LEARNING SPATIO-TEMPORAL REPRESENTATIONS USING SPIKE-BASED BACKPROPAGATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Spiking neural networks (SNNs) offer a promising alternative to current artificial neural networks to enable low-power event-driven neuromorphic hardware. However, training SNNs remains a challenge primarily because of the complex non-differentiable neuronal behavior arising from their spike-based computation. In this paper, we propose an algorithm to train spiking autoencoders on regenerative learning tasks. A sigmoid approximation is used in place of the Leaky Integrate-and-Fire neuron's threshold based activation during backpropagation to enable differentiability. The loss is computed on the membrane potential of the output layer, which is then backpropagated through the network at each time step. These spiking autoencoders learn meaningful spatio-temporal representations of the data, across two modalities - audio and visual. We demonstrate audio to image synthesis in a spike-based environment by sharing these spatio-temporal representations between the two modalities. These models achieve very low reconstruction loss, comparable to ANNs, on MNIST and Fashion-MNIST datasets, and while converting TI-46 digits audio samples to MNIST images.

1 INTRODUCTION

In recent years, Artificial Neural Networks (ANNs have become powerful computation tools for complex tasks such as pattern recognition, classification and function estimation problems (LeCun et al., 2015). They have an "activation" function as their neuron (compute unit). These functions are mostly *sigmoid*, *tanh*, or *ReLU* (Nair & Hinton, 2010) and are very different from a biological neuron. Spiking neural networks (SNNs), on the other hand, are recognized as the "third generation of neural networks" (Maass, 1997), with their "spiking" neuron model (or compute unit) much closely mimicking a biological neuron. They have a biologically plausible architecture that can potentially achieve high computational power and efficient neural implementation (Ghosh-Dastidar & Adeli, 2009; Maass, 2015). However, training methods for these spiking neural networks (Jin et al., 2018; Sengupta et al., 2018) are still in an early development stage, where each method comes with its own advantages and challenges.

For any neural network, the first step of learning is the ability to encode the input into meaningful representations. We investigate how input spike trains can be processed and encoded into meaningful hidden representations in a spatio-temporal format of output spike trains which can be used to recognize and regenerate the original input. Autoencoders are very effective tools to learn underlying representations of data, especially visual data (Vincent et al., 2008). Their simple two-layer structure makes them easy to train as well. In the domain of SNNs, autoencoders provide an exciting opportunity for implementing unsupervised feature learning.

One way to train spiking autoencoders is by using Spike Timing Dependent Plasticity (STDP)(Sjöström & Gerstner, 2010), an unsupervised local learning rule in which the weight update of a synapse is dependent on when the pre-neuron and post-neuron of the synapse spike with respect to each other. Burbank (2015) and Tavanaei et al. (2018) use STDP based learning rules to train autoencoders on MNIST and natural images. However, STDP, being unsupervised and localized, still fails to train SNNs to perform at par with ANNs. Another approach is derived from ANN backpropagation; the average firing rate of the output neurons is used to compute the global loss (Bohte et al., 2002; Lee et al., 2016). Rate-coded loss fails to include spatio-temporal information of the network, as the network response is accumulated over time to compute the loss. Wu et al. (2018)

applied backpropagation through time (BPTT) (Werbos, 1990), while Jin et al. (2018) proposed a hybrid backpropagation technique to incorporate the temporal effects. However, it continues to be a challenge to accurately map the time-dependent neuronal behavior with a time-averaged rate coded loss function.

Apart from firing rate, spiking neurons can be characterized by their internal state, referred to as the membrane potential (V_{mem}). The V_{mem} changes over time depending on the input to the neuron, and whenever it exceeds a threshold, the neuron generates a spike. Thus the firing rate of the spiking neuron is regulated by its membrane potential. Panda & Roy (2016) first presented a backpropagation algorithm for spiking autoencoders that uses V_{mem} of the output neurons to compute the loss of the network. They proposed an approximate gradient descent based algorithm to learn hierarchical representations in stacked convolutional autoencoders. In this work, we compute the loss of the network using V_{mem} of the output neurons, and we incorporate BPTT (Werbos, 1990) to compute the gradients, as membrane potential integrates over time, and use a *sigmoid* function approximation for the discontinuous step function relation between neuron output and it's internal state (V_{mem}).

Generally, an autoencoder can learn the hidden representations of data belonging to one modality only. However, the information surrounding us presents itself in multiple modalities - visual, audio, sensory perception. We learn to associate sound, visuals and other sensory stimuli to one object. For example, "apple" when shown as an image, or as text or heard as an audio holds the same meaning for us. A learning system should be capable of learning shared representation of multimodal data (Srivastava & Salakhutdinov, 2012). Wysoski et al. (2010) proposed a bimodal SNN model that performs person authentication using speech and visual (face) signals. In this work, we explore the possibility of two sensory inputs - audio and visual, of the same object, learning a shared representation using multiple autoencoders, and then use this shared representation to synthesize images from audio samples.

The main contributions of this work are

1. We propose an algorithm to train spiking neural networks that computes loss based on membrane potential of the output neurons. This loss is then backpropagated through the layers, with sigmoid approximation for the derivative of the neuron activation with respect to its membrane potential. The algorithm is verified by training autoencoders on MNIST and Fashion-MNIST and benchmarked against ANNs trained with mini-batch stochastic gradient descent (SGD).
2. We demonstrate that in a spike-based environment, inputs can be transformed into compressed spatio-temporal spike maps, which can be then be utilized to reconstruct the input later, or can be transferred across network models, and data modalities. A spiking autoencoder is used to generate compressed spatio-temporal spike maps of images (MNIST). For audio-to-image synthesis, a spiking audiocoder learns to map audio samples to compressed spike map representations, which are then converted back to images with high fidelity using the spiking autoencoder. This is the first work to perform audio to image synthesis in a spike-based environment.
3. We observe that in presence of sparse/quantized data, a spiking autoencoder can potentially outperform an ANN in reconstruction task, as seen by training spiking audiocoders on single/two bit spatio-temporal compressed spike maps of MNIST images. This could be a result of the ability of SNNs to learn with sparse data because of the temporal nature of their computation that is carried out over several time steps.

The paper is organized in the following manner: In Section 2, the neuron model, the network structure and notations are introduced. The backpropagation algorithm is explained in detail. This followed by the Section 3 where the performance of these spiking autoencoders is evaluated against several benchmarks. We illustrate our Audio to Image synthesis model and evaluate it for converting TI-46 digits audio samples to MNIST images. Finally, in Section 4, we conclude the paper with discussion on this work and its future prospects.

2 TRAINING SPIKING AUTOENCODERS

2.1 INPUT AND NEURON

A spiking neural network differs from a conventional ANN in two main aspects - inputs and activation functions. For an image classification task, for example, an ANN would typically take the raw pixel values as input. However, in SNNs, inputs are binary spike events that happen over time. There are several methods for input encoding in SNNs currently in use, such as rate encoding, rank order coding and temporal coding (Wu et al., 2007). One of the most common methods is rate encoding, where each pixel is mapped to a neuron that produces a Poisson spike train, and its firing rate is proportional to the pixel value. In this work, every pixel value of $0 - 255$ is scaled to a value between $[0, 1]$ and a corresponding Poisson spike train of fixed duration, with a pre-set maximum firing rate, is generated (Fig.1).

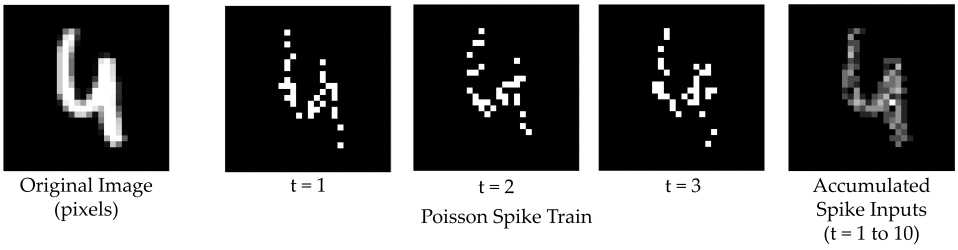


Figure 1: The input image is converted into a spike map over time. At each time step neurons spike with a probability proportional to the corresponding pixel value at their location. These spike maps, when summed over several time steps, resemble the original input

The neuron model is that of a leaky integrate-and-fire (LIF) neuron. The membrane potential (V_{mem}) is the internal state of the neuron that gets updated at each time step based on the input of the neuron, $Z^{[t]}$ (eq. 1).The output activation ($A^{[t]}$) of the neuron depends on whether V_{mem} reaches a threshold (V_{th}) or not. At the time instant when $V_{mem} \geq V_{th}$, the neuron spikes (eq. 2). At any time instant, the output of the neuron is 0 if it has not spiked, or 1 if it has spiked. The leak factor is determined by a constant α. After a neuron spikes, it's membrane potential is reset to 0. Fig. 2b illustrates a typical neuron's behavior over time in an SNN.

$$V_{mem}^{[t]} = (1 - \alpha)V_{mem}^{[t-1]} + Z^{[t]} \tag{1}$$

$$A^{[t]} = \begin{cases} 0, & V_{mem}^{[t]} < V_{th} \\ 1, & V_{mem}^{[t]} \geq V_{th} \end{cases} \tag{2}$$

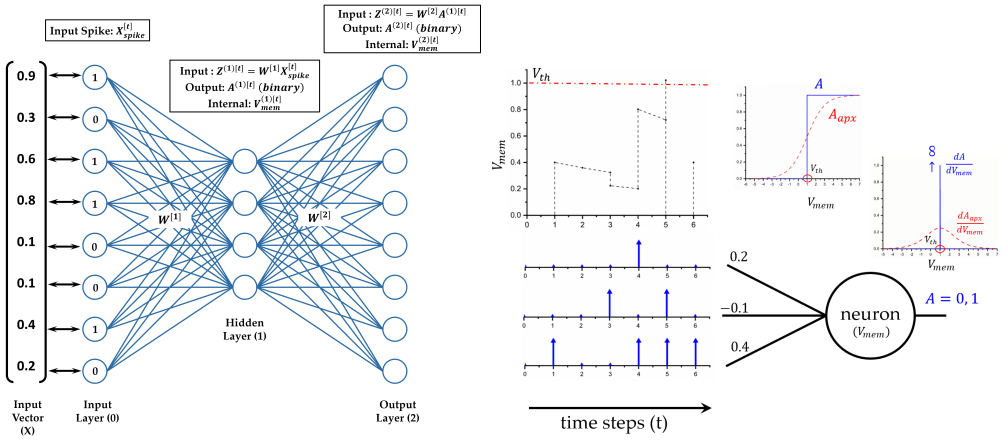
The activation function (eq. 2) is non-differentiable with respect to V_{mem}, and hence we cannot take its derivative during backpropagation. For backpropagation, the derivative of $A^{[t]}$ is approximated as the derivative of a sigmoid, ($A_{apx}^{[t]}$), which is centered around V_{th} (eq. 3, 4).

$$A_{apx}^{[t]} = \frac{1}{1 + \exp(-(V_{mem}^{[t]} - V_{th}))} \tag{3}$$

$$\frac{\partial A^{[t]}}{\partial V_{mem}^{[t]}} \approx \frac{\partial A_{apx}^{[t]}}{\partial V_{mem}^{[t]}} = \frac{\exp(-(V_{mem}^{[t]} - V_{th}))}{(1 + \exp(-(V_{mem}^{[t]} - V_{th})))^2} \tag{4}$$

2.2 NETWORK MODEL

The autoencoder used here is a two layer fully connected feed forward network. To evaluate our proposed training algorithm, we have used two datasets - MNIST (LeCun et al., 1998) and Fashion MNIST (Xiao et al., 2017). For both the datasets the input and the output layers have 784 neurons each, which is the input size of the datasets. The number of $layer^{(1)}$ neurons varies with dataset. The input neurons ($layer^{(0)}$) are mapped to the image pixels in a one-to-one manner and generate



(a) A two layer feed-forward spiking neural network at any given arbitrary time instant

(b) A leaky integrate and fire (LIF) neuron model with 3 synapses/weights at its input

Figure 2: The operation of a Spiking Neural Network:(a) The input vector is mapped one-to-one to the input neurons($layer^{(0)}$). The input value governs the firing rate of the neuron, i.e. number of times the neuron output is 1 in a given duration. (b) The membrane potential of the neuron integrates over time (with leak). As soon as it crosses V_{th}, the neuron output changes to 1, and V_{mem} is reset to 0. For taking derivative during backpropagation, a sigmoid approximation is used for the neuron activation

the Poisson spike trains. These autoencoders later form the building blocks of the audio-to-image synthesis network. The description of the network and the notation used throughout the paper is given in Fig. 2a.

2.3 Backpropagation using Membrane Potential

In this work, loss is computed using the membrane potential of output neurons (Panda & Roy, 2016) at every time step and then it's gradient with respect to weights is backpropagated for weight update. The input image is provided to the network as 784×1 binary vector over T time steps, represented as $X^{(t)}_{spike}$. At each time step the desired membrane potential of the output layer is calculated (eq. 5). The loss is the difference between the desired membrane potential and the actual membrane potential of the output neurons. Additionally a masking function can be used that helps us focus on specific neurons at a time. The mask used here is bitwise *XOR* between expected spikes ($X^{[t]}_{spike}$) and output spikes ($A^{(2)[t]}$) at a given time instant. The mask only preserves the error of those neurons that either were supposed to spike but did not spike, or were not supposed to spike, but spiked. It sets the loss to be zero for all other neurons. We observed that masking is essential for training in spiking autoencoder as shown in Fig. 3b.

$$O^{[t]} = V_{th}.X^{[t]}_{spike} \tag{5}$$

$$mask = bitXOR(X^{[t]}_{spike}, A^{(2)[t]}) \tag{6}$$

$$Error = E = mask.(O^{[t]} - V^{(2)[t]}_{mem}) \tag{7}$$

$$Loss = L = \frac{1}{2}|E|^2 \tag{8}$$

The weight gradients, $\frac{\partial W}{\partial L}$, are computed by back-propagating loss in the two layer network as depicted in Fig. 2a. We derive the weight gradients below.

$$\frac{\partial L}{\partial V^{(2)[t]}_{mem}} = -E \tag{9}$$

From eq. 1,

$$\frac{\partial V_{mem}^{(2)[t]}}{\partial W^{(2)}} = (1-\alpha)\frac{\partial V_{mem}^{(2)[t-1]}}{\partial W^{(2)}} + \left[A^{(1)[t]}\right]^T \tag{10}$$

From eq. 9 - 10 and applying the chain rule.

$$\frac{\partial L}{\partial W^{(2)}} = \frac{\partial L}{\partial V_{mem}^{(2)[t]}}\frac{\partial V_{mem}^{(2)[t]}}{\partial W^{(2)}} = -E\left[(1-\alpha)\frac{\partial V_{mem}^{(2)[t-1]}}{\partial W^{(2)}} + \left[A^{(1)[t]}\right]^T\right] \tag{11}$$

From eq. 1,

$$\frac{\partial V_{mem}^{(2)[t]}}{\partial Z^{(2)[t]}} = I \tag{12}$$

From 9 and 12, we obtain the local error of $layer^{(2)}$ with respect to the overall loss which is back-propagated to $layer^{(1)}$.

$$\delta_2 = \frac{\partial L}{\partial Z^{(2)[t]}} = I(-E) = -E \tag{13}$$

Next, the gradients for $layer^{(1)}$ are calculated.

$$\frac{\partial Z^{(2)[t]}}{\partial A^{(1)[t]}} = W^{(2)} \tag{14}$$

From eq. 3 - 4

$$\frac{\partial A^{(1)[t]}}{\partial V_{mem}^{(1)[t]}} \approx \frac{\partial A_{apx}^{(1)[t]}}{\partial V_{mem}^{(1)[t]}} = \frac{\exp(-(V_{mem}^{(1)[t]} - V_{th}))}{(1+\exp(-(V_{mem}^{(1)[t]} - V_{th})))^2} \tag{15}$$

From eq. 1,

$$\frac{\partial V_{mem}^{(1)[t]}}{\partial W^{(1)}} = (1-\alpha)\frac{\partial V_{mem}^{(1)[t-1]}}{\partial W^{(1)}} + \left[X_{spike}^{[t]}\right]^T \tag{16}$$

From 13 - 16,

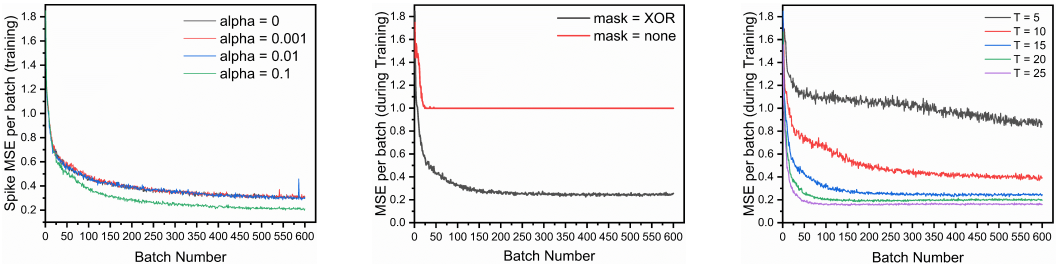
$$\frac{\partial L}{\partial W^{(1)}} = \frac{\partial L}{\partial V_{mem}^{(1)[t]}}\frac{\partial V_{mem}^{(1)[t]}}{\partial W^{(1)}} = \left[\left[W^{(2)}\right]^T\delta_2 \circ \frac{\partial A^{(1)[t]}}{\partial V_{mem}^{(1)[t]}}\right]\left[(1-\alpha)\frac{\partial V_{mem}^{(1)[t-1]}}{\partial W^{(1)}} + \left[X_{spike}^{[t]}\right]^T\right] \tag{17}$$

Thus, equations 11 and 17 show how gradients of the loss function with respect to weights are calculated. For weight update, we use mini-batch gradient descent and a weight decay value of 1e-5. We implement Adam optimization (Kingma & Ba, 2014), but the first and second moments of the weight gradients are averaged over time steps per batch (and not averaged over batches). We store $\frac{\partial V_{mem}^{(l)[t]}}{\partial W^{(l)}}$ of the current time step for use in next time step. The initial condition is, $\frac{\partial V_{mem}^{(l)[0]}}{\partial W^{(l)}} = 0$. If a neuron spikes, it's membrane potential is reset and therefore we reset $\frac{\partial V_{mem}^{(l,m)[t]}}{\partial W^{(l)}}$ to 0 as well, where l is the layer number and m is the neuron number.

3 Experiments

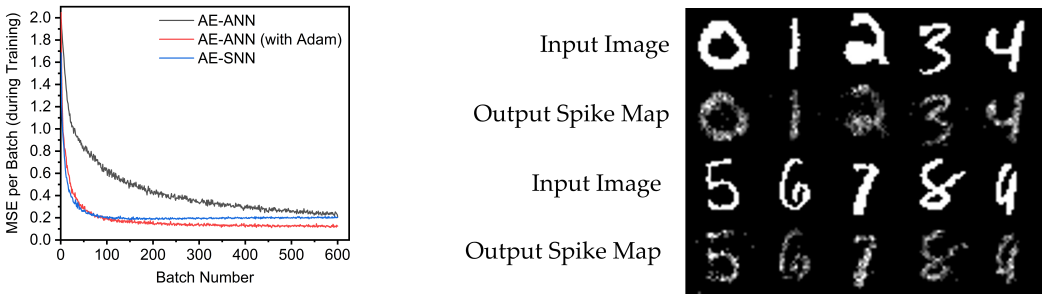
3.1 Regenerative Learning with Spiking Autoencoders

For MNIST, a 784-196-784 fully connected network is used. The spiking autoencoder (AE-SNN) is trained for 1 epoch with a batch size of 100, learning rate 5e-4, and a weight decay of 1e-4. The threshold, V_{th}, is set to 1. We use define two metrics for network performance. Spike-MSE is the mean square error between the input spike map and the output spike map, both summed over the entire duration. MSE is the mean square error between the input image and output spike map summed over the entire duration. Both are normalized, zero mean and unit variance, and then the mean square error is computed. The duration of inference is kept same as the training duration of the network. It is observed in Fig. 3a that a leaky neuron (LIF) performs better than a neuron without any leak ($\alpha = 0$), i.e an Integrate-and-Fire (IF) neuron. We use Spike-MSE as the comparison metric, to observe how well the autoencoder can recreate the input spike train. Going forth, we set the leak coefficient at 0.1 for all subsequent simulations. Fig. 3b shows that using a mask function is essential for training this type of network. Without a masking function, all of the 784 neurons are being forced to have membrane potential of 0 or V_{th}, which makes training difficult. With a masking



(a) Effect of leak coefficient (b) Effect of Mask function (c) Effect of spike train duration

Figure 3: The AE-SNN (784-196-784) is trained over MNIST (60,000 training samples, batch size = 100) and we study the impact of (a) leak, (b) mask, and (c) input spike train duration.



(a) Spiking autoencoder (AE-SNN) versus AE-ANNs (trained with/ without Adam)

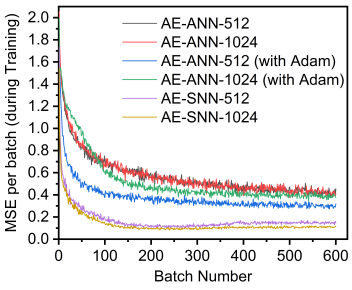
(b) Regenerated images from test set for AE-SNN (input spike duration = 15, leak = 0.1)

Figure 4: AE-SNN trained on MNIST (training examples = 60,000, batch size = 100)

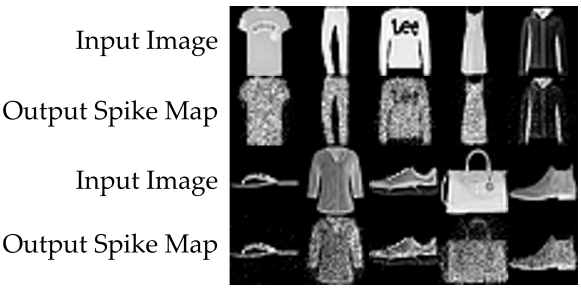
function, the neurons that are not supposed to fire, are just being trained to not have a membrane potential that exceeds the threshold, which is a more relaxed rule to enforce. Increasing the duration of the input spike train improves the performance as shown in Fig.3c. However, it comes at the cost of increased training time as backpropagation is done at each time step, as well as increased inference time. We settle for an input time duration of 15 as a trade-off between MSE and time taken to train and infer for the next set of simulations.

For comparison with ANNs, a network (AE-ANN) of same size (784×196×784) is trained with SGD, both with and without Adam optimizer (Kingma & Ba, 2014) on MNIST for 1 epoch with a learning rate of 0.1, batch size of 100, and weight decay of 1e-4. When training the AE-SNN, the first and second moments of the gradients are computed over sequential time steps within a batch (and not across batches). Thus it is not analogous to the AE-ANN trained with Adam, where the moments are computed over batches. Hence, we compare our network with both variants of the AE-ANNs. The AE-SNN achieves better performance than the AE-ANN trained without Adam; however it lacks behind AE-ANN optimized Adam as shown in Fig. 4a. Some of the reconstructed MNIST images are depicted in Fig. 4b. One important thing to note is that the AE-SNN is trained at every time step, hence there are 15× more backpropagation steps as compared to an AE-ANN. However at every backpropagation step, the AE-SNN only backpropagates the error vector of a single spike map, which is very sparse, and carries less information than the error vector of the AE-ANN.

Next, the spiking autoencoder is next evaluated on Fashion-MNIST dataset (Xiao et al., 2017). It is similar to MNIST, and is composed of 28×28 gray-scale images (60,000 training, 10,000 testing) of clothing items belonging to 10 distinct classes. We test our algorithm on two network sizes: 784-512-784 (AE-SNN-512) and 784-1024-784 (AE-SNN-1024). The AE-SNNs are compared against AE-ANNs of the same sizes (AE-ANN-512, AE-ANN-1024) in Fig. 5a. For the AE-SNNs, the duration of input spike train is 60, leak coefficient is 0.1, and learning rate is set at 5e-4. The networks are trained for 1 epoch, with a batch size of 100. The longer the spike duration, the better



(a) AE-SNN (784×(512/1024)×784) versus AE-ANNs (trained with/without Adam)



(b) Regenerated images from test set for SNN-1024

Figure 5: AE-SNN trained on Fashion-MNIST (training examples = 60,000, batch size = 100)

would be the spike image resolution. For a duration of 60 time steps, a neuron can spike anywhere between zero to 60 times, thus allowing 61 gray-scale levels. Some of the generated images by AE-SNN-1024 are displayed in Fig. 5b. The AE-ANNs are trained for 1 epoch, batch size 100, learning rate 5e-3 and weight decay 1e-4. For Fashion-MNIST, the AE-SNNs exhibited better performance than AE-ANNs as shown in Fig. 5a. This is an interesting observation, where the better performance comes at the increased effort of per-batch training. Also it exhibits such behavior on only this dataset, and not on MNIST (4a). The spatio-temporal nature of training over each time step could possibly train the network to learn the details in an image better. We also observed that, for both datasets, MNIST and Fashion-MNIST, the AE-SNN converges faster than AE-ANNs trained without Adam, and converges at almost the same time as an AE-ANN trained with Adam. The proposed spike-based backpropagation algorithm is able to bring the AE-SNN performance at par, or even better, than AE-ANNs.

3.2 Audio to Image Synthesis using Spiking Auto-Encoders

3.2.1 Dataset

For audio samples, the 0-9 digits subset of TI-46 speech corpus (Liberman et al., 1993) is used. The dataset is composed of read utterances of 16 speakers. It has total 4136 audio samples. The dataset is divided into 3500 train samples and 636 test samples, maintaining an 85%/15% train/test ratio. The audio clips were preprocessed using Auditory Toolbox (Slaney, 1998). They were converted to spectrograms having 39 frequency channels over 1500 time steps. The spectrogram is then converted into a 58500×1 vector of length 58500. This vector is then mapped to the input neurons ($layer^{(0)}$) of the audiocoder, which then generate Poisson spike trains over the given training interval. Images are taken from the MNIST, (LeCun et al., 1998), a dataset of handwritten digits. Two multimodal datasets are prepared as described below. The testing set is kept same for both datasets, composed of 636 audio samples.

1. **Dataset A**: 10 unique images of the 10 digits is manually selected (1 image per class) and audio samples are paired with the image belonging to their respective classes (one-image-per-audio-class)
2. **Dataset B**: Each audio sample of the training set is paired with a randomly selected image (of the same label) from the MNIST dataset (one-image-per-audio-sample).

3.2.2 Network Model

The principle of stacked autoencoders is used to perform audio-to-image synthesis. An autoencoder is built of two sets of weights; the $layer^{(1)}$ weights ($W^{(1)}$) encodes the information into a "hidden state" of a different dimension, and the second layer ($W^{(2)}$) decodes it back to it's original representation. We first train a spiking autoencoder on MNIST dataset. We use the AE-SNN as trained in Fig. 4a. Using $layer^{(1)}$ weights ($W^{[1]}$) of this AE-SNN, we generate "hidden-state" representations of the images in the training set of the multimodal dataset. These hidden-state representations are

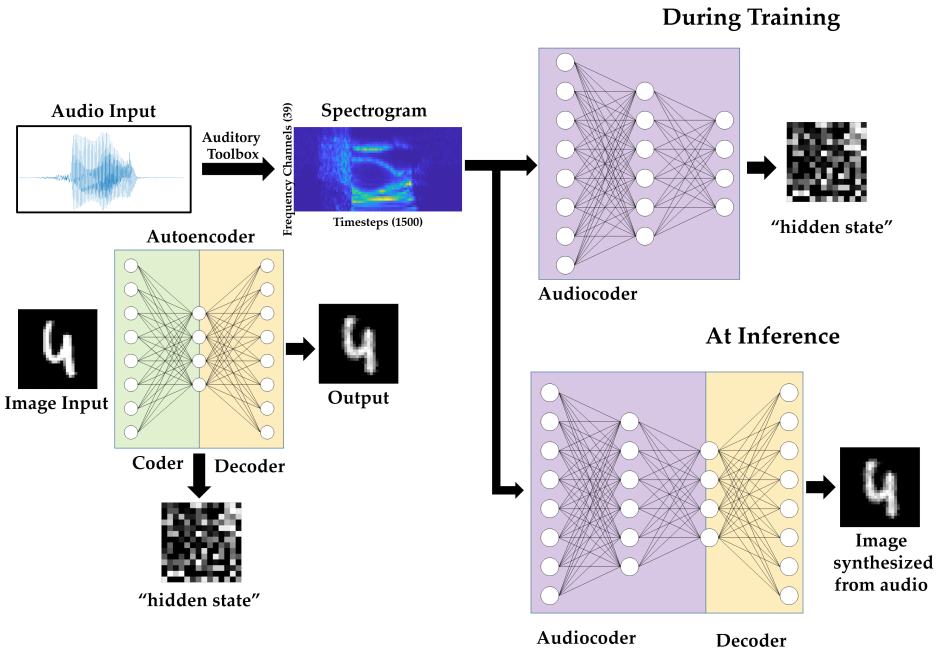


Figure 6: Audio to Image synthesis model using an Autoencoder trained on MNIST images, and an Audiocoder trained to convert TI-46 digits audio samples into corresponding hidden state of the MNIST images.

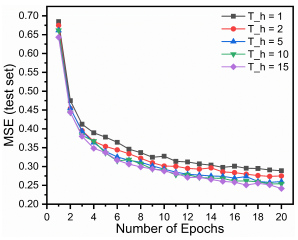
spike trains of a fixed duration. Then we construct an audiocoder: a two layer spiking network that converts spectrograms to this hidden state representation. The audiocoder is trained with membrane potential based backpropagation as described in Section 2.3. The generated representation, when fed to the "decoder" part of the autoencoder, gives us the corresponding image. The network model is illustrated in Fig. 6

3.2.3 RESULTS

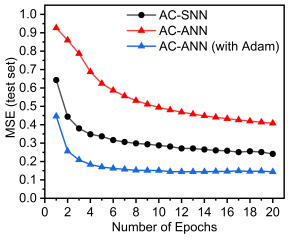
The AE-SNN used for audio-to-image synthesis task is trained using the following parameters: batch size of 100, learning rate 5e-4, leak coefficient 0.1, weight decay 1e-4, input spike train duration 15, and number of epochs 1. We use Dataset A and Dataset B (as described in section 3.2.1) to train and evaluate our audio-to-image synthesis model. The images that were paired with the training audio samples are converted to Poisson spike trains (duration 15 time steps) and fed to the AE-SNN, which generates a 196×15 corresponding bitmap as the output of $layer^{(1)}$ (Fig. 2a). This spatio temporal representation is stored. Instead of storing the entire duration of 15 time steps, one can choose to store a subset, such as first 5 or 10 time steps. We use T_h to denote the saved hidden state's duration. This stored spike map serves as the target spike map for training the audiocoder, which is a 58500×2048×196 fully connected network. The spectrogram (39×1500) of the audio samples was converted to 58500×1 vector which is mapped one-to-one to the input neurons($layer^{(0)}$). These input neurons then generate Poisson spike trains for 60 time steps. The target map was of T_h time steps was shown repeatedly over this duration. The audiocoder is trained over 20 epochs, with a learning rate of 5e-5 and leak coefficient of 0.1. Weight decay is set at 1e-4 and the batch size is 50. Once trained, the audiocoder is then merged with $W^{(2)}$ of AE-SNN to create the audio-to-image synthesis model (Fig. 6).

For Dataset A, we compare the images generated by audio samples of a class against the MNIST image of that class to compute the MSE. In case of Dataset B, each audio sample of the train set is paired with an unique image. For calculating training set MSE, we compare the paired image and the generated image. For testing set, the generated image of an audio sample is compared with all the training images in the dataset, and the lowest MSE is recorded. The output spike map is normalized

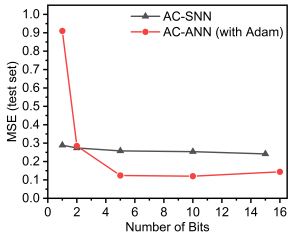
and compared with the normalized MNIST images, as was done previously. Our model gives lower MSE for Dataset A compared to Dataset B (Fig 8a), as it is easier to learn just one representative image for a class. The network trained with Dataset A generates very good identical images for audio samples belonging to a class. In comparison the network trained on Dataset B generates a blurry image, thus indicating that it has learned to associate the underlying shape and structure of the digits, but has not been able to learn finer details better. This is because the network is trained over multiple different images of the same class, and it learns what is common among them all. Fig. 8b displays the generated output spike map for the two models trained over Dataset A and B for 50 different test audio samples (5 of each class).



(a) Reconstruction loss of the audio-to-image synthesis model for varying T_h

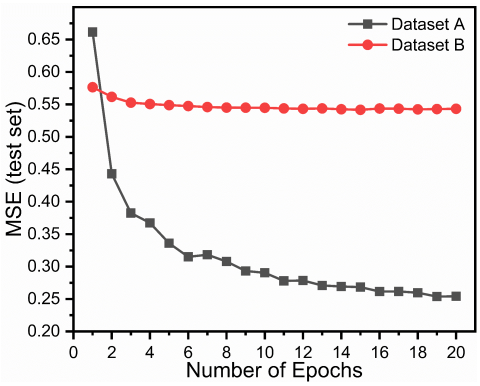


(b) Audiocoder (AC-SNN) ($T_h = 15$) vs AC-ANN (16 bit full precision)

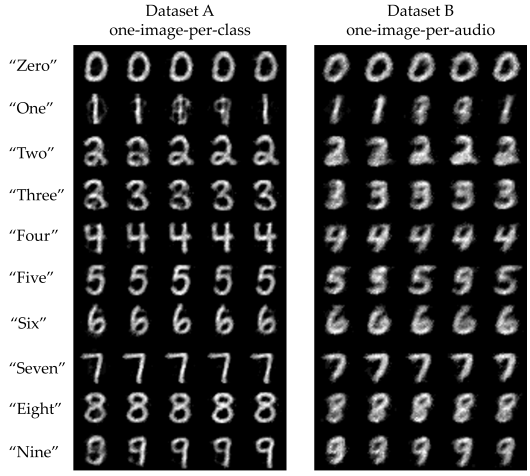


(c) Effect of training with reduced hidden state representation on AC-SNN and AC-ANN model

Figure 7: The audiocoder (AC-SNN/AC-ANN) is trained over Dataset A, while the autoencoder (AE-SNN/AE-ANN) is fixed. MSE is reported on the overall audio-to-image synthesis model composed of AC-SNN/ANN and AE-SNN/ANN



(a) AC-SNNs are trained on Dataset A and Dataset B (T_h = 10))



(b) Images synthesized from different test audio samples (5 per class) when the network is trained with Datasets A and B

Figure 8: The performance of the Audio to Image synthesis model on the two datasets - A and B

The duration (T_h) of stored "hidden state" spike train was varied from 15 to 10, 5, 2, and 1. A spike map at a single time step is a 1-bit representation. The AE-SNN compresses an 784×8 bit representation into 196×T_h-bit representation. For T_h = 15, 10, 5, 2, and 1, the compression is 2.1×, 3.2×, 6.4×, 16× and 32× respectively. Even when the AC-SNN is trained with a much smaller "hidden state", the AE-SNN is able to reconstruct the images without much loss. In Fig. 7a we observe the reconstruction loss (test set) over epochs for training using different lengths of hidden state.

For comparison, we initialize an ANN audiocoder (AC-ANN) of size 58500×2048×196. The AE-ANN trained over MNIST in section 3.1 is used to convert the images of the multimodal dataset (A/B) to 196×1 "hidden state" vectors. Each element of this vector is 16 bit full precision number. In case of AE-SNN, the "hidden state" is represented $196 \times T_h$ bit map. For comparison, we quantize the equivalent hidden state vector into 2^{T_h} levels. The AC-ANN is trained using these quantized hidden state representations with the following learning parameters: learning rate 1e-4, weight decay 1e-4, batch size 100, epochs 20. Once trained, the ANN audio-to-image synthesis model is built by combining AC-ANN and $layer^{(2)}$ weights ($W^{(2)}$) of AE-ANN. The AC-ANN is trained with/without Adam optimizer is paired with the AE-ANN trained with/without Adam optimizer respectively. In Fig. 7b, we see that our spiking model achieves a performance in between the two ANN models, a trend we have observed earlier while training autoencoders on MNIST. In this case, the AC-SNN is trained with T_h as 15, while AC-ANNs are trained without any output quantization; both are trained on Dataset A. In Fig. 7c, we observe the impact of quantization for the ANN model and the corresponding impact of lower T_h for SNN. For higher hidden state bit precision, the ANN model outperforms the SNN one. However for extreme low precision case, number of bits = 2, and 1, the SNN performs better. This could possibly be attributed to the temporal nature of SNN, where the computation is event-driven and spread out over several time steps.

Table 1: Summary of results obtained for the 3 tasks - Autoencoder on MNIST, Autoencoder on Fashion-MNIST, and Audio to Image conversion (T = input duration for SNN)

Dataset	Network Size	Epochs	T	Loss (MSE) (test)		
				SNN	ANN	ANN (with Adam)
MNIST	784-196-784	1	15	0.357	0.226	**0.122**
Fashion-MNIST	784-512-784	1	60	**0.178**	0.416	0.300
	784-1024-784	1	60	**0.140**	0.418	0.387
Audio-to-Image A	58500-2048-196/196-784	20	30	0.254	0.408	**0.144**
Audio-to-Image B	58500-2048-196/196-784	20	30	**0.543**	0.611	0.556

4 Discussion and Conclusion

In this work, we proposed an algorithm to train spiking networks, and in Table 1, we have summarized the results of this work[1][2]. The proposed algorithm brings SNN performance at par with ANNs for the given tasks. We demonstrate that spiking autoencoders can be used to generate reduced-duration spike maps ("hidden state") of an input spike train, which are a highly compressed version of the input, and they can be utilized across applications. This is also the first work to demonstrate audio to image synthesis in spiking domain. While training these autoencoders, we made a few important and interesting observations; the first one is the importance of bit masking of the output layer. Trying to steer the membrane potentials of all the neurons is extremely hard to optimize, and selectively correcting only incorrectly spiked neurons makes training easier. This could be applicable to any spiking neural network with a large output layer. Second, while the AE-SNN is trained with spike durations of 15 time steps, we can use hidden state representations of much lower duration to train our audiocoder with negligible loss in reconstruction of images for the audio-to-image synthesis task. In this task, the ANN model trained with Adam outperformed the SNN one when trained with full precision "hidden state". However, at ultra-low precision, the hidden state loses it's meaning in ANN domain, but in SNN domain, the network can still learn from it. This observation raises important questions on the ability of SNNs to possibly compute with less data. While sparsity during inference has always been an important aspect of SNNs, in this work, we explored how SNNs can be used to compress information into compact spatio-temporal representations and then reconstruct that information back from it. Another interesting observation is that we can potentially train autoencoders and stack them to create deeper spiking networks with greater functionalities. This could be an alternative approach to training deep spiking networks. Thus, this work sheds light on the interesting behavior of spiking neural networks, their ability to generate compact spatio-temporal representations of data, and offers a new training paradigm for learning meaningful representations of complex data.

[1]Table 1: Audio-to-Image A: SNN: $T_h = 15$, ANN : no quantization for hidden state

[2]Table 1: Audio-to-Image B: SNN: $T_h = 10$, ANN : no quantization for hidden state

REFERENCES

Sander M Bohte, Joost N Kok, and Han La Poutre. Error-backpropagation in temporally encoded networks of spiking neurons. *Neurocomputing*, 48(1-4):17–37, 2002.

Kendra S Burbank. Mirrored stdp implements autoencoder learning in a network of spiking neurons. *PLoS computational biology*, 11(12):e1004566, 2015.

Samanwoy Ghosh-Dastidar and Hojjat Adeli. Spiking neural networks. *International journal of neural systems*, 19(04):295–308, 2009.

Yingyezhe Jin, Peng Li, and Wenrui Zhang. Hybrid macro/micro level backpropagation for training deep spiking neural networks. *arXiv preprint arXiv:1805.07866*, 2018.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436, 2015.

Jun Haeng Lee, Tobi Delbruck, and Michael Pfeiffer. Training deep spiking neural networks using backpropagation. *Frontiers in neuroscience*, 10:508, 2016.

M Liberman, R Amsler, K Church, E Fox, C Hafner, J Klavans, M Marcus, B Mercer, J Pedersen, P Roossin, et al. Ti 46-word. *Philadelphia (Pennsylvania): Linguistic Data Consortium*, 1993.

Wolfgang Maass. Networks of spiking neurons: the third generation of neural network models. *Neural networks*, 10(9):1659–1671, 1997.

Wolfgang Maass. To spike or not to spike: that is the question. *Proceedings of the IEEE*, 103(12): 2219–2224, 2015.

Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pp. 807–814, 2010.

Priyadarshini Panda and Kaushik Roy. Unsupervised regenerative learning of hierarchical features in spiking deep networks for object recognition. In *Neural Networks (IJCNN), 2016 International Joint Conference on*, pp. 299–306. IEEE, 2016.

Abhronil Sengupta, Yuting Ye, Robert Wang, Chiao Liu, and Kaushik Roy. Going deeper in spiking neural networks: Vgg and residual architectures. *arXiv preprint arXiv:1802.02627*, 2018.

Jesper Sjöström and Wulfram Gerstner. Spike-timing dependent plasticity. *Spike-timing dependent plasticity*, 35(0):0–0, 2010.

Malcolm Slaney. Auditory toolbox. *Interval Research Corporation, Tech. Rep*, 10:1998, 1998.

Nitish Srivastava and Ruslan Salakhutdinov. Learning representations for multimodal data with deep belief nets. In *International conference on machine learning workshop*, volume 79, 2012.

Amirhossein Tavanaei, Timothée Masquelier, and Anthony Maida. Representation learning using event-based stdp. *Neural Networks*, 2018.

Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pp. 1096–1103. ACM, 2008.

Paul J Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 1990.

QingXiang Wu, Martin McGinnity, Liam Maguire, Bredan Glackin, and Ammar Belatreche. Learning mechanisms in networks of spiking neurons. In *Trends in Neural Computation*, pp. 171–197. Springer, 2007.

Yujie Wu, Lei Deng, Guoqi Li, Jun Zhu, and Luping Shi. Spatio-temporal backpropagation for training high-performance spiking neural networks. *Frontiers in neuroscience*, 12, 2018.

Simei Gomes Wysoski, Lubica Benuskova, and Nikola Kasabov. Evolving spiking neural networks for audiovisual information processing. *Neural Networks*, 23(7):819–835, 2010.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.