Draft of a paper latter accepted and published : Please cite as: R. Dogaru and I. Dogaru, "RD-CNN: A Compact and Efficient Convolutional Neural Net for Sound Classification," *2020 International Symposium on Electronics and Telecommunications (ISETC)*, Timisoara, 2020, pp. 1-4, doi: 10.1109/ISETC50328.2020.9301110.

RD-CNN: A Compact and Efficient Convolutional Neural Net for Sound Classification

Radu Dogaru
Dept. of Applied and Information Engineering
University "Politehnica" of Bucharest
Bucharest, Romania
radu.dogaru@upb.ro

Ioana Dogaru
Dept. of Applied and Information Engineering
University "Politehnica" of Bucharest
Bucharest, Romania
ioana.dogaru@upb.ro

***Abstract*— Classification of signals (sounds, biomedical, etc.) occurs in various circumstances, often requiring low-complexity implementations for various resources-constrained platforms such as mobile or embedded systems within the IoT framework. Herein we describe preliminary results for a novel recognition system where a compact and fast transform, the reaction-diffusion transform (RDT), is used to generate spectral images. Such images are then processed into a novel type of compact convolution neural network (called NL-CNN) where nonlinear convolution is emulated. The ESC-50 environmental sound database with 50 sound categories was considered for parameter tuning and performance evaluation. Despite its low complexity, a reasonable well accuracy is demonstrated (up to 76.5%) close to the human accuracy reported on the same dataset and better in both accuracy and complexity when compared to similar approaches reported in the literature and based on mel-cepstrum. These results provide a basis for a novel and effective method for time-series recognition in general.**

Keywords—machine learning, convolution neural network, signal recognition, spectrograms, resource-constrained systems.

I. INTRODUCTION

With the wide development of machine learning solutions for various signal problems two main problems arise: i) the improvement of the functional performance, often given as the "accuracy" i.e. the fraction of correctly recognized samples in a test set; ii) low-complexity and low-power implementation solutions, while keeping good accuracy (while accepting some sacrifice in performance when compared with the most accurate model). Our work reports preliminary results for a novel approach in signal recognition, with two original elements: i) a specific nonlinear transform, called RDT (reaction-diffusion transform) is used to transform signal sequences representing labeled instances of certain signals (they can have arbitrary size) into spectral images; ii) the use of a specially designed convolution neural network (denoted here NL-CNN, and explained in detail in Section II) with 3 macro-layers where non-linear convolution (NL) [15,16] is emulated using a cascade of *nl* readily available Keras/Tensorflow[1] layers (each level having a traditional (linear) convolution followed by a ReLu nonlinearity). The result is a low complexity (evaluated in terms of parameters and computing latency) classifier, yet capable to provide state of the art accuracy. Consequently, it replaces conveniently more traditional resource-constrained CNNs.

The RDT (reaction-diffusion transform) to be detailed next in Section III was first proposed in [1] and has a simple (multiplier-free) computation algorithm where a special role plays the absolute value nonlinearity. Basically it was inspired from a descriptor used to evaluate emergent dynamics in cellular automata [2]. Using variable delays (one delay for each "spectral" component) the RDT computes a sort of spectral value associated with each delay, although it is not based on the traditional (linear) Fourier spectrum definition. Given the use of delays in a series of power of two {1,2,4, …} a relatively compact spectral representation with up to 10 spectral lines (a *vector spectrogram*), yet containing the meaningful information for recognition, is obtained for signal segments with thousands of samples. Such vectors are then joined for all segments in the labeled signal instance and the result is an *spectral image* (also called image spectrogram) of the type shown in Fig. 3. One dimensional RDT combined with simple classifiers (SVM, ELM) was already tested with very good results in many signal recognition problems (voice [3][4][5], EEG in [6], and ultrasonic echoes [7]) and recently a functional implementation on a low complexity platform was demonstrated [5] achieving 100% recognition accuracy for the simple task of 6 vocal commands recognition of the same user. Besides, it is important to note the good latency for a resource-constrained platform (giving a RTF=real time factor of 0.36 i.e. a 1.6 seconds voice signal was recognized in 0.51 seconds) which allows reacting in real time to the audio commands. In this respect RDT outperforms traditional mel-cepstrum based methods such as [8] where RTF is close or larger than 1. For more complex sound recognition tasks, inspired from [9] where 2D mel-cepstrum spectrograms were submitted to a CNN, this work proposes the use of 2-dimensional spectral images (equivalent of image spectrograms) generated by RDT followed by a compact yet efficient NL-CNN (further described in Section II) as a signal recognizer. Among signal recognition tasks, sound recognition is an important one with many approaches under development [10]. Consequently, to demonstrate the RD-CNN performance, results are presented in Section IV for the ESC-50 dataset [11] our method gives 76.5% accuracy (and thus favorably compared with 64.5% reported by [9]). Conclusions and further research perspectives are given in

[1] https://keras.io/

Section V. Note that accuracy doubled when image spectrograms were used instead of simple 1D vector features, giving thus an interesting perspective for applying the proposed method to a wider variety of labeled signals and time-series.

II. The Classifier Arhitecture and its Implementation

The RD-CNN classifier architecture combines a reaction-diffusion transform (RDT) based processor with the proposed NL-CNN. The whole system was implemented using the Keras/Tensorflow libraries running on Google Colab[2] and its structure is depicted in Fig.1. While the RDT processor is detailed in the next section, here we focus on the NL-CNN architecture. It is a relatively light architecture which expands from work in [12]. It was previously tested with very good performance on various classic image datasets (for instance, it can achieve 90.9% accuracy on the well known CIFAR10 dataset, as can be verified in [17]). The NL-CNN is based on 3 macro-layers, each macro-layer being characterized by 4 specific structural hyper-parameters: *(nl,nf,c1,c2)* where *nl* is the macro-layer nonlinearity (for instance *nl1* refers to macro-layer 1 as detailed next), *nf* is the number of 2D convolution filters, while *c1* and *c2* represent the convolution kernel sizes on both directions of the image spectrogram to be processed by the NL-CNN). Directional asymmetric convolutions *(c1,c2)=(5,3)* instead *(c1,c2)=(3,5)* or symmetric *(c1,c2)=(3,3)* provided the best accuracy.

The *spectral images*, each with a size *(Ns,m)* are computed for each labeled signal instance as detailed in Fig.3 and training / testing batches are provided to the NL-CNN in the usual "channels-last" format 4-sized tensor. Several models and their evaluations are provided in [17]. The *global average pooling* layer is used to flatten the output from the last convolution macro-layer. Dropout (with a factor of 0.4) in convolution improves slightly the performance. Batch Normalization (BN) after each macro-layer gives also some light improvement. Without using them the performance reduces to about 70% on the ESC-50 set. Various other "consacrated" architectures (including ResNet56 and MobileNet, where tested instead NL-CNN, but with lower results on accuracy and much larger training times). It is likely that a sophisticated model employing 99% on CIFAR10 will also improve slightly the result here, but it was out of our scope, namely a resource-constrained solution.

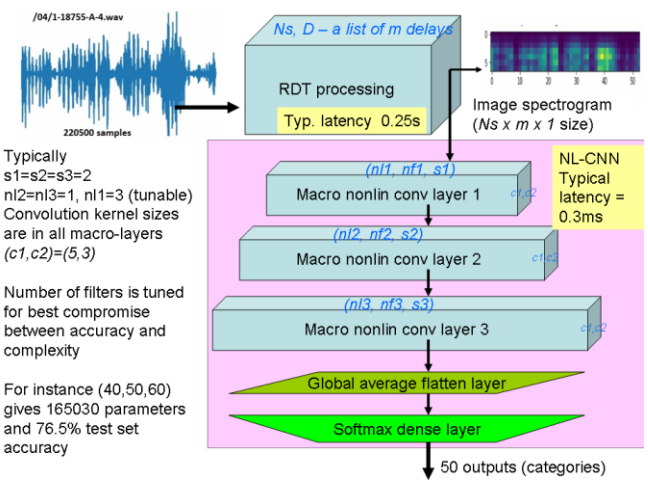

Fig. 1. The structure of the NL-CNN model and its use in the context of the RD-CNN signal classifier system.

The structure of the first macro-layer as plotted by the Keras utilities is shown in Fig.2, here for the case *nl1=3*.

After each convolution layer a nonlinear layer (typically *relu()*) is applied, thus emulating a form of nonlinear convolution [15][16] for the entire macro-layer. Our experiments with various datasets demonstrate that choosing *nl>1* (*nl1=3* for most cases, in the first macro-layer, *nl2=2* in the second and *nl3=1* in the output macro-layer) gives important accuracy improvements when compared to the simple case *nl=0* which is equivalent to the 3-layers L-CNN model in [12].

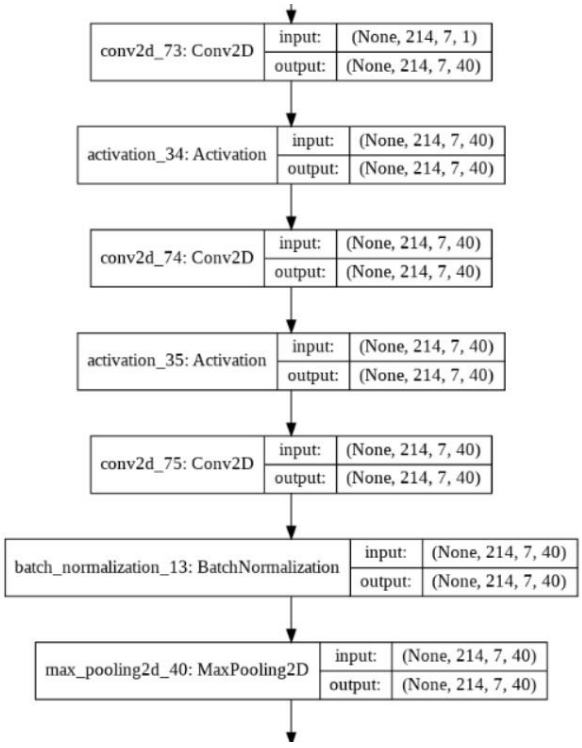

Fig. 2. The structure of a macro-layer emulating nonlinear convolution with the given number of filters (here *nf=40*). In this case the degree of nonlinearity is *nl=3*. The spectral image processed here is of (217,7) size.

Each macro-layer has a 2D-maxpool layer at the output, acting also as nonlinear layer (unlike average pooling, max

[2] https://colab.research.google.com/

provides nonlinearity), with stride *s=2* to reduce the image dimensionality in each macro-layer. The optimal pooling kernel size was selected *p=4* on both dimensions.

III. Spectral Images Generated by RDT

In order to introduce the RDT concept in a simple manner, Fig. 3 represents the most important steps and parameters of the RDT-module (depicted in Fig.1) which is used to associate an *image spectrogram* (or spectral image) to a given input signal sequence. In [17] code for the RDT based spectral images as well as processing examples are provided.

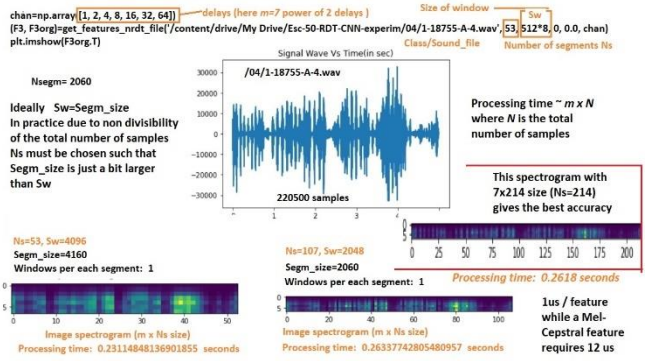

Fig. 3. RDT processor and its parameters to produce spectral images.

The *signal sequence* (provided here as a .wav file in a subdirectory associated to its category – 04 in this case), is divided into a given number *Ns* of *signal sequences*. In the above figure the Python function *get_features_nrdt()* has *Ns=53* as its parameter (leading to the leftmost spectral image). Given the total number of *N=220500* samples in the 5 seconds sound signal it follows that each segment (signal sequence) will have *N/Ns*= 4160 samples. Another parameter of the RDT processor is the window signal size *Sw*. According to the RDT model in [1] this is supposed to be a power of 2. For instance if *Sw=512* the integer number of windows fitting into a *signal segment* is *4160/512* = 8. The RDT transform is applied for each of these windows and the resulting *m-sized vectors* are averaged to obtain a single vector-spectrogram associated to the segment. In practice it was found that optimal performance is obtained when there is only 1 window signal per segment, consequently for a given choice of *Ns*, the window size *Sw* is computed to fulfill the above, with a window size smaller but almost equal (extra samples are not considered in computation) with the segment size. For the above example, *Ns=53 and Sw=512*8=4096* is one choice. But other choices are possible (as indicated by the several image spectrograms in Fig.3) these parameters need tuning to optimize the overall accuracy. Finally the third parameter of the RDT processor is *m* defining the set of delays $D = \{d_0, d_1, \dots d_{m-1}\}$ used in computing the *m* components of the RDT vector spectrogram associated to a window signal *s(t)*. In its original RDT approach [1] delays are powers of 2 i.e. $d_k = 2^k$ but since [6] we also consider arbitrary integer number *m* and values for delays provided that the maximal delay is no more than half of the *window signal* size. This arbitrary choice has the benefit of allowing a fine tuning of performance but on the other hand requires an intensive search among various possibilities. Consequently, here we used the original (power of two) approach where the parameter *m* is automatically computed as $m = \lfloor log_2(Sw) \rfloor - 3$. In the case with *Sw=4096* it follows that *m=9*. Given the *k-th* delay value, the associated spectral value is computed as follows:

$$S_k = \frac{1}{Sw-d_{k-1}} \sum_{i=d_k}^{Sw-d_{k-1}} |s(t-d_k) + s(t+d_k) - 2s(t)| \quad (1)$$

Note that there are only simple operators and computations with a very good degree of parallelism (one can compute independently spectral values per each *k* and given segment in the signal) making this operator suitable mostly for FPGA implementations in hybrid embedded systems such as PynQ-Z1[3] when low power is desired. Herein the implementation based on the NUMPY library is not optimized for speed and, as seen in Fig. 1, gives the dominant latency in the entire system. However, when compared to the time needed to compute a mel-cepstrum (MFC) feature on the same platform, the speed was about 12 times better for the RDT approach.

IV. Experimental Results

The RDT functions to generate image spectrograms and their associated datasets as well as the NL-CNN implementation and training were implemented as notebooks and their run was done using the Google COLAB (GPU runtime). Herein the dataset ESC-50 [11] was used, stored in Google Drive in a specific folder where sub-folders for each of the 50 classes. Image-spectrogram train and test datasets for various choices of the RDT parameters were initially generated as .mat files such that 1600 samples were used for training and the remaining 400 for testing (randomly chosen for each class) and then NL-CNN networks were optimized for improved accuracy and small number of parameters. The training setup was: *Adam()* optimizer, batch-size of 5 and 300 epochs training (with an average of 1-2 seconds per training epoch on the GPU runtime in the Google Colab).

A. Parameter optimization

Model parameters, accuracy and latencies for several models and RDT setups are summarized in Table I where *Ns and m* are the sizes of the spectral images.

TABLE I. Parameter optimization for the RDT transform and NL-CNN in order to maximize accuracy with a small number of parameters

RDT spectral images parameters *(Ns,m)*	***NL-CNN parameters Nf=(nf1,nf2,nf3) (c1,c2)=(3,3) if not other specified***	***NL-CNN test accuracy (%)***	***Latency for prediction (ms)***
(214,7)	165030 Nf=(40,50,60) (c1,c2)=(5,3)	76.5	0.36 (11 on CPU)
(214,7)	100590 Nf=(40,50,60)	72.75	0.15
(53,9)	85550 Nf=(40,50,60)	67.75	0.085
(53,9)	12610 Nf=(16,16,16)	55	0.076
(53,9)	40434 Nf=(16,32,64) No Batch Norm.	63.75	0.082
(214,7)	74950	22	0.05

[3] https://reference.digilentinc.com/reference/programmable-logic/pynq-z1/start

RDT spectral images parameters (Ns,m)	*NL-CNN parameters Nf=(nf1,nf2,nf3) (c1,c2)=(3,3) if not other specified*	*NL-CNN test accuracy (%)*	*Latency for prediction (ms)*
	Single output layer, no 2D convolutions (Adaline) No BN or drop		

From the above table it follows even for small number of parameters accuracies larger than 60% can be achieved while the best result (76.5%) was obtained for the setup in the first line. For reference, a simple output layer applied to the spectral image gives only 22%. This result indicates that the RDT feature itself without using a proper CNN does not provide state of the art accuracy for such complex problems. On the other hand, similar problems were reported in [11] using a more sophisticated (Random Forest) approach applied to the feature vector.

B. Comparison with other reported solutions

In order to compare our result with other solutions, a good starting point is the github[4] of ESC-50 author [11], where state of the art results on the ESC-50 dataset are reported. As indicated in [11] no more than 44% accuracy is obtained with the baseline simplest model combining mel-cepstrum analysis with Random Forest classifiers (somehow confirmed by our last line in Table I). On the other hand, the human accuracy reported on this dataset [11] is 81.3% just a bit higher, when compared to our result. Unfortunately most of the solutions in [11] give little or no information about the computational complexity, making thus difficult a comparison with regards to this important aspect in our scope. The best result reported so far [13] has 9.75% better accuracy than ours but requires a relatively complex solution.

TABLE II. MODELS TO IDENTIFIY THE MOST EFFICIENT MNIST LB-CNN MODEL

Model	**Accuracy (%)**	**Complexity (latency)**	**Notes**
Our best	**76.5**	Around 0.3 s	160k param. CNN
[11] baseline	44.3%	N.A.	Random Forest + Mel Cepstral
Human accuracy [11]	81.3%	---	
Best result so far [13]	86.5%	N.A.	
CNN Mel-spectrograms baseline [9]	64.50%	60x41 image spectrograms	Similar approach (spectrograms+CNN)
Similar performance [14]	76	N.A. Likely more complex than ours	10 layers CNN with up to 256 filters

The result in [9] where a very similar approach was used (2D spectrograms followed by a CNN) is worse than ours. It indicates that the use of low-complexity RDT instead of mel-cepstrum is a valid alternative with a lot of potential in signal recognition.

V. CONCLUSIONS

A low complexity signal recognition system, RDT-CNN, is proposed, and preliminary results in operating it on the ESC-50 dataset [11] are reported. Our preliminary results indicates that given the reduced complexity (latency of the RDT processing is around 0.27 seconds on a CPU runtime while the latency of the best NL-CNN model is 0.011 seconds on CPU), the accuracy of 76.5% represents a good result for a resources-constrained environment. The above times usually multiply by 10 on such platforms indicating that such a recognition system is capable to react in real time (under 3 seconds) for signals lasting 5 seconds. It s achieved in a system combining RDT processors (previously used with good results in a variety o tasks when combined with classic machine learning techniques) with a particular kind of CNN denoted here NL-CNN, thus approaching the 81.3% human recognition accuracy. The dominant latency is given by the RDT spectral image computations, but its value is reasonably low and can be further improved. The use of convolutional neural networks in recognizing spectral images was found important, similarly to [9], in addition to the very low complexity spectral transform (found to be 12 times faster than mel-cepstrum on the same platform). Such hybrid approaches (RDT + CNN) reveal important improvements in such signal processing tasks and consequently we expect very good results in applying this technique to other classes of signals (e.g. those considered in [3-7]). For CNN, our choice was NL-CNN ensuring good performance (as compared to more traditional CNN) with a relatively light structure. While efficient implementations solutions for resource constrained systems exists (e.g. in the framework of TFLITE libraries) for Keras/Tensorflow models (as the resulted ones) further investigation on how the RDT processor can be implemented (for instance, using the Pl FPGA array in PynQ-Z1 boards) may lead to dramatic reduction of latencies thus making this solution widely attractive for any signal recognition task to be performed in resources-constrained platforms.

Many aspects are to be further investigated, particularly the latency evaluation of solutions reporting similar performance and employing techniques (e.g. augmentation or other as proposed in [13], additional tuning of hyperparameters) to improve the performance while keeping implementation complexity minimal.

REFERENCES

[1] R. Dogaru, "Fast and Efficient Speech Signal Classification with a Novel Nonlinear Transform," 2007 International Symposium on Information Technology Convergence (ISITC 2007), Joenju, 2007, pp. 43-47

[2] R. Dogaru, Systematic design for emergence in cellular nonlinear networks – with applications in natural computing and signal processing, Springer-Verlag, Berlin Heidelberg, 2008.

[3] R. Dogaru, "A Low Complexity Algorithm for Isolated Sound Recognition using Neural Networks," 2007 International Symposium on Signals, Circuits and Systems, Iasi, 2007, pp. 1-4

[4] M. Bucurica, I. Dogaru and R. Dogaru, "Novel Applications of Complexity Inspired RDT Transform for Low Complexity Embedded Speech Recognition in Automotive Environments," 2017 21st International Conference on Control Systems and Computer Science (CSCS), Bucharest, 2017, pp. 375-378.

[5] I. Dogaru, D. Stan and R. Dogaru, "Compact Isolated Speech Recognition on Raspberry-Pi based on Reaction Diffusion Transform," 2019 6th International Symposium on Electrical and Electronics Engineering (ISEEE), Galati, Romania, 2019, pp. 1-4.

[6] R. Dogaru and I. Dogaru, "A low complexity solution for epilepsy detection using an improved version of the reaction-diffusion transform," 2017 5th International Symposium on Electrical and Electronics Engineering (ISEEE), Galati, 2017, pp. 1-6.

[4] https://github.com/karolpiczak/ESC-50

[7] M. Bucurica, I. Dogaru and R. Dogaru, "A low complexity method for analyzing acoustic echo signals in the framework of a cellular automata virtual environment", U.P.B. Sci. Bull., Series C, Vol. 80, Iss. 2, 2018 ISSN 2286-3540.

[8] M. Walid, S. Bousselmi, K. Dabbabi and A. Cherif, "Real-Time Implementation of Isolated-Word Speech Recognition System on Raspberry Pi 3 Using WAT-MFCC ", in IJCSNS International Journal of Computer Science and Network Security, VOL.19 No.3, March 2019.

[9] K. J. Piczak, "Environmental sound classification with convolutional neural networks," 2015 IEEE 25th International Workshop on Machine Learning for Signal Processing (MLSP), Boston, MA, 2015, pp. 1-6

[10] H. Purwins, B. Li, T. Virtanen, J. Schlüter, S. Chang and T. Sainath, "Deep Learning for Audio Signal Processing," in IEEE Journal of Selected Topics in Signal Processing, vol. 13, no. 2, pp. 206-219, May 2019

[11] K. J. Piczak, "ESC: Dataset for Environmental Sound Classification", Proc. 23rd ACM Int. Conf. Multimedia, pp. 1015-1018, Oct. 2015. https://github.com/karolpiczak/ESC-50

[12] R. Dogaru and I. Dogaru, "BCONV - ELM: Binary Weights Convolutional Neural Network Simulator based on Keras/Tensorflow, for Low Complexity Implementations," 2019 6th International Symposium on Electrical and Electronics Engineering (ISEEE), Galati, Romania, 2019, pp. 1-6

[13] H. B. Sailor, D. M. Agrawal and H. A. Patil, "Unsupervised filterbank learning using convolutional restricted Boltzmann machine for environmental sound classification", Proc. INTERSPEECH, pp. 3107-3111, Aug. 2017.

[14] Y. Tokozume, Y. Ushiku, and T. Harada, “Learning from between-class examples for deep sound recognition,” in Proc. Int. Conf. Learn. Representations, 2018.

[15] A. Farina, A. Bellini and E. Armelloni, "Non-Linear convolution: a new approach for the auralization of distorting systems", in Audio Engineering Society 110'th Convention, May 2001, Amsterdam.

[16] G. Zoumpourlis, A. Doumanoglou, N. Vretos, and P. Daras., “Non-linear convolution filters for CNN-based learning”. In ICCV, 2017.

[17] R. Dogaru and I. Dogaru, Github support for this paper: models and a notebook to evaluate them: https://github.com/radu-dogaru/NL-CNN-RDT-based-sound-classification- 2020.