CONTINUOUS CNN FOR NONUNIFORM TIME SERIES

Hui Shi[†]*, Yang Zhang*[⋆]*, Hao Wu*[‡]*, Shiyu Chang*[⋆]*, Kaizhi Qian*[⋆]*, Mark Hasegawa-Johnson*[‡]*, Jishen Zhao*[†]

[†] University of California, San Diego [⋆] MIT-IBM Watson AI Lab
[‡] University of Illinois at Urbana-Champaign

ABSTRACT

CNN for time series data implicitly assumes that the data are uniformly sampled, whereas many event-based and multimodal data are nonuniform or have heterogeneous sampling rates. Directly applying regular CNN to nonuniform time series is ungrounded, because it is unable to recognize and extract common patterns from the nonuniform input signals. In this paper, we propose the Continuous CNN (CCNN), which estimates the inherent continuous inputs by interpolation, and performs continuous convolution on the continuous input. The interpolation and convolution kernels are learned in an end-to-end manner, and are able to learn useful patterns despite the nonuniform sampling rate. Results of several experiments verify that CCNN achieves a better performance on nonuniform data, and learns meaningful continuous kernels.

Index Terms— deep learning, time series, convolutional neural network, signal processing

1. INTRODUCTION

Convolutional neural network (CNN), together with recurrent neural network (RNN), is among the most popular deep learning architectures to process time series data, especially after the success of WaveNet [1] on audio data. However, both CNN and RNN rest on the assumption that both the input and output data are sampled uniformly. However, many time-series data are event-based and thus not uniform in time, such as stock price [2], social media data [3] and health care data [4].

Recently, research have been done on RNN structures for nonuniform time series. Some works [5, 6, 7] use continuous-time dynamical system methods to design RNN structures. Phased-LSTM [8] and Time-LSTM [9] introduce a time gate that passes the data at a certain frequency. Similar ideas can be found in Clockwork RNN [10] and DilatedRNN [11].

For CNN, although the models with multiple spatial or temporal resolutions are proposed for image segmentation [12], object detection [13, 14, 15] and image super-resolution [16, 17, 18], they still require uniform sampling rate within a single patch. Few CNN-based approaches are able to deal with truly nonuniform data, except the paradigm to directly feed nonuniform data feature vectors with appended sampling interval information to a regular CNN [19].

The problem is that, without the uniform sampling assumptions, the application of the regular CNN is ungrounded, and thus the performance is compromised. This is because one major justification of CNN is that the filters/kernels are able to extract useful patterns from input signals. But if the sampling rate varies, the traditional CNN implementations will no longer be able to recognize the same pattern. An alternative approach is to convert the time series to uniform by interpolation before applying. But existing interpolation schemes may not be able to recover signal well enough.

Motivated these challenges, we propose Continuous CNN (CCNN), a generalization to CNN for nonuniform time series. CCNN estimates the implicit continuous signal by interpolation, yet performs continuous convolution on the continuous signal. As a result, CCNN is capable of capturing the useful patterns in the implicit input signal under nonuniform sampling rate. Furthermore, the interpolation and convolution kernels are not preset, but rather learned in an end-to-end manner.

The proposed CCNN is well supported by works on spiking neural networks (SNN) [20], which mimic how human brains process information. The inputs to SNNs are spike chains with nonuniform intervals that carry information. An important class SNNs [21, 22, 23] convolves the input with continuous kernel functions, which is similar to the key step of CCNN. However, CCNN differs from SNN in two aspects. First, for SNN, the input information resides in time intervals, not in the inputs values; the goal of the SNN convolution is to extract time interval information. In contrast, the input information for CCNN resides in input values, not time intervals; the goal of the continuous convolution is to remove the impact of nonuniform sampling. Second, CCNN learns the kernel functions in a data-driven manner, whereas SNN employs predefined kernel functions.

2. THE CCNN ALGORITHM

Our problem is formulated as follows. Given a nonuniform input sequence $x(t_1), x(t_2), \cdots, x(t_N) \in \mathcal{X}_{\text{in}}$, where the input time stamps $t_n \in \mathcal{T}_{\text{in}}$ can be distributed nonuniformly, our goal is to design a continuous convolutional layer that can produce output, $y(t_{out})$, for any *arbitrary* output time t_{out}.

The proposed CCNN solves the problem via two steps:
(1) interpolation to recover the continuous signal $\hat{x}(t)$;

(2) *continuous* convolution on $\hat{x}(t)$, producing the final output $y(t_{out})$.

Furthermore, rather than applying a preset interpolation, CCNN learns the interpolation kernel and the convolution kernel in an end-to-end manner.

Reconstruct Continuous Signal. CCNN reconstructs the underlying continuous input signal, $\hat{x}(t)$, by interpolating among nonuniform input samples.

$$\hat{x}(t) = \sum_{i=1}^{N} x(t_i) I(t - t_i; \mathcal{T}_{\text{in}}, \mathcal{X}_{\text{in}}) + \varepsilon(t; \mathcal{T}_{\text{in}}, \mathcal{X}_{\text{in}}) \tag{1}$$

where the first term is the interpolation term, and $I(\cdot)$ is the *interpolation kernel*; the second term is the *error correction* term. For the first term, a form analogous to the Parzen window approach [24] is used. Considering the versatility of $I(\cdot)$, the interpolation algorithms representable by Eq. (1) are vast.

Continuous Convolution. Analogous to a standard CNN layer, after the continuous input is estimated by interpolation, the CCNN layer performs a continuous convolution to produce the final output.

$$y(t_{out}) = [\hat{x}(t) * C(t)]|_{t=t_{out}} + b \tag{2}$$

where $*$ denotes continuous convolution, $C(t)$ is the *convolution kernel*, and b is bias. Combining Eqs. (1) and (2) gives

$$\begin{aligned} y(t_{out}) =& \sum_{i=1}^{N} x(t_i) \left[I(t_{out} - t_i; \mathcal{T}_{\text{in}}, \mathcal{X}_{\text{in}}) * C(t) \right] \\ &+ [\varepsilon(t; \mathcal{T}_{\text{in}}, \mathcal{X}_{\text{in}}) * C(t) + b]|_{t=t_{out}} \\ =& \sum_{i=1}^{N} x(t_i) K(t_{out} - t_i; \mathcal{T}_{\text{in}}, \mathcal{X}_{\text{in}}) + \beta(t_{out}; \mathcal{T}_{\text{in}}, \mathcal{X}_{\text{in}}) \end{aligned} \tag{3}$$

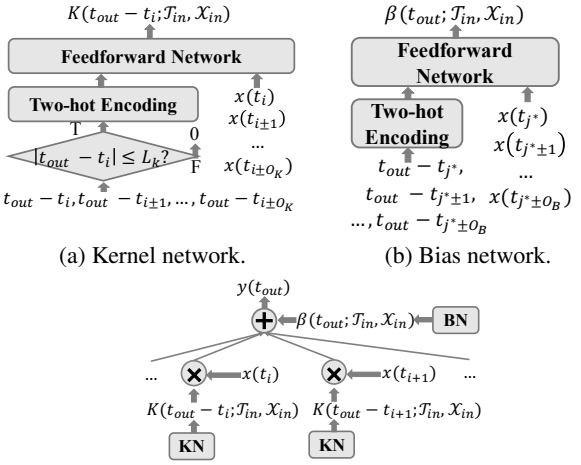
where $K(t_{out}; \mathcal{T}_{\text{in}}, \mathcal{X}_{\text{in}})$ is the collapsed kernel function, representing the combined effect of interpolation and convolution; $\beta(t_{out}; \mathcal{T}_{\text{in}}, \mathcal{X}_{\text{in}})$ is the collapsed bias function, representing the combined effect of error correction and convolution. Eq. (3) shows that learning the interpolation and convolution kernels and errors is now simplified into learning the collapsed kernel and bias functions.

3. THE CCNN MODEL

A CCNN layer has three parts: the kernel network learning $K(\cdot)$, the bias network learning $\beta(\cdot)$, and the main network producing the final output via Eq. (3).

The Kernel Network. Kernel network uses a feedforward network to represent the kernel function (shown in Fig. 1(a)), based on the fact that a neural network can represent any function given enough layers and nodes [25]. In order to regularize the complexity, a few assumptions on $K(\cdot)$ are introduced:

• *Stationarity and Finite Dependency:* The dependency of $K(\cdot)$ on $\mathcal{T}_{\text{in}}$ is relative to the output time t_{out}, and is constrained to among the adjacent time stamps, i.e.



(a) Kernel network. (b) Bias network.

(c) Compute $y(t)$ by collapsed interpolation and convolution.

Fig. 1: CCNN structure.

$$K(t_{out} - t_i; \mathcal{T}_{\text{in}}, \mathcal{X}_{\text{in}}) = K(\{t_{out} - t_{i\pm k}, x(t_{i\pm k})\}_{k=0:O_K}) \tag{4}$$

where $\{t_{out} - t_{i\pm k}, x(t_{i\pm k})\}_{k=0:O_K}$ denotes the set of $t_{out} - t_{i\pm k}$ and $x(t_{i\pm k})$ where k runs from 0 to O_K, and O_K is the order of the kernel network.

• *Finite Kernel Length:* The collapsed kernel function has finite length.

$$K(t_{out} - t_i; \mathcal{T}_{\text{in}}) = 0, \forall |t_{out} - t_i| > L_K \tag{5}$$

where L_K is the kernel length. This assumption implies the interpolation and the convolutional kernels both have finite length. While many interpolation kernels do have finite length (*e.g.* linear interpolation), others do not (*e.g.* sinc interpolation). Nevertheless, most infinite-length interpolation kernels, including sinc interpolation, have tapering tails, and thus truncation on both sides still provides good approximations. Regarding the convolutional kernel, the finite length assumption naturally extends from the standard CNN.

The Bias Network. For the bias network, a similar stationarity and finite dependency assumption is applied as follows.

$$\begin{aligned} \beta(t_{out}; \mathcal{T}_{\text{in}}, \mathcal{X}_{\text{in}}) = \beta(\{t_{out} - t_{j^*\pm k}, x(t_{j^*\pm k})\}_{k=0:O_B}), \\ \text{where, } t_{j^*} = \underset{t_j \in \mathcal{T}_{\text{in}}}{\operatorname{argmin}} |t_j - t_{out}| \end{aligned} \tag{6}$$

t_{j^*} is the closest input time stamp to output time t_{out}, and O_B denotes the order of the bias network. Fig. 1(b) shows the bias network, which is also a feedforward network.

Causal Setting. For causal tasks, current output should not depend on future input, and therefore the $t_{out} - t_{i\pm k}$ terms that are greater than 0, as well as the corresponding $x(t_{i\pm k})$, are removed from Eq. (4). Similarly, $t_{out} - t_{j^*\pm k}$ that are greater than 0, as well as the corresponding $x(t_{j^*\pm k})$, are removed from Eq. (6). Also, the condition bound in Eq. (5), *i.e.* $|t_{out} - t_i| > L_K$, is replaced with $t_{out} - t_i > L_K$ or $t_{out} - t_i < 0$.

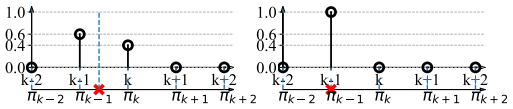


Fig. 2: Two-hot encoding illustration.

Two-Hot Encoding. The kernel and bias functions can be complicated functions of the input times, so model complexity and convergence can be serious concerns. Therefore, we introduce a two-hot encoding scheme based on [26] for vectorizing the input time intervals without losing information.

Denote the time interval value to encode as Δt. The two-hot encoding scheme partitions a range into $d-1$ intervals, whose edges are $\pi_1, \pi_2, \cdots, \pi_d$, and the interval width $\delta = \pi_k - \pi_{k-1}$ is a constant value. The two-hot encoding of Δt is a vector of length $d-1$, the elements of which are decided as: when Δt falls in an interval, the two elements corresponding to its two edges are lit. Formally, denote the encoded vector as $\boldsymbol{g}$, and suppose Δt falls in interval $[\pi_k, \pi_{k+1})$. Then

$$\boldsymbol{g}_k = \frac{\pi_{k+1} - \Delta t}{\delta}, \boldsymbol{g}_{k+1} = \frac{\Delta t - \pi_k}{\delta}; \boldsymbol{g}_l = 0, \forall l \notin \{k, k+1\} \tag{7}$$

where $\boldsymbol{g}_k$ denotes the k-th element of $\boldsymbol{g}$. Fig. 2 gives an intuitive visualization of the encoding process.

4. EVALUATION

4.1. Auto-Regression Task on Simulated Data

To reveal the misspecification error introduced by interpolation methods with the pre-set kernels, we compare the performance of CCNN and other adaptions of CNN/RNN structures for nonuniform time series on the prediction of the $x(t_{N+1})$, given the previous nonuniform samples $x(t_1) \cdots x(t_N)$.

Datasets and Configuration. The synthetic datasets are generated from three standard time series: Sine, Mackey-Glass (MG), and Lorenz. Shown in Table 1. The standard time series functions are introduced as $x(t)$. We set the parameters of the three sequences as follows: (1) $T = 5$ in Sine; (2) $\beta = 0.2$, $\tau = 17$, $n = 10$, and $\gamma = 0.2$ in MG; (3) $\sigma = 10$, $r = 28$, $b = 8/3$ in Lorenz. Sine signal is sampled by a Poisson process with mean of 1. MG and Lorenz are two chaotic time series which do not have closed-form solutions to their delay differential equations. Thus the Runge-Kutta method [29] is applied to obtain a uniform sequence with sampling interval Δt, which are then subsampled into nonuniform sequences by choosing M ($M < N$) sample points at random. We set $M = 42$, $N = 14$;$\Delta t = 2$ in MG, and $\Delta t = 0.05$ in Lorenz.

All the networks are trained with Adam optimizer [30] and mean squared error (MSE) loss. The training batch size is 20. The number of training steps is determined by validation. The validation set size is 10,000.

Table 1: Synthetic continuous signals. $\dot{x}$ denotes dx/dt

Sine	$x(t) = sin\left(\frac{2\pi t}{T}\right)$
Mackey-Glass (MG) [27]	$\dot{x}(t) = \beta\frac{x(t-\tau)}{1+x(t-\tau)^n} - \gamma x(t)$
Lorenz [28]	$\dot{x}(t) = \sigma(y(t) - x(t))$ $\dot{y}(t) = -x(t)z(t) + rx(t) - y(t)$ $\dot{z}(t) = x(t)y(t) - bz(t)$

Table 2: MSE of prediction on simulated data ($\times 10^{-2}$).

Alg.	Sine	MG	Lorenz
CNN	46.0 (8.22)	12.8 (3.92)	9.90 (3.33)
CNNT	20.2 (7.65)	3.50 (1.29)	5.97 (2.41)
CNNT-th	8.44 (4.58)	3.00 (1.21)	8.37 (3.24)
ICNN	0.72 (0.83)	0.72 (0.42)	4.22 (2.27)
RNNT	36.1(12.9)	8.15(3.32)	13.4(3.95)
RNNT-th	19.5(6.48)	8.48(3.11)	13.9(4.36)
CCNN	0.88 (0.61)	2.46 (0.89)	3.93 (1.73)
CCNN-th	**0.42** (0.36)	**0.53** (0.97)	**3.25** (1.67)

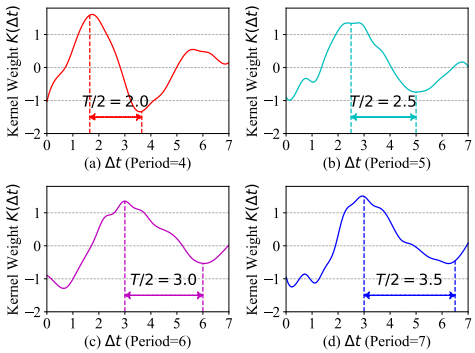


Fig. 3: The learned continuous kernel function

Models. The following algorithms are compared:

• *CCNN:* The first layer is a CCNN layer takes both the time intervals and the signal sequence. Then the sequence is resampled onto a uniform time interval. CCNN is set to the causal setting as of Sec. 3.

• *CNN:* data are directly fed into a regular CNN.

• *CNNT:* The time intervals are appended to the input data, which are fed to a regular CNN.

• *ICNN:* data are interpolated to uniform before being fed to a regular two-layer CNN. Piecewise constant, linear, quadratic, cubic spline and sinc kernels are evaluated, and only the best performance is reported.

• *RNNT:* the time intervals are appended to the input data, then are fed into a vanilla RNN.

All the networks use ReLU activations after the first layer. For CNN, ICNN, and CNNT, the convolution kernel length of each layer is set to 7. For ICNN, the input signal is interpolated at timestamps $t_{N+1} - k$, $k = 1, \cdots, 13$ to form a uniform sequence. For CCNN, the output time stamps of the first layer are also $t_{N+1} - k$. The kernel length $L_K = 3$. The second layer of CCNN is a regular convolutional layer with kernel length 7. These configurations ensure that all the neural networks have the same *expected* receptive field size of 13. *-th* indicates time information is two-hot encoded in the model.

Results and Analysis. There are three observations from Table 2. *First*, CCNN-th outperforms the other baselines in terms of prediction accuracy. *Second*, interpolation methods (ICNN and CCNN) generally outperform the other baselines,

particularly CNNT. This again shows that interpolation is more reasonable for dealing with nonuniform time series than simply appending the time intervals. Furthermore, preset interpolation algorithms (ICNNs) can rarely match CCNN that has the flexibility to learn its own interpolation kernel. *Third*, two-hot encoding usually improves performance.

Kernel Analysis. We visualized the learned collapsed continuous kernels $K(t_{out} - t_i; \mathcal{T}_{\text{in}}, \mathcal{X}_{\text{in}})$ by repeating the experiment with different Sine signals, $T = 4, 5, 6, 7$. The CCNN remains in the same configuration except that the number of filter is 1, so that the $K(t_{out} - t_i; \mathcal{T}_{in}, \mathcal{X}_{in})$ outputs scalars. Fig. 3 shows the learned continuous kernel functions, each of which is a sine-like function with estimated period equaling the underlying signal period. The result convinces of that the advantage of CCNN origins from the ability of learning the interpolation and convolution from the data.

4.2. Speech Interpolation

The speech interpolation task involves restoring the high-resolution speech signal from the downsampled signal. To mitigate the complexity in directly working on speech waveforms, the sampling rate of the high-resolution speech is set to 4 kHz, and that of the downsampled signals is 2 kHz. Three different downsampling schemes are tested: 1) *uniform filtered*, uniformly downsamples the speech to 2 kHz after passing an anti-aliasing filter. 2) *uniform unfiltered*, uniformly downsamples the signal without the anti-aliasing filter. 3) *nonuniform*, randomly preserves half of the speech samples, and thus the resulting signal has an average sampling rate of 2 kHz.

Dataset. Our dataset consists of one hour of lab-recorded speech of one speaker reading structured composite sentences. The training, validation and test are split 8:1:1. The high-resolution speech is chunked into 40-point sequences without overlap, and the downsampled speech into 20-point sequences.

Models and Configurations. We additionally include a spectrum-based benchmark, Speech DNN, from [31]. Since the phase spectrum estimate of Speech DNN can be inaccurate, we introduced another version, called Speech DNN with a cheated phase (CP), where the ground truth phase spectrum is applied. Note that this version is given too much oracle information to be fairly compared with.

• *CCNN-th:* The timestamps are normalized such that the sampling interval of the original 4 kHz speech is 0.5. The first layer outputs the uniform timestamps at the rate of 4 kHz. The bias and kernel network have one layer, with $O_B = 28$, $O_K = 7$, $L_K = 10$ and 16 hidden filters. The two-hot encoding time interval $\delta = 0.5$. The second layer is a 1×28 CNN layer.

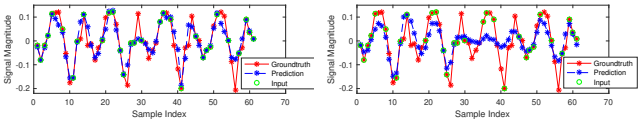
• *ICNN:* the ICNN interpolates the low-resolution speech into 4kHz before it is fed to a CNN which contains two 1×28 convolutional layers with 80 hidden filters, similar to [16].

• *Speech DNN & Speech DNN CP:* Speech DNN converts both the high resolution and downsampled speech into amplitude spectra using FFT with 64ms window length and 48ms window shift. Speech DNN has 3 hidden layers, each of which has 1024 hidden nodes.

Table 3: Signal-to-Noise Ratio (dB) in Speech Interpolation.

Alg.	Uniform		Non-uniform
	filtered	unfiltered	
Speech DNN	9.13	-*	-*
Speech DNN (CP)	**13.64**	-*	-*
ICNN	9.74	7.49	2.83
CCNN-th	9.61	**7.80**	**6.58**

* Speech DNN does not work with non-filtered down-sampled signals and nonuniformly down-sampled signals.



(a) By CCNN (b) By ICNN

Fig. 4: Restored speech compared with ground truth.

Results and Analysis. Tab. 3 shows the Signal-to-Noise Ratio (SNR) of the signals recovered. The Speech DNN with cheated phase yields the best SNR, because it uses the phase information from ground truth. However, the Speech DNN without cheated phase has similar performance to CNN and CCNN. Comparing CCNN and ICNN, they have similar SNR under uniform sampling cases, which verifies that both architectures have similar representation power given uniform data. However, CCNN has much higher SNR than ICNN in nonuniform case. One important reason is that CCNN, by construction, is aware of whether neighboring samples are dense or sparse, and outputs robust interpolation kernels accordingly, despite the variation in sampling patterns.

Fig. 4 shows an example of the restored signal from the nonuniform samples by CCNN and ICNN respectively. CCNN can restore some spikes (*e.g.* ones at 25 and 40) even without an input sample point in the spike, because CCNN can learn the continuous kernels and restore the original spikes. CNN model does poorly in restoring spikes even when there are input sample points in the spikes.

5. CONCLUSION

In this paper, we have introduced CCNN for nonuniform time series with two takeaways. First, interpolation before continuous convolution is shown to be a reasonable way for nonuniform time series. Second, learning task-specific kernels in a data-driven way significantly improves the performance. There are two promising directions. First, we have focused on 1D convolution, but this framework can be generalized to multi-dimensional nonuniform data. Second, while the computational complexity is similar for CCNN and CNN, the runtime of the former is much longer, because of the lack of parallelization. Fast implementation of CCNN is thus another research direction.

6. REFERENCES

[1] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu, "Wavenet: A generative model for raw audio," *arXiv preprint arXiv:1609.03499*, 2016.

[2] Ramazan Gençay, Michel Dacorogna, Ulrich A Muller, Olivier Pictet, and Richard Olsen, *An Introduction to High-frequency Finance*, Academic press, 2001.

[3] Shiyu Chang, Yang Zhang, Jiliang Tang, Dawei Yin, Yi Chang, Mark A Hasegawa-Johnson, and Thomas S Huang, "Positive-unlabeled learning in streaming networks," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2016, pp. 755–764.

[4] Alistair EW Johnson, Tom J Pollard, Lu Shen, H Lehman Liwei, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark, "Mimic-iii, a freely accessible critical care database," *Scientific data*, vol. 3, pp. 160035, 2016.

[5] Barak A Pearlmutter, "Learning state space trajectories in recurrent neural networks," *Learning*, vol. 1, no. 2, 2008.

[6] Ken-ichi Funahashi and Yuichi Nakamura, "Approximation of dynamical systems by continuous time recurrent neural networks," *Neural Networks*, vol. 6, no. 6, pp. 801–806, 1993.

[7] Gert Cauwenberghs, "An analog VLSI recurrent neural network learning a continuous-time trajectory," *IEEE Transactions on Neural Networks*, vol. 7, no. 2, pp. 346–361, 1996.

[8] Daniel Neil, Michael Pfeiffer, and Shih-Chii Liu, "Phased LSTM: Accelerating recurrent network training for long or event-based sequences," in *Advances in Neural Information Processing Systems*, 2016, pp. 3882–3890.

[9] Yu Zhu, Hao Li, Yikang Liao, Beidou Wang, Ziyu Guan, Haifeng Liu, and Deng Cai, "What to do next: Modeling user behaviors by Time-LSTM," in *Twenty-Sixth International Joint Conference on Artificial Intelligence (IJICAI)*, 2017, pp. 3602–3608.

[10] Jan Koutnik, Klaus Greff, Faustino Gomez, and Juergen Schmidhuber, "A clockwork RNN," in *International Conference on Machine Learning*, 2014, pp. 1863–1871.

[11] Shiyu Chang, Yang Zhang, Wei Han, Mo Yu, Xiaoxiao Guo, Wei Tan, Xiaodong Cui, Michael Witbrock, Mark A Hasegawa-Johnson, and Thomas S Huang, "Dilated recurrent neural networks," in *Advances in Neural Information Processing Systems*, 2017, pp. 76–86.

[12] Mohsen Ghafoorian, Nico Karssemeijer, Tom Heskes, IWM van Uder, Frank-Erik de Leeuw, Elena Marchiori, Bram van Ginneken, and Bram Platel, "Non-uniform patch sampling with deep convolutional neural networks for white matter hyperintensity segmentation," in *Biomedical Imaging (ISBI), 2016 IEEE 13th International Symposium on*. IEEE, 2016, pp. 1414–1417.

[13] Forrest Iandola, Matt Moskewicz, Sergey Karayev, Ross Girshick, Trevor Darrell, and Kurt Keutzer, "Densenet: Implementing efficient convnet descriptor pyramids," *arXiv preprint arXiv:1404.1869*, 2014.

[14] Christian Szegedy, Alexander Toshev, and Dumitru Erhan, "Deep neural networks for object detection," in *Advances in Neural Information Processing Systems*, 2013, pp. 2553–2561.

[15] Pierre Sermanet, David Eigen, Xiang Zhang, Michaël Mathieu, Rob Fergus, and Yann LeCun, "Overfeat: Integrated recognition, localization and detection using convolutional networks," *arXiv preprint arXiv:1312.6229*, 2013.

[16] Jiwon Kim, Jung Kwon Lee, and Kyoung Mu Lee, "Accurate image super-resolution using very deep convolutional networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 1646–1654.

[17] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang, "Image super-resolution using deep convolutional networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 2, pp. 295–307, 2016.

[18] Zhaowen Wang, Ding Liu, Jianchao Yang, Wei Han, and Thomas Huang, "Deep networks for image super-resolution with sparse prior," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 370–378.

[19] Junbo Zhang, Yu Zheng, and Dekang Qi, "Deep spatio-temporal residual networks for citywide crowd flows prediction.," in *AAAI*, 2017, pp. 1655–1661.

[20] Wolfgang Maass, "Networks of spiking neurons: the third generation of neural network models," *Neural networks*, vol. 10, no. 9, pp. 1659–1671, 1997.

[21] Chris Eliasmith and Charles H Anderson, *Neural Engineering: Computation, Representation, and Dynamics in Neurobiological Systems*, MIT press, 2004.

[22] Jonathan Tapson and André van Schaik, "Learning the pseudoinverse solution to network weights," *Neural Networks*, vol. 45, pp. 94–100, 2013.

[23] Jonathan C Tapson, Greg Kevin Cohen, Saeed Afshar, Klaus M Stiefel, Yossi Buskila, Tara Julia Hamilton, and André van Schaik, "Synthesis of neural networks for spatio-temporal spike pattern recognition and processing," *Frontiers in Neuroscience*, vol. 7, pp. 153, 2013.

[24] Emanuel Parzen, "On estimation of a probability density function and mode," *The Annals of Mathematical Statistics*, vol. 33, no. 3, pp. 1065–1076, 1962.

[25] Kurt Hornik, "Approximation capabilities of multilayer feedforward networks," *Neural Networks*, vol. 4, no. 2, pp. 251–257, 1991.

[26] Andrew Adams, Jongmin Baek, and Myers Abraham Davis, "Fast high-dimensional filtering using the permutohedral lattice," in *Computer Graphics Forum*. Wiley Online Library, 2010, vol. 29, pp. 753–762.

[27] Leon Glass and Michael C Mackey, "Pathological conditions resulting from instabilities in physiological control systems," *Annals of the New York Academy of Sciences*, vol. 316, no. 1, pp. 214–235, 1979.

[28] Edward N Lorenz, "Deterministic nonperiodic flow," *Journal of the Atmospheric Sciences*, vol. 20, no. 2, pp. 130–141, 1963.

[29] Przemyslaw Bogacki and Lawrence F Shampine, "A 3 (2) pair of runge-kutta formulas," *Applied Mathematics Letters*, vol. 2, no. 4, pp. 321–325, 1989.

[30] Diederik Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[31] Kehuang Li and Chin-Hui Lee, "A deep neural network approach to speech bandwidth expansion," in *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*. IEEE, 2015, pp. 4395–4399.