# Oscillatory Fourier Neural Network: A Compact and Efficient Architecture for Sequential Processing

**Bing Han, Cheng Wang, and Kaushik Roy**

School of Electrical and Computer Engineering, Purdue University
610 Purdue Mall, West Lafayette, IN 47907
han183,wang4700,kaushik@purdue.edu

## Abstract

Tremendous progress has been made in sequential processing with the recent advances in recurrent neural networks. However, recurrent architectures face the challenge of exploding/vanishing gradients during training, and require significant computational resources to execute back-propagation through time. Moreover, large models are typically needed for executing complex sequential tasks. To address these challenges, we propose a novel neuron model that has cosine activation with a time varying component for sequential processing. The proposed neuron provides an efficient building block for projecting sequential inputs into spectral domain, which helps to retain long-term dependencies with minimal extra model parameters and computation. A new type of recurrent network architecture, named Oscillatory Fourier Neural Network, based on the proposed neuron is presented and applied to various types of sequential tasks. We demonstrate that recurrent neural network with the proposed neuron model is mathematically equivalent to a simplified form of discrete Fourier transform applied onto periodical activation. In particular, the computationally intensive back-propagation through time in training is eliminated, leading to faster training while achieving the state of the art inference accuracy in a diverse group of sequential tasks. For instance, applying the proposed model to sentiment analysis on IMDB review dataset reaches 89.4% test accuracy within 5 epochs, accompanied by over 35x reduction in the model size compared to LSTM. The proposed novel RNN architecture is well poised for intelligent sequential processing in resource constrained hardware.

## Introduction

Recently, artificial neural networks, especially various types of recurrent neural networks (RNN) have demonstrated significant success in sequential processing tasks such as natural language processing (NLP)(Yin et al. 2017). However, complex sequential processing tasks desires increasingly large RNN models, and the training of RNNs encounters the notorious challenge of exploding or vanishing gradients when information across long time need to be preserved and processed. In particular, back-propagation through time (BPTT) applied on weight matrices associated with hidden states such as $W_{hh}$ need to unroll gradient calculations throughout large number of time steps. BPTT is not only prone to exponentially decaying or growing of the propagated values but also consuming significant computation resources over the recurrent steps. Although mitigating architectures, such as long-short term memory (LSTM) and gated recurrent units (GRU) have been proposed, these modified RNN models typically contain more training parameters, consumes longer time to train, and may still suffer the loss of gradients in presence of very long input sequences. On the other hand, it has been observed that extracting and understanding meaningful patterns at different time scales can be important for sequential processing tasks such as text sentiment analysis or body activity recognition. For instances, patterns in movie reviews could emerge at the word level, sentence/clause level, and paragraph level that altogether determine the outcome of the final sentiment of a review. In order to learn the features at various scales from sequential data, incorporating spectral analysis such as Fourier Transform in RNNs have received growing attention (Lee-Thorp et al. 2021), (Tamkin, Jurafsky, and Goodman 2020). Typically, the input of temporal series can be fed into certain type of spectral processing such as Fourier transform, where frequency-domain outcome can be obtained and further analyzed by the subsequent neural network blocks.

In this work, we propose a new type of neuron with time varying cosine activation (termed TV-Cosine neuron), and construct an RNN architecture, named oscillatory Fourier neural network (O-FNN), for efficient learning for sequential tasks. Intuitively, the proposed neuron model projects sequential input data into a *phase* of oscillating neuron, in contrast to conventional neurons such as ReLU or tanh that modulate the *magnitude* of activation depending on the input. Moreover, each neuron in the proposed architecture is a cosine activation applied onto a superposition of input signal and time dependent rotating term of certain frequency. The hidden state vector of the O-FNN accumulates the outcome of cosine activation across all time steps before being fed into the final layer of the network. Following this approach, no backpropagation through time is needed during the backward pass of training, and both forward and backward pass can process the sequential data in a fully parallel manner. We will show that after accumulating the cosine activation through time, the hidden states will equivalently represent a modified discrete Fourier transform of sine and cosine neu-

ral activation. The major contribution is summarized as follows:

- We propose **a new type of neuron with time varying cosine activation** for sequential processing. We demonstrate that RNN with such time-varying activation is mathematically equivalent to a simplified form of discrete Fourier transform. Due to the usage of cosine activation, the transform of data into frequency domain is more computationally efficient compared to applying Fourier transform on ReLU or sigmoidal neurons.
- We propose **a new type of RNN architecture, O-FNN**, that is fully parallelizable in both forward and backward passes. In particular, since the backward propagation is not unrolled in time, the issue of exponential explosion or decay of the gradient values across long range of time steps with conventional activation functions is eliminated.
- We show that O-FNN architecture is capable of achieving better performance on a plethora of sequential processing tasks with more compact models and high computational efficiency in comparison with regular RNN/LSTM. The characteristics of smaller memory footprint and faster training make the O-FNN especially suitable for deployment in resource-constrained hardware such as IOT and battery-powered edge devices.

## Related Work

While periodic activation functions have been proposed as early as 1980s (Lapedes and Farber 1987; McCaughan 1997), learning with such periodic activation has received limited attention from the research community (Sopena, Romero, and Alquezar 1999; Wong, Leung, and Chang 2002). In (Sopena, Romero, and Alquezar 1999), the authors show that with proper range of initial weight values, a multi-layer perceptron using sinusoidal neurons in the form of $\sin(WX + b)$ improves accuracy and trains faster compared to its sigmoidal counterpart on some small datasets. Recently, sinusoidal activation function in implicit neural representations demonstrated improved capability of processing complex spatial-temporal signals and their derivative (Sitzmann et al. 2020). Authors in (Ramachandran, Zoph, and Le 2017) conduct network architecture search and identify neuron with partial sinusoidal behavior as one of the top candidates for activation functions for typical image classification tasks. Note that training networks with periodical activation may be challenging due to the possibility of having numerous local minima in the landscape of loss function (Sopena, Romero, and Alquezar 1999; Parascandolo, Huttunen, and Virtanen 2017).

Regarding applying periodical activation in RNNs, (Sopena and Alquezar 1994) reports improvement in learning when sine instead of sigmoid activation function is used in the last fully connected layer of a simple RNN trained for a next-character prediction task. The authors of (Parascandolo, Huttunen, and Virtanen 2017) also observe that the periodical activation can be beneficial for training RNNs and LSTMs on some algorithmic tasks, showing faster learning and possibly better accuracy. Moreover, oscillatory neuron dynamics is exploited in implementing coupled oscillator recurrent neural networks in (Rusch and Mishra 2021), demonstrating great potential of oscillating neurons for processing complex sequential data.

Usage of periodical activation is related to Fourier transforms which also involve extraction of information and features in the frequency domains. In particular, Fourier transforms, especially discrete Fourier Transforms (DFT), have been adopted successfully for RNNs to execute sequential processing (Koplon and Sontag 1997) or make predictions (Zhang and Chan 2000). Fourier transforms are typically done on input signals to facilitate the learning of spectral features. More recently, the idea of leveraging Fourier transforms for extracting spectral information is also explored in Natural Language Processing (NLP) tasks. Authors in (Zhang et al. 2018) proposes using Fourier basis to summarize the statistics of hidden states through past time steps in RNNs. (Tamkin, Jurafsky, and Goodman 2020) applies spectral filters similar to DFT to the activations of individual neurons in BERT (Devlin et al. 2018) language model, aiming for extracting information changing at different time scales in texts.

## Proposed Approach

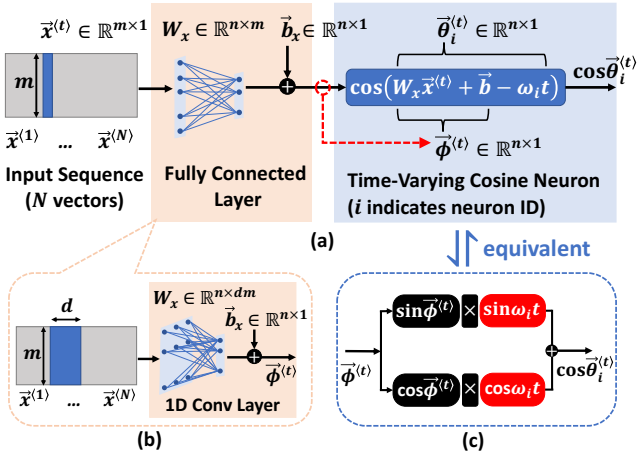


Figure 1: (a) Time-Varying Cosine neuron with fully-connected input layer. (b) One dimensional convolution layer can also be used as input layer. (c) TV-Cosine neuron is mathematically equivalent to projecting sine and cosine activations respectively onto $\sin\omega_i t$ and $\cos\omega_i t$ channels of discrete frequencies.

### Time-Varying Cosine (TV-Cosine) Neuron for Sequential Processing

We propose temporal artificial neuron named Time-Varying Cosine (TV-Cosine) neuron. As shown in Fig.1(a), the proposed neuron has a cosine activation, which has an input $\theta_i^{\langle t\rangle}$ obtained from a superposition of a fully connected layer's output $\phi^{\langle t\rangle}$ and a time-varying phase modulating term $\omega_i t$. Mathematically, each TV-Cosine neuron is equivalent to a sine neuron and a cosine neuron respectively, projected to a

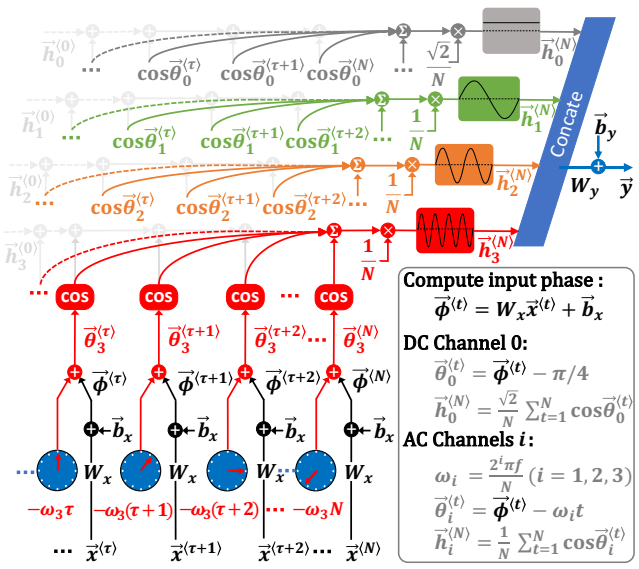


Figure 2: O-FNN forward propagation (full parallelism).

sine and a cosine oscillating functions, as shown in Fig.1(c). Each TV-Cosine neuron has a specific oscillating frequency $\omega_i$, where the subscript $i$ indicates the frequency channel. The frequencies for different channels $\omega_i$ are hyper parameters of the proposed model. In sequential processing tasks, the fully connected input layer in Fig.1(a) can also be replaced with one dimensional convolutional layer as shown in Fig.1(b), in which, $d$ consecutive input vectors are fed to the TV-Cosine neuron at each time-step. We observed that using 1D convolutional layer with a small slicing window ($d = 3$) and small stride ($s = 1$) slightly improves the accuracy, at the cost of slightly increasing the number of weight parameters.

## Forward pass in O-FNN

We propose O-FNN architecture consisting of TV-Cosine neurons for efficient learning of sequential tasks. A simple example of O-FNN containing four TV-Cosine neurons with distinctive channel frequencies is shown in Fig.2. The forward flow of O-FNN is unrolled in time as shown along the horizontal direction. Each channel traces and accumulates outputs of one TV-Cosine neuron across all time-steps. The final hidden states of all neurons are concatenated before further processing by a fully connected read-out layer. As discussed earlier, each TV-Cosine neuron equivalently projects activations of sine and cosine activations onto oscillating sine and cosine channels with various frequencies. Hence, the forward flow of O-FNN in Fig.2 is mathematically equivalent to a special form of Discrete Fourier Transform (DFT).

TV-Cosine neurons in O-FNN can be categorized into one DC neuron and a few AC neurons depending on the neuron input. For example, at each time-step $t$, an input phase $\vec{\phi}^{\langle t \rangle}$ is computed as shown by Eq.1. Input to DC neuron equals $\vec{\phi}^{\langle t \rangle}$ subtracts a constant value $\frac{1}{4}\pi$ as shown by Eq.2. Inputs to AC neurons equal $\vec{\phi}^{\langle t \rangle}$ subtracts a time-driven phase modulating term $\omega_i t$ as shown by Eq.3. The angular velocity $\omega_i$ of different AC neurons are computed by Eq.4, in which $f$ is a hyper-parameter used to determine the running speed of the clock in each AC neuron.

$$\vec{\phi}^{\langle t \rangle} = W_x \vec{x}^{\langle t \rangle} + \vec{b}_x \quad (t = 1, ..., N) \tag{1}$$

$$\vec{\theta}_0^{\langle t \rangle} = \vec{\phi}^{\langle t \rangle} - \frac{1}{4}\pi \quad (t = 1, ..., N) \tag{2}$$

$$\vec{\theta}_i^{\langle t \rangle} = \vec{\phi}^{\langle t \rangle} - \omega_i t \quad (t = 1, ..., N) \tag{3}$$

$$\omega_i = \frac{2^i \pi f}{N} \quad (i = 1, 2, 3) \tag{4}$$

The forward flow of O-FNN is a fully parallelizable architecture as shown in Fig.2. Concretely, neuronal activation of all time steps can be computed in parallel if the whole input sequence is already known. The process of forward propagation can be explained as follows. First, all hidden states are initialized to zero. Next, input at each time step can be fed into the parallelizable architecture, and the cosine activation from each time step can be computed following Eq.1 - 4. Subsequently, the final states of DC and AC neurons in O-FNN are computed by summing and averaging of cosine neurons' outputs over all time steps, following Eq.5(a) and Eq.6(a). The summation for the AC-channel neurons outputs is equivalent to combining a sine neuron's DST and cosine neuron's DCT terms. Finally, the resulting hidden coefficients are concatenated and fed to the final read-out layer as shown by Eqs.7 and 8. Note that, by subtracting a constant phase $\frac{1}{4}\pi$ in Eq.2 (and multiplying a coefficient of $\sqrt{2}$), we manage to obtain a unified formal expression of hidden states of both DC and AC channels.

$$\vec{h}_0^{\langle N \rangle} = \frac{\sqrt{2}}{N}\left(\vec{h}_0^{\langle 0 \rangle} + \sum_{t=1}^{N} \cos \vec{\theta}_0^{\langle t \rangle}\right) \tag{5a}$$

$$= \frac{1}{N}\sum_{t=1}^{N}\left(\sin \vec{\phi}^{\langle t \rangle} + \cos \vec{\phi}^{\langle t \rangle}\right) \tag{5b}$$

$$\vec{h}_i^{\langle N \rangle} = \frac{1}{N}\left(\vec{h}_i^{\langle 0 \rangle} + \sum_{t=1}^{N} \cos \vec{\theta}_i^{\langle t \rangle}\right) \tag{6a}$$

$$= \frac{1}{N}\sum_{t=1}^{N}\left[\sin \vec{\phi}^{\langle t \rangle} \sin(\omega_i t) + \cos \vec{\phi}^{\langle t \rangle} \cos(\omega_i t)\right] \tag{6b}$$

(in which $i = 1, 2, 3$ for AC channels)

$$\vec{h}_{cat}^{\langle N \rangle} = \text{concate}\left[\vec{h}_0^{\langle N \rangle}; \vec{h}_1^{\langle N \rangle}; \vec{h}_2^{\langle N \rangle}; \vec{h}_3^{\langle N \rangle}\right] \tag{7}$$

(concatenate along dimension 1)

$$\vec{y} = W_y \vec{h}_{cat}^{\langle N \rangle} + \vec{b}_y \tag{8}$$

The compact model size and high computational efficiency stem from the fact that O-FNN does not require the square matrix $W_{hh}$ and the associated matrix multiplication

operations $\vec{h}^{\langle t\rangle} = W_{hh}\left[\vec{h}^{\langle t-1\rangle} + f(W_x\vec{x}^{\langle t\rangle} + \vec{b}_x)\right]$ that occur at every time-step in regular RNNs. The superior accuracy of O-FNN can be attributed to operating in the frequency domain. The DC and low frequency AC channels efficiently capture long time dependencies from sequential data, whereas the high frequency AC channels provide short-term memory. Compared to approaches that perform spectrum analysis on raw input data or ReLU/Sigmoid neurons, our approach eliminates multiplication operations required to perform the transformation from time to frequency domain. For example, we can perform a regular DFT on neuron with activation function $f$ (ReLU or Sigmoid). However, as described by Eqs.9 (a) and (b), at least one multiplication is required at each time-step to complete the transformation. On the other hand, only addition operations are used in our approach to perform the special form of DFT in Fig.2 using TV-Cosine neurons and O-FNN architecture in Fig.2. The required cosine computation can be implemented using a look-up table. The required extra memory space $O(N)$ ($N$ is sequence length) is negligible compared to the trainable parameters.

$$\vec{h}_i^{\langle N\rangle} = \frac{1}{N}\sum_{t=1}^{N}\left[f\left(\vec{\phi}^{\langle t\rangle}\right)\sin(\omega_i t) + f\left(\vec{\phi}^{\langle t\rangle}\right)\cos(\omega_i t)\right] \tag{9a}$$

$$= \frac{\sqrt{2}}{N}\sum_{t=1}^{N}\left[f\left(\vec{\phi}^{\langle t\rangle}\right)\cos\left(\omega_i t - \frac{\pi}{4}\right)\right] \tag{9b}$$

(in which $i = 1, 2, 3$ for AC channels)

The number of AC neurons (channels) used in O-FNN is a hyper-parameter. According to our simulation results, using only two to three AC neurons are sufficient to obtain the SOTA accuracies across various datasets we tested. Increasing the number of AC neurons beyond four causes accuracy degradation due to over-fitting. As can be seen in the result section, O-FNN requires far fewer neurons and smaller memory footprint while achieving superior accuracy than today's SOTA models.

## Backward pass in O-FNN

The back-propagation of O-FNN is shown in Fig.3. During forward propagation, the input sequence $\vec{x}^{\langle 1\rangle}, ..., \vec{x}^{\langle N\rangle}$, the concatenated final hidden states $\vec{h}_{cat}^{\langle N\rangle} \in \mathbb{R}^{4n\times 1}$, and the inputs $\vec{\theta}_i^{\langle t\rangle}$ to all TV-Cosine neurons at all time-steps have been cached and will be used in back-propagation. Gradient w.r.t. output $\frac{\partial L}{\partial \vec{y}} \in \mathbb{R}^{d\times 1}$ first propagates through the fully connected read-out layer and the parameters in this layer can be updated using Formulas 10 and 11, where $\eta$ is learning rate. Next as shown in Formula 12, the weight matrix $W_y \in \mathbb{R}^{d\times 4n}$ of the read-out layer is decomposed into four sub-matrices $W_{y[i]} \in \mathbb{R}^{d\times n}$, where $(i = 0, ..., 3)$. Gradients w.r.t each one of the backward channels can be computed as $\left[W_{y[i]}\right]^T \frac{\partial L}{\partial \vec{y}}$ as shown in Fig.3. No BPTT is used in O-FNN because all gradients w.r.t. input bias and weights at all time-steps and channels can be computed in a fully parallel fashion as shown by Eqs.13 and 14, respectively. Finally,

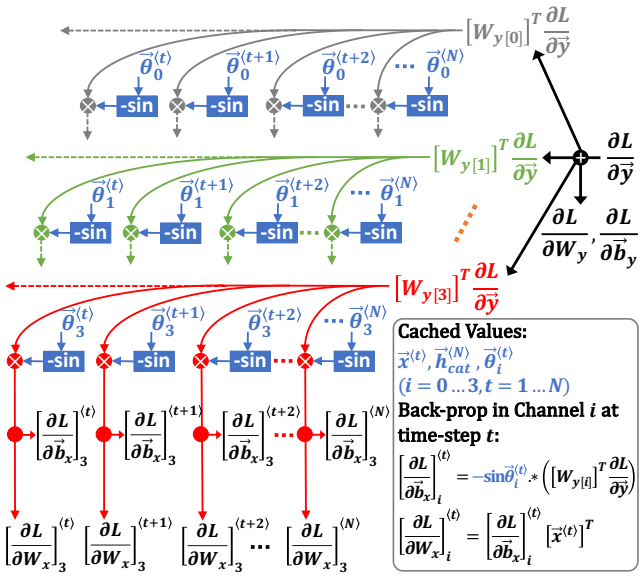


Figure 3: O-FNN backward propagation (full parallelism).

the parameters of input layer are updated using the averaged gradients across all time-steps and channels as indicated by 15 and 16.

$$\vec{b}_y \leftarrow \vec{b}_y + \eta\frac{\partial L}{\partial \vec{y}} \tag{10}$$

$$W_y \leftarrow W_y + \eta\frac{\partial L}{\partial \vec{y}}\left[\vec{h}_{cat}^{\langle N\rangle}\right]^T \tag{11}$$

$$\text{concate}\left[W_{y[0]}; W_{y[1]}; W_{y[2]}; W_{y[3]}\right] \leftarrow W_y \tag{12}$$

(concatenate along dimension 1)

$$\left[\frac{\partial L}{\partial \vec{b}_x}\right]_i^{\langle t\rangle} = -\sin\vec{\theta}_i^{\langle t\rangle} .* \left(\left[W_{y[i]}\right]^T \frac{\partial L}{\partial \vec{y}}\right) \tag{13}$$

(in which $.*$ is element-wise multiplication)

$$\left[\frac{\partial L}{\partial W_x}\right]_i^{\langle t\rangle} = \left[\frac{\partial L}{\partial \vec{b}_x}\right]_i^{\langle t\rangle}\left[\vec{x}^{\langle t\rangle}\right]^T \tag{14}$$

$$\vec{b}_x \leftarrow \vec{b}_x + \frac{\eta}{4N}\sum_{i=0}^{3}\sum_{t=1}^{N}\left[\frac{\partial L}{\partial \vec{b}_x}\right]_i^{\langle t\rangle} \tag{15}$$

$$W_x \leftarrow W_x + \frac{\eta}{4N}\sum_{i=0}^{3}\sum_{t=1}^{N}\left[\frac{\partial L}{\partial W_x}\right]_i^{\langle t\rangle} \tag{16}$$

By eliminating hidden state weight matrix $W_{hh}$ and the time consuming BPTT in the backward pass, training O-FNN can be faster with considerable improvement in energy efficiency. Moreover, the occurrence of exploding and vanishing gradients due to multiplicative errors across numerous time steps is also prevented in absence of BPTT. Unlike GRUs/LSTMs, which employ extra states and gates

to retain memory for long-term dependencies, O-FNN resorts to transforming the input sequences to the frequency domain, and focuses on training the network based on the spectral information. Since O-FNN is designed to accumulate equal contribution from all time-steps, the updates of parameters is determined by the averaged gradients, leading to an equivalent "mini-batch" effect during training. Such equivalent "mini-batch gradient descent" based training in O-FNN might be able to explain the fast convergence with O-FNN compared to other SOTA sequential models, as is demonstrated in the Results and Discussion Section.

## Results and Discussion

We will discuss the tuning of hyper-parameters for optimizing the network architecture. Specifically we investigate the impact of the base frequency $f$ and the total number of channels ($\#chs$) on the model's learning capability. Then we will present the results of applying the proposed O-FNN to a diverse group of learning tasks. We conducted experiments on an NVIDIA GeForce GTX 1080 GPU. We perform hyper-parameters tuning based on a grid search. The best performing results is based on the highest mean values of validation accuracy averaged over 10 random initialization of trainable parameters. We used an exponential decaying learning with an initial learning rate 1e-3 for all the experiments. For the IMDB task, an exponential decaying factor 0.3 is used, and 0.7 is used for other tasks.

### Hyper-parameters Tuning and Understanding

We start with optimizing hyper-parameters for the IMDB sentiment classification task. IMDB dataset contains 50K movie reviews from IMDB users. The sentiments of the reviews written by IMDB users are labeled as either positive ("1") or negative ("0") (Maas et al. 2011). The binary sentiment classification task is to differentiate a user review being positive or negative. The 50K dataset is split evenly for training and testing (25K for each). As for word embedding, we apply the pretrained 50d Glove to vectorize the vocabulary (Brochier, Guille, and Velcin 2019). As is shown in Fig.4(a), adding AC channels of various base frequency $f$ improves the learning performance compared to the case of using DC channel only, corroborating the importance of capturing AC features for learning the long sequences of texts. For IMDB dataset, it is found that base frequency of $f$ around 1.0 works well. Increasing $f$ above 2.0 hurts the performance, possibly due to the loss of low-frequency information which is critical for learning long-time dependencies. It is also observed that increasing the total number of channels from 3 to 5 further improve the test accuracy with a more stabilized learning curve versus number of epochs. Fig.4(b) summarizes the results of hyper-parameter tuning for IMDB dataset. The high efficiency of the Time-Varying Cosine neuron is clearly demonstrated, as only 3-5 channels are sufficient to extract a rich set of features in the time-domain. Adding more channels (above 5) shows no benefit while increasing the model size.

In addition to text classifications, we also evaluate the tuning of hyper-parameters for the task of image classification. We look into digit classification on permuted sequential MNIST dataset (psMNIST). The psMNIST is a modification of sequential MNIST (Le, Jaitly, and Hinton 2015) (sMNIST), where a fixed permutation is applied to the stream of individual pixels based on MNIST digits (Lecun et al. 1998). The psMNIST provides sequences of pixels with a sequence length T = 784, which is a few times longer than the sequence length of IMDB reviews (averaged at about 250 words). As is shown in Fig.4(d), we observe improvement of performance as AC channels are added into the network, similarly to the observation from the IMDB experiment. Moreover, it is found that higher base frequency is desirable to cope with the increased sequence length in psMNIST. As is further discussed in the following, the choice of optimized base frequency is dependent on the characteristics of dataset such as the sequence length and the ratio of pattern length to total sequence length. As is summarized in Fig.4(e), the best performance occurs when number of channels equals 3 with the base AC frequency $f$=2.0. The observed degradation of test accuracy for number of channels higher than 3 is due to overfitting, as a large number of neurons are connected to process single-pixel sequences.

### Results on Various Datasets

We apply the O-FNN to various sequential learning tasks, with hyper-parameters optimized for each dataset and task. In addition to IMDB and permuted sequential MNIST as discussed above, we further apply the O-FNN model to process noise padded CIFAR-10 (Chang et al. 2019) and human activity recognition (HAR-2) datasets (Anguita et al. 2012)(Kusupati et al. 2018). The results of O-FNN on the four datasets are summarized in Table 1. For each task, we compare the O-FNN with a few baseline models such as various types of LSTM ((Pandey 2020)) and GRU ((Dey and Salem 2017)) as well as some recent results from the literature (such as coRNN(Rusch and Mishra 2021)) for comparison. As is shown in Fig.4(c) and Fig.4(f), the proposed O-FNN reaches the highest test accuracy with faster convergence on both IMDB and sMNIST benchmark experiments. For IMDB, we focus the comparison with models that have a relatively small number (128) of hidden units. We observe that O-FNN achieves significantly higher accuracy with 32$\times$ fewer neurons, 35$\times$ fewer parameters and being 10$\times$ faster in training. As for the psMNIST benchmark, our proposed model reaches higher than 93% test accuracy within 3 epochs, outperforming both LSTM (van der Westhuizen and Lasenby 2018) (Helfrich, Willmott, and Ye 2018) and GRU (Hafner et al. 2017) (Chang et al. 2017). The proposed O-FNN reaches the state of the art performance at sMNIST, while outperforming all single-layer RNN architectures at processing the challenging permuted sMNIST.

The noise padded CIFAR-10 experiment poses great challenges for models to learn temporal dependence over the large number of time steps, due to the fact that the padded input sequences contain Gaussian noise in 968 out of 1000 time steps while only 32 time steps contain the serialized CIFAR10 images (Krizhevsky and Hinton 2009). From Table 1, we observe that O-FNN outperforms other RNN architectures on this benchmark, while requiring only 65k parameters. We found that a higher base frequency ($f$=8.0) is

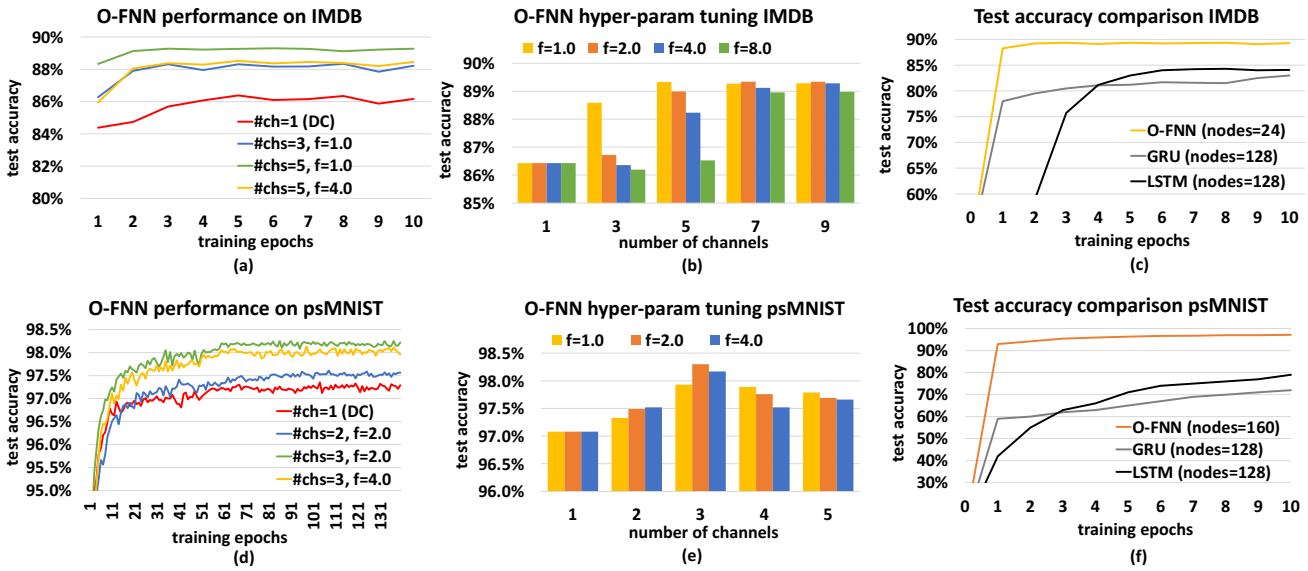


Figure 4: (a) O-FNN performance vs training epochs on IMDB dataset (b) Hyper-parameters tuning on IMDB dataset (c) Test accuracy comparison on IMDB dataset (d) O-FNN performance vs training epochs on psMNIST dataset (e) Hyper-parameters tuning on psMNIST dataset (f) Test accuracy comparison on psMNIST dataset.

| Task | Model | Test accuracy(%) | # Units | # Params | # Epochs |
|---|---|---|---|---|---|
| IMDB sentiment analysis | LSTM (Campos et al. 2018) | 86.8 | 128 | 220k* | - |
| | GRU (Dey and Salem 2017) | 84.8 | 128 | 99k | 100 |
| | ReLUGRU (Dey and Salem 2017) | 85.2 | 128 | 99k | 100 |
| | coRNN (Rusch and Mishra 2021) | 87.4 | 128 | 46k | 100 |
| | **O-FNN [This Work]** | **89.4** | **24** | **6k** | **5** |
| | **compact O-FNN [This Work]** | **88.6** | **4** | **1k** | **5** |
| sMNIST / permuted sMNIST | LSTM (Helfrich, Willmott, and Ye 2018) | 98.9 / 92.9 | 256 | 270k | - |
| | GRU (Chang et al. 2017) | 99.1 / 94.1 | 256 | 200k | - |
| | Dilated GRU (Chang et al. 2017) | 99.2 / 94.6 | 256 | 20k | - |
| | coRNN (Rusch and Mishra 2021) | 99.4 / 97.3 | 256 | 134k | 100 |
| | **O-FNN [This Work]** | **99.3 / 98.3** | **48 / 160** | **11k / 29k** | **54 / 86** |
| Noise padded CIFAR-10 | LSTM (Kag, Zhang, and Saligrama 2020) | 11.6 | 128 | 64k | - |
| | Incremental RNN (Kag, Zhang, and Saligrama 2020) | 54.5 | 128 | 11.5k | - |
| | Lipshitz RNN (Erichson et al. 2020) | 59 | 256 | 134k | 100 |
| | coRNN (Rusch and Mishra 2021) | 59 | 128 | 46k | 120 |
| | **O-FNN [This Work]** | **60.1** | **128** | **65k** | **39** |
| HAR-2 | LSTM (Kag, Zhang, and Saligrama 2020) | 93.7 | 64 | 16k | - |
| | GRU (Kag, Zhang, and Saligrama 2020) | 93.6 | 75 | 16k | - |
| | FastGRNN-LSQ (Kusupati et al. 2018) | 95.6 | 80 | 7.5k† | 300 |
| | coRNN (Rusch and Mishra 2021) | 97.2 | 64 | 9k | 250 |
| | **O-FNN [This Work]** | **96.3** | **64** | **3k** | **57** |
| | **compact O-FNN [This Work]** | **94.7** | **16** | **0.8k** | **77** |

Table 1: O-FNN in comparison with references on multiple benchmark data sets. (Numbers marked with $*$ and † are obtained from (Rusch and Mishra 2021) and (Kag, Zhang, and Saligrama 2020) respectively).

| Models | mean | standard deviation |
|---|---|---|
| IMDB (24 units) | 88.85% | 0.49% |
| sMNIST (48 units) | 99.01% | 0.15% |
| psMNIST (160 units) | 97.73% | 0.68% |
| Noise pdded CIFAR-10 (128 units) | 59.22% | 0.98% |
| HAR-2 (64 units) | 95.87% | 0.57% |

Table 2: Mean and standard deviation of O-FNN performance for experiments based on 10 re-trainings using random initialization of the trainable parameters.

needed for the model to learn informative features in presence of highly concentrated input data over a small fraction of total sequence (i.e. only first 32 actual image input from 1000 total steps). The intuition is that high frequency channels are more capable of extracting localized pattern formed by input from adjacent time steps, while low frequency channels tend to get an averaged effect from input over large numbers of steps. To this effect, it is challenging for low frequency channels to learn meaningful features since noises take a dominating proportion in those long sequences. Our study demonstrates that tuning the hyper-

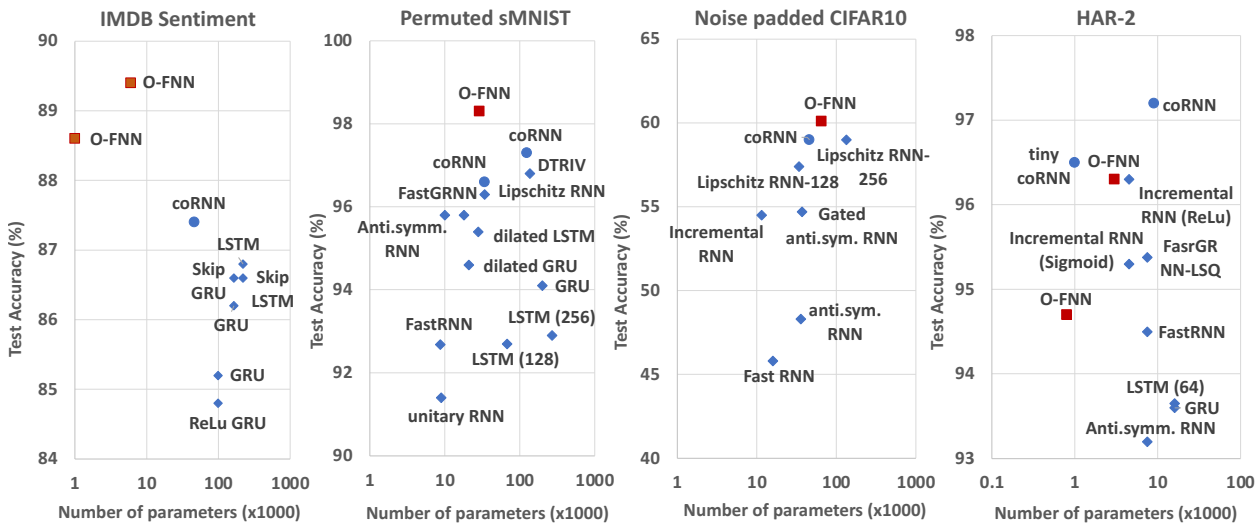


Figure 5: Summary of test accuracy versus number of parameters of O-FNN in comparison with recent references. Reference points are collected from (Arjovsky, Shah, and Bengio 2016), (Dey and Salem 2017),(Chang et al. 2017) (Campos et al. 2018),(Helfrich, Willmott, and Ye 2018), (Kusupati et al. 2018),(Lezcano Casado 2019),(Chang et al. 2019),(Kag, Zhang, and Saligrama 2020), (Erichson et al. 2020), and (Rusch and Mishra 2021).

parameter $f$ can provide a powerful knob for accommodating the imbalanced distribution of input information among noise padding in long sequences.

As for the HAR-2 dataset, we can see that O-FNN can reach close to SOTA accuracy with fewer parameters compared to most of the references. Morevoer, we observe that even the compact version of O-FNN remain to perform at about 95% test accuracy with less than 1K parameters. Therefore, the compact O-FNN model can be well poised for applications in edge applications and IOT devices.

Figure 5 summarizes the accuracy and number of parameters of the proposed O-FNN model in comparison with recent references. In plots of model test accuracy vs model size, a trade-off is typically found when the accuracy of a model increases with number of parameters. Therefore, intrinsic gain in the capability of a model will be demonstrated if the position of the model in this plot can shift up vertically. Specifically, we observe that O-FNN is overall on the top/left side of most references across all four datasets.The overall high accuracy across various tasks can be attributed to the capability of learning features at various time scales through different frequency channels. Based on the trade-off plots, O-FNN is significantly more efficient than all the references at IMDB sentiment analysis task. As for permuted sMNIST and noise-padded CIFAR10, O-FNN outperforms the most recent references, while the separation is smaller. On HAR-2, both O-FNN and the recently proposed coRNN perform well, suggesting a good potential of utilizing neurons with oscillatory dynamics in processing temporal signals. The compact size of O-FNN is partially attributed to the absence of the $W_{hh}$ matrix in regular RNNs, as well as the fact that our simplified Discrete Fourier Transform requires only a few frequency channels each associated with one Time-Varying Cosine neuron. The observation of fast convergence in training across various tasks can be attributed to O-FNN's fully parallelizable architecture, which makes the error back-propagation more direct than regular RNNs that requires recursive computations through time.

## Conclusion

A novel RNN architecture (O-FNN) based on a Time-Varying Cosine neuron model is proposed for sequential processing. Compared to conventional Fourier transform on raw input data and ReLU/sigmoidal activations, the proposed architecture provides a computationally efficient approach to extracting frequency-domain information from sequential data. Moreover, the O-FNN by design have full parallelism in both forward and backward passes, leading to significant simplification in training while eliminating the issue of exploding/vanishing gradients encountered by RNNs. In contrast with GRU/LSTM models, which require various types of gates and additional memory/cell states to retain long-term memory, our proposed architecture has extremely compact model size, while achieving the retention of information over long sequences through the learning of frequency-domain feature. We show that O-FNN is capable of handling long time dependencies with significantly smaller model size and lower computational cost, while retaining superior accuracy, faster convergence, and better error resiliency than the State-of-The-Art across various types of sequential tasks. Future avenues of work could extend the current O-FNN to address many-to-many tasks such as machine translation, or image captioning. Leveraging the potential of fast training and having compact models, O-FNN can also be further explored to process complex tasks such as video analysis.

## Acknowledgments

The research was funded in part by C-BRIC, one of six centers in JUMP, a Semiconductor Research Corporation (SRC) program sponsored by DARPA, Vannevar Bush Faculty fellowship, National Science Foundation, Army Research Laboratory and Sandia National Laboratory.

## References

Anguita, D.; Ghio, A.; Oneto, L.; Parra, X.; and Reyes-Ortiz, J. L. 2012. Human Activity Recognition on Smartphones Using a Multiclass Hardware-Friendly Support Vector Machine. In Bravo, J.; Hervás, R.; and Rodríguez, M., eds., *Ambient Assisted Living and Home Care*, 216–223. Berlin, Heidelberg: Springer Berlin Heidelberg. ISBN 978-3-642-35395-6.

Arjovsky, M.; Shah, A.; and Bengio, Y. 2016. Unitary Evolution Recurrent Neural Networks. In Balcan, M. F.; and Weinberger, K. Q., eds., *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, 1120–1128. New York, New York, USA: PMLR.

Brochier, R.; Guille, A.; and Velcin, J. 2019. Global Vectors for Node Representations. In *The World Wide Web Conference*, WWW '19, 2587–2593. New York, NY, USA: Association for Computing Machinery. ISBN 9781450366748.

Campos, V.; Jou, B.; Giró-i Nieto, X.; Torres, J.; and Chang, S.-F. 2018. Skip RNN: Learning to Skip State Updates in Recurrent Neural Networks. In *International Conference on Learning Representations*.

Chang, B.; Chen, M.; Haber, E.; and Chi, E. H. 2019. AntisymmetricRNN: A Dynamical System View on Recurrent Neural Networks. In *International Conference on Learning Representations*.

Chang, S.; Zhang, Y.; Han, W.; Yu, M.; Guo, X.; Tan, W.; Cui, X.; Witbrock, M. J.; Hasegawa-Johnson, M.; and Huang, T. S. 2017. Dilated Recurrent Neural Networks. *CoRR*, abs/1710.02224.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Dey, R.; and Salem, F. M. 2017. Gate-variants of Gated Recurrent Unit (GRU) neural networks. In *2017 IEEE 60th International Midwest Symposium on Circuits and Systems (MWSCAS)*, 1597–1600.

Erichson, N. B.; Azencot, O.; Queiruga, A.; and Mahoney, M. W. 2020. Lipschitz Recurrent Neural Networks. *CoRR*, abs/2006.12070.

Hafner, D.; Irpan, A.; Davidson, J.; and Heess, N. 2017. Learning Hierarchical Information Flow with Recurrent Neural Modules. In Guyon, I.; Luxburg, U. V.; Bengio, S.; Wallach, H.; Fergus, R.; Vishwanathan, S.; and Garnett, R., eds., *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Helfrich, K.; Willmott, D.; and Ye, Q. 2018. Orthogonal Recurrent Neural Networks with Scaled Cayley Transform. In Dy, J.; and Krause, A., eds., *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, 1969–1978. PMLR.

Kag, A.; Zhang, Z.; and Saligrama, V. 2020. RNNs Incrementally Evolving on an Equilibrium Manifold: A Panacea for Vanishing and Exploding Gradients? In *International Conference on Learning Representations*.

Koplon, R.; and Sontag, E. D. 1997. Using Fourier-neural recurrent networks to fit sequential input/output data. *Neurocomputing*, 15(3-4): 225–248.

Krizhevsky, A.; and Hinton, G. 2009. Learning multiple layers of features from tiny images. *Master's thesis, Department of Computer Science, University of Toronto*.

Kusupati, A.; Singh, M.; Bhatia, K.; Kumar, A.; Jain, P.; and Varma, M. 2018. FastGRNN: A Fast, Accurate, Stable and Tiny Kilobyte Sized Gated Recurrent Neural Network. *Proceedings of the 32nd International Conference on Neural Information Processing SystemsDecember*, 9031–9042.

Lapedes, A.; and Farber, R. 1987. Nonlinear signal processing using neural networks: Prediction and system modelling.

Le, Q. V.; Jaitly, N.; and Hinton, G. E. 2015. A Simple Way to Initialize Recurrent Networks of Rectified Linear Units. *CoRR*, abs/1504.00941.

Lecun, Y.; Bottou, L.; Bengio, Y.; and Haffner, P. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11): 2278–2324.

Lee-Thorp, J.; Ainslie, J.; Eckstein, I.; and Ontañón, S. 2021. FNet: Mixing Tokens with Fourier Transforms. *CoRR*, abs/2105.03824.

Lezcano Casado, M. 2019. Trivializations for Gradient-Based Optimization on Manifolds. In Wallach, H.; Larochelle, H.; Beygelzimer, A.; d'Alché-Buc, F.; Fox, E.; and Garnett, R., eds., *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Maas, A. L.; Daly, R. E.; Pham, P. T.; Huang, D.; Ng, A. Y.; and Potts, C. 2011. Learning Word Vectors for Sentiment Analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, 142–150. USA: Association for Computational Linguistics. ISBN 9781932432879.

McCaughan, D. 1997. On the properties of periodic perceptrons. In *Proceedings of International Conference on Neural Networks (ICNN'97)*, volume 1, 188–193 vol.1.

Pandey, H. 2020. A Comparative Analysis of Recurrent Neural Networks and 1D Convolution on IMDb Movie Review Dataset. In *GitHub repository*.

Parascandolo, G.; Huttunen, H.; and Virtanen, T. 2017. Taming the waves: sine as activation function in deep neural networks. https://openreview.net/forum?id=Sks3zF9eg.

Ramachandran, P.; Zoph, B.; and Le, Q. V. 2017. Searching for activation functions. *arXiv preprint arXiv:1710.05941*.

Rusch, T. K.; and Mishra, S. 2021. Coupled Oscillatory Recurrent Neural Network (coRNN): An accurate and (gradient) stable architecture for learning long time dependencies. In *International Conference on Learning Representations*.

Sitzmann, V.; Martel, J.; Bergman, A.; Lindell, D.; and Wetzstein, G. 2020. Implicit neural representations with periodic activation functions. *Advances in Neural Information Processing Systems*, 33.

Sopena, J.; and Alquezar, R. 1994. Improvement of learning in recurrent networks by substituting the sigmoid activation function. In *International Conference on Artificial Neural Networks*, 417–420. Springer.

Sopena, J.; Romero, E.; and Alquezar, R. 1999. Neural networks with periodic and monotonic activation functions: a comparative study in classification problems. In *1999 Ninth International Conference on Artificial Neural Networks ICANN 99. (Conf. Publ. No. 470)*, volume 1, 323–328 vol.1.

Tamkin, A.; Jurafsky, D.; and Goodman, N. 2020. Language Through a Prism: A Spectral Approach for Multiscale Language Representations. *arXiv preprint arXiv:2011.04823*.

van der Westhuizen, J.; and Lasenby, J. 2018. The unreasonable effectiveness of the forget gate. arXiv:1804.04849.

Wong, K.-w.; Leung, C.-s.; and Chang, S.-j. 2002. Handwritten digit recognition using multilayer feedforward neural networks with periodic and monotonic activation functions. In *Object recognition supported by user interaction for service robots*, volume 3, 106–109. IEEE.

Yin, W.; Kann, K.; Yu, M.; and Schütze, H. 2017. Comparative Study of CNN and RNN for Natural Language Processing. *CoRR*, abs/1702.01923.

Zhang, J.; Lin, Y.; Song, Z.; and Dhillon, I. 2018. Learning long term dependencies via fourier recurrent units. In *International Conference on Machine Learning*, 5815–5823. PMLR.

Zhang, Y.-Q.; and Chan, L.-W. 2000. Forenet: Fourier recurrent networks for time series prediction. In *International Conference on Neural Information Processing*, pp 576 – 582.