WaveSense: A Spiking Neural Network Inspired by the WaveNet Architecture for Keyword Spotting

Anonymous Author(s)
Affiliation
Address
email

Abstract

Ultra-low power local signal processing is a crucial aspect for edge applications on always-on devices. Neuromorphic processors emulating spiking neural networks show great computational power while fulfilling the limited power budget as needed in this domain. In this work we develop a spiking neural network inspired by the WaveNet architecture which is based on simple neural dynamics, fixed time-constants and is strictly feed-forward and hence is particularly well suited for a neuromorphic implementation. We test the capabilities of this model on several datasets for keyword-spotting. The results show that the proposed network beats the state-of-the-art of other spiking neural networks and reaches near state-of-the-art performance of artificial neural networks such as CNNs and LSTMs.

Keywords— spiking, keyword spotting, temporal processing, streaming, audio, neurompрphic, wavenet, wavesense, always-on, low-power

1 Introduction

Local signal processing is an important component of the computational pipeline for Internet-of-Things (IoT) devices equipped with a range of sensors like audio, video, and motion sensing. A significant range of these sensors capture signals comprising temporal features. Ideally these features need to be extracted by an on-board processor before decision making or relaying the pre-processed information for further processing. Processing temporal signals is often computationally challenging and requires large amounts of memory and power, especially in always-on scenarios. Neuromorphic [13] processors with spiking-neural networks have shown promise in this domain as ultra-low power compact solutions [10, 2, 14, 8, 6, 12].

In this work we propose WaveSense a Spiking Neural Network (SNN) model suitable for efficient neuromorphic implementations while retaining high accuracy on temporal data streams. This work bridges the performance gap between Artificial Neural Networks (ANNs) and SNNs for temporal tasks. Crucially, the proposed model

- accepts spike streams and not 'buffered frames' as input,
- requires no delays in its connectivity,
- utilizes a very simple spiking neuron model - Leaky Integrate and Fire (LIF) neuron - without the need of any additional adaptive mechanisms,
- does not require recurrent connectivity (recurrent connectivity can often be difficult to tune or train).
- produces a high classification performance.

Recently several works have shown how to build efficient SNNs with accuracies equivalent to ANNs [7, 18]. In these studies, spiking neurons are used in rate mode with equivalent response curves to ReLU activations, to transfer weights from pre-trained ANNs to SNNs. This approach therefore completely neglects the temporal capabilities of spiking neurons. On the other hand, surrogate gradient methods enable directly training SNN using Back Propagation Through Time (BPTT) [15]. Shrestha and Orchard [21] for instance show temporal processing

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

on temporal gesture recognition task and show a good performance on a visual task. Similar approaches have already been investigated on audio tasks [1, 26, 5]. Bellec et al. [1] demonstrate classification results on TIMIT dataset using long time constants, a complex learning strategy (Deep-R) and require significant computational resources to train. Wu et al. [26] train SNNs for automatic speech recognition tasks in a tandem approach with an ANNs. This training pipeline integrated a language model and pronunciation model which goes beyond the capabilities of the neuromorphic system. The same authors showed in an earlier study the capabilities of a SNN in combination with a self-organized map to learn to recognize digits using the TIDIGIT dataset [25].

Blouw et al. [3] demonstrate high accuracy in a audio classification task using dense networks with spectrograms as inputs. This approach requires passing the frequency data of previous time steps (defined by the spectrogram time window) in every sample presentation to the network. Kugele et al. [11] show that by matching ANN roll-out delays to the propagation delays in SNN, the resultant networks can demonstrate a high accuracy on vision based spatio-temporal classification tasks. Implementation of delays in neuromorphic hardware requires additional memory resources to store and deliver spikes in a delayed fashion and could be potentially quite expensive. Yin, Corradi, and Bohté [27] use a Spiking Recurrent Neural Network (SRNN) architecture and demonstrate that by utilizing *adaptive* LIF neurons and learning the time constants, these networks can perform temporal classification tasks in a sequential manner fairly well. The authors demonstrate the effective use of spiking neural networks and show a significant increase in power efficiency. Unfortunately, having fine tuned time constants in low-power neuromorphic hardware can often be challenging especially while using fixed precision numerical representations and computations.

We propose a novel network architecture for SNN that does not require buffering or delays and can directly process spiking data from event-based sensor using simple LIF neurons. Our architecture is derived from first principles and inspired by the WaveNet [16] architecture that does not necessitate learning the time constants of the system but could be defined as the task demands. In addition we also propose an efficient training strategy and a corresponding loss-function that is suitable for streaming based models, in particular models that could be run in real-time with neuromorphic hardware.

The first key aspect of the WaveNet architecture is the use of a multi-layer causal dilated convolutions. The *causal* refers to the use of data from the past, *dilated* refers to a sparse kernel and the *convolution* is along the time axis. Stacking such convolutional operations along multiple layers enables the network to have a long temporal memory. A second aspect is that it eliminates the need for sliding window based inference/prediction and minimizes the number of computations within the network when operating on a continuous stream of data.

The WaveNet architecture is very amenable to general purpose micro-processors and micro-controllers but it still requires a reasonable amount of memory and non-linear computations such as $tanh$ and $sigmoid$ which are fairly complex (within the context of ultra-low power devices) and this inturn requires higher engergy and power requirements. Neuromorphic technology promises to bring the energy required for these tasks down by utilizing SNNs to perform ultra-low power computation. So far, while neuromorphic devices have been demonstrated to operate at an extremely low power, they have fallen short in demonstrating computational performance that is on-par or comparable to state of the art ANNs for temporal tasks, most prominently in the audio domain.

WaveSense model proposed here aims to bridge this gap. For the results in this work, we focus on audio tasks as spatio-temporal tasks without loss of generality. Sec. 2 details all the methods used for audio data pre-processing, conversion to spikes and the details of the network architecture. In Sec. 3 we demonstrate the computational capability of this network over several audio datasets for key-word spotting task. We compare our results to the state of the art SNNs and ANNs. We conclude in Sec. 4 where we discuss the implications and impact of this work and potential areas where this work could be utilized.

Most importantly all the code used to generate the reported results have been made open-source and can be found at `http://XXXXXXXXXXXXXX`. We believe this will enable the research community to explore other avenues that could take advantage of our work.

2 Materials and Methods

2.1 Neuron Model

In this work we used Leaky Integrate and Fire (LIF) neuron model with synaptic time constant τ_s and membrane time constant τ_v. The sub-threshold dynamics of this neuron are described below.

$$\dot{v}(t) = -v(t)/\tau_v + i_s \tag{1}$$

$$\dot{i}_s(t) = -i_s(t)/\tau_s + \sum w_j s_j(t) \tag{2}$$

where v is the membrane potential, i_s is the synaptic current, w the synaptic weight and s is the input spike train.

In-order to optimize simulation time and computational efficiency, we make some alterations. Unlike a traditional LIF which resets to a resting potential upon reaching threshold θ, we *subtract* a fixed value θ from the membrane potential. Further more, if the membrane potential increases beyond $N\theta$, where N is an arbitrary positive integer, then this neuron produces N spikes, and proportionally $N\theta$ is *subtracted* from the membrane potential.

$$s(t) = \begin{cases} [v(t)/\theta], & \text{if } v(t) \geq \theta \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

The use of a mechanism for generating multiple spikes enables the computation to be more robust to the choice of simulation time steps. This enables us to choose a relatively large time step for our simulations. In practice we observe that some neurons occasionally *do* produce multiple spikes. (Producing multiple spikes in a single simulation time-step is not necessary if one chooses a small enough time step.)

For further simulation efficiency the LIF neurons were simulated using the Spike Response model (SRM) [9].

$$v(t) = \sum_j w_j (\epsilon * s_j)(t) + (\nu * s)(t) \tag{4}$$

where

$$\epsilon_s(t) = e^{(-t/\tau_s)} \tag{5}$$
$$\epsilon_v(t) = e^{(-t/\tau_v)} \tag{6}$$
$$\nu(t) = -\theta\, e^{(-t/\tau_v)} \tag{7}$$
$$\epsilon(t) = (\epsilon_s * \epsilon_v) \tag{8}$$

We use exponential kernels for synaptic $\epsilon_s(t)$ and membrane dynamics $\epsilon_v(t)$ and derive the Post Synaptic Potential (PSP) kernel $\epsilon(t)$. The refractory kernel $\nu(t)$ is also a negative exponential kernel with the same time constant τ_v as the membrane potential. The symbol $*$ denotes a convolution operation.

As spike generation is non-differentiable, we use a surrogate gradient [15]. Several profiles for the surrogate gradients have been proposed in literature. We use a modified exponential kernel for the surrogate gradient function. In order to accommodate the multi-spike behavior of the neuron, we choose a periodic exponential function (Figure 1) as the surrogate gradient. This function peaks as the membrane potential approaches multiples of neuron spiking threshold θ. Intuitively, this gradient function maximizes the impact of a parameter when the neuron is close to spiking or has just spiked and is a variant of the exponential gradient function. An extreme simplification of the periodic exponential would be a Heaviside function [1].

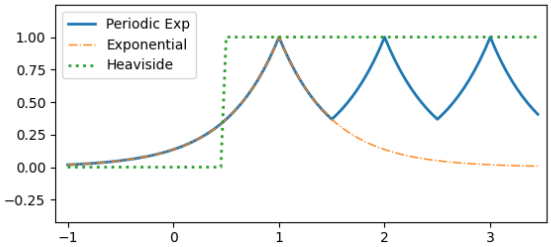

Figure 1: The surrogate gradient of the spiking neuron as a function of the (pre-spike) membrane potential.

The simulations were run with the library 'sinabs' [19][2] using an adaptation of SLAYER [21].

[1] The heaviside function like a Rectified Linear Unit (ReLU) has a range of membrane potentials where the gradient is 0 and could potentially prevent the network from learning at low activity levels.

[2] GNU AGPL v3 License

2.2 Network description

We take inspiration from the work of Coucke et al. [4] who uses the WaveNet architecture [16] for classification of continuous audio streams. The WaveNet architecture provides a prescription for distributing temporal memory and computation across layers without repeated presentation of previous input data.

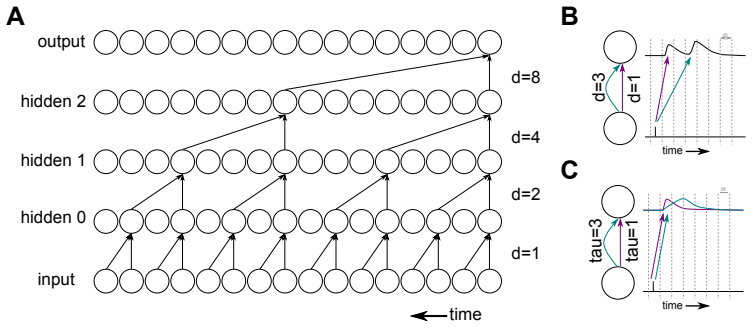

Figure 2: Relegating the job of delays in dilated convolutions to synaptic dynamics.

The original ANN WaveNet model comprises of a few different computational building blocks. We translate each of these building blocks to SNNs as follows.

Dilated Temporal Convolutions are the crux of what makes WaveNets efficient. The dilation enables convolution over a large time window (or sample length) while still only using a small number of parameters. These convolutions perform a weighted-accumulation of information from different points in time, separated by the *dilation* parameter. This is done by storing previous activations in ANNs, and naively, this could be mimicked by replacing it by synaptic transmission delays. In this work, we observe that neuron and synaptic dynamics could be seen as proxies for temporal convolutional processing as shown in Figure 2. Figure 2B shows the contributions of each projection with a kernel size 2 would look like when realized with delays. Alternatively, if we choose an appropriate set of synaptic time constants τ_s, this is shown in Figure 2C. While quantitatively, these are different, qualitatively both these approaches provide the ability to transform and project information in the temporal domain [20]. This is the key insight we leverage to design our final model.

Rectified Linear Unit (ReLU) activations can be *approximated* by spiking neurons because a spiking neuron can only produce spikes if the membrane potential crosses a threshold and is silent otherwise [7].

Non-linear activations like $tanh$ and $sigmoid$ cannot be efficiently translated to SNNs. So we choose to replace these activations with SNNs activations (potentially ignoring the benefits of filtering and gating). In the original model, these two activations are preceded by two sets of weights, where as in our SNN we only use one weighted projection. (See Figure 3)

Residual connections and summation ($+$) are realized by a synaptic connection.

The WaveSense model is built upon these building blocks as shown in Figure 3. It comprises of several 'blocks', each of which comprises of three spiking neuron layers. The first spiking layer receives inputs filtered by two separate synapses with different time constants τ_s and weights. The time constants τ_s of the slow projections in each of these blocks are chosen such that they span a range of values relevant to the task. The number of blocks is chosen such that the sum of all these time constants is proportional to the temporal memory demanded by the task. This layer projects to the second spiking layer. Additionally a third spiking layer in each of these blocks projects to a 'hidden' layer followed by a non-spiking 'low pass' readout layer. The output of this block is the summation of its input (residual connection) and the output of the second layer. These 'blocks' are connected in a feed forward manner. The non-spiking 'low-pass'(LP) layer simply acts as a weighted low pass filter on the spikes of the 'hidden' layer. This is equivalent to the synapse of a spiking neuron (without the neuron's membrane potential or the spiking dynamics) and does not require any extra components unavailable to spiking neurons. The choice of leaving the output layer to be non-spiking is to enable a smooth, continuous valued readout useful for faster learning using BPTT.

2.3 Datasets

In order to evaluate the efficacy of the proposed model, we train and test it against several open-source publicly available audio datasets.

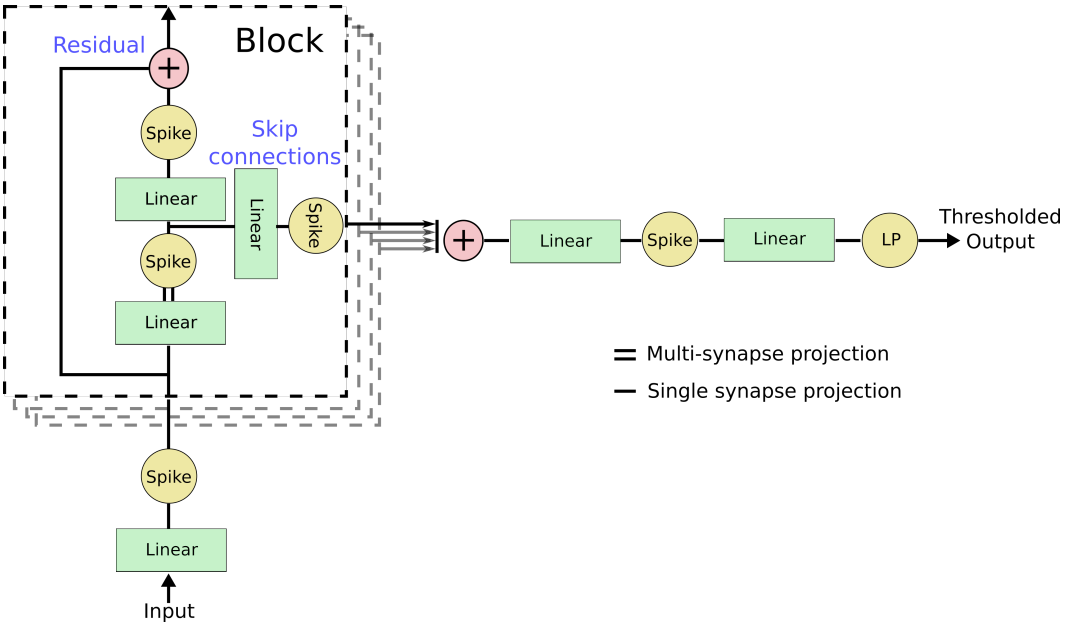

Figure 3: The WaveSense model prescribed in this work is a theoretical adaptation of the WaveNet architecture [16] based on first principles.

2.3.1 Aloha Dataset

The Aloha dataset [3] is a small collection of audio samples containing the keyword 'Aloha' and several distractors such as 'take a load off'. As the dataset is very small, only $\sim$ 2000 samples, we augmented the samples using the MUSAN noise dataset [22]. For that end we standardized the sample length of each utterance in the training and validation set to five seconds and added randomly selected background noise data with a signal-to-noise ratio (SNR) of 5 dB to the training data.

2.3.2 Hey Snips Dataset

The 'Hey Snips' dataset [4] for wake phrase spotting distinguishes between two classes. The positive class contains 11'000 utterances from over 2'000 speakers of the wake phrase 'Hey Snips' while the negative (or distractor) class contains over 86'000 negative examples from more than 6'000 speakers. We split the data into a training-, validation- and test-set as provided by the authors of the dataset. We standardized the sample length of each utterance in the training and validation set to five seconds. As the dataset is already very large, we did not augment the data with noise.

2.3.3 Speech Commands Dataset

The Speech Commands [24] describe a dataset containing 35 keywords uttered in total 105'000 times from over 2'600 speakers. The keywords contain the numbers 0 - 10, commands such as "stop", "go", "left" and "right" as well as other words like "Marvin", "Sheila", etc. This dataset was initially designed for keyword spotting in a limited vocabulary and the intended experiment is to detect 10 commands (plus silence) out of all 35 keywords (12 classes in total). Nevertheless, there are studies training models and showing results on all 35 classes [5]. We augment the training set with noise data from the MUSAN dataset using a SNR of 5 dB just as we do for the Aloha dataset.

2.4 Pre-processing

All datasets we use in this work contain many wave audio files of a few seconds length. The data is pre-processed in several stages:

- **Noise augmentation** The training data is augmented with noise from the MUSAN noise dataset using a SNR of 5dB (except for the HeySnips data).
- **Length standardization and pre-amplification** The noise augmented waveform is cut into a standard length (dependent on the dataset) and the amplitude is normalized.

- **Band-pass filters** The audio is then passed through 64 Butterworth bandpass filters 2nd order. The bandpass filters are distributed in Mel-scale between 100Hz and $8k$Hz.
- **Rectification** The response of the 64 bandpass filters is rectified using a full-wave rectifier.
- **Spike conversion and binning** The output of the rectifier is applied as direct input to the membrane of 64 simplified LIF neurons resulting in a rate code. The spike trains are binned into 10ms timesteps allowing multiple spikes per timestep.

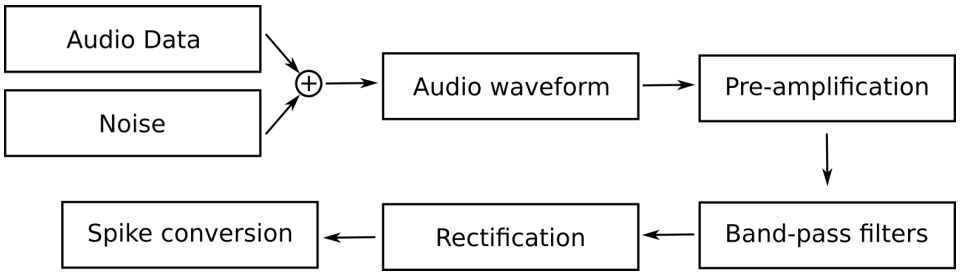

Figure 4: Data pre-processing pipeline figure.

2.5 Training Method

In order to train the parameters of SNN (See Sec. 2.2) we use BPTT. In particular we aim to be able to deploy the network in streaming mode *ie.* the model receives the data stream directly generated from a sensor without any frame based (sliding window) buffering. This requires us to employ an appropriate loss function.

Often in a classification task, the output class can be determined by computing cross-entropy loss on the sum of the outputs over the sample length for each output neuron. While this would yield a good classification accuracy, the magnitude of the output trace at 'a given point in time' is not indicative of the network prediction. This is not ideal for models being run in streaming mode.

2.5.1 Peak Loss

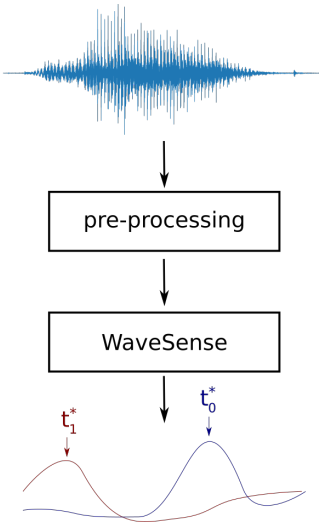

Figure 5: Visualization of the peak loss with peak times t_c^*.

Typically for streaming models, a signal is predicted as belonging to a certain class when the corresponding output trace exceeds a 'detection threshold'. This approach is also ideal for always-on neuromorphic systems. We therefore design our loss function to reflect this detection mechanism and train our neural networks. We determine the peaks of the output traces and use only the activation values at the peaks to compute the cross entropy loss (see Figure 5) similar to *max-over-time loss* [5].

Consequently the loss is computed as follows:

$$L_{CE} = -\sum_{c} \lambda_c \log(p_c) \tag{9}$$

where λ_c yields 1 if class label c corresponds to the current input and 0 otherwise. p_c is the prediction probability by the neural network that the current input belongs to class c. It is calculated by a *softmax* operation as shown below.

$$p_c = \frac{e^{\hat{\mathbf{y}}_c}}{\sum_i e^{\hat{\mathbf{y}}_i}} \tag{10}$$

where $\hat{\mathbf{y}}$ are the 'logits' produced by the neural network.

For temporal tasks, the input $\mathbf{x} = x^T = x^{1\ldots T}$ and the output (logits) $\hat{y}$ of the neural network are time-series over time T.

$$\hat{y}^t = f(x^t|\Theta, s^t) \tag{11}$$

where f is the transformation of the neural network, Θ are the network parameters and s^t is the internal state of the network at time t. In *peak-loss* we pass the peak of each output trace to the *softmax* function. The peaks are calculated as follows:

$$\hat{\mathbf{y}}_c = \max(y_c^T) = y_c^{t_c^*} \tag{12}$$

where $t_c^* = \text{argmax}(y_c^T)$ is the 'peak time', the time of maximal activation of output trace c (see Figure 5).

2.5.2 Spiking activity regularization

The activity of LIF neurons can change dramatically during the learning process. This is particularly true while using the multi-spike neuron model as we do in this WaveSense implementation. This could potentially eliminate the benefit of using spiking neurons by spiking at a high firing rate at every time step and consequently have no sparsity. This could also lead to high energy utilization of the network in a neuromorphic implementation.

In order to limit the activity of these neurons and maintain sparse activity, we include an activity regularizer term in our loss function [23].

$$L_{act} = (N_{spk}^{\dagger}/(T \cdot N_{neurons}))^2 \tag{13}$$

where the activation loss L_{act} is dependent on the total *excess* number of spikes $N_{spk}^{\dagger}$ produced by the network with a population size $N_{neurons}$ in response to a input of length T time steps. $N_{spk}^{\dagger}$ is given as:

$$N_{spk}^{\dagger} = \sum \sum_{i} N_i^t \Theta(N_i^t - 1) \tag{14}$$

is the sum of spikes from all neurons N_i exceeding 1 in each time bin t (Θ is a heaviside function).

Finally the loss function is given as:

$$L = L_{CE} + \alpha L_{act} \tag{15}$$

where α was chosen to be 0.01.

3 Results

In order to validate and verify that sufficient information from the input is retained after pre-processing and conversion to spikes, we train a state-of-the-art WaveNet classifier on the datasets considered in this work and check that we can obtain a high accuracy. We implement a non-spiking dilated Convolutional Neural Network (CNN) to replicate the WaveNet architecture very similar to that described in Coucke et al. [4] and Oord et al. [16] (see Section 2.2 for details).

We train this ANN on the HeySnips, Aloha and Speechcommands datasets and compare our results to those reported in literature [4, 3, 5]. The results obtained from this network are then used as baseline to evaluate the performance of the proposed SNN.

Table 1: Aloha result model size and resource comparison.

Publication	#Neurons	#Parameters	Accuracy
Blouw et al. [3]	541	172800	95.8
This work	864	18482	98.0 ± 1.1

3.1 Aloha dataset

In order to compare our model to other SNN implementations in the keyword spotting domain, we trained our WavseSense on the Aloha dataset [3]. Table 1 shows the network and memory resources the proposed model utilizes in comparison to the work demonstrated in Blouw et al. [3]. With an average accuracy of 98.0% with a standard deviation of 1.1% the model presented in this work performs significantly better while at the same time requiring a significantly smaller number of parameters. The best runs of the WaveSense model yielded 99.5% accuracy which is equal to the performance of the ANN model. It is important to note that the key focus of the work by Blouw et al. [3] is to benchmark energy and power consumption and not model performance.

3.2 HeySnips dataset

On the HeySnips dataset, our implementation of the WaveNet reaches an accuracy of 99.8% on the clean dataset. In Coucke et al. [4], the authors do not report any accuracy number but rather report the false rejection rate (FRR) of 0.12% for a fixed false alarm per hours (FAPH) of 0.5. In order to compare our results more accurately, we implement the same metrics; our WaveNet implementation reaches 0.95 FAPH and a 0.8% FRR on the test set. These results are slightly worse than the results reported by Coucke et al. [4] but that is expected as we do not apply the same specific methods to improve performance such as "End-Of-Keyword labeling" and "masking". Without those methods and without gating, the FRR reported by Coucke et al. [4] drops to 0.98%. On the other hand, our WaveNet implementation reaches similar or even better results than the CNN and LSTM reported by Coucke et al. [4]. This fact shows that our pre-processing method indeed extracts sufficient information from the input such that a neural network can reach very high accuracy. Hence, we train a spiking version of the WaveNet architecture (WaveSense), as described in 2.2, on the same data.

In the WaveSense model we do not use any gating mechanism, a kernel size of 2 and only 8 layers; much less compared to the 24 layers and kernel size of 3 as used in our WaveNet implementation and the WaveNet implementation by Coucke et al. [4]. The memory in our model is still long enough as the WaveSense implements the dilations using synaptic dynamics with long time constants but the number of parameters drops from 47090 to 13042. Despite the low number of parameters and quantization from spiking activations, the WaveSense model acheives an average accuracy of 99.6% over 11 runs (only drops by 0.2%) . Our best run of the WaveSense model yielded the same accuracy (of 99.8%) as our WaveNet implementation. With an FRR $= 1.0\%$ and FAPH $= 1.34$ the performance is indeed lower than the WaveNet, but it is comparable to that of LSTM and CNN as reported by Coucke et al. [4].

3.3 Speechcommands dataset

We also trained WaveSense on the Speechcommands dataset. We evaluated our model by training it to classify all 35 classes in the dataset. In a study by Perez-Nieves et al. [17] in which the authors investigate the impact of heterogenity of time constants on the performance, the best model reached $\sim 57.3\%$ accuracy on the same dataset. In Cramer et al. [5] the best performing SNN is a recurrent network which yields $\sim 50.9\%$ accuracy of all 35 classes. In the same study, also an LSTM and CNN are trained on the same data resulting in an accuracy of $\sim 73\%$ resp. $\sim 77.7\%$. The WaveSense model reaches an average accuracy of 79.6% over 11 runs (best 80.0%) which is significantly higher than the best SNN described in previous studies. Notably, WaveSense performs better than the reported LSTM and CNN [5].

4 Discussion and Conclusion

While the results demonstrated here are obtained using a fixed set of time constants, it is conceivable that according to the constraints of the neuromorphic hardware, an appropriate network could be trained to obtain qualitatively similar results. This holds true even for mixed-signal neuromorphic devices [10] with programmable weights and tune-able time constants. Because the algorithm provides a recipe for how to choose the time constants in the network, even if a neuromorphic substrate has a limited range of time constants, a number of layers with an appropriate combination (sum) of time constants can always be chosen to fit the temporal task. This is in stark contrast to recurrent neural networks that often require a tight balance between excitation and inhibition and long time constants [1, 27].

Table 2: A comparison of model performance for various datasets and network architectures.

Publication	Dataset	Accuracy (%)	Architecture
Coucke et al. [4]	HeySnips	FRR 0.12 FAPH 0.5	WaveNet
Coucke et al. [4]	HeySnips	FRR 2.09 FAPH 0.5	LSTM
Coucke et al. [4]	HeySnips	FRR 2.51 FAPH 0.5	CNN
This work	HeySnips	99.8 (FRR 0.8 FAPH 0.95)	WaveNet
This work	HeySnips	99.6 ± 0.1 (FRR 1.0 FAPH 1.34)	SNN
Cramer et al. [5]	SpeechCommands(35)	50.9 ± 1.1	SNN
Cramer et al. [5]	SpeechCommands(35)	73 ± 0.1	LSTM
Cramer et al. [5]	SpeechCommands(35)	77.7 ± 0.2	CNN
Perez-Nieves et al. [17]	SpeechCommands(35)	57.3 ± 0.4	SNN
This work	SpeechCommands(35)	87.6	WaveNet
This work	SpeechCommands(35)	79.6 ± 0.1	SNN
Blouw et al. [3]	Aloha	93.8	SNN
This work	Aloha	99.5	WaveNet
This work	Aloha	98.0 ± 1.1	SNN

The choice of time constants and number of layers is informed by the total temporal memory required by the task. We choose them in a similar fashion to that of WaveNet with time constants increasing with factors of 2 and such that the sum of all the time constants is proportional to τ_{task}. Typically we observe that a proportionality of 2.5 is suitable with a kernel size of 2. The proportionality factor is the length of time after which the effect of a PSP is negligible. This also translates to compact networks with fewer parameters for the same amount of temporal memory (at the same time resolution). In other words, given a network, the temporal memory of a given task can be computed as follows:

$$\tau_{task} \approx 2.5 \sum_i \tau_s^i \tag{16}$$

where i is the list of all the layers in the WaveSense network.

While the results reported here are significantly high, we believe this can be further improved by improving the loss function. For instance the *max-over-time* loss computed only during the presence of a keyword as opposed to the entire sample [4] has been shown to improve performance of such models. Furthermore, a thorough architecture search could potentially result in a better combination of time constants τ_m and τ_s, number of channels, kernel sizes etc.

A crucial factor in adopting a model is ease of training, deployment and power efficiency. By utilizing simple LIF neurons, we take full advantage of their computational efficiency [27] in additional to sparse computations afforded by SNNs. While training SNNs is relatively slow on CPUs and GPUs, utilizing the SRM in combination with the SLAYER algorithm [21], we are able to train at a relatively high speed. All experimental results reported in this manuscript were performed on a single NVIDIA 1080 Ti with a few hours of run time per experiment. We further improve upon this efficiency with a custom fork of the SLAYER implementation [3]. The resulting models while accurate within the SRM framework, are not *identical* to simulations based on LIF neurons, supported by most digital neuromorphic devices. But we find that they are a close approximation and a quick retraining can recover the model's performance using LIF neurons.

The WaveNet architecture requires storing activations of each of its layers depending on their kernel size and dilation value: $N_{buf} \propto (k-1) \cdot d + 1$. In contrast, WaveSense does not buffer any spikes(activations) from the past explicitly. Instead the information is retained in the neuron and synaptic states: $N_{buf} \propto k + 1$. This makes WaveSense extremely efficient in terms of memory utilization in contrast to WaveNet.

The results demonstrated here show that the WaveSense architecture is suitable for audio classification tasks and show a promising performance improvement in comparison to prior state-of-the-art. Audio signals, after they are pre-processed are equivalent to a population of neurons producing spike patterns with complex spatio-temporal correlations. We argue therefore, that the results presented here can be extended to other modalities of sensory data such as ECG, PPG, machine vibrations or DVS data.

This work we believe could contribute towards a future with a ubiquitous abundance of always-on audio and other sensory devices responding to user commands. This could lead to potential misuse of the technology for surveillance. Thankfully, neuromorhic algorithms such as the one proposed here require specialized neuromorphic hardware to take full advantage. If the availability of such hardware could be regulated, we hope that the society can benefit from this technology while protecting itself from misuse.

[3] https://XXXXXXXXXXXXX

References

[1] Guillaume Bellec et al. “Long short-term memory and learning-to-learn in networks of spiking neurons”. In: *arXiv preprint arXiv:1803.09574* (2018).

[2] Ben Varkey Benjamin et al. “Neurogrid: A mixed-analog-digital multichip system for large-scale neural simulations”. In: *Proceedings of the IEEE* 102.5 (2014), pp. 699–716.

[3] Peter Blouw et al. *Benchmarking Keyword Spotting Efficiency on Neuromorphic Hardware*. 2018. arXiv: 1812.01739 [cs.LG].

[4] Alice Coucke et al. *Efficient keyword spotting using dilated convolutions and gating*. 2018. arXiv: 1811.07684 [cs.LG].

[5] Benjamin Cramer et al. “The Heidelberg Spiking Data Sets for the Systematic Evaluation of Spiking Neural Networks”. In: *IEEE Transactions on Neural Networks and Learning Systems* (2020).

[6] Mike Davies et al. “Loihi: A neuromorphic manycore processor with on-chip learning”. In: *Ieee Micro* 38.1 (2018), pp. 82–99.

[7] Peter U Diehl et al. “Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing”. In: *2015 International joint conference on neural networks (IJCNN)*. ieee. 2015, pp. 1–8.

[8] Steve B Furber et al. “The spinnaker project”. In: *Proceedings of the IEEE* 102.5 (2014), pp. 652–665.

[9] Wulfram Gerstner. “A framework for spiking neuron models: The spike response model”. In: *Handbook of Biological Physics*. Vol. 4. Elsevier, 2001, pp. 469–516.

[10] Giacomo Indiveri et al. “Neuromorphic silicon neuron circuits”. In: *Frontiers in neuroscience* 5 (2011), p. 73.

[11] Alexander Kugele et al. “Efficient Processing of Spatio-Temporal Data Streams With Spiking Neural Networks”. In: *Frontiers in Neuroscience* 14 (2020), p. 439. ISSN: 1662-453X. DOI: 10.3389/fnins.2020.00439. URL: https://www.frontiersin.org/article/10.3389/fnins.2020.00439.

[12] Qian Liu et al. “Live demonstration: face recognition on an ultra-low power event-driven convolutional neural network ASIC”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*. 2019.

[13] Carver Mead. “Neuromorphic electronic systems”. In: *Proceedings of the IEEE* 78.10 (1990), pp. 1629–1636.

[14] Paul A Merolla et al. “A million spiking-neuron integrated circuit with a scalable communication network and interface”. In: *Science* 345.6197 (2014), pp. 668–673.

[15] Emre O. Neftci, Hesham Mostafa, and Friedemann Zenke. *Surrogate Gradient Learning in Spiking Neural Networks*. 2019. arXiv: 1901.09948 [cs.NE].

[16] Aaron van den Oord et al. *WaveNet: A Generative Model for Raw Audio*. 2016. arXiv: 1609.03499 [cs.SD].

[17] Nicolas Perez-Nieves et al. “Neural heterogeneity promotes robust learning”. In: *bioRxiv* (2021), pp. 2020–12.

[18] Bodo Rueckauer et al. “Conversion of continuous-valued deep networks to efficient event-driven networks for image classification”. In: *Frontiers in neuroscience* 11 (2017), p. 682.

[19] Sadique Sheik and Martino Sorbaro. *Project Title*. https://sinabs.ai/. 2013.

[20] Sadique Sheik et al. “Emergent Auditory Feature Tuning in a Real-Time Neuromorphic VLSI System”. In: *Frontiers in Neuroscience* 6 (2012), p. 17. ISSN: 1662-453X. DOI: 10.3389/fnins.2012.00017. URL: https://www.frontiersin.org/article/10.3389/fnins.2012.00017.

[21] Sumit Bam Shrestha and Garrick Orchard. “SLAYER: Spike Layer Error Reassignment in Time”. In: *Advances in Neural Information Processing Systems 31*. Ed. by S. Bengio et al. Curran Associates, Inc., 2018, pp. 1419–1428. URL: http://papers.nips.cc/paper/7415-slayer-spike-layer-error-reassignment-in-time.pdf.

[22] David Snyder, Guoguo Chen, and Daniel Povey. *MUSAN: A Music, Speech, and Noise Corpus*. arXiv:1510.08484v1. 2015. eprint: 1510.08484.

[23] Martino Sorbaro et al. “Optimizing the energy consumption of spiking neural networks for neuromorphic applications”. In: *Frontiers in neuroscience* 14 (2020), p. 662.

[24] Pete Warden. "Speech commands: A dataset for limited-vocabulary speech recognition". In: *arXiv preprint arXiv:1804.03209* (2018).

[25] J Wu et al. *A spiking neural network framework for robust sound classification. Front. Neurosci. 12 (2018)*. 2018.

[26] Jibin Wu et al. "A Tandem Learning Rule for Effective Training and Rapid Inference of Deep Spiking Neural Networks". In: *arXiv e-prints* (2019), arXiv–1907.

[27] Bojian Yin, Federico Corradi, and Sander M Bohté. "Effective and efficient computation with multiple-timescale spiking recurrent neural networks". In: *International Conference on Neuromorphic Systems 2020*. 2020, pp. 1–8.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] The potential impact of this work extends from a broader discussion on the potential impact of low-power keyword spotting technologies.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes]
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]