Keyword Transformer: A Self-Attention Model for Keyword Spotting

Axel Berg[1,2,*], *Mark O'Connor*[1,*], *Miguel Tairum Cruz*[1]

[1]Arm ML Research Lab, [2]Lund University

{axel.berg, mark.oconnor, miguel.tairum-cruz}@arm.com

Abstract

The Transformer architecture has been successful across many domains, including natural language processing, computer vision and speech recognition. In keyword spotting, self-attention has primarily been used on top of convolutional or recurrent encoders. We investigate a range of ways to adapt the Transformer architecture to keyword spotting and introduce the Keyword Transformer (KWT), a fully self-attentional architecture that exceeds state-of-the-art performance across multiple tasks without any pre-training or additional data. Surprisingly, this simple architecture outperforms more complex models that mix convolutional, recurrent and attentive layers. KWT can be used as a drop-in replacement for these models, setting two new benchmark records on the Google Speech Commands dataset with 98.6% and 97.7% accuracy on the 12 and 35-command tasks respectively.[1]

Index Terms: speech recognition, keyword spotting, Transformers

1. Introduction

Recent works in machine learning show that the Transformer architecture, first introduced by Vaswani et al. [1], is competitive not only in language processing, but also in e.g. image classification, [2, 3, 4], image colorization [5], object detection [6], video classification [7] and multi-agent spatiotemporal modeling [8]. This can be seen in the light of a broader trend, where a single neural network architecture generalizes across many domains of data and tasks.

Attention mechanisms have also been explored for speech recognition [9, 10, 11], but only as an extension to other architectures, such as convolutional or recurrent neural networks.

Inspired by the strength of the simple Vision Transformer (ViT) model [2] in computer vision and by the techniques that improves its data-efficiency [3], we propose an adaptation of this architecture for keyword spotting and find that it matches or outperforms existing models on the much smaller Google Speech Commands dataset [12] without additional data.

We summarize our main contributions as follows:

1. An investigation into the application of the Transformer architecture to keyword spotting, finding that applying self-attention is more effective in the time domain than in the frequency domain.
2. We introduce the Keyword Transformer, as illustrated in Figure 1, a fully self-attentional architecture inspired by ViT [2] that can be used as a drop-in replacement for existing keyword spotting models and visualize the effect of the learned attention masks and positional embeddings.
3. An evaluation of this model across several tasks using the Google Speech Commands dataset with comparisons to state-of-the-art convolutional, recurrent and attention-based models.
4. An analysis of model latency on a mobile phone, showing that the Keyword Transformer is competitive in edge use cases.

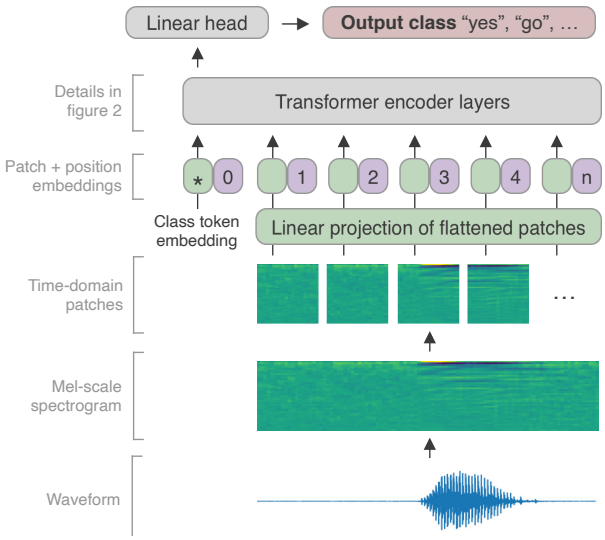

Figure 1: *The Keyword Transformer architecture. Audio is preprocessed into a mel-scale spectrogram, which is partitioned into non-overlapping patches in the time domain. Together with a learned class token, these form the input tokens for a multilayer Transformer encoder. As with ViT [2], a learned position embedding is added to each token. The output of the class token is passed through a linear head and used to make the final class prediction.*

2. Related Work

2.1. Keyword Spotting

Keyword spotting is used to detect specific words from a stream of audio, typically in a low-power always-on setting such as smart speakers and mobile phones. To achieve this, audio is processed locally on the device. In addition to detecting target words, classifiers may also distinguish between "silence" and "unknown" for words or sounds that are not in the target list.

In recent years, machine learning techniques, such as deep (DNN), convolutional (CNN) and recurrent (RNN) neural networks, have proven to be useful for keyword spotting. These networks are typically used in conjunction with a preprocessing pipeline which extracts the mel-frequency cepstrum coefficients (MFCC) [13]. Zhang et al. [14] investigated sev-

*Equal contribution.

[1]Preprint. Submitted to INTERSPEECH.

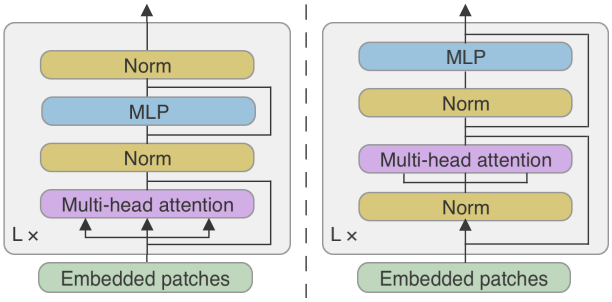

Figure 2: *The PostNorm (left) and PreNorm (right) Transformer encoder architectures. KWT uses a PostNorm encoder.*

eral small-scale network architectures and identified depthwise-separable CNN (DS-CNN) as providing the best classification accuracy tradeoff for memory footprint and computational resources. Other works have improved upon this result using synthesized data [15], temporal convolutions [16, 17], and self-attention [9].

Rybakov et al. [10] created a framework for training streaming models that continually accept input and achieved a new state of the art result on Google Speech Commands using MHAtt-RNN, a non-streaming CNN, RNN and multi-headed (MH) self-attention model.

2.2. Self-Attention and the Vision Transformer

Dosovitskiy et al. introduced the Vision Transformer (ViT) [2] and showed that Transformers can learn high-level image features by computing self-attention between different image patches. This simple approach outperformed CNNs but required pre-training on large datasets. Touvroun et al. [3] improved data efficiency using strong augmentation, careful hyperparameter tuning and token-based distillation.

While self-attention has previously been used on top of convolutional and recurrent feature extractors for keyword spotting, to the best of our knowledge fully-attentional models based on the Transformer architecture [1] have not been investigated. Our approach is inspired by ViT, in the sense that we use patches of the audio spectrogram as input. We restrict ourselves to a non-streaming setting in this work, noting that others have previously investigated extensions of Transformers to a streaming setting [18].

3. The Keyword Transformer

3.1. Model Architecture

Let $X \in \mathbb{R}^{T \times F}$ denote the output of the MFCC spectrogram, with time windows $t = 1, ..., T$ and frequencies $f = 1, ..., F$. The spectrogram is first mapped to a higher dimension d, using a linear projection matrix $W_0 \in \mathbb{R}^{F \times d}$ in the frequency domain. In order to learn a global feature that represents the whole spectrogram, a learnable class embedding $X_{\text{class}} \in \mathbb{R}^{1 \times d}$ is concatenated with the input in the time-domain. Then a learnable positional embedding matrix $X_{\text{pos}} \in \mathbb{R}^{(T+1) \times d}$ is added, such that the input representation fed into the Transformer encoder is given by

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

The projected frequency-domain features are then fed into a sequential Transformer encoder consisting of L multi-head attention (MSA) and multi-layer perceptron (MLP) blocks. In the l:th Transformer block, queries, keys and values are calculated as $Q = X_l W_Q$, $K = X_l W_K$ and $V = X_l W_V$ respectively, where $W_Q, W_K, W_V \in \mathbb{R}^{d \times d_h}$ and d_h is the dimensionality of each attention-head. The self attention (SA) is calculated as

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

The MSA operation is obtained by linearly projecting the concatenated output, using another matrix $W_P \in \mathbb{R}^{kd_h \times d}$, from the k attention heads.

$$\text{MSA}(X_l) = [\text{SA}_1(X_l); \text{SA}_2(X_l); ...; \text{SA}_k(X_l)]W_P \tag{3}$$

In our default setting, we use the PostNorm [1] Transformer architecture as shown in Figure 2, where the Layernorm (LN) [19] is applied after the MSA and MLP blocks, in contrast to the PreNorm [20] variant, where LN is applied first. This decision is discussed further in the ablation section. As is typical for Transformers, we use GELU [21] activations in all MLP blocks.

In summary, the output of the l:th Transformer block is given by

$$\tilde{X}_l = \text{LN}(\text{MSA}(X_{l-1}) + X_{l-1}), \qquad l = 1, ..., L \tag{4}$$

$$X_l = \text{LN}(\text{MLP}(\tilde{X}_l) + \tilde{X}_l), \qquad l = 1, ..., L \tag{5}$$

At the output layer, the class embedding is fed into a linear classifier. Our approach treats time windows in a manner analogous to the handling of image patches in ViT. Whereas in ViT, the self-attention is computed over image patches, the attention mechanism here takes place in the time-domain, such that different time windows will attend to each other in order to form a global representation in the class embedding.

The model size can be adjusted by tuning the parameters of the Transformer. Following [3], we fix the number of sequential Transformer encoder blocks to 12, and let $d/k = 64$, where d is the embedding dimension and k is the number of attention heads. By varying the number of heads as $k = 1, 2, 3$, we end up with three different models as shown in Table 1.

Table 1: *Model parameters for the KWT architecture.*

Model	dim	mlp-dim	heads	layers	# parameters
KWT-1	64	256	1	12	607K
KWT-2	128	512	2	12	2,394K
KWT-3	192	768	3	12	5,361K

3.2. Knowledge Distillation

As introduced by [22], knowledge distillation uses a pre-trained teacher's predictions to provide an auxiliary loss to the student model being trained. We follow [3] and implement this by appending an additional learned distillation token to the input. At the output layer the distillation token is fed into a linear classifier and trained using the hard labels predicted by the teacher.

In contrast to [22], this teacher is typically weaker than the student and in contrast to [23] the teacher receives the same augmentation of the input as the student. In contrast to both, the hard predictions of the teacher labels are used as targets. Let Z_{sc} be the logits of the student class token, Z_{sd} be the logits of the student distillation token and Z_{t} be the logits of the teacher

Table 2: *Hyperparameters used in all experiments.*

Training	
Training steps	23,000
Batch size	512
Optimizer	AdamW
Learning rate	0.001
Schedule	Cosine
Warmup epochs	10
Regularization	
Weight decay	0.1
Label smoothing	0.1
Dropout	0

Pre-processing	
Time window length	30 ms
Time window stride	10 ms
#DCT Features	40
Data augmentation	
Time shift [ms]	[-100, 100]
Resampling	[0.85, 1.15]
Background vol.	0.1
#Time masks	2
Time mask size	[0,25]
#Frequency masks	2
Frequency mask size	[0,7]

model. The overall loss becomes

$$\mathcal{L} = \frac{1}{2}\mathcal{L}_{\mathrm{CE}}(\psi(Z_{\mathrm{sc}}), y) + \frac{1}{2}\mathcal{L}_{\mathrm{CE}}(\psi(Z_{\mathrm{sd}}), y_{\mathrm{t}}), \qquad (6)$$

where $y_{\mathrm{t}} = \mathrm{argmax}(Z_{\mathrm{t}})$ are the hard decision of the teacher, y are the ground-truth labels, ψ is the softmax function and $\mathcal{L}_{\mathrm{CE}}$ is the cross-entropy loss. At inference time the class and distillation token predictions are averaged to produce a single prediction. In all experiments, we use MHAtt-RNN as a teacher and denote hard-distillation models with KWT ⚗.

4. Experiments

4.1. Keyword Spotting on Google Speech Commands

We provide experimental results on the Google Speech Commands dataset V1 and V2 [12]. Both datasets consist of 1 second long audio snippets, sampled at 16 kHz, containing utterances of short keywords recorded in natural environments. V1 of the dataset contains 65,000 snippets of 30 different words, whereas V2 contains 105,000 snippets of 35 different words. The 12-label classification task uses 10 words: "up", "down", "left", "right", "yes", "no", "on", "off", "go", and "stop", in addition to "silence" and "unknown", where instances of the latter is taken from the remaining words in the dataset, whereas the 35-label task uses all available words. We use the same 80:10:10 train/validation/test split as [12, 14, 10] for side-by-side comparisons. We adhere as closely as possible to the evaluation criteria of [10], and for each experiment, we train the model three times with different random initializations.

As our intention is to explore the extent to which results using Transformers from other domains transfer to keyword spotting, we follow the choices and hyperparameters from [3] as closely as possible, with the notable exception that we found increasing weight decay from 0.05 to 0.1 to be important. Furthermore, we use the same data pre-processing and augmentation policy as in [10], which consists of random time shifts, resampling, background noise, as well as augmenting the MFCC features using SpecAugment [24]. We train our models over the same number of total input examples as MHAtt-RNN (12M) to allow a fair comparison. For clarity, the hyperparameters used in all experiments are reported in Table 2.

The results are shown in Table 3, where for our own results, we report a 95% confidence interval for the mean accuracy over all three model evaluations. Our best models match or surpass the previous state-of-the-art accuracies, with significant improvements on both the 12-label and 35-label V2-datasets. In general, Transformers tend to benefit more from large amounts of data, which could explain why KWT does not outperform MHAtt-RNN on the smaller V1-dataset. Nevertheless, we also note that knowledge distillation is effective in improving the accuracy of KWT in most scenarios.

Table 3: *Accuracy on Speech Commands V1 [25] and V2 [26].*

Model	V1-12	V2-12	V2-35
DS-CNN [14]	95.4		
TC-ResNet [16]	96.6		
Att-RNN [9]	95.6	96.9	93.9
MatchBoxNet [17]	97.48 ±0.11	97.6	
Embed + Head [15]		97.7	
MHAtt-RNN [10]	97.2	98.0	
Res15 [27]		98.0	96.4
MHAtt-RNN (Ours)	**97.50** ±0.29	98.36 ±0.13	97.27 ±0.02
KWT-3 (Ours)	97.24±0.24	**98.54** ±0.17	97.51 ±0.14
KWT-2 (Ours)	97.36 ±0.20	98.21 ±0.06	97.53 ±0.07
KWT-1 (Ours)	97.05 ±0.23	97.72 ±0.01	96.85 ±0.07
KWT-3 ⚗ (Ours)	**97.49** ±0.15	**98.56** ±0.07	97.69 ±0.09
KWT-2 ⚗ (Ours)	97.27 ±0.08	98.43 ±0.08	**97.74** ±0.03
KWT-1 ⚗ (Ours)	97.26 ±0.18	98.08 ±0.10	96.95 ±0.14

4.2. Ablation Studies

We investigate different approaches to self-attention by varying the shapes of the MFCC spectrogram patches that are fed into the Transformer. Using our default hyperparameters, the spectrogram consists of 98 time windows, containing 40 mel-scale frequencies. Our baseline uses time-domain attention, but we also investigate frequency-domain attention and intermediate steps where rectangular patches are used. We find time-domain attention to perform best, as shown in Figure 3. This is in agreement with previous findings that temporal convolutions work well for keyword spotting [16], since the first projection layer of our model can be interpreted as a form of convolution.

We also investigate the use of PreNorm and PostNorm and found that the latter improves performance for keyword spotting in our experiments. This is contrary to previous findings on other tasks [28], where PreNorm has been shown to yield better results and we encourage further work to explore the role of normalization in Transformers across different domains.

4.3. Attention Visualization

In order to examine which parts of the audio signal the model attends to, we propagate the attention weights of each Trans-

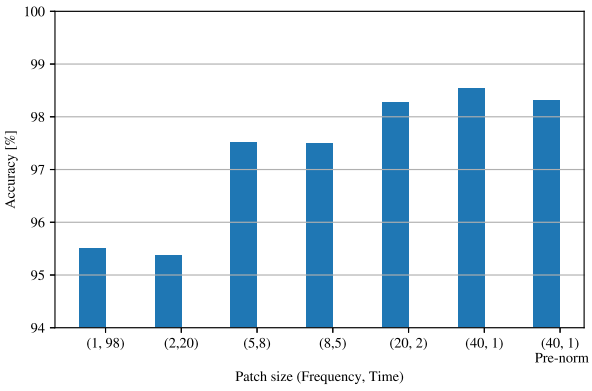

Figure 3: *Accuracy on Speech Commands V2-12 using KWT-3 with different patch sizes.*

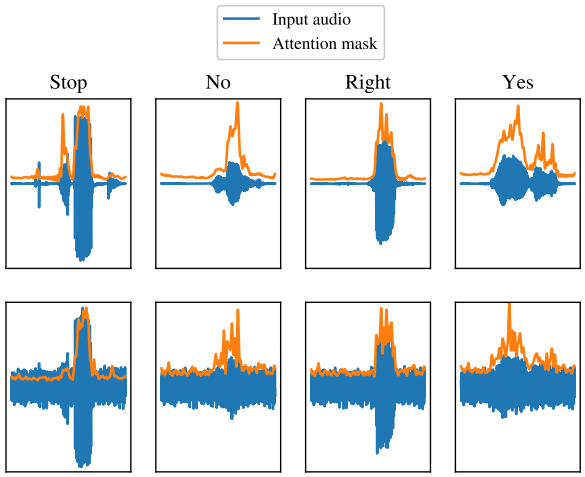

Figure 4: *The learned attention mask, propagated from the input to the class token, overlaid on four different audio snippets, without (top) and with (bottom) background noise.*

former layer from the input to the class token by averaging the attention weights over all heads. This produces a set of attention weights for each time window of the input signal. Figure 4 shows the attention mask overlayed on the waveform of four different utterances. It can be seen that the model is able to pay attention to the important parts of the input while effectively suppressing background noise.

We also study the position embeddings of the final model by analyzing their cosine similarity, as shown in Figure 5. Nearby position embeddings exhibit a high degree of similarity and distant embeddings are almost orthogonal. This pattern is less emphasized for time windows near the start and the beginning of the audio snippets. We hypothesize that this is either because words are typically in the middle of each snippet and therefore relative position is more important there, or because the audio content at the start and end is less distinguishable.

4.4. Latency Measurements

We converted our KWT models, DS-CNN (with stride) [14], TC-ResNet [16] and MHAtt-RNN [10] to Tensorflow (TF) Lite

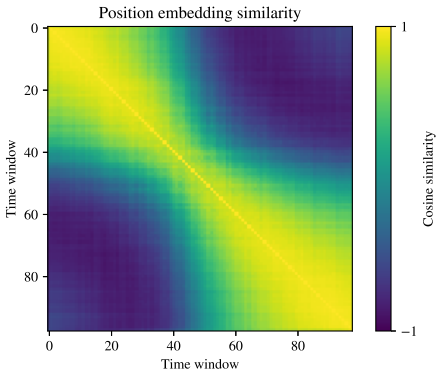

Figure 5: *Cosine similarities of the learned position embeddings of KWT.*

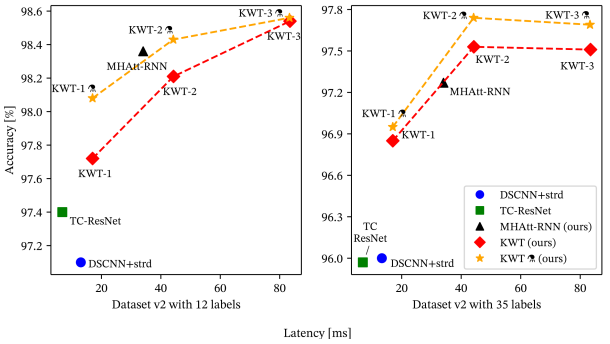

Figure 6: *Latency and accuracy of processing a whole 1 sec audio, on a single thread on a mobile phone.*

format to measure inference latency on a OnePlus 6 mobile device based on the Snapdragon 845 (4x Arm Cortex-A75, 4x Arm Cortex-A55). For latency comparisons we report accuracy figures for the Google Speech Commands V2 with 12 labels and 35 labels [26, 10]. The TFLite Benchmark tool [29] is used to measure latency, defined by the processing time of a single 1 second speech sequence. For each model, we do 10 warmup runs followed by 100 inference runs, capturing the average latency on a single thread.

In Figure 6 we observe that Transformer-based models are competitive with the existing state-of-the-art despite being designed with no regard to latency. There is a broad body of research on optimizing Transformer models — of particular note is the replacement of layer normalization and activations in [30] that decreases latency by a factor of three. Our findings here suggest that many of these results could be leveraged in the keyword spotting domain to extend the practicality of these models.

5. Conclusion

In this paper we explore the direct application of Transformers to keyword spotting, using a standard architecture and a principled approach to converting the audio input into tokens.

In doing so we introduce KWT, a fully-attentional model that matches or exceeds the state-of-the-art over a range of keyword spotting tasks with real-world latency that remains competitive with previous work.

These surprising results suggest that Transformer research in other domains offers a rich avenue for future exploration in this space. In particular we note that Transformers benefit from large-scale pre-training [2], have seen 5.5x latency reduction through model compression [30] and can obtain up to 4059x energy reduction through sparsity and hardware codesign [31]. Such improvements would make a meaningful impact on many keyword spotting applications and we encourage future research in this area.

6. Acknowledgements

This work was partially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP), funded by the Knut and Alice Wallenberg Foundation. We thank Matt Mattina for supporting this work, Magnus Oskarsson for his feedback and comments, and Oleg Rybakov and Hugo Touvron for sharing their code with the community.

7. References

[1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 2017, pp. 6000–6010.

[2] 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.

[3] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. Jégou, "Training data-efficient image transformers & distillation through attention," *arXiv preprint arXiv:2012.12877*, 2020.

[4] L. Yuan, Y. Chen, T. Wang, W. Yu, Y. Shi, F. E. Tay, J. Feng, and S. Yan, "Tokens-to-Token ViT: Training Vision Transformers from Scratch on ImageNet," *arXiv preprint arXiv:2101.11986*, 2021.

[5] M. Kumar, D. Weissenborn, and N. Kalchbrenner, "Colorization Transformer," *arXiv preprint arXiv:2102.04432*, 2021.

[6] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," in *European Conference on Computer Vision*. Springer, 2020, pp. 213–229.

[7] D. Neimark, O. Bar, M. Zohar, and D. Asselmann, "Video Transformer Network," *arXiv preprint arXiv:2102.00719*, 2021.

[8] M. A. Alcorn and A. Nguyen, "baller2vec: A Multi-Entity Transformer For Multi-Agent Spatiotemporal Modeling," *arXiv preprint arXiv:1609.03675*, 2021.

[9] D. C. de Andrade, S. Leo, M. L. D. S. Viana, and C. Bernkopf, "A neural attention model for speech command recognition," *arXiv preprint arXiv:1808.08929*, 2018.

[10] O. Rybakov, N. Kononenko, N. Subrahmanya, M. Visontai, and S. Laurenzo, "Streaming Keyword Spotting on Mobile Devices," *Proc. Interspeech 2020*, pp. 2277–2281, 2020.

[11] A. Gulati, J. Qin, C.-C. Chiu, N. Parmar, Y. Zhang, J. Yu, W. Han, S. Wang, Z. Zhang, Y. Wu *et al.*, "Conformer: Convolution-augmented Transformer for Speech Recognition," *Proc. Interspeech 2020*, pp. 5036–5040, 2020.

[12] P. Warden, "Speech commands: A dataset for limited-vocabulary speech recognition," *arXiv preprint arXiv:1804.03209*, 2018.

[13] 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*, vol. 28, no. 4, pp. 357–366, 1980.

[14] Y. Zhang, N. Suda, L. Lai, and V. Chandra, "Hello edge: Keyword spotting on microcontrollers," *arXiv preprint arXiv:1711.07128*, 2017.

[15] J. Lin, K. Kilgour, D. Roblek, and M. Sharifi, "Training keyword spotters with limited and synthesized speech data," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 7474–7478.

[16] S. Choi, S. Seo, B. Shin, H. Byun, M. Kersner, B. Kim, D. Kim, and S. Ha, "Temporal Convolution for Real-Time Keyword Spotting on Mobile Devices," *Proc. Interspeech 2019*, pp. 3372–3376, 2019.

[17] S. Majumdar and B. Ginsburg, "MatchboxNet: 1D Time-Channel Separable Convolutional Neural Network Architecture for Speech Commands Recognition," *Proc. Interspeech 2020*, pp. 3356–3360, 2020.

[18] X. Chen, Y. Wu, Z. Wang, S. Liu, and J. Li, "Developing Real-time Streaming Transformer Transducer for Speech Recognition on Large-scale Dataset," *arXiv preprint arXiv:arXiv:2010.11395*, 2020.

[19] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *arXiv preprint arXiv:1607.06450*, 2016.

[20] 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.

[21] D. Hendrycks and K. Gimpel, "Gaussian error linear units (GELUs)," *arXiv preprint arXiv:1606.08415*, 2016.

[22] G. Hinton, O. Vinyals, and J. Dean, "Distilling the Knowledge in a Neural Network," in *NIPS Deep Learning and Representation Learning Workshop*, 2015. [Online]. Available: http://arxiv.org/abs/1503.02531

[23] Q. Xie, M.-T. Luong, E. Hovy, and Q. V. Le, "Self-Training With Noisy Student Improves ImageNet Classification," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[24] D. S. Park, W. Chan, Y. Zhang, C.-C. Chiu, B. Zoph, E. D. Cubuk, and Q. V. Le, "SpecAugment: A Simple Data Augmentation Method for Automatic Speech Recognition," *Proc. Interspeech 2019*, pp. 2613–2617, 2019.

[25] "Speech commands dataset v1." [Online]. Available: http://download.tensorflow.org/data/speech_commands_v0.01.tar.gz

[26] "Speech commands dataset v2." [Online]. Available: https://storage.googleapis.com/download.tensorflow.org/data/speechcommands_v0.02.tar.gz

[27] R. Vygon and N. Mikhaylovskiy, "Learning Efficient Representations for Keyword Spotting with Triplet Loss," *arXiv preprint arXiv:2101.04792*, 2021.

[28] T. Q. Nguyen and J. Salazar, "Transformers without tears: Improving the normalization of self-attention," *arXiv preprint arXiv:1910.05895*, 2019.

[29] "Tflite model benchmark tool." [Online]. Available: https://github.com/tensorflow/tensorflow/tree/master/tensorflow/lite/tools/benchmark

[30] Z. Sun, H. Yu, X. Song, R. Liu, Y. Yang, and D. Zhou, "MobileBERT: a Compact Task-Agnostic BERT for Resource-Limited Devices," *arXiv preprint arXiv:2004.02984*, 2020.

[31] H. Wang, Z. Zhang, and S. Han, "SpAtten: Efficient Sparse Attention Architecture with Cascade Token and Head Pruning," *arXiv preprint arXiv:arXiv:2012.09852*, 2021.