DCT-SNN: Using DCT to Distribute Spatial Information over Time for Low-Latency Spiking Neural Networks

Isha Garg* Sayeed Shafayet Chowdhury* Kaushik Roy
Purdue University, West Lafayette, IN, USA 47907
gargi, chowdh23, kaushik@purdue.edu

Abstract

Spiking Neural Networks (SNNs) offer a promising alternative to traditional deep learning, since they provide higher computational efficiency due to event-driven information processing. SNNs distribute the analog values of pixel intensities into binary spikes over time. However, the most widely used input coding schemes, such as Poisson based rate-coding, do not leverage the additional temporal learning capability of SNNs effectively. Moreover, these SNNs suffer from high inference latency which is a major bottleneck to their deployment. To overcome this, we propose a time-based encoding scheme that utilizes Discrete Cosine Transform (DCT) to reduce the number of timesteps required for inference (DCT-SNN). DCT decomposes an image into a weighted sum of sinusoidal basis images. At each time step, a single frequency base, taken in order and modulated by its corresponding DCT coefficient, is input to an accumulator that generates spikes upon crossing a threshold. We use the proposed scheme to train DCT-SNN, a low-latency deep SNN with leaky-integrate-and-fire neurons using surrogate gradient descent based backpropagation. We achieve top-1 accuracy of 89.94%, 68.30% and 52.43% on CIFAR-10, CIFAR-100 and TinyImageNet, respectively using VGG architectures. Notably, DCT-SNN performs inference with 2-14X reduced latency compared to other state-of-the-art SNNs, while achieving comparable accuracy to their standard deep learning counterparts. The dimension of the transform allows us to control the number of timesteps required for inference. Additionally, we can trade-off accuracy with latency in a principled manner by dropping the highest frequency components during inference. The code is publicly available.[1]

1. Introduction

Deep Learning networks have tremendously improved state-of-the-art performance for many tasks such as object detection, classification and natural language processing [7, 12, 19]. However, such architectures are extremely energy-intensive [22] and require custom architectures and training methodologies for edge deployment [14]. To address this, Spiking Neural Networks (SNNs) have emerged as a promising alternative to traditional deep learning architectures [25, 31]. SNNs are bio-plausible networks inspired from the learning mechanisms observed in mammalian brains. They are analogous in structure to standard networks, but perform computation in the form of spikes instead of fully analog values, as done in standard networks. In this paper, we refer to standard networks as Analog Neural Networks (ANNs) to distinguish them from their spiking counterparts with digital (spiking) inputs. The input and the correspondingly generated activations in SNNs are all binary spikes and inference is performed by accumulating the spikes over time. This can be visualized as distributing the one step inference of ANNs into a multi-step, very sparse inference scheme in the SNN.

The primary source of energy efficiency of SNNs comes from the fact that very few neurons spike at any given timestep. This event driven computation and the replacement of every multiply–accumulate (MAC) operation in the ANN by an addition in SNN allows SNNs to infer with lesser energy. This energy benefit can be further enhanced using custom SNN implementations with architectural modifications [17]. Li *et al.* [23] have released a spiking version of the CIFAR-10 dataset based on inputs from neuromorphic sensors. IBM has designed a non-commercial processor 'TrueNorth' [2], and Intel has designed its equivalent 'Loihi' [6], that can train and infer on SNNs. Blouw *et al.* [3] have shown SNNs implemented on Loihi to be two orders of magnitude more efficient than an equivalent ANN running on GPU for keyword spotting. However, the higher inference latency in SNNs due to accumulation of spikes over timesteps remains a challenge. Energy efficiency at the cost of too high a latency would still hamper real-time deployment. Consequently, reduction of timesteps required for inference in SNNs is an active field of research. One of the major factors that affect the number

*equal contribution
[1] https://github.com/SayeedChowdhury/dct-snn

of timesteps needed is the encoding scheme that converts pixels into spikes over the timesteps. Currently, the most common encoding scheme is Poisson spike generation [32], where the spikes at the input are generated as a Poisson spike train, with the mean spiking rate proportional to the pixel intensity. This scheme does not encode anything meaningful in the temporal axis, with each timestep being the same as any other. Moreover, networks trained using this scheme suffer from high inference latency [32]. Temporal coding schemes such as phase [18] or burst [27] coding have been introduced to better encode temporal information into the spike trains, but still incur high latency and require a large number of spikes for inference. Another related temporal method, time-to-first-spike (TTFS) coding [28, 44], limits the number of spikes per neuron but the high latency problem still persists. Relative timing of spikes to encode information has been used in [5], but the results are only reported for simple tasks like MNIST and its scalability to deeper architectures such as VGG and more complex datasets like CIFAR remains unclear.

In this paper, we propose a novel encoding scheme to convert pixels into spikes over time. The proposed scheme utilizes a block-wise matrix multiplication to decompose spatial information into a weighted sum of bases, and then reverses the transform to allow reconstruction of the input over multiple timesteps. These bases, taken one per timestep, modulated by the weights from the forward transform are then presented to the spike generating layer. The spike generator sums the contribution of all bases seen until the current timestep, as shown in Figure 1. Though any invertible matrix can be utilized as the transform, the ideal transform follows the properties of energy compaction and orthonormality of bases as outlined in Section 3.1. We motivate Discrete Cosine Transform (DCT) as the ideal choice, since it is data independent, with orthogonal bases ordered by their contribution to spectral energy. Each timestep gets the information corresponding to a single base, starting from the zero frequency component at the first timestep. Each subsequent step refines the input representation progressively. At the end of the cycle, the entire pixel value has passed through the spike generating neuron. Thus, this methodology successfully distributes the pixel value over all the timesteps in a meaningful manner. Choosing the appropriate dimensions of the transform provides a fine grained control on the number of timesteps used for inference. We use the proposed scheme to train DCT-SNN, and observe that it reduces the timesteps needed to infer an image taken from CIFAR-10, CIFAR-100 and TinyImageNet datasets from 100 to 48, 125 to 48 and 250 to 48, respectively, while achieving comparable accuracy to the state-of-the-art Poisson encoded SNNs [30]. Additionally, ordering the frequency bases being input at each timestep provides a principled way of trading off accuracy for a reduced number of timesteps during inference, if desired, by dropping the least important (highest frequency) components.

To summarize, the main contributions of this work are-

- A novel input encoding scheme for SNNs is introduced wherein each timestep of computation encodes distinct information, unlike other rate-encoding methods.
- The proposed encoding scheme is used to train DCT-SNN, which is able to infer with 2-14X lower timesteps compared to other state-of-the-art SNNs, while achieving comparable accuracy.
- The proposed technique is, to the best of our knowledge, the first work that leverages frequency domain learning for SNNs on vision applications.
- To the best of our knowledge, this is the first work that orders timesteps by significance to reconstruction. This provides an option to trade-off accuracy for faster inference by trimming some later frequency components, which is non-trivial to perform in other SNNs.

2. Related Works

Learning ANNs in the frequency domain. Successful learning for vision tasks in the frequency domain has been demonstrated in ANNs in several works. These utilize the DCT coefficients directly available from JPEG compression method [36] without performing the decompression steps. Conventional CNNs were used with DCT coefficients as input for image classification in [35] and [29]. Ehrlich and Davis [11] proposed a model conversion algorithm to apply pretrained spatial domain networks to JPEG images. Wavelet features are utilized in [37] to train CNN-based classifiers. However, these methods suffer a small accuracy degradation compared to learning in spatial domain. DCT features were used effectively for large scale classification and instance segmentation tasks in [41]. Although such frequency domain approaches have proved fruitful in ANNs, it is unexplored in SNNs despite the conversion of spatial bases of the image to temporal bases in the frequency domain being intuitively related to distributing the analog pixel values in ANNs to spikes over time in SNNs. There exist three prominent line of works for training SNNs, namely using spike-timing-dependent plasticity rules (STDP) [8], ANN-SNN conversion [9, 33] and training an SNN from scratch [34, 39]. While STDP-based local learning [8, 42] is more bio-plausible, scaling such algorithms beyond MNIST type of tasks has been challenging. Hence, the following discussion focuses mainly on conversion and backpropagation based works.

ANN-SNN Conversion. The most common approach of training rate-coded deep SNNs is to first train an ANN and then convert it to an SNN for finetuning [4, 9, 33].

Usually, the ANNs are trained with some limitations to facilitate this, such as not using bias, batch-norm or average pooling layers, though some works are able to bypass these constraints [32]. To convert ANNs to SNNs successfully, it is critical to adjust the threshold of Integrate-and-Fire (IF) / Leaky IF (LIF) neurons properly. Sengupta *et al.* [33] recommend computing the layerwise thresholds as the maximum pre-activation of the neurons. This results in high accuracy but incurs high inference latency (about 1000 timesteps). Alternatively, Rueckauer *et al.* [32] choose a certain percentile of the pre-activation distribution as the threshold, reducing inference latency and improving robustness. The difference between these works and ours lie in the significance we attach to the timesteps.

Backpropagation from Scratch and Hybrid Training. Another approach to training SNNs is learning from scratch using backpropagation, which is challenging due to the non-differentiability of the spike function at the time of spike. Additionally, this training takes a long time to converge. Surrogate gradient based optimization [26] has been utilized to circumvent this issue and implement backpropagation in SNNs effectively [15, 21]. Surrogate gradient based backpropagation on the membrane potential at only a single timestep was proposed in [43]. Shrestha and Orchard [34] compute the gradients using the difference between the membrane potential and the threshold, but only demonstrate on MNIST using shallow architectures. Wu *et al.* [39] perform backpropagation through time (BPTT) on SNNs with a surrogate gradient defined on the membrane potential as it is continuous-valued. Overall, SNNs trained with BPTT using such surrogate-gradients have been shown to achieve high accuracy and low latency ($\sim$100-125 timesteps), but the training is very compute intensive compared to conversion techniques. Rathi *et al.* [30] propose a combination of both methods, where a pre-trained ANN serves as initialization for subsequent surrogate gradient learning in the SNN domain. The hybrid approach improves upon conversion by reducing latency and speeding up convergence. However, this is orthogonal to the encoding scheme and can be used to improve the performance of any rate-coded scheme. In this work, we adopt the hybrid training method to train the SNNs. The key distinction of our method lies in how the pixel values are encoded over time, which is described next.

3. Encoding Scheme

An ideal encoding scheme to convert pixel values into spikes over time should capture relevant information in the temporal statistics of the data. Additionally, the total spike activity over all the timesteps at the input neuron should correspond to the pixel intensity. Our encoding scheme deconstructs the image into a weighted sum of basis functions. We invert this transform to reconstruct the image over time steps. Each basis function, taken one per timestep and modulated by the weights from the deconstruction is input to an Integrate-and-Fire (IF) neuron, which accumulates the input over timesteps and fires when accumulation crosses its threshold.

3.1. A Generic 1-D Transform to Distribute Pixels over Time

1-D transformation. For simplicity, we first consider a one dimensional transform over the entire input pixel space. Let us consider a single d-dimensional image, $X \in \mathbb{R}^{1\times d}$. We transform this image using a transform matrix T into a new coordinate system, where $T \in \mathbb{R}^{d\times d}$. The transformed vector, $Y = XT$, where $Y \in \mathbb{R}^{1\times d}$, contains the coefficients of the image in the new coordinate system. This is shown pictorially for $d = 5$ in Figure 1. Assume that T is a full rank matrix and let us consider $T^{-1} \in \mathbb{R}^{d\times d}$, the inverse transformation matrix that takes us back into the original coordinate system. For reasons clarified shortly, assume that T is an orthonormal matrix with its inverse equal to its transpose, $T^{-1} = T'$. The forward transform represents deconstruction of the input X into a weighted sum of basis vectors, represented by the rows of T^{-1}, or columns of T as shown in Figure 1. These bases are referenced by $T_n, n = 1, 2....d$. If we input one basis function per timestep to the SNN, we get intermediate representations of the input at timestep t, $X^{(t)}$ by modulating the t-th basis by its corresponding weight from the forward transform, summed over all previous timesteps. Summing over all bases allows us to reconstruct X. Mathematically,

$$X^{(t)} = \sum_{n=1}^{t} y_n T_n \qquad \text{and} \qquad X^{(d)} = X, \tag{1}$$

The analog value of $X^{(t)}$ is the input to the spike generator at each timestep and is converted to spikes using IF neurons as shown in Figure 2. Hence, we have successfully distributed the input X over d timesteps, with each timestep carrying information over our chosen bases. In the next section, we discuss the desirable properties of the bases for deconstructing X.

Desirable Properties of the Basis Vectors. The columns of the transform matrix T contain the bases to deconstruct X. Since we use one basis per timestep, we want each base to offer non-interfering information about X. This is captured by the **orthonormality** constraint on T. Orthonormal columns avoid cancellation of information between timesteps, and relate the forward and reverse transforms by a transpose operation. The second constraint on T is that the bases be ranked by a measure of the information they carry. This allows each basis function to successively refine the representation per timestep. It is desirable to have the bulk of the information focused in the earlier timesteps, with fine-grained information added by the later steps. This **ordering of bases** allows us to

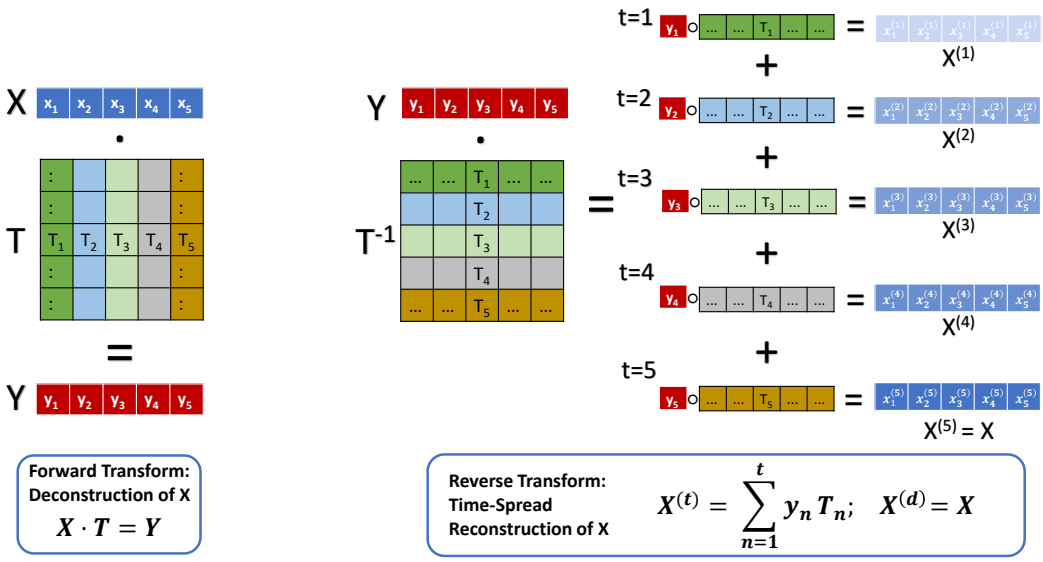

Figure 1: 1-D Encoding Scheme: On the left we show the **Forward Transform**. T represents the transform matrix that takes the input X into an intermediate coordinate system, resulting in representation Y. On the right, we show the **Inverse Transform** that uses Y to reconstruct X over time. Here $T^{-1} = T'$. The input image X is reconstructed progressively at each timestep by summing the basis vectors T_n modulated by the corresponding coefficient y_n over all previous timesteps. Since there are 5 bases shown here, X requires 5 timesteps for reconstruction.

drop bases in a principled manner to trade-off accuracy for latency during inference.

Transforms that Satisfy Constraints. There are two widely used transforms that satisfy these properties: the **DCT** transform [1] and the Karhunen–Loève transform [10], also known as **Principal Component Analysis (PCA)**. DCT decomposes an image into a linear combination of sinusoidal frequencies, ranked by spectral energy. PCA uses the eigenvectors of the covariance matrix of the inputs as the bases, ranked by the amount of variance they explain. DCT is commonly used in JPEG Compression and PCA in dimensionality reduction, by approximating the later components. However, PCA results in dataset dependent bases, whereas the DCT bases are pre-determined, avoiding extra computation. The 1-D DCT transform uses the following equation to take the pixel values x_n into DCT coefficients X_k using sinusoidal bases.

$$X_k = \sum_{n=0}^{N-1} x_n \cos\left[\frac{\pi}{N}\left(n+\frac{1}{2}\right)k\right] \qquad k = 0,\ldots,N-1. \tag{2}$$

The sinusoidal bases can be entered as the columns of a transformation matrix T. The forward transform is then computed as $Y = TXT'$ and the reverse transform is computed as $X = T'YT$. A comparison of the results for DCT, PCA, a random orthonormal transform without ranked bases, and a random non-orthonormal non-ranked transform is shown in Table 1. For the rest of the paper, we use the dataset agnostic DCT. Additionally, the conversion of spatial to temporal frequency in DCT lends itself intuitively to the concept of distributing spatial information in ANNs into spikes over time.

3.2. 2-D Discrete Cosine Transform

We now extend the scheme to 2-D. The 2-D DCT is just the 1-D DCT applied first along the width channel and then along the length channel. Images are high dimensional, resulting in large transformation matrices T. This is undesirable since the number of DCT bases (or the dimension of T) dictates the number of timesteps required to reconstruct the image. To tackle this, similar to JPEG compression [36], we first convert the image from RGB to YCbCr domain, and then perform 2-D DCT on blocks of size $n \times n$, getting n^2 ordered frequency components. We replace the $n \times n$ pixel block with the equivalently reshaped frequency coefficients. An $n \times n$ block requires n^2 timesteps for perfect reconstruction of the pixel block. Small values of n allow us to reconstruct the images by summing over only a few basis images. In standard JPEG compression, 8×8 blocks are used, resulting in the 64 basis images shown in Figure 3. The bases obtained from PCA on the training dataset of CIFAR-10 are also shown. Empirically, we find that block sizes of 4×4 converge to the best accuracy with the lowest number of timesteps. We usually need to run 3 cycles to achieve convergence to best accuracy, amounting to $4 \times 4 \times 3 = 48$ timesteps. In each of the 3 cycles, we repeat the 16 DCT coefficients and bases, to allow time for spike propagation to the deeper layers. This is discussed in further detail in Section 4. Unlike the JPEG

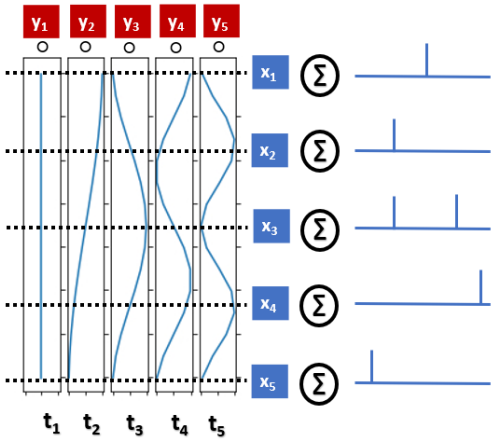

Figure 2: Spike generation with 1-D DCT basis functions input per timestep (shown vertically). The neuron spikes when the accumulated value crosses a threshold.

compression scheme, we utilize an overlapped scheme to improve accuracy, where the blocks of pixels overlap. This is equivalent to performing convolution with a kernel size of 4 and a stride of 2, and increases the input dimensions by $4\times$. To counter this, we add an extra 2×2 average pooling layer before the linear layers.

3.3. Conversion from ANN and Threshold Selection

In our scheme, the SNN trains on intermediate pixel representations. Hence, we utilize an ANN trained with pixels (rather than DCT coefficients) for initialization. The threshold of the IF neuron at the spike generator significantly affects the timesteps required for spike propagation to deeper layers. This IF neuron, as shown in Fig. 2, receives the bases modulated by the DCT coefficients and accumulates them over timesteps, firing when the accumulation crosses the threshold. We allow both positive and negative spikes to account for the positive and negative cycles of the sinusoidal bases. Similar to the hidden layer neurons, the threshold is chosen as a percentile of the accumulation at the spike generator neurons. We obtain best results by using 6.5 and 93.5 percentile of the accumulation as thresholds for negative and positive spikes, respectively.

4. Experiments and Results

We implement DCT-SNN by incorporating the proposed encoding scheme with surrogate-gradient based learning using LIF neurons. Starting with a pretrained ANN, we copy the weights to the SNN and select the 99.9 percentile of the pre-activation distribution at each layer as its threshold. The details of the learning methodology and hyperparameters of training are given in the supplementary in sections 1 and 2, respectively. The implementation is provided as part of the supplementary material.

Choice of Transformation. We first analyze the performance of DCT-SNN trained on different choices of transformation matrices (denoted as T in section 3.1). Table 1 shows the results for a VGG5 network trained on CIFAR-10. With a random T, the network does converge but with much lower accuracy than the ANN. Next, to avoid interference between different bases, we use a random orthonormal T. Table 1 shows that the accuracy improves by $\sim 20\%$ compared to non-orthonormal case. However, this choice of T does not perform energy compaction. Ranking bases by their contribution to reconstruction allows us to trade-off accuracy for latency during inference. To incorporate this, we experiment with the transformation matrix generated by performing PCA on 4×4 blocks of the inputs from the training dataset. While this satisfies both the desired properties and gives the best performance, it is a data-dependent transform. Therefore, we utilize the fixed DCT matrix, and find that it performs at par with PCA, while additionally being data-agnostic. For all subsequent analysis, we use DCT as the choice of transformation.

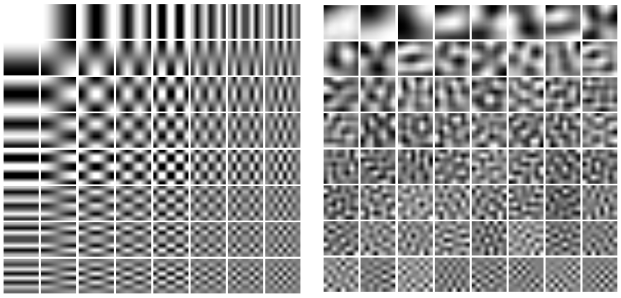

Figure 3: 8×8 2-D DCT bases (left) and PCA bases for CIFAR-10 (right). The DCT bases are ranked in a zig-zag fashion starting from top left to the bottom right and the PCA bases are ranked from left to right and top to bottom.

Effect of Block Size and Overlap. Having chosen DCT to determine the bases of our encoding scheme, we tune the block size and stride. The results are shown in Fig. 4. 'DCT-x' denotes a network trained on inputs transformed with DCT of blocksize x, and 'ov' refers to overlap among the DCT blocks. Reducing the blocksize from 16 to 4 improves accuracy consistently. Moreover, since a blocksize of x requires x^2 timesteps to pass one information of cycle, smaller blocksizes benefit from a lesser requirement of timesteps per cycle. The results on different block sizes with timesteps required in parenthesis are shown in Fig. 4. We empirically find that DCT-2 is unable to converge, and that overlapped version of DCT-4 with a stride of 2 outperforms all other cases. Hence, we utilize this scheme for all further experiments.

Number of Cycles for Information Propagation. The next design parameter is the number of timesteps per forward pass. The performance of DCT-SNN trained with different timesteps is shown in Fig. 5. In the scheme DCT-

Table 1: Accuracy of VGG5 on CIFAR-10, with DCT block size = 4×4

Transform Matrix	Accuracy (%)
Random	64.7
Unranked Orthonormal	83.3
PCA	83.8
DCT	83.5

Table 2: Accuracy(%), with timesteps indicated in parenthesis. -p and -d represent training with pixels and DCT coefficients, respectively

Configuration	VGG9 CIFAR-10	VGG11 CIFAR-100	VGG13 TinyImageNet
ANN-p	91.3	69.7	56.9
ANN-d	90.4	66.4	45.5[a] 53.1[b]
SNN-p	90.1 (175) 88.9 (100)	67.8 (125)	53 (250)
SNN-d	88.2 (100)	65.1 (125)	44.6 (250)
DCT-SNN	89.94 (48)	68.3 (48)	52.43 (125) 51.45 (48)

[a] ANN without batchnorm and maxpool to facilitate conversion
[b] ANN with batchnorm and maxpool

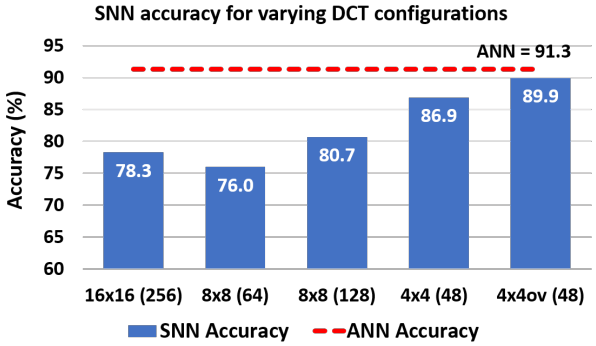

Figure 4: Accuracy(%) for VGG9 on CIFAR10 with varying DCT blocksize (timesteps)

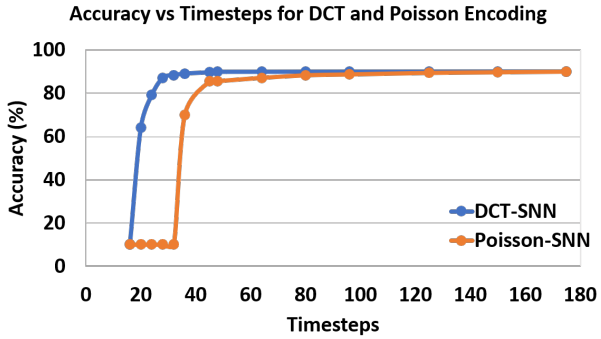

Figure 5: Accuracy(%) for VGG9 on CIFAR10 with varying timesteps

4, one full cycle amounts to 16 timesteps. The network converges to 89.94% accuracy with 48 timesteps and 88.41% accuracy with just 32 timesteps. Since performance saturates after 3 cycles (48 timesteps), we fixed 48 as the number of timesteps to train DCT-SNN on CIFAR-10 and CIFAR-100. However, for deeper networks and larger datasets, larger timesteps might yield further improvements, as shown in Table 2 for TinyImagenet with VGG13. Notably, the accuracy of Poisson-encoded networks drops severely below 45 timesteps (Fig. 5), whereas DCT-SNN suffers a minimal drop even with 28 timesteps. In particular, Poisson does not converge under 32 timesteps, whereas we achieve less than 2% accuracy drop at 32 timesteps.

Results on CIFAR and TinyImageNet. The experimental results using the proposed scheme for CIFAR and TinyImageNet datasets are shown in Table 2. DCT-SNN performs comparably to SNNs trained with Poisson-encoded pixels, but requires lesser than half the timesteps. Additionally, DCT-SNN outperforms SNNs trained on Poisson-encoded DCT coefficients, presumably due to the reconstruction of pixels over time. To demonstrate the scalability of DCT-SNN, we apply it to TinyImagenet. While SNNs with Poisson-encoded pixels require $\sim$ 250 timesteps to converge, our method achieves comparable accuracy in 125 timesteps. Allowing for a 1% drop in accuracy, our method converges with even 48 timesteps.

Performance Comparison. We compare our performance with reported results for different state-of-the-art SNNs in Table 3. DCT-SNN performs better than or comparably to the reported accuracy of the these methods, while achieving lower inference latency. Wu *et al.* [39] report CIFAR10 results with 30 timesteps for a shallow network with 2 convolutional and 2 fully-connected layers with 50.7% accuracy. We implement the same net with DCT-SNN and achieve 68.1% accuracy with 28 timesteps. Next, we compare with methods that expose analog pixel intensities directly to the first convolutional layer, instead of spikes. In a subsequent work, Wu *et al.* [40] achieve 90.53% accuracy on CIFAR-10 using just 12 timesteps on a network with 5 convolutional and 2 fully connected layers. However, in [40], after each conv layer, the binary activations go through a channel-wise normalization which makes the binary activations essentially analog. We believe that the analog computation at each layer makes this network closer to ANNs than SNNs, resulting in the significant reduction in timesteps,

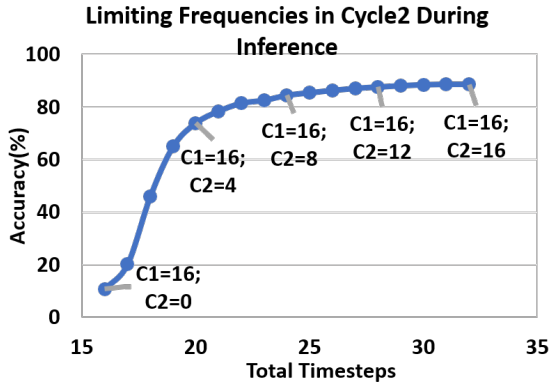

Figure 6: Accuracy-latency tradeoff during inference; VGG9 trained on CIFAR-10 with all 16 frequencies for 48 timesteps. During inference, cycle1 uses all 16 frequencies, cycle2 uses limited, ordered frequencies.

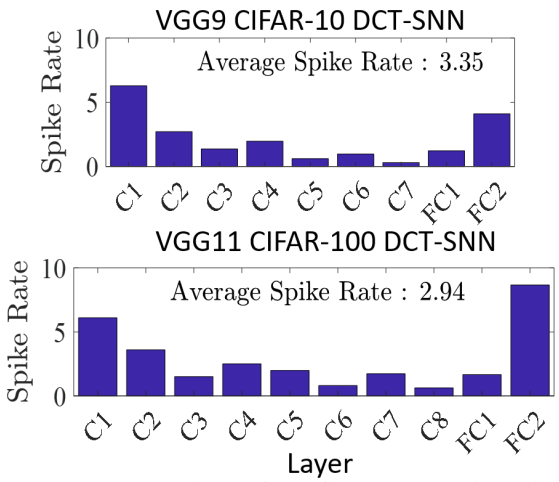

Figure 7: DCT-SNN layerwise spike rate. C and FC denote Conv and Fully Connected layers, respectively.

especially since their network is shown to converge to a good accuracy in a single timestep.

We also compare with works that train ANNs on DCT coefficients. Authors of [11] report ANNs with 72.5% and 38.5% accuracy on CIFAR-10 and CIFAR-100, respectively and authors of [35] report 86.35% accuracy on CIFAR-10. DCT-SNN reaches upto $\sim 90\%$ accuracy on CIFAR-10, as seen in Table 2, when DCT is performed on blocks of size 4×4 with an overlap of 2, unlike the standard JPEG scheme of 8×8 with no overlap. However, DCT-SNN does not train in the frequency domain unlike [11], since after passing the modulated bases through the network, an equivalent of the input image in the pixel domain is passed through the SNN. To verify that our method is trainable via backpropagation from scratch, we trained a VGG9 SNN from scratch for CIFAR-10, which gives 84.9% accuracy with 48 timesteps. A more detailed comparison with other encoding schemes is shown in the supplementary section 5.

Accuracy-Latency Trade-off. The ranking of bases in our scheme allows us to drop the least significant components. In Fig. 6, we show the effect the ranking of bases has on accuracy by performing inference on VGG9 DCT-SNN trained on CIFAR-10 for 48 timesteps using all 16 frequencies. A minimum of 16 timesteps (1 cycle) are required for spike propagation to the deeper layers, and hence any configuration with timesteps lesser than 16 cannot infer correctly. We provide 2 cycles of inputs on the test data. The first cycle uses all 16 components, and the next adds successively higher frequencies. Due to the fact that the bulk of the information is contained in the earlier timesteps, we are able to get good accuracy (73.9% out of 88.6%) with just the first 4 bases. Successive components add more refined information, and therefore the accuracy saturates, as evident from Fig. 6. To the best of our knowledge, this is the first work that demonstrates a principled trade-off between inference accuracy and latency on a trained network. Results for networks trained with limited frequencies are shown in the supplementary section 3. The effect of changing the order of inputting frequencies is shown in the supplementary section 6.

Computational Efficiency. The floating-point (FP) MAC operations in ANN are replaced by FP additions in SNN. The cost of a MAC operation ($4.6pJ$) is $5.1\times$ compared to an addition ($0.9pJ$) [13] in 45nm CMOS technology. The expressions representing the computational cost in the form of operations per layer in an ANN, $\#\text{ANN}_{\text{ops}}$, are given in the supplementary section 4. The number of operations per layer in an equivalent DCT-SNN are related to $\#\text{ANN}_{\text{ops}}$ by the layer's spike-rate.

$$\#\text{DCT-SNN}_{\text{ops, L}} = \text{spike rate}_{\text{L}} \times \#\text{ANN}_{\text{ops, L}}, \quad (3)$$

where $\text{spike rate}_{\text{L}}$ is the average number of spikes per neuron per image over all timesteps in layer L. The layerwise spike rates for CIFAR-10 and CIFAR-100 using DCT-SNN are shown in Fig. 7. The overall average spike rate across all layers for both cases is well below 5.1 (relative cost of MAC to addition), indicating the energy benefits of DCT-SNN over the corresponding ANN. For DCT-SNN, the additional cost of 2 full precision matrix multiplications in the forward and reverse preprocessing transforms are denoted as $\text{Encoder}_{\text{ops}}$. This computation is needed for only one cycle (16 timesteps), since the other 2 cycles just repeat the same bases and coefficients. The overhead is negligible when compared to the number of operations over all the layers across all timesteps.

We compute the energy benefits of DCT-SNN over ANN, $\alpha = \frac{\text{E}_{\text{ANN}}}{\text{E}_{\text{DCT-SNN}}}$ as,

$$\alpha = \frac{\sum_L \#\text{ANN}_{\text{ops,L}} * 4.6}{\#\text{Encoder}_{\text{ops}} * 4.6 + \sum_L \#\text{DCT-SNN}_{\text{ops,L}} * 0.9} \quad (4)$$

Table 3: Comparison of DCT-SNN to other reported results. SGB denotes Surrogate-Gradient Based backprop, Hybrid denotes pretrained ANN followed by SNN fine-tuning, TTFS denotes Time-To-First-Spike scheme, TL denotes tandem learning and (xC, yL) denotes an architecture with x Conv layers and y Linear layers.

Reference	Dataset	Training	Architecture	Accuracy(%)	Timesteps
Hunsberger and Eliasmith [16]	CIFAR10	Conversion	2C, 2L	82.95	6000
Cao et al. [4]	CIFAR10	Conversion	3C, 2L	77.43	400
Sengupta et al. [33]	CIFAR10	Conversion	VGG16	91.55	2500
Lee et al. [21]	CIFAR10	SGB	VGG9	90.45	100
Rueckauer et al. [32]	CIFAR10	Conversion	4C, 2L	90.85	400
Rathi et al. [30]	CIFAR10	Hybrid	VGG9	90.50	100
Park et al. [28]	CIFAR10	TTFS	VGG16	91.40	680
Park et al. [27]	CIFAR10	Burst-coding	VGG16	91.40	1125
Kim et al. [18]	CIFAR10	Phase-coding	VGG16	91.20	1500
Wu et al. [39]	CIFAR10	SGB	2C, 2L	50.70	30
Wu et al. [40]	CIFAR10	SGB	5C, 2L	90.53	12
Wu et al. [38]	CIFAR10	TL(LIF)	5C, 2L	89.04	8
This work	CIFAR10	DCT-SNN	VGG9	89.94	48
Lu and Sengupta [24]	CIFAR100	Conversion	VGG15	63.20	62
Rathi et al. [30]	CIFAR100	Hybrid	VGG11	67.90	125
Park et al. [28]	CIFAR100	TTFS	VGG16	68.80	680
Park et al. [27]	CIFAR100	Burst-coding	VGG16	68.77	3100
Kim et al. [18]	CIFAR100	Phase-coding	VGG16	68.60	8950
This work	CIFAR100	DCT-SNN	VGG11	68.30	48
Sengupta et al. [33]	TinyImagenet	Conversion	VGG16	48.60	2500
Kundu et al. [20]	TinyImagenet	Hybrid	VGG16	51.92	150
This work	TinyImagenet	DCT-SNN	VGG13	52.43	125

The obtained values of α are 1.52 and 1.74 for VGG9-CIFAR10 and VGG11-CIFAR100, respectively, showing that DCT-SNN improves energy efficiency over its ANN counterpart. Similar to [28], the cost of memory access has not been considered in this evaluation, since it depends on the hardware architecture and system configurations.

5. Conclusion

Bio-plausible SNNs derive efficiency from the sparsity of spikes per timestep and event-driven computation, but suffer from high inference latency. The most widely used Poisson based rate encoding scheme does not encode meaningful information into the temporal axis of SNNs, and requires a large number of timesteps for inference. To address this, we propose a new encoding scheme that can be used to distribute spatial pixel information over timesteps in an ordered fashion. It utilizes a linear transform in the form of an invertible matrix, with columns that serve as basis of representation distribution. The input pixels are reconstructed over time by summing these bases modulated by the intermediate coefficients. At each step, we feed in the modulated bases to an integrate-and-fire neuron at the input layer. As we cycle through all bases over timesteps, the neurons cumulatively receive the total pixel values. The ideal properties of the bases are orthonormality to avoid interference, and ordering by contribution to pixel reconstruction. DCT meets these conditions, while also being dataset-agnostic. We get best performance with 2-D DCT on 4×4 blocks of the input, resulting in 16 basis frequencies. We show that DCT-SNN trained by passing these 16 bases through the SNN cyclically a few times achieves comparable accuracy to their ANN counterparts, with less than half the number of inference timesteps compared to other state-of-the-art SNNs. Additionally, ranking these bases allows us to drop the least important bases (and therefore, timesteps). This principled trade-off between inference accuracy and latency is a promising direction for deploying SNNs on edge devices.

6. Acknowledgements

This work was supported in part by the Center for Brain Inspired Computing (C-BRIC), one of the six centers in JUMP, a Semiconductor Research Corporation (SRC) program sponsored by DARPA, by the SRC, the National Science Foundation, Intel Corporation, the DoD Vannevar Bush Fellowship, and by the U.S. Army Research Laboratory and the U.K. Ministry of Defence under Agreement Number W911NF-16-3-0001.

References

[1] Nasir Ahmed, T_ Natarajan, and Kamisetty R Rao. Discrete cosine transform. *IEEE transactions on Computers*, 100(1): 90–93, 1974.

[2] F. Akopyan, J. Sawada, A. Cassidy, R. Alvarez-Icaza, J. Arthur, P. Merolla, N. Imam, Y. Nakamura, P. Datta, G. Nam, B. Taba, M. Beakes, B. Brezzo, J. B. Kuang, R. Manohar, W. P. Risk, B. Jackson, and D. S. Modha. Truenorth: Design and tool flow of a 65 mw 1 million neuron programmable neurosynaptic chip. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 34(10):1537–1557, 2015. doi: 10.1109/TCAD.2015.2474396.

[3] Peter Blouw, Xuan Choo, Eric Hunsberger, and Chris Eliasmith. Benchmarking keyword spotting efficiency on neuromorphic hardware. In *Proceedings of the 7th Annual Neuro-inspired Computational Elements Workshop*, pages 1–8, 2019.

[4] Yongqiang Cao, Yang Chen, and Deepak Khosla. Spiking deep convolutional neural networks for energy-efficient object recognition. *International Journal of Computer Vision*, 113(1):54–66, 2015.

[5] Iulia M Comsa, Thomas Fischbacher, Krzysztof Potempa, Andrea Gesmundo, Luca Versari, and Jyrki Alakuijala. Temporal coding in spiking neural networks with alpha synaptic function. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8529–8533. IEEE, 2020.

[6] Mike Davies, Narayan Srinivasa, Tsung-Han Lin, Gautham Chinya, Yongqiang Cao, Sri Harsha Choday, Georgios Dimou, Prasad Joshi, Nabil Imam, Shweta Jain, et al. Loihi: A neuromorphic manycore processor with on-chip learning. *IEEE Micro*, 38(1):82–99, 2018.

[7] Li Deng and Yang Liu. *Deep learning in natural language processing*. Springer, 2018.

[8] Peter U Diehl and Matthew Cook. Unsupervised learning of digit recognition using spike-timing-dependent plasticity. *Frontiers in computational neuroscience*, 9:99, 2015.

[9] Peter U Diehl, Daniel Neil, Jonathan Binas, Matthew Cook, Shih-Chii Liu, and Michael Pfeiffer. Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing. In *2015 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. ieee, 2015.

[10] R Dony et al. Karhunen-loeve transform. *The transform and data compression handbook*, 1:1–34, 2001.

[11] Max Ehrlich and Larry S Davis. Deep residual learning in the jpeg transform domain. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3484–3493, 2019.

[12] Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal processing magazine*, 29(6):82–97, 2012.

[13] Mark Horowitz. 1.1 computing's energy problem (and what we can do about it). In *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, pages 10–14. IEEE, 2014.

[14] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[15] Dongsung Huh and Terrence J Sejnowski. Gradient descent for spiking neural networks. In *Advances in Neural Information Processing Systems*, pages 1433–1443, 2018.

[16] Eric Hunsberger and Chris Eliasmith. Spiking deep networks with lif neurons. *arXiv preprint arXiv:1510.08829*, 2015.

[17] Xiping Ju, Biao Fang, Rui Yan, Xiaoliang Xu, and Huajin Tang. An fpga implementation of deep spiking neural networks for low-power and fast classification. *Neural Computation*, 32(1):182–204, 2020.

[18] Jaehyun Kim, Heesu Kim, Subin Huh, Jinho Lee, and Kiyoung Choi. Deep neural networks with weighted spikes. *Neurocomputing*, 311:373–386, 2018.

[19] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[20] Souvik Kundu, Gourav Datta, Massoud Pedram, and Peter A Beerel. Spike-thrift: Towards energy-efficient deep spiking neural networks by limiting spiking activity via attention-guided compression. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 3953–3962, 2021.

[21] Chankyu Lee, Syed Shakib Sarwar, Priyadarshini Panda, Gopalakrishnan Srinivasan, and Kaushik Roy. Enabling spike-based backpropagation for training deep neural network architectures. *Frontiers in Neuroscience*, 14, 2020.

[22] Da Li, Xinbo Chen, Michela Becchi, and Ziliang Zong. Evaluating the energy efficiency of deep convolutional neural networks on cpus and gpus. In *2016 IEEE international conferences on big data and cloud computing (BDCloud), social computing and networking (SocialCom), sustainable computing and communications (SustainCom)(BDCloud-SocialCom-SustainCom)*, pages 477–484. IEEE, 2016.

[23] Hongmin Li, Hanchao Liu, Xiangyang Ji, Guoqi Li, and Luping Shi. Cifar10-dvs: an event-stream dataset for object classification. *Frontiers in neuroscience*, 11:309, 2017.

[24] Sen Lu and Abhronil Sengupta. Exploring the connection between binary and spiking neural networks. *arXiv preprint arXiv:2002.10064*, 2020.

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

[26] Emre O Neftci, Hesham Mostafa, and Friedemann Zenke. Surrogate gradient learning in spiking neural networks. *IEEE Signal Processing Magazine*, 36:61–63, 2019.

[27] Seongsik Park, Seijoon Kim, Hyeokjun Choe, and Sungroh Yoon. Fast and efficient information transmission with burst spikes in deep spiking neural networks. In *2019 56th ACM/IEEE Design Automation Conference (DAC)*, pages 1–6. IEEE, 2019.

[28] Seongsik Park, Seijoon Kim, Byunggook Na, and Sungroh Yoon. T2fsnn: Deep spiking neural networks with time-to-first-spike coding. *arXiv preprint arXiv:2003.11741*, 2020.

[29] Bulla Rajesh, Mohammed Javed, Shubham Srivastava, et al. Dct-compcnn: A novel image classification network using jpeg compressed dct coefficients. In *2019 IEEE Conference on Information and Communication Technology*, pages 1–6. IEEE, 2019.

[30] Nitin Rathi, Gopalakrishnan Srinivasan, Priyadarshini Panda, and Kaushik Roy. Enabling deep spiking neural networks with hybrid conversion and spike timing dependent backpropagation. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=B1xSperKvH`.

[31] Kaushik Roy, Akhilesh Jaiswal, and Priyadarshini Panda. Towards spike-based machine intelligence with neuromorphic computing. *Nature*, 575(7784):607–617, 2019.

[32] Bodo Rueckauer, Iulia-Alexandra Lungu, Yuhuang Hu, Michael Pfeiffer, and Shih-Chii Liu. Conversion of continuous-valued deep networks to efficient event-driven networks for image classification. *Frontiers in neuroscience*, 11:682, 2017.

[33] Abhronil Sengupta, Yuting Ye, Robert Wang, Chiao Liu, and Kaushik Roy. Going deeper in spiking neural networks: Vgg and residual architectures. *Frontiers in neuroscience*, 13:95, 2019.

[34] Sumit Bam Shrestha and Garrick Orchard. Slayer: Spike layer error reassignment in time. In *Advances in Neural Information Processing Systems*, pages 1412–1421, 2018.

[35] Matej Ulicny and Rozenn Dahyot. On using cnn with dct based image data. In *Proceedings of the 19th Irish Machine Vision and Image Processing conference IMVIP*, 2017.

[36] G. K. Wallace. The jpeg still picture compression standard. *IEEE Transactions on Consumer Electronics*, 38(1):xviii–xxxiv, 1992.

[37] Travis Williams and Robert Li. Advanced image classification using wavelets and convolutional neural networks. In *2016 15th IEEE international conference on machine learning and applications (ICMLA)*, pages 233–239. IEEE, 2016.

[38] Jibin Wu, Yansong Chua, Malu Zhang, Guoqi Li, Haizhou Li, and Kay Chen Tan. A tandem learning rule for efficient and rapid inference on deep spiking neural networks. *arXiv*, pages arXiv–1907, 2019.

[39] Yujie Wu, Lei Deng, Guoqi Li, Jun Zhu, and Luping Shi. Spatio-temporal backpropagation for training high-performance spiking neural networks. *Frontiers in neuroscience*, 12:331, 2018.

[40] Yujie Wu, Lei Deng, Guoqi Li, Jun Zhu, Yuan Xie, and Luping Shi. Direct training for spiking neural networks: Faster, larger, better. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 1311–1318, 2019.

[41] Kai Xu, Minghai Qin, Fei Sun, Yuhao Wang, Yen-Kuang Chen, and Fengbo Ren. Learning in the frequency domain. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1740–1749, 2020.

[42] Qi Xu, Jianxin Peng, Jiangrong Shen, Huajin Tang, and Gang Pan. Deep covdensesnn: A hierarchical event-driven dynamic framework with spiking neurons in noisy environment. *Neural Networks*, 121:512–519, 2020.

[43] Friedemann Zenke and Surya Ganguli. Superspike: Supervised learning in multilayer spiking neural networks. *Neural computation*, 30(6):1514–1541, 2018.

[44] Lei Zhang, Shengyuan Zhou, Tian Zhi, Zidong Du, and Yunji Chen. Tdsnn: From deep neural networks to deep spike neural networks with temporal-coding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 1319–1326, 2019.