Towards a General Purpose CNN for Long Range Dependencies in ND

David W. Romero [*1] David M. Knigge [*2] Albert Gu [3] Erik J. Bekkers [2] Efstratios Gavves [2]
Jakub M. Tomczak [1] Mark Hoogendoorn [1]

Abstract

The use of Convolutional Neural Networks (CNNs) is widespread in Deep Learning due to a range of desirable model properties which result in an efficient and effective machine learning framework. However, performant CNN architectures must be tailored to specific tasks in order to incorporate considerations such as the input length, resolution, and dimentionality. In this work, we overcome the need for problem-specific CNN architectures with our *Continuous Convolutional Neural Network* (CCNN): a single CNN architecture equipped with continuous convolutional kernels that can be used for tasks on data of arbitrary resolution, dimensionality and length without structural changes. Continuous convolutional kernels model long range dependencies at every layer, and remove the need for downsampling layers and task-dependent depths needed in current CNN architectures. We show the generality of our approach by applying the same CCNN to a wide set of tasks on sequential (1D) and visual data (2D). Our CCNN performs competitively and often outperforms the current state-of-the-art across all tasks considered.

1. Introduction

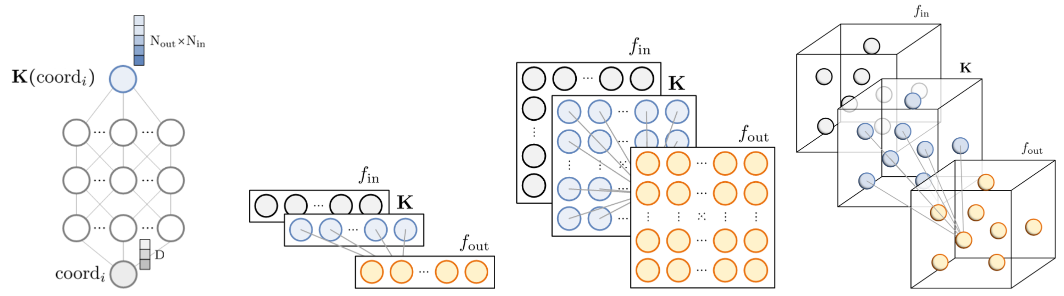
Convolutional Neural Networks (LeCun et al., 1998) (CNNs) are a class of Deep Learning models widely used for machine learning applications. Their popularity stems from their high performance and efficiency which has led them to achieve state-of-the-art in several applications across sequential (Abdel-Hamid et al., 2014; Van Den Oord et al., 2016), visual (Krizhevsky et al., 2012; Simonyan & Zisserman, 2014) and high-dimensional data (Schütt et al., 2017; Wu et al., 2019). Nevertheless, an important limitation of CNNs –and Neural Networks in general– is that their architectures must be tailored towards particular applications in order to handle different data lengths, resolutions and dimensionalities. This, in turn, has led to an extensive number of task-specific CNN architectures (Oord et al., 2016; Bai et al., 2018; Simonyan & Zisserman, 2014; Szegedy et al., 2015; Ronneberger et al., 2015; He et al., 2016; Qi et al., 2017; Wu et al., 2019).

Data can come at many different lengths, e.g., images can be 32x32 or 1024x1024 and audio can easily be 16000 per second. The problem with standard CNNs is that their convolutional kernels are *local*, which requires a custom architecture for each length with carefully chosen strides and pooling layers to capture full context. In addition, many types of data are inherently continuous in nature and have the same semantic meaning at different resolutions, e.g., images can be captured at arbitrary resolutions and have identical semantic content, and audio can be arbitrarily sampled at 16kHz or 44.1kHz and still sound the same to human ears. Nevertheless, conventional CNNs are bound to resolution and cannot be used across resolutions due to the *discrete nature* of their convolutional kernels. Both problems are further exacerbated when considering data of different dimensionality with the same CNN, e.g., sequential (1D), visual (2D) and high-dimensional data (3D, 4D), as different dimensionalities operate at different characteristic lengths and resolutions, e.g., a second of audio easily has length 16000 which strongly contrasts with the size of images in benchmark datasets (Krizhevsky et al., 2009; Deng et al., 2009).

Towards a general-purpose CNN architecture. In this work, we aim to construct a single CNN architecture that can be used on data of arbitrary resolutions, lengths and dimensionalities. Standard CNNs require task-specific architectures due to the discrete nature of their convolutional kernels which binds the kernels to specific data resolutions and makes them ill-suited to model global context due to the large amount of parameters required to construct large discrete convolutional kernels. Consequently, in order to construct a general-purpose CNN architecture it is crucial to develop a resolution agnostic convolutional layer able to model long range dependencies in a parameter efficient manner.

[*]Equal contribution [1]Vrije Universiteit Amsterdam, The Netherlands [2]University of Amsterdam, The Netherlands [3]Stanford University, CA, USA. Correspondence to: David W. Romero <d.w.romeroguzman@vu.nl>, David M. Knigge <d.m.knigge@uva.nl>.

Workshop on *Continuous Time Methods for Machine Learning* (ICML 2022). Copyright 2022 by the authors.



(a) (b) Sequential data (c) Visual data (d) Point clouds

Figure 1. Continuous convolutional kernels. A continuous convolutional kernel is parameterized with a small neural network $\mathrm{G}_{\mathrm{Kernel}}$ that receives coordinates $\mathbf{c}_i \in \mathbb{R}^{\mathrm{D}}$ as input and outputs the value of the convolutional kernel at that position $\mathbf{K}(\mathbf{c}_i)$=$\mathrm{G}_{\mathrm{Kernel}}(\mathbf{c}_i) \in \mathbb{R}^{\mathrm{N}_{\mathrm{in}} \times \mathrm{N}_{\mathrm{out}}}$ (1a). The continuous parameterization of $\mathbf{K}$ allows the convolutional layer to *(i)* model long range dependencies, *(ii)* handle irregularly sampled data, and *(iii)* be used across different resolutions. Additionally, changing the dimensionality of the coordinates $\mathbf{c}_i$ can be used to construct convolutional kernels for sequential (1b), visual (1c), and higher dimensional data (1d) with the same kernel generator network.

The need for continuous parameterizations. Discrete convolutional kernels are defined with $\mathrm{N}_{\mathrm{out}} \times \mathrm{N}_{\mathrm{in}}$ independent learnable weights at each kernel position. Hence, large convolutional kernels require a large number of parameters and conventional CNNs rely on local convolutional kernels in combination with task-dependent depth values and pooling layers in order to model long range dependencies. Alternatively, we can construct *continuous* convolutional kernels through use of a small neural network that maps positions to the value of the kernel at those positions (Romero et al. (2022b), Fig. 1a). This approach decouples the size of the convolutional kernel from the number of parameters required to construct it, thus allowing the construction of arbitrary long kernels in a parameter efficient manner. Moreover, this parameterization overcomes the discrete nature of standard kernels and allows for the construction of resolution agnostic convolutional kernels that operate on coordinates of arbitrary resolution. Consequently, the same kernel generator network –and thus the same CNN– can be used regardless of the input length and resolution. Furthermore, the same kernel generator network can be used to construct convolutional kernels for sequential D=1 (Fig. 1b), visual D=2 (Fig. 1c) and higher dimensional tasks D≥3 (Fig. 1d) simply by changing the dimensionality of the input coordinates. In summary, the properties of *Continuous Convolutional Kernels* allow for the construction of a single CNN architecture that can be used across data lengths, resolutions and dimensionalities.

Contributions.

- We present the *Continuous CNN* (CCNN): a simple, general purpose CNN that can be used across data resolutions and dimensionalities without structural modifications. Our CCNN matches and often surpasses the state-of-the-art on several sequential (1D) and visual tasks (2D), as well as on tasks with irregularly-sampled data and test-time resolution changes.
- To this end, we provide several improvements for existing continuous CNN methods (Romero et al., 2022b;a) that allow them to match current state-of-the-art methods, e.g., S4 (Gu et al., 2022). Our improvements include changes to the initialization of the kernel generator networks, and modifications to the convolutional layers and the overall structure of the CNN.

2. Continuous Kernel Convolutions

Continuous kernel convolutions (Romero et al., 2022b) parameterize convolutional kernels as continuous functions by using a small neural network $\mathrm{G}_{\mathrm{Kernel}} : \mathbb{R}^{\mathrm{D}} \rightarrow \mathbb{R}^{\mathrm{N}_{\mathrm{out}} \times \mathrm{N}_{\mathrm{in}}}$ as a kernel generator network. This network maps a coordinate $\mathbf{c}_i \in \mathbb{R}^{\mathrm{D}}$ to the value of the convolutional kernel at that position: $\mathrm{G}_{\mathrm{Kernel}}(\mathbf{c}_i) \in \mathbb{R}^{\mathrm{N}_{\mathrm{out}} \times \mathrm{N}_{\mathrm{in}}}$ (Fig. 1a). By passing a vector of K coordinates $[\mathbf{c}_i]_{i \in [1,\dots,\mathrm{K}]}$ through $\mathrm{G}_{\mathrm{Kernel}}$, a convolutional kernel $\mathbf{K}$ of equal size can be constructed, i.e., $\mathbf{K}$=$[\mathrm{G}_{\mathrm{Kernel}}(\mathbf{c}_i)]_{i \in [1,\dots,\mathrm{K}]}$. Subsequently, a convolution operation takes place between an input signal $\mathbf{x} : \mathbb{R}^{\mathrm{D}} \rightarrow \mathbb{R}^{\mathrm{N}_{\mathrm{in}}}$ and the generated convolutional kernel $\mathbf{K} : \mathbb{R}^{\mathrm{D}} \rightarrow \mathbb{R}^{\mathrm{N}_{\mathrm{out}} \times \mathrm{N}_{\mathrm{in}}}$ to construct an output feature representation $\mathbf{y} : \mathbb{R}^{\mathrm{D}} \rightarrow \mathbb{R}^{\mathrm{N}_{\mathrm{out}}}$. That is: $\mathbf{y}$=$\mathrm{ConvNd}(\mathbf{K}, \mathbf{x})$.

2.1. Properties

A general operation for arbitrary data dimensionalities. By changing the dimensionality D of the input coordinates $\mathbf{c}_i$, the kernel generator network $\mathrm{G}_{\mathrm{Kernel}}$ can be used to construct convolutional kernels of arbitrary dimensionality. Consequently, the same operation can be used to process sequential D=1, visual D=2 and higher dimensional data D≥3.

Parameter and computation efficient modelling of long range dependencies at every layer. We can use the kernel generator network $\mathrm{G}_{\mathrm{Kernel}}$ to construct convolutional kernels as big as the input signal in order to model long range dependencies at every layer, i.e., $\mathbf{K}$=$[\mathrm{G}_{\mathrm{Kernel}}(\mathbf{c}_i)]_{i \in [1,\dots,\mathrm{K}]}$; K=len($\mathbf{x}$). The number of parameters in $\mathrm{G}_{\mathrm{Kernel}}$ is independent from the length of the convolutional kernel, and thus kernels of arbitrary size can be constructed under a fixed parameter count. Convolutions with large convolutional kernels can be efficiently computed using the *convolution theorem*, which states that a convolution in the time domain equals a pointwise product in the frequency domain.

Irregularly-sampled data. For some applications, the input $\mathbf{x}$ may not be on a regular grid, e.g., medical data. Discrete convolutional kernels are ill-suited for such applications as their value is only known at some preset positions and not for arbitrary coordinates $\mathbf{c}_i$. Contrarily, continuous kernels are defined everywhere and thus can handle irregular data natively.

Equivalent responses across input resolutions. If the input signal $\mathbf{x}$ undergoes a resolution change, e.g., audio initially observed at 8KHz is now observed at 16KHz, convolving with a discrete convolutional kernel would yield different responses, as the kernel would cover a different subset of the input at each resolution. On the other hand, continuous kernels are resolution agnostic, and thus able to recognize an input regardless of its resolution. When presenting an input at a different resolution, e.g., higher resolution, it is sufficient to pass a finer grid of coordinates through the kernel generator network in order to construct the same kernel at the corresponding resolution. For a signal $\mathbf{x}$ and a continuous convolutional kernel $\mathbf{K}$ sampled at resolutions $\mathrm{r}^{(1)}$ and $\mathrm{r}^{(2)}$, the convolution at both resolutions are approximately equal up to a factor proportional to the resolution change (Romero et al., 2022b):

$$\mathrm{ConvNd}\left(\mathbf{K}_{\mathrm{r}^{(2)}}, \mathbf{x}_{\mathrm{r}^{(2)}}\right) \approx \left(\mathrm{r}^{(1)}/\mathrm{r}^{(2)}\right)^{\mathrm{D}} \mathrm{ConvNd}\left(\mathbf{K}_{\mathrm{r}^{(1)}}, \mathbf{x}_{\mathrm{r}^{(1)}}\right). \tag{1}$$

Learning of hyperparameters. A promising property of continuous kernels is that they enable the learning of parameters that must otherwise be treated as hyperparameters in CNNs with discrete kernels. FlexConvs (Romero et al., 2022a), for instance, define their convolutional kernels as the product of a kernel generator network $\mathrm{G}_{\mathrm{kernel}}$ and a trimmed Gaussian mask $\mathrm{Gauss}_{\mu,\sigma}$ with learnable parameters μ, σ, i.e., $\mathbf{K}(\mathbf{c}_i)$=$\mathrm{G}_{\mathrm{kernel}}(\mathbf{c}_i) \cdot \mathrm{Gauss}_{\mu,\sigma}(\mathbf{c}_i)$. The Gaussian mask defines the size of the kernel and thus, by learning the mask, one can effectively learn the size of the convolutional kernel during training. In concurrent work we observe that a similar strategy can be used to additionally learn the depth and width of neural networks.

3. The Continuous Convolutional Neural Network: Modelling Long Range Dependencies in ND

An improved residual block with continuous kernel convolutions. Recent works have shown that residual blocks (He et al., 2016) can be strongly improved by changing the nonlinearities used and the position of the normalization layers within the blocks (Xiong et al., 2020; Liu et al., 2022). Based on these observations, we modify the FlexNet architecture (Romero et al., 2022a) with a residual network composed of blocks similar to those of S4 networks (Gu et al., 2022). The CCNN architecture is shown in Figure 2.

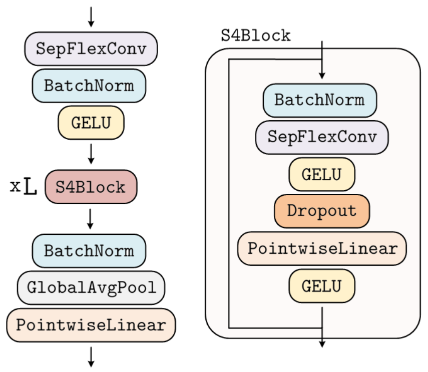


Figure 2. The Continuous CNN architecture.

Depthwise separable continuous kernel convolutions. Separable convolutions have long been shown to improve the parameter and computational efficiency of CNNs (Rigamonti et al., 2013; Sifre & Mallat, 2014). More recently, their usage has shown improvement over conventional convolutions in CNNs (Chollet, 2017; Tan & Le, 2019; Knigge et al., 2021; Liu et al., 2022), due to the separation of spatial and channel dimensions, which reduces the computational and parameter complexity of the convolution and allows for wider networks and higher performance.

Based on these observations we construct a depth-wise separable version of FlexConv (Romero et al., 2022a), in which a channel-wise convolution is computed with a kernel generated by a kernel generator network $\mathrm{G}_{\mathrm{Kernel}} : \mathbb{R}^{\mathrm{D}} \rightarrow \mathbb{R}^{\mathrm{N}_{\mathrm{in}}}$, followed by a point-wise convolution from N_{in} to $\mathrm{N}_{\mathrm{out}}$. This change allows for the construction of much a wider CCNN –from 30 to 110 hidden channels– without increasing the parameter or computation complexity of the network.

Proper initialization of the kernel generator network $\mathrm{G}_{\mathrm{Kernel}}$. We observe that the kernel generator networks in previous works are not properly initialized for their purpose of parameterizing convolutional kernels (Schütt et al., 2017; Wu et al., 2019). Upon initialization, one would like the variance of the input and the output of a convolutional layer to remain equal to avoid exploding and vanishing gradients, i.e., $\mathrm{Var}(\mathbf{x})$=$\mathrm{Var}(\mathbf{y})$. As such, convolutional kernels are initialized to have variance $\mathrm{Var}(\mathbf{K})$=$\texttt{gain}^2/(\texttt{in\_channels} \cdot \texttt{kernel\_size})$, with a $\texttt{gain}$ that depends on the nonlinearity used (He et al., 2015). Nevertheless, neural networks are initialized such that the unitary variance of the input is preserved at the output. Consequently, when used as a kernel generator network, a standard initialization method leads the kernel to have unitary variance, i.e., $\mathrm{Var}(\mathbf{K})$=1. As a result, CNNs using neural networks as kernel generator networks experience a layer-wise growth in the variance of the feature representations proportional to $\texttt{in\_channels} \cdot \texttt{kernel\_size}$. For example, we observe that the logits of CKCNNs (Romero et al., 2022b) and FlexNets (Romero et al., 2022a) lie in the order of $1\mathrm{e}^{19}$ upon initialization. This is undesirable as it can lead to unstable training and the need for low learning rates.

Table 1. Pixel-level 1D image classifcation.

	SIZE	SMNIST	PMNIST	SCIFAR10
LSTM	70K	87.2	85.7	-
GRU	70K	96.2	87.3	-
IndRNN	83K	99.0	96.0	-
DilRNN	44K	98.0	96.1	-
HiPPO-RNN	500K	-	98.30	-
r-LSTM	500K	98.4	95.2	72.2
TCN	70K	99.0	97.2	-
TrellisNet	8M	99.20	98.13	73.42
Transformer	500K	98.9	97.9	62.2
LSSL	7.8M	99.53	**98.76**	84.65
S4	7.8M	**99.63**	98.70	**91.13**
CKCNN	98K	99.31	98.00	62.25
CKCNN-Big	1M	99.32	98.54	63.74
FlexTCN-6	375K	99.62	98.63	80.82
$\text{CCNN}_{4,110}$	200K	$\underline{\mathbf{99.72}}$	**98.82**	**90.30**
$\text{CCNN}_{6,380}$	2M	$\underline{\mathbf{99.72}}$	$\underline{\mathbf{98.84}}$	$\underline{\mathbf{93.08}}$

Table 2. Speech classification.

	SIZE	MFCC	RAW	0.5X
Transformer	500K	90.75	-	-
Performer		80.85	30.77	30.68
ODE-RNN		65.90	-	-
NRDE		89.80	16.49	15.12
ExpRNN		82.13	11.60	10.80
LipschitzRNN		88.38	-	-
LSSL	300K	93.58	-	-
S4	300K	93.96	**98.32**	**96.30**
WaveGAN-D	26.3M	-	96.25	-
CKConv	105K	95.30	71.66	65.96
FlexTCN-6	373K	**97.67**	91.73	-
$\text{CCNN}_{4,110}$	200K	**95.01**	**98.34**	**96.22**
$\text{CCNN}_{6,380}$	2M	$\underline{\mathbf{97.98}}$	$\underline{\mathbf{98.44}}$	$\underline{\mathbf{96.44}}$

Table 3. 2D image classification.

	SIZE	CIFAR10	CIFAR100	STL10
ResNet-44	660K	92.90	71.15	-
ResNet-18	11.2M	**94.92**	$\underline{\mathbf{77.50}}$	**81.04**
Parabolic CNN	502K	88.5	64.8	77.0
Hamiltonian CNN	264K	89.3	64.9	78.3
CKConv	630K	86.8	-	-
FlexNet-6	670K	92.2	-	-
$\text{CCNN}_{4,110}$	200K	**92.78**	66.86	**81.80**
$\text{CCNN}_{6,380}$	2M	$\underline{\mathbf{95.20}}$	**73.16**	$\underline{\mathbf{83.00}}$

2D LONG RANGE ARENA TASKS		
	2DIMAGE	2DPATHFINDER
$\text{CCNN}_{4,110}$	**89.48**	**94.80**
$\text{CCNN}_{6,380}$	$\underline{\mathbf{91.12}}$	$\underline{\mathbf{96.00}}$

Table 4. Long Range Arena.

	SIZE	LISTOPS	TEXT	RETRIEVAL	IMAGE	PATHFINDER	PATH-X	AVG.
Transformer	0.5M	36.37	64.27	57.46	42.44	71.40	-	53.66
Reformer		37.27	56.10	53.40	38.07	68.50	-	50.56
BigBird		36.05	64.02	59.29	40.83	74.87	-	57.17
Linear Trans.		16.13	65.90	53.09	42.34	75.30	-	50.46
Performer		18.01	65.40	53.82	42.77	77.05	-	51.18
FNet		35.33	65.11	59.61	38.67	77.80	-	54.52
Nystromförmer		37.15	65.52	79.56	41.58	70.94	-	57.46
Luna-256		37.25	64.57	79.29	47.38	77.72	-	59.37
S4		$\underline{\mathbf{58.35}}$	**76.02**	$\underline{\mathbf{87.09}}$	**87.26**	**86.05**	$\underline{\mathbf{88.10}}$	$\underline{\mathbf{80.48}}$
$\text{CCNN}_{4,110}$	200K	**44.85**	**83.59**	-	**87.62**	**91.36**	-	76.86
$\text{CCNN}_{6,380}$	2M	**43.60**	$\underline{\mathbf{84.08}}$	-	$\underline{\mathbf{88.90}}$	$\underline{\mathbf{91.51}}$	-	**77.02**

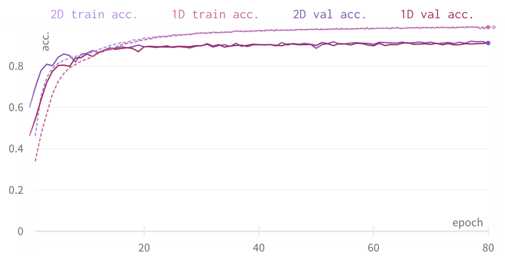


Figure 3. 1D vs. 2D image classification.

To solve this problem, we require that the variance at the output of $\mathrm{G}_{\mathrm{Kernel}}$ equals $\texttt{gain}^2/(\texttt{in\_channels}\cdot\texttt{kernel\_size})$ and not 1. To this end and inspired by Chang et al. (2020), we re-weight the last layer of the kernel generator network by $\texttt{gain}/\sqrt{\texttt{in\_channels}\cdot\texttt{kernel\_size}}$. As a result, the variance at the output of the kernel generator network follows the initialization of conventional convolutional kernels, and the logits of CCNNs present unitary variance upon initialization.

4. Experiments and discussion

Our goal is to construct a single model that can be applied to data of arbitrary length, resolution and dimensionality. We construct two CCNNs of different sizes: $\text{CCNN}_{4,110}$ (4 blocks, 110 channels) and $\text{CCNN}_{6,380}$ (6 blocks, 380 channels) and validate them on several sequential (1D) and visual (2D) benchmark datasets. For sequential data, we consider 1D pixel-level image classification (Le et al., 2015; Chang et al., 2017), speech classification (Warden, 2018) and the Long Range Arena (LRA) (Tay et al., 2021) benchmark, which evaluates the capacity of models to describe long range dependencies. For visual data, we consider 2D classification of images of different size (Krizhevsky et al., 2009; Coates et al., 2011). A complete description of the datasets used is given in Appx. A. The hyperparameters used in our experiments as well as additional descriptions of our models are reported in Appx. B.[1]

Results. As shown in Tabs. 1-4, our CCNN models perform well across all tasks considered. In fact, CCNNs set a new state of the art on multiple LRA tasks as well as 1D CIFAR10 pixel classification and raw speech classification on Speech Commands, while often being (much) smaller than competitive approaches.

The importance of modelling long range dependencies on ND. In principle, we could consider all tasks as a sequential task in which no 2D structure is considered. This is done, for instance with S4 (Gu et al., 2022) due to the complexity of defining state spaces in multidimensional spaces. Nevertheless, this comes at the cost of throwing away important information regarding the nature of the data. Contrarily, CCNNs can be easily defined on multidimensional spaces simply by changing the dimensionality of the coordinates going into the kernel generator networks. Interestingly, we observe that by considering the 2D nature of the Image and Pathfinder tasks in the LRA benchmark, much better results can be obtained (Tab. 3). In PathFinder with 2D images, our largest CCNN obtains an accuracy of 96.00 outperforming the previous state of the art by a margin of almost 10% points and performing remarkably better than the CCNN on flattened images. In addition, we observe that models trained on the original 2D data show faster convergence than their sequential counterparts (Fig. 3).

We note that 2D CNNs with small convolutional kernels, e.g., ResNet-18, were unable to solve Pathfinder due to the lack of fine-grained global context modelling resulting from intermediate pooling layers. This was also seen by Gu et al. (2020).

[1]Our code is publicly available at `github.com/david-knigge/ccnn`

5. Conclusion

We propose the *Continuous Convolutional Neural Network*: a single CNN architecture able to model long range dependencies on data of arbitrary length, resolution and dimensionality. Key to this development is the replacement of discrete convolutional kernels used in standard CNNs with Continuous Convolutional Kernels. With a single architecture, our CCNN performs competitively and often outperforms the current state of the art across a variety of sequential and visual tasks.

Acknowledgements

This work is co-supported by the Qualcomm Innovation Fellowship granted to David W. Romero. David W. Romero sincerely thanks Qualcomm for his support. David W. Romero is financed as part of the Efficient Deep Learning (EDL) programme (grant number P16-25), partly funded by the Dutch Research Council (NWO). David Knigge is partially funded by Elekta Oncology Systems AB and a RVO public-private partnership grant (PPS2102).

This work was carried out on the Dutch national infrastructure with the support of SURF Cooperative.

References

Abdel-Hamid, O., Mohamed, A.-r., Jiang, H., Deng, L., Penn, G., and Yu, D. Convolutional neural networks for speech recognition. *IEEE/ACM Transactions on audio, speech, and language processing*, 22(10):1533–1545, 2014.

Bai, S., Kolter, J. Z., and Koltun, V. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*, 2018.

Biewald, L. Experiment tracking with weights and biases, 2020. URL `https://www.wandb.com/`. Software available from wandb.com.

Chang, O., Flokas, L., and Lipson, H. Principled weight initialization for hypernetworks. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=H1lma24tPB`.

Chang, S., Zhang, Y., Han, W., Yu, M., Guo, X., Tan, W., Cui, X., Witbrock, M., Hasegawa-Johnson, M. A., and Huang, T. S. Dilated recurrent neural networks. *Advances in neural information processing systems*, 30, 2017.

Chollet, F. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1251–1258, 2017.

Coates, A., Ng, A., and Lee, H. An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pp. 215–223. JMLR Workshop and Conference Proceedings, 2011.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Falcon et al., W. Pytorch lightning. *GitHub. Note: https://github.com/PyTorchLightning/pytorch-lightning*, 3, 2019.

Fathony, R., Sahu, A. K., Willmott, D., and Kolter, J. Z. Multiplicative filter networks. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=OmtmcPkkhT`.

Gu, A., Dao, T., Ermon, S., Rudra, A., and Ré, C. Hippo: Recurrent memory with optimal polynomial projections. *Advances in Neural Information Processing Systems*, 33:1474–1487, 2020.

Gu, A., Goel, K., and Re, C. Efficiently modeling long sequences with structured state spaces. In *International Conference on Learning Representations*, 2022. URL `https://openreview.net/forum?id=uYLFoz1vlAC`.

He, K., Zhang, X., Ren, S., and Sun, J. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034, 2015.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Kidger, P., Morrill, J., Foster, J., and Lyons, T. Neural controlled differential equations for irregular time series. *Advances in Neural Information Processing Systems*, 33:6696–6707, 2020.

Knigge, D. M., Romero, D. W., and Bekkers, E. J. Exploiting redundancy: Separable group convolutional networks on lie groups. *arXiv preprint arXiv:2110.13059*, 2021.

Krizhevsky, A., Hinton, G., et al. Learning multiple layers of features from tiny images. 2009.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

Le, Q. V., Jaitly, N., and Hinton, G. E. A simple way to initialize recurrent networks of rectified linear units. *arXiv preprint arXiv:1504.00941*, 2015.

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

Liu, Z., Mao, H., Wu, C.-Y., Feichtenhofer, C., Darrell, T., and Xie, S. A convnet for the 2020s. *arXiv preprint arXiv:2201.03545*, 2022.

Loshchilov, I. and Hutter, F. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

Loshchilov, I. and Hutter, F. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Oord, A. v. d., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., and Kavukcuoglu, K. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.

Qi, C. R., Su, H., Mo, K., and Guibas, L. J. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 652–660, 2017.

Rigamonti, R., Sironi, A., Lepetit, V., and Fua, P. Learning separable filters. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2754–2761, 2013.

Romero, D. W., Bruintjes, R.-J., Tomczak, J. M., Bekkers, E. J., Hoogendoorn, M., and van Gemert, J. Flexconv: Continuous kernel convolutions with differentiable kernel sizes. In *International Conference on Learning Representations*, 2022a. URL `https://openreview.net/forum?id=3jooF27-0Wy`.

Romero, D. W., Kuzina, A., Bekkers, E. J., Tomczak, J. M., and Hoogendoorn, M. CKConv: Continuous kernel convolution for sequential data. In *International Conference on Learning Representations*, 2022b. URL `https://openreview.net/forum?id=8FhxBtXSl0`.

Ronneberger, O., Fischer, P., and Brox, T. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pp. 234–241. Springer, 2015.

Schütt, K., Kindermans, P.-J., Sauceda Felix, H. E., Chmiela, S., Tkatchenko, A., and Müller, K.-R. Schnet: A continuous-filter convolutional neural network for modeling quantum interactions. *Advances in neural information processing systems*, 30, 2017.

Sifre, L. and Mallat, S. Rigid-motion scattering for texture classification. *arXiv preprint arXiv:1403.1687*, 2014.

Simonyan, K. and Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Sitzmann, V., Martel, J., Bergman, A., Lindell, D., and Wetzstein, G. Implicit neural representations with periodic activation functions. *Advances in Neural Information Processing Systems*, 33:7462–7473, 2020.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9, 2015.

Tan, M. and Le, Q. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pp. 6105–6114. PMLR, 2019.

Tay, Y., Dehghani, M., Abnar, S., Shen, Y., Bahri, D., Pham, P., Rao, J., Yang, L., Ruder, S., and Metzler, D. Long range arena : A benchmark for efficient transformers. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=qVyeW-grC2k.

Van Den Oord, A., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A. W., and Kavukcuoglu, K. Wavenet: A generative model for raw audio. *SSW*, 125:2, 2016.

Warden, P. Speech commands: A dataset for limited-vocabulary speech recognition. *arXiv preprint arXiv:1804.03209*, 2018.

Wu, W., Qi, Z., and Fuxin, L. Pointconv: Deep convolutional networks on 3d point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9621–9630, 2019.

Xiong, R., Yang, Y., He, D., Zheng, K., Zheng, S., Xing, C., Zhang, H., Lan, Y., Wang, L., and Liu, T. On layer normalization in the transformer architecture. In *International Conference on Machine Learning*, pp. 10524–10533. PMLR, 2020.

Yadan, O. Hydra - a framework for elegantly configuring complex applications. Github, 2019. URL https://github.com/facebookresearch/hydra.

Supplementary Material

Towards a General Purpose CNN for Long Range Dependencies in ND

A. Dataset description

Sequential and Permuted MNIST. The MNIST dataset (LeCun et al., 1998) consists of 70K gray-scale 28×28 handwritten digits divided into training validation and test sets of 60K and 10K samples, respectively. For validation purposes, the training dataset is further divided into training and validation sets of 55K and 5K samples, respectively.

The sequential MNIST dataset (sMNIST) presents MNIST images as a sequence of 784 pixels for digit classification. Consequently, good predictions require the model to preserve long-term dependencies up to 784 steps in the past. The permuted MNIST dataset (pMNIST) incorporates an additional level of difficulty by permuting the order of all sMNIST sequences with a random permutation. Resultantly, models can no longer rely on local information for the construction of their features and the importance of modelling long-term dependencies becomes more pronounced.

CIFAR10, CIFAR100 and Sequential CIFAR10. The CIFAR10 dataset (Krizhevsky et al., 2009) consists of 60K real-world 32×32 RGB images uniformly drawn from 10 classes divided into training and test sets of 50K and 10K samples, respectively. The CIFAR100 dataset (Krizhevsky et al., 2009) is similar to the CIFAR10 dataset, with the difference that the images are now uniformly drawn from 100 different classes. For validation purposes, the training dataset of both CIFAR10 and CIFAR100 are further divided into training and validation sets of 45K and 5K samples, respectively.

Analogously to the sMNIST dataset, the sequential CIFAR10 (sCIFAR10) dataset presents CIFAR10 images as a sequence of 1024 pixels for image classification. This dataset is more difficult than sMNIST, as *(i)* larger memory horizons are required to successfully solve the task, and *(ii)* more complex structures and intra-class variations are present in the images.

Speech Commands. The Speech Commands dataset (Warden, 2018) consists of 105809 one-second audio recordings of 35 spoken words sampled at 16kHz. Following Kidger et al. (2020), we extract 34975 recordings from ten spoken words to construct a balanced classification problem. We refer to this dataset as *Raw Speech Commands*. In addition, we use the preprocessing steps of Kidger et al. (2020) and extract mel-frequency cepstrum coefficients from the raw data. The resulting dataset, referred to as *MFCC Speech Commands*, consists of time series of length 161 and 20 channels.

Long Range Arena. The Long Range Arena benchmark (Tay et al., 2021) consists of 6 tasks with lengths 1K-16K steps encompassing modalities and objectives that require similarity, structural, and visuospatial reasoning. The `Pathfinder`, `Path-X` and `Image` tasks are similar in nature to the sMNIST and sCIFAR10 tasks. These tasks consists of classification tasks performed on images that are treated as sequences.

The `Image` task corresponds to the sequential CIFAR10 dataset with the only difference that the CIFAR10 images are treated as gray-scale images. The `Pathfinder` and `Path-X` tasks are binary tasks in which binary images are provided and the model must predict whether the two points in the images are connected with a line or not –see Fig. 4 for an example–. The difference between both datasets is their resolution. Whereas `Pathfinder` has images of size 32×32, `Path-X` has images of size 128×128. It is important to mention that these tasks are so difficult that even if treated as 2D signals, CNNs without global receptive fields are unable to solve them (Gu et al., 2022).

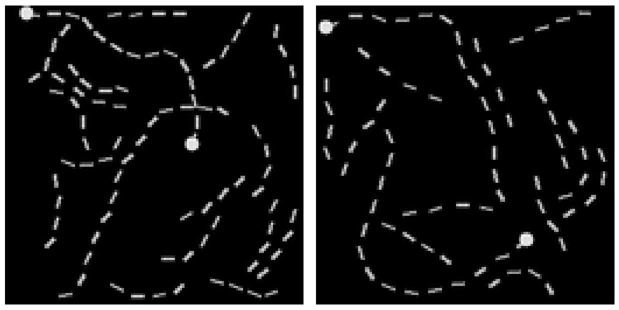


Figure 4. Positive and negative samples from the `Path-X` dataset

STL-10. The STL-10 dataset (Coates et al., 2011) is a subset of the ImageNet dataset (Krizhevsky et al., 2012) consisting of 13,000 96×96 real-world RGB images uniformly drawn from 10 classes divided into training and test sets of 5K and 8K images, respectively. For validation purposes, the training dataset is further divided into training and validation sets of 4,500 and 500 samples, respectively.

Table 5. Best hyperparameters found for CCNN$_{4,140}$ on all tasks considered.

	w_0	**Dropout**	**Learning Rate**	**Weight Decay**	**Batch Size**	**Epochs**
sMNIST	2976.49	0.1	0.01	1e-6	100	210
pMNIST	2985.63	0.2	0.02	0	100	210
sCIFAR10	2386.49	0.0	0.02	0	50	210
Speech Commands (raw)	1295.61	0.2	0.02	1e-6	20	160
Speech Commands (mfcc)	750.18	0.2	0.02	1e-6	100	110
ListOps	784.66	0.1	0.001	1e-6	50	60
Text	2966.60	0.2	0.001	1e-5	50	60
Image	4005.15	0.2	0.01	0	50	210
Pathfinder	2272.56	0.2	0.01	0	100	210
CIFAR10	1435.77	0.1	0.02	0.0001	50	210
CIFAR100	3521.55	0.1	0.02	0.0001	50	210
STL10	954.28	0.1	0.02	0	64	210
2DImage	2085.43	0.2	0.02	1e-6	50	210
2DPathfinder	1239.14	0.1	0.01	0	100	210

Table 6. Best hyperparameters found for CCNN$_{6,380}$ on all tasks considered.

	w_0	**Dropout**	**Learning Rate**	**Weight Decay**	**Batch Size**	**Epochs**
sMNIST	2976.49	0.1	0.01	0	100	210
pMNIST	2985.63	0.2	0.02	0	100	210
sCIFAR10	4005.15	0.25	0.01	0	50	210
Speech Commands (raw)	1295.61	0.2	0.02	1e-6	20	160
Speech Commands (mfcc)	750.18	0.2	0.02	1e-6	100	110
ListOps	784.66	0.25	0.001	0	50	60
Text	2966.60	0.3	0.02	0	50	60
Image	4005.15	0.1	0.01	0	50	210
Pathfinder	2272.56	0.1	0.01	1e-6	100	210
CIFAR10	1435.77	0.15	0.02	0	50	210
CIFAR100	679.14	0.2	0.02	0	50	210
STL10	954.28	0.1	0.01	1e-6	64	210
2DImage	2306.08	0.2	0.02	0	50	210
2DPathfinder	3908.32	0.2	0.01	0	100	210

B. Experimental details

B.1. General remarks

Code repository and logging. Our code is written in `PyTorch`. We utilize `wandb` (Biewald, 2020) `hydra` (Yadan, 2019) and `pytorch-lightning` (Falcon et al., 2019) for logging and code structuring. Our experiments are performed on NVIDIA TITAN RTX, A6000 and A100 GPUs, depending on the size of the datasets and inputs considered, and our code is publicly available at `github.com/david-knigge/ccnn`.

The kernel generator network $\mathrm{G}_{\mathrm{Kernel}}$. Our kernel generator network is parameterized as a 3-layer MAGNet (Romero et al., 2022a) with 32 hidden units for the CCNN$_{4,140}$ models, and 64 hidden units for the larger CCNN$_{6,380}$ models. The output size of the kernel generator network corresponds to the input channels of each layer in the network.

Normalized relative positions. The kernel generator network $\mathrm{G}_{\mathrm{Kernel}}$ can, in principle, receive arbitrary coordinates as input. However, considering unitary step-wise relative positions, i.e., 0, 1, 2, ... , N, can be problematic from a numerical stability perspective as N may grow very large, e.g., N=16000 for the Speech Commands dataset. Consequently, based on insights from the Implicit Neural Representations, e.g., Sitzmann et al. (2020); Fathony et al. (2021), we normalize the coordinates such that they lie in the space $[-1, 1]^{\mathrm{D}}$ for D-dimensional kernels. To this end, we map largest unitary positions seen during training $[0, N]$ to a uniform linear space in $[-1, 1]$.

B.2. Hyperparameters and training details

Optimizer and learning rate scheduler. All our models are optimized with AdamW (Loshchilov & Hutter, 2017) in combination with a cosine annealing learning rate scheduler (Loshchilov & Hutter, 2016) and a linear learning rate warm-up stage of 10 epochs.

Best hyperparameters found. We perform hyperparameter search on the learning rate, dropout rate, weight decay, and ω_0 of our CCNNs for each task considered.[2] The best hyperparameters found are reported in Tables 5 and 6.

[2] ω_0 serves as a prior on the variance of the data that is fitted with several types of implicit neural representations, e.g., SIRENs (Sitzmann et al., 2020), MFNs (Fathony et al., 2021), etc.