X_{class}; XW_0] + X_{pos} \tag{1}$$

The encoder includes two types of blocks: multi-head attention and MLP-GELU blocks. The time windows are the objects of the attention (i.e., time windows are attended by other time windows). The output of the multi-head attention blocks is linearly projected using the projection matrix W_p. The equations (originally developed by *berg*) that describe these relationships for the lth transformer block are shown in 2. For more detail on the attention equations, please see Berg et al. [2021]. Outputs from both the multi-head attention blocks and MLP blocks are run through Layernorm (post-norm scheme).

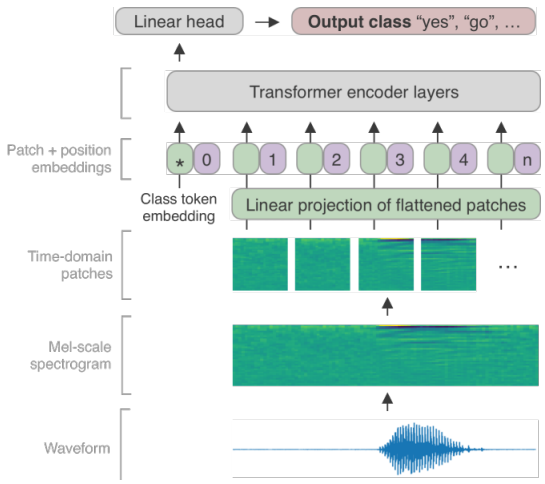

Figure 2: Keyword Transformer architecture Berg et al. [2021]

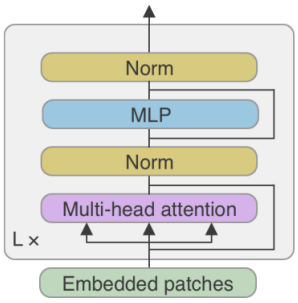

Figure 3: The Keyword Transformer uses the PostNorm architecture for its encoder block

$$SA(X_l) = \text{Softmax}\left(\frac{QK^T}{\sqrt{d_h}}\right)V \tag{2}$$

where Q are the queries, K are the keys, V are the values. Q, K, and V are given by:

$$Q = X_l W_Q \tag{3}$$

$$K = X_l W_K \tag{4}$$

$$V = X_l W_V \tag{5}$$

The outputs of the lth transformer block are given by:

$$\tilde{X}_l = LN(MSA(X_{l-1}) + X_{l-1}), \qquad l = 1, ..., L \tag{6}$$

$$X_l = LN(MLP(\tilde{X}_l) + \tilde{X}_l), \qquad l = 1, ..., L \tag{7}$$

4.2.3 Evaluation metric

The loss function used is the cross entropy between the predicted class probabilities and the true class labels for an audio sequence. The accuracy is calculated as the number of sequence predictions that have the correct class. This evaluation was chosen by the authors of the original paper and therefore, we continued with it in order to make direct comparisons with the original architecture in terms of validation accuracy. To compare the latency of the different model architectures, we chose to compute the training time per epoch and inference time per input audio sequence of the model in seconds and milliseconds. The training time is the time taken for the model to train on the whole training data set once, while inference time is the time taken by the model to receive an input and provide an output prediction.

5 Experiments

Our experiments to investigate latency improvements focus on modifying the transformer block. For all experiments, we used a Tesla T4 GPU on an AWS EC2 instance.

5.1 Replacing the activation function

As per the official documentation of Tensorflow, GELU activation is computed using the following equation -

$$x * P(X <= x) = 0.5 * x * (1 + erf(x/\sqrt{2})) \tag{8}$$

where P(X) lies in N(0, 1) and erf(x) is the Gaussian error function given by

-

$$erf(x) = \frac{2}{\sqrt{\pi}} \int_0^x x^{-t^2} \, dt \tag{9}$$

This Gaussian error function has no direct closed form solution and therefore requires further approximations. This makes GELU activation slow to compute.

Swish

Swish [Ramachandran et al., 2017] is a relatively new activation function that was announced by the Google Brain team in 2017 and has been shown to outperform ReLU and its variations in deeper model architectures across various domains. Swish is unbounded above and bounded below and is also non-monotonic and self-gated. This is the main difference in its behavior over activation functions like ReLU and GELU, enhancing the expression of input data and weight to be learned while also being very simple to compute.

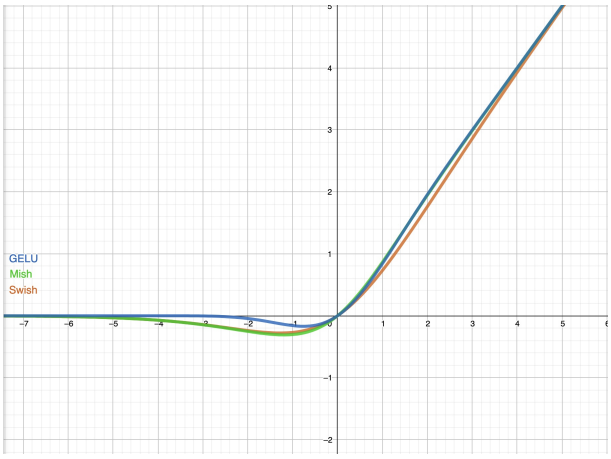

Figure 4: Activation functions in comparison

Swish's equation is shown below

$$f(x) = x * \text{sigmoid}(\beta * x) \tag{10}$$

Besides being faster in computation, it also makes sense to use Swish because Swish is essentially an approximation of GELU when the β is set to 1.702 in the Swish equation. This allows Swish to preserve and replicate the properties of GELU, allowing a speed up in the model training and inference time while also retaining a comparable performance in terms of accuracy from the original model architecture.

Mish

Mish [Misra, 2020] furthermore was inspired by Swish and came out in 2019, which was able to beat GELU and Swish in a lot of different tasks as well. While Mish borrows a lot of properties from Swish, the authors of the original paper hypothesized that it is able to outperform Swish because of the higher smoothness at nearly all points on the curve, allowing for better information propagation deeper into the neural network, and thus better accuracy and generalization.

$$f(x) = x * \tanh(\ln(1 + e^x)) \tag{11}$$

Mish shares a lot of common properties with Swish, such as being unbounded above zero and bounded below zero with a non-monotonic curve and having a self-gated mechanism, allowing it to have a behavior similar to Swish while having shown better performance yield for certain deep learning tasks empirically as stated above.

5.2 Replacing the normalization

One area of focus in the field of optimizing transformers has been on efficient normalizations. Sun et al. [2020] introduced the MobileBERT, a compact version of BERT optimized for mobile devices with low resources. MobileBERT replaced BERT's layer normalization with a normalization they termed NoNorm.

5.2.1 NoNorm

NoNorm is an element-wise linear transformation, as expressed in Equation 12 where $\odot$ represents the Hadamard product. MobileBERT reported a three times reduction in latency with NoNorm. We investigate if similar improvements can be found for the Keyword Transformer.

$$\text{NoNorm}(h) = \gamma \odot h + \beta \tag{12}$$

6 Results and Discussion

Model	Val Accuracy (%)	Inference time reduction (%)	Training time reduction (%)
Baseline	97.633	-	-
Swish	97.289	3.869	3.645
Mish	97.426	0.707	0.936
NoNorm	97.174	3.211	2.321
NoNorm and Swish	**96.53**	**10.175**	**11.059**
NoNorm and Mish	96.53	2.99	4.9

Table 1: Collection of experiment results.

Our experiment results are shown in Table 1. We found that the combination of replacing layer normalization with NoNorm and GELU with Swish resulted in the greatest reduction in latency. It is worth pointing out that it also showed the largest decrease in the validation accuracy.

One particularly interesting result is manifested in the inference time difference exhibited by the combination of NoNorm and Swish. When only NoNorm is applied, the latency reduction is 3.211%. When only Swish is applied, the latency reduction is 3.869%. However, when both NoNorm and Swish are applied, the latency reduction is 10.175Ċlearly, the latency reduction from the latter is greater than the sum of their individual contributions. There's no immediately apparent explanation for this interesting result, and determining one is outside the scope of this work. However, one might conjecture that the model's architecture allows for a multiplicative effect of these two individual impacts. Determining whether

this is true and, if so, the reason for it is a topic that can be explored in future work.

7 Conclusion

Keyword spotting is a task that is crucial to the success of voice-enabled edge devices. Since 2021, the Transformer architecture has been regarded as the state of the art in this field. Our work assessed the impact of small changes to the Transformer architecture in an attempt to improve its latency while retaining accuracy. From our experiments, we conclude that replacing layer normalizations and GELU activation with the simpler NoNorm and Swish activation decreases inference time by around 10% while causing only a minor decrease in test accuracy.

Since the introduction of the Keyword Transformer, there have been many other Transformer-based architectures used for keyword spotting. Our work did not test those architectures. We are also aware that there are other techniques to improve latency (pruning is a particularly notable technique). As a next step, various pruning methodologies could be experimented with to understand the accuracy-latency trade-off.

8 Code

https://github.com/Sanjan611/keyword-transformer

References

Jay G. Wilpon, Lawrence R. Rabiner, Chin-Hui Lee, , and E.R. Goldman. Automatic recognition of keywords in unconstrained speech using hidden markov models. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 1990. URL `https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=103088`.

Guoguo Chen, Carolina Parada, , and Georg Heigold. Small-footprint keyword spotting using deep neural networks. *Proc. ICASSP*, 2014. URL `https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/42537.pdf`.

Chandra Dhir Devang Naik Ashish Shrivastava, Arnav Kundu and Oncel Tuzel. Optimize what matters: Training dnn-hmm keyword spotting model using end metric. *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021. URL `https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9414797`.

Axel Berg, Mark O'Connor, and Miguel Tairum Cruz1. Keyword transformer: A self-attention model for keyword spotting. *INTERSPEECH 2021*,

2021. URL https://www.isca-speech.org/archive/pdfs/interspeech_2021/berg21_interspeech.pdf.

S. Davis and P. Mermelstein. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE transactions on acoustics, speech, and signal processing*, 1980. URL https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1163420.

Liangzhen Lai Yundong Zhang, Naveen Suda and Vikas Chandra. Hello edge: Keyword spotting on microcontrollers. *arXiv preprint arXiv:1711.07128*, 2017. URL https://arxiv.org/pdf/1711.07128v3.pdf.

Jinkyu Lee Byeonggeun Kim, Simyung Chang and Dooyong Sung. Broadcasted residual learning for efficient keyword spotting. *arXiv preprint arXiv:2106.04140*, 2021. URL https://arxiv.org/pdf/2106.04140v2.pdf.

Somshubra Majumdar and Boris Ginsburg. Matchboxnet: 1d time-channel separable convolutional neural network architecture for speech commands recognition. *arXiv preprint arXiv:2004.08531*, 2020.

Roman Vygon and Nikolay Mikhaylovskiy. Learning efficient representations for keyword spotting with triplet loss. *International Conference on Speech and Computer. Springer, Cham*, 2021. URL https://arxiv.org/pdf/2101.04792.pdf.

Martin Loesener Da Silva Viana Douglas Coimbra de Andrade, Sabato Leo and Christoph Bernkopf. A neural attention model for speech command recognition. *arXiv preprint arXiv:1808.08929, 2018.*, 2018. URL https://arxiv.org/pdf/1808.08929v1.pdf.

Oleg Rybakov, Natasha Kononenko, Niranjan Subrahmanya, Mirko Visontai, and Stella Laurenzo. Streaming keyword spotting on mobile devices. *arXiv preprint arXiv:2005.06720*, 2020. URL https://arxiv.org/pdf/2005.06720v2.pdf.

Heung-Seon Oh Deokjin Seo and Yuchul Jung. Wav2kws: Transfer learning from speech representations for keyword spotting. *IEEE Access 9*, 2021. URL https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9427206.

Pete Warden. Speech commands: A dataset for limited-vocabulary speech recognition, 2018. URL https://arxiv.org/abs/1804.03209.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, and etc. An image is worth 16x16 words: Transformers for image recognition at scale. *International Conference on Learning Representations*, 2021. URL https://arxiv.org/pdf/2010.11929.pdf.

Prajit Ramachandran, Barret Zoph, and Quoc V. Le. Swish: A self-gated activation function. *arXiv:1710.05941v1 [cs.NE] 16 Oct 2017*, 2017. URL https://arxiv.org/pdf/1710.05941v1.pdf.

Diganta Misra. Mish: A self regularized non-monotonic activation function. *ArXiv:1908.08681v3 [cs.LG] 13 Aug 2020*, 2020. URL `https://arxiv.org/pdf/1908.08681.pdf`.

Zhiqing Sun, Hongkun Yu, Xiaodan Song, Renjie Liu, Yiming Yang, and Denny Zhou. Mobilebert: a compact task-agnostic bert for resource-limited devices. *arXiv preprint arXiv:2004.02984*, 2020.