NEUROEVOLUTION FOR SOUND EVENT DETECTION IN REAL LIFE AUDIO: A PILOT STUDY

Christian Kroos and Mark D. Plumbley

Centre for Vision, Speech and Signal Processing (CVSSP)
University of Surrey
Guildford, Surrey, GU2 7XH, UK
{c.kroos, m.plumbley}@surrey.ac.uk

ABSTRACT

Neuroevolution techniques combine genetic algorithms with artificial neural networks, some of them evolving network topology along with the network weights. One of these latter techniques is the NeuroEvolution of Augmenting Topologies (NEAT) algorithm. For this pilot study we devised an extended variant (joint NEAT, J-NEAT), introducing dynamic cooperative co-evolution, and applied it to sound event detection in real life audio (Task 3) in the DCASE 2017 challenge. Our research question was whether small networks could be evolved that would be able to compete with the much larger networks now typical for classification and detection tasks. We used the wavelet-based deep scattering transform and k-means clustering across the resulting scales (not across samples) to provide J-NEAT with a compact representation of the acoustic input. The results show that for the development data set J-NEAT was capable of evolving small networks that match the performance of the baseline system in terms of the segment-based error metrics, while exhibiting a substantially better event-related error rate. In the challenge, J-NEAT took first place overall according to the F1 error metric with an F1 of 44.9% and achieved rank 15 out of 34 on the ER error metric with a value of 0.891. We discuss the question of evolving versus learning for supervised tasks.

Index Terms— Sound event detection, neuroevolution, NEAT, deep scattering transform, wavelets, clustering, co-evolution

1. INTRODUCTION

Neuroevolution algorithms evolve artificial neural networks using genetic algorithms (see [1] for an overview). They have been successfully applied for finding the solution policy in intricate reinforcement tasks such as guiding a robot through a maze [2] or autonomous computer game playing [3]. Almost exclusively these tasks are simulated or situated in virtual environments, where environmental information is accessible without being affected by noise. Neuroevolution performance in real world tasks is unclear and many real world equivalents of the simulated tasks might be even out of reach due to the time consuming nature of artificial evolution. For instance, for the scenario of a wheeled robot driving autonomously through a physical maze, robots might not be available in large enough numbers and/or take weeks to complete a single full evaluation run. In this study we employ neuroevolution in an heretofore unfamiliar task, sound event detection with noisy real-world data in the form of the DCASE 2017 challenge Task 3 (sound event detection in real life audio) dataset.

The research leading to this submission was funded by EPRSC grant EP/N014111/1.

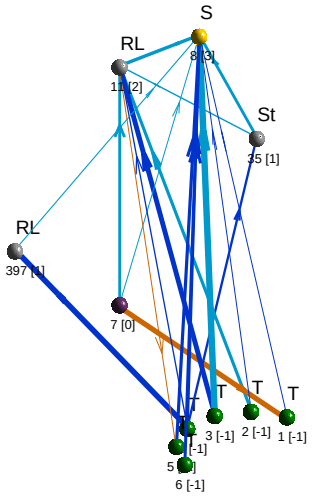

Figure 1: Structure of the first network of the ensemble with the best training performance on the full data set for the 'people walking' detector. Blue lines represent forward connections (light blue negative, dark blue positive weights), red/orange lines recurrent connections (orange negative, red positive weights). The relative magnitude of the weight is indicated by line width. Input nodes are depicted in green, bias nodes in dark purple and the output node in yellow. Each node has its identification number next to it and in square brackets the layer to which it was assigned. There is also a letter coding the activation function, where S = sigmoid, St = a steeper sigmoid function used in NEAT, T = tanh, I = identity, R = rectified linear, RL = leaky rectified linear and M = softmax.

Neuroevolution can be applied for only evolving the weights of neural networks or for evolving the topology and the weights together. The latter class of methods is known as TWEANNs, Topology and Weight Evolving Artificial Neural Networks. There are two major encoding approaches: In indirect coding the code refers to rules on how to construct the network phenotype, in direct encoding all neurons and synapses are explicitly specified in the genome. In this study we used a variant of the NEAT algorithm (NeuroEvolution of Augmenting Topologies) [4, 5], which uses direct encoding. NEAT starts with a minimal network consisting of input, bias and output nodes and then grows the networks using crossover (mating) and mutations. It protects evolving, more complex networks that

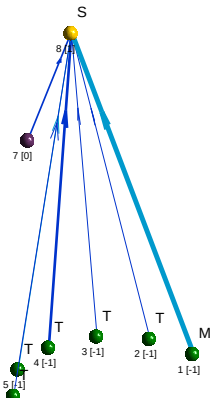

Figure 2: Structure of the second network of the ensemble with the best training performance on the full data set for the 'people walking' detector. See the caption of Figure 1 for the legend.

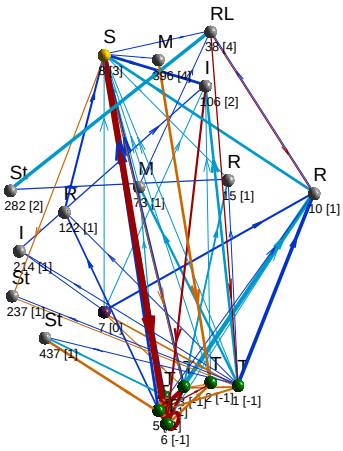

Figure 3: Structure of the third network of the ensemble with the best training performance on the full data set for the 'people walking' detector. See the caption of Figure 1 for the legend.

are likely to have lower fitness at the beginning through speciation: Individual networks are categorised into species at each generation based on their similarity and subsequently have to compete only with other networks within their own species.

A variant of NEAT, *Learning-NEAT* [6], has been used in classification of small data sets (e.g. Fisher's Iris flower dataset). L-NEAT's hybrid training scheme combines learning with evolution by incorporating backpropagation into NEAT. For the two tested datasets good performance was reported. However, the research appears to have not not been followed up upon. Another variant, termed *Layered NEAT*, closer to the original NEAT, was used for shale lithofacies prediction in geology [7] with good results.

The current study makes makes three major contributions: The application of neuroevolution to sound event detection, a novel way of feature extraction (though based on established procedures, the deep scattering transform and k-means clustering) and a unique extension of the NEAT algorithm introducing dynamic cooperative co-evolution (for use of co-evolution in other neuroevolution methods see e.g. [8] or [9]). The latter two contributions are to a large part a consequence of the need to adapt the neuroevolution procedure to the new task. In fact, the straightforward formulation of an offline detection and classification task had also to be revised: Instead of considering the entire input data set (or at least substantial parts of it as in the mini-batch procedure in deep learning), our starting concept included a sample-by-sample step-wise evaluation within each generation, with the fittest individuals (producing the least errors) progressing the farthest and the evaluation of unfit networks stopped early. For practical reasons, the step-wise procedure had to be later parallelised and the number of steps reduced and fixed in order to take advantage of the speed of matrix computations over loop executions in our implementation language of choice Matlab (The MathWorks, Inc).

2. METHOD

2.1. Feature extraction

Most current neuroevolution algorithms can evolve only small networks within a reasonable time frame compared to even moderately sized current classification networks with a single hidden layer let alone deep neural networks. This is certainly true for the NEAT algorithm (but see HyperNeat [10, 11] for an approach to fundamentally change this limitation). For classification and detection tasks this entails that a compact feature representation is required even on the expense of losing detail information. Due to the computationally intense procedure of evaluating hundreds of individual networks over hundreds of generations the temporal resolution of the acoustic input has also to be kept relatively low, at least so long as standard desktop computers are used. The combined requirements call for a representation that preserves high-frequency properties of the data despite averaging over time and reduces the parameters to a small number of components characteristic for the data. The wavelet-based deep scattering transform [12, 13] fulfils the first constraint. It computes in a cascade multiple orders of coefficients that are locally translation invariant. The second order coefficients preserve transient phenomena such as the amount of attack despite averaging over larger window sizes. The deep scattering transform results, however, in a large number of coefficients for each time slice, e.g., in our case with audio default settings and a window size of 0.372 s in 520 coefficients. To reduce the number of components we applied k-means clustering. In the clustering, the dimensions were switched: The observations (samples) were treated as input variables for the clustering, while the dimensions (scattering coefficients for the different scales) were treated as observations on these variables. The resulting k by n matrix (with n being the number of samples) of the cluster centroids constitutes a low-dimensional representation of the data consisting of the major modes of the scattering scale contributions.

The deep scattering transform was computed using the *ScatNet* toolbox [14] for Matlab, progressing file by file through the raw audio files of the development data set, using the first channel of the stereo files. We used a window size of 14^2 samples and kept all other parameters at the default values recommended by the creators for audio signals [15] (two orders, $q1 = 8$ and $q2 = 1$). We re-normalised and log-transformed the coefficients using the routines provided by the toolbox and then concatenated all training data for the clustering. In the clustering, we set k to 17 and used the squared Euclidean distance as distance measure (keep in mind that distances were computed on the original observations, i.e. samples). We re-normalised the resulting matrix containing the centroids by

subtracting the mean and dividing it by its standard deviation - separately for each component. We also calculated the channel differences from the raw spectral information to become the 18th input component: The spectrum was computed with the same temporal resolution as the deep scattering transform using the *rastamat* toolbox [16]. For each of the 8193 spectral components the difference between the two channels was calculated and then averaged to arrive at a single number characterising the channel differences.

Since our target resolution was 1 Hz (matched to the DCASE challenge's segment length), we re-sampled the combined input data with Matlab's *resample* function.

2.2. J-NEAT - a novel coperative co-evolution extension of the NEAT algorithm

For the event classification and detection task we devised a modified version of the NeuroEvolution and Augmenting Topology (NEAT) algorithm [4, 5]. The modifications can be divided into two classes:

- Modifications that adapted aspects of the original NEAT algorithm within its general paradigm;
- Modification that extended NEAT and changed its nature.

In the first class fall many changes that were undertaken to stabilise essential hyper-parameters, i.e., make them independent from each other and from the input value range, as well as providing more control over the rate of evolutionary weight increase and decrease. Due to space limitations we cannot describe them here in detail and focus on the more crucial second class, modifications that changed some of the core mechanisms of NEAT and transformed it into a new algorithm which we named J-NEAT for Joint Neuro-Evolution of Augmenting Topologies.

1. New nodes added by the mutation process do not intersect existing connections (synapses). If a new node is added, it is not constraint by the current synapses of the network, but established independently with two new synapses connecting it to existing nodes (or a single synapse if it is a new bias node). Additionally, new synapses between existing nodes are added in the mutation process.
2. The number of offspring for each crossover pair is not fixed to a single individual, but made dependent on the combined fitness of the two parent individuals. Higher relative fitness leads to a higher number of offspring in accordance with principles from biological evolution.
3. Contrary to standard NEAT, synapses and nodes can also be removed in the mutation process.
4. While in NEAT the activation function is fixed, we subject the choice for each node to the evolutionary process (except for the output nodes). During mutation a new activation function is selected from a pre-defined set (including sigmoid, RLU, leaky RLU, softmax, identity and tanh). This applies to both new nodes and existing nodes. The latter in the form of activation function mutations, albeit occurring only with a relatively low probability.
5. Following [7], we determine network layers by the longest path of forward connections to reach a node - not only because it enables analysing the network structure, but also because in our implementation it is required in order to be able to distinguish forward from recurrent synapses. The additional computational effort is reduced by only partially determining layer assignments (that is, only for the affected nodes) when a new node or synapse is added and reanalysing the full network only when necessary, e.g., when a node or a synapse is removed. We designed a fast recursive algorithm for this task.
6. We included standard recurrence. Wang et al. [7] showed that the recurrence calculation in the standard implementation of NEAT is not correctly working, revisiting nodes several times, leading to higher computational costs and incorrect output values. Unfortunately, their suggested alternative implementation solves only the revisiting problem, but does not realise recurrence in the standard way. It only creates a looped calculation of the current node states, which might be helpful in itself, but differs from recurrence in the typical definition, which requires the previous states (output from the previous evaluation step) to be considered. In our approach synapses that connect nodes of the same layer or connect from a higher to a lower layer are defined as recurrent and work with the previous states of the source nodes. Layer-wise the impact of the recursive nodes is computed first and then the impact of the forward nodes before finally the activation functions are applied.
7. To break the complex classification problem into smaller partial problems, we introduced cooperative co-evolution. Several populations exist concurrently and evolve simultaneously, coupled loosely through cooperation: Each of them gets a part of the input at each sample point, but the individuals from the populations solve the overall task by cooperating across population boundaries. In our set-up, there are three populations and they receive each a third of the input, that is, initially randomly chosen 6 values per sample out of the 18 components we obtained in the feature extraction. After determining the output for the current sample, ensembles are created based on their present energy (see below): triplets consisting of one selected individual from each population. The output of each individual network is considered as the probability that the target event was detected and is treated as a confidence value. Is is averaged within ensemble, but with higher confidence values (closer to 1 or 0) boosted by:

$$\tilde{w}_k = 1 - 4\,p_k\,(1 - p_k), \tag{1}$$

$$\tilde{p}_k = (p_k - 0.5)\,\tilde{w}_k, \tag{2}$$

$$p_o = \frac{\frac{1}{N}\sum_{i=1}^{N}\tilde{p}_i}{\frac{1}{N}\sum_{i=1}^{N}\tilde{w}_i} + 0.5, \tag{3}$$

where p_k is the estimate of ensemble member k (from population k), N the number of ensemble members (populations), $\tilde{w}$ the boosting factor, $\tilde{p}$ the adjusted estimate and p_o the averaged adjusted final output. This allows individual members of the ensemble with high confidence to override the influence of the others if these are weighing in for the opposite decision but are not too far from the 0.5 chance level border. Based on whether the result is a true positive, true negative, false positive or false negative rewards and penalties are given equally to all members of the ensemble and then converted into an energy measure. Ensembles are dissolved and reformed at the next evaluation sample and, thus,

also not carried over to the next generation. However, the individual members keep their acquired energy.

We used three populations with each 400 individuals. For each sound event class a separate neuroevolution run was conducted, i.e., all networks had always only one output node. Each run started with all individuals possessing a minimal, fully forward-connected network, consisting of six input nodes, one bias node and one output node with randomly assigned weights (drawn from a standard uniform distribution). The activation function of the output node was set to the sigmoid function. For implementation reasons input nodes possess an activation function, too. In the beginning we set them to the identity function, but in test runs the evolutionary process often replaced them by the hyperbolic tangent function in successful networks. We therefore decided to make $tanh$ the default start input activation function.

In each generation and each evaluation step we evaluated 250 randomly selected samples simultaneously. Their selection indices were determined by drawing in each generation consecutive distance values from a normal distribution with mean 20 and standard deviation 1.79 and accumulating them (e.g., values 18, 26 and 15 resulting in the selection of the 18^{th}, 44^{th} and 59^{th} sample). Each evaluation step progressed then by one sample, keeping all the distances intact. Forty-four steps were taken per generation, which meant that each sample point was on average evaluated 2.6 times. A complete run for a single class consisted of 500 generation. The best fitness value achieved over the course of the 500 generations was stored and the ensemble which accomplished it taken as the final classifier network. The results were aggregated over classes and folds and the final evaluation conducted using the Python routine provided by the DCASE 2017 Task 3 organisers [17].

For comparison, we also ran J-NEAT with no co-evolution, employing only a single population and each network receiving the entire input per sample. Finally, to gauge the impact of the feature extraction method and create a minimal node-size classifier using learning, we also applied a simple single-layer feed-forward network with no hidden nodes. The learning rate was set to 0.2. No regularisation techniques were used.

3. RESULTS

Figure 1 to 3 show as an example the three networks that formed the ensemble with the best training performance on the full development data set for the 'people walking' detector. Table 1 summarises the performance results across the four-fold validation and Table 2 shows the results from the challenge evaluation [18].

Table 1: Performance on the development test data set showing both segment- and event-based evaluation results.

Method	Seg. ER	Seg. F1	Ev. ER	Ev. F1
Baseline	0.72	51.40	3.30	6.74
J-NEAT ensemble	0.73	49.24	1.46	6.46
J-NEAT plain	0.72	50.55	1.37	5.66
Single-layer FFN	0.69	56.47	1.40	5.85

4. DISCUSSION

One could question the wisdom of using neuroevolution for tasks that are in principle solvable with supervised learning, presenting no obstacles for backpropagation of the error through the network. Our interest in neuroevolution stems from the desire to develop parsimonious neural network-based detectors and classifiers, consisting ideally only of a few nodes. These small systems could be used in, for instance, autonomous micro-robots such as insect-like airborn robots. A powerful all-purpose (hard-coded) feature extracting system might be available, but the detectors and classifiers would need to be of minimal size: In real-world navigation and other task solving a large number of them would be required, but the number of available neurons would be very limited. Neuroevolution using TWEANNs offers the potential to find small network solutions without a human experimenter having to specify network size and topology.

Table 2: Challenge evaluation.

Method	Segment-based value		Rank	
	ER	F1	ER	F1
DCASE Baseline	0.936	42.8	19	8
J-NEAT ensemble	0.898	44.9	15	**1**
J-NEAT plain	0.891	41.6	14	12
Single-layer FFN	1.014	43.8	28	3

The results using the development data set demonstrated that neuroevolution techniques can evolve small networks able to compete with the much bigger network of the baseline. Here they were, however, still outperformed by a minimal human-designed learning network. The picture changes when taking the challenge results into account. On these previously unseen data – never touched upon in the system development – abstraction and regularisation shortcomings become evident. The two J-NEAT systems performed approximately equally according to the ER metric. Both were above average of all 34 submitted systems and substantially outperformed our learning simple single-layer network. When looking at the complementary F1 metric the ensemble-based J-NEAT system clearly got the upper hand over the plain J-NEAT system and indeed performed better than all other submitted systems.

With caution this can interpreted as evidence for the ability of ensemble-based J-NEAT classifiers to tune in on specific characteristics of some but not all of the involved audio event classes and avoid to a certain degree interference from overlapping sound events in this difficult polyphonic task of the DCASE 2017 challenge. The challenge results certainly establish J-NEAT as a serious alternative to the DNN approaches, which dominated not just this challenge, but by now reign over most of machine learning.

Future work will investigate task dependency problems more closely. For static co-evolution it is advantageous for the subordinate tasks to be as independent as possible [19]. If the input is split to form subtasks as in our classification and detection system, independence is not likely. However, in our system ensemble assignments change dynamically at each evaluation step. It is therefore unclear whether greater independence would improve the results. We will also explore alternatives to the current approach, for instance by providing all populations with the full input, but still using cooperative co-evolution in the form of dynamic cross-population ensembles. A specialisation of individual networks on parts of the input will then be left to evolution as well. Another alternative will be to enable symbiotic relationships between networks of different populations through synapses connecting networks across the population barrier.

5. REFERENCES

[1] D. Floreano, P. Dürr, and C. Mattiussi, "Neuroevolution: from architectures to learning," *Evolutionary Intelligence*, vol. 1, no. 1, pp. 47–62, 2008.

[2] J. Lehman and K. O. Stanley, "Exploiting open-endedness to solve problems through the search for novelty," in *Proceedings of the Eleventh International Conference on Artificial Life (Alife XI)*, 2008, pp. 329–336.

[3] M. Hausknecht, J. Lehman, R. Miikkulainen, and P. Stone, "A neuroevolution approach to general Atari game playing," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 6, no. 4, pp. 355–366, 2014.

[4] K. O. Stanley and R. Miikkulainen, "Evolving neural networks through augmenting topologies," *Evolutionary Computation*, vol. 10, no. 2, pp. 99–127, 2002.

[5] ——, "Competitive coevolution through evolutionary complexification," *Journal of Artificial Intelligence Research*, vol. 21, pp. 63–100, 2004.

[6] L. Chen and D. Alahakoon, "NeuroEvolution of augmenting topologies with learning for data classification," in *International Conference on Information and Automation (ICIA 2006)*. IEEE, 2006, pp. 367–371.

[7] G. Wang, G. Cheng, and T. R. Carr, "The application of improved NeuroEvolution of Augmenting Topologies neural network in Marcellus Shale lithofacies prediction," *Computers & Geosciences*, vol. 54, pp. 50–65, 2013.

[8] F. Gomez, J. Schmidhuber, and R. Miikkulainen, "Accelerated neural evolution through cooperatively coevolved synapses," *Journal of Machine Learning Research*, vol. 9, no. May, pp. 937–965, 2008.

[9] Y. Liu, X. Yao, and T. Higuchi, "Evolutionary ensembles with negative correlation learning," *IEEE Transactions on Evolutionary Computation*, vol. 4, no. 4, pp. 380–387, 2000.

[10] J. Gauci and K. O. Stanley, "Autonomous evolution of topographic regularities in artificial neural networks," *Neural Computation*, vol. 22, no. 7, pp. 1860–1898, 2010.

[11] K. Stanley and P. Verbancsics, "Constraining connectivity to encourage modularity in HyperNEAT," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2011)*, 2011.

[12] J. Andén and S. Mallat, "Deep scattering spectrum," *IEEE Transactions on Signal Processing*, vol. 62, no. 16, pp. 4114–4128, 2014.

[13] S. Mallat, "Group invariant scattering," *Communications on Pure and Applied Mathematics*, vol. 65, no. 10, pp. 1331–1398, 2012.

[14] L. Sifre, M. Kapoko, E. Oyallon, and V. Lostanlen, *Scatnet: a MATLAB toolbox for scattering networks*, 2013.

[15] "ScatNet - Quickstart for audio processing," online web resource. [Online]. Available: http://www.di.ens.fr/data/software/scatnet/quickstart-audio/

[16] D. P. W. Ellis, "PLP and RASTA (and MFCC, and inversion) in Matlab," 2005, online web resource. [Online]. Available: http://www.ee.columbia.edu/~dpwe/resources/matlab/rastamat/

[17] A. Mesaros, T. Heittola, A. Diment, B. Elizalde, A. Shah, E. Vincent, B. Raj, and T. Virtanen, "DCASE 2017 challenge setup: tasks, datasets and baseline system," in *Detection and Classification of Acoustic Scenes and Events 2017 Workshop (DCASE2017)*, submitted.

[18] "DCASE 2017 - Task 3 (Sound event detection in real life audio) - results," online web resource. [Online]. Available: http://www.cs.tut.fi/sgn/arg/dcase2017/challenge/task-sound-event-detection-in-real-life-audio-results

[19] R. Chandra, M. Frean, and M. Zhang, "On the issue of separability for problem decomposition in cooperative neuro-evolution," *Neurocomputing*, vol. 87, pp. 33–40, 2012.