USING AN EVOLUTIONARY APPROACH TO EXPLORE CONVOLUTIONAL NEURAL NETWORKS FOR ACOUSTIC SCENE CLASSIFICATION

Technical Report

Christian Roletscheck, Tobias Watzka

Augsburg University
Human Centered Multimedia
Augsburg, 86159, Germany
rolle.roletscheck@t-online.de, tobias.watzka@gmail.com

ABSTRACT

The successful application of modern deep neural networks is heavily reliant on the chosen architecture and the selection of the appropriate hyperparameters. Due to the large amount of parameters and the complex inner workings of a neural network, finding a suitable configuration for a respective problem turns out to be a rather complex task for a human. In this paper we propose an evolutionary approach to automatically generate a suitable neural network architecture for any given problem. A genetic algorithm is used to generate and evaluate a variety of deep convolutional networks. We take the DCASE 2018 Challenge as an opportunity to evaluate our algorithm on the task of acoustic scene classification. The best accuracy achieved by our approach was 74.7% on the development dataset.

Index Terms— Evolutionary algorithm, genetic algorithm, convolutional neural networks, acoustic scene classification

1. INTRODUCTION

Deep learning techniques have already proven their capability to achieve outstanding performance in solving various classification tasks, it is therefore reasonable to use them for acoustic scene classification also. This trend can be clearly seen in the DCASE Challenge of 2016 [1] and 2017 [2]. However, finding a suitable network architecture and corresponding hyperparameters remains a challenge. Most of these neural networks have been developed through targeted research by experts. Therefore the motivation for this work lies in improving the mostly complex trial and error design process by hand. To facilitate or spare the user the development of a neural network, evolutionary algorithms can be used as optimization methods, which usually find workable solutions in an acceptable time. For this reason, we have developed an algorithm that automatically generates convolutional neural networks. Our self-adaptive evolutionary algorithm uses a genetic representation and creates deep neural networks from ground up. Thus, we named it "DeepSelf-Adaptive-Genetic-Algorithm" short DeepSAGA.

2. GENETIC ALGORITHM

The principle procedure [3, Cap. 3.1] of an evolutionary algorithm (EA) consists of generating an initial population and its evaluation. After some cycles have elapsed, the EA will terminate as soon as the termination criterion is met. A cycle consists of the steps, selection of parents, recombination, mutation, evaluation of offsprings and selection of individuals to form the population of the next cycle. In the entire course of this work, the term session refers to the holistic process of an EA (from initialization to termination).

The creation of an executable EA instance requires the specification of its parameters. The resulting values not only influence the finding of an optimal solution, but also the efficiency. Finding suitable EA instance parameters can be made easier using parameter control [3, Chap. 7.3]. *Self-Adaptive* is one of the possible *Parameter-Control* techniques. The parameters to be adapted represent a part of the genetics and are thus part of the evolutionary search space. Therefore, DeepSAGA has the potential to adapt the algorithm to the problem while solving the problem itself [3, Chap. 8]. The following subsections describe the implementation of the components that form DeepSAGA.

2.1. Representation and definition

To enhance readability, we have used a representation analogous to biological genetics. Within biological terminology, a **genome** contains all **chromosomes** and represents the entire genetic of a living being. A chromosome is a bundle of several **genes** contained in the organism, whereby genes determine the different characteristics of that organism. An **allele** is a concrete expression of a gene.

In our work, a genome represents the genetic of a neural network and describes its characteristics (its architecture and hyperparameters). In the terminology of evolutionary algorithms our genome is referred to as **genotype**, since it is part of the evolutionary search space. The transformation of a genotype into a solution (the so called **phenotype**) for the original problem is called decoding, in our case it is the process of creating and training the network according to the characteristics described in the genome.

The Figure 1 lists our chromosomes contained in each genome. The **Conv-Block** is made up of at least one so called **Gene-Bundle**. For each Gene-Bundle a convolutional layer followed by a max-pooling layer will be added to a neural network architecture. It therefore contains information (genes) about the number of filters, filter-shape and filter-stride. In addition each Gene-Bundle contains genes indicating if optional layers for zero-padding, dropout and batch-normalization are included. Finally a global-average-pooling or flatten layer can be used to connect the Conv-Block with the output or further classification layers.

Thanks to..

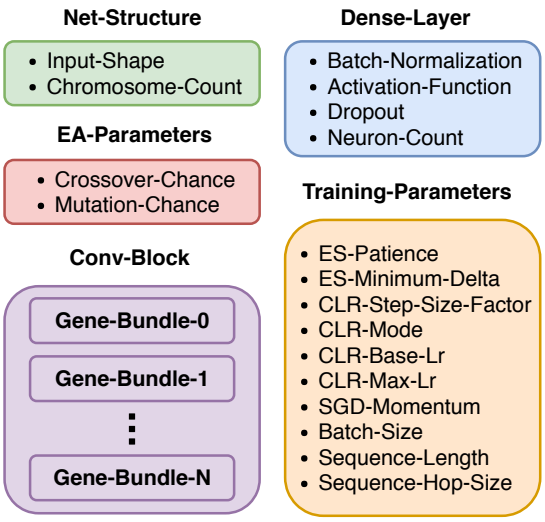

Figure 1: Detailed overview of all chromosomes and the genes listed in a genome. ES stands for early stopping and CLR for cyclic-learning-rate.

In our chosen representation model an allele for the **Batch-Size-Gene** could be, for example, the integer number 128. Finally Figure 2 illustrates the overall design of our genome.

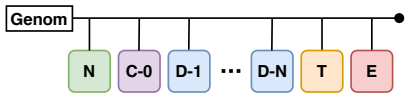

Figure 2: Illustration of a genome. The square like objects are representatives for the chromosomes Net-Structure, Conv-Block, Dense-Layer, Training- and EA-Parameters.

2.2. Fitness function

A fitness function specifies the quality of a genotype by assigning a fitness value to it. In our case the focus was mainly on the accuracy of a neural network. However, to speed up the evolutionary search process, the number of training epochs of a network was also taken into account. As a result our $score$ value represents the total fitness of a population member, meaning the higher the $score$ the better the quality of a genotype. The following formula illustrates the utilized fitness function:

$$\mathbf{score} = 0.98 * accuracy + 0.02 * \frac{epoch_{limit} - epoch}{epoch_{limit}} \quad (1)$$

In this context, $epoch_{limit}$ stands for the maximum number of epochs a net is permitted as training time and $epoch$ for the number of epochs with which the net was actually trained. The distribution with 98% on accuracy and 2% on the other half, seems to be a solid approach and is solely based on own empirical observations.

2.3. Population

A population contains possible solution candidates, the individuals (genotypes). A steady state model [3, Chap. 5.1] is used, to manage the population. In this model, only a part of the population is changed. Thus old individuals are replaced by new individuals (the offsprings). To promote diversity and the self-adaptive property, the population size is dynamic. However, since the available resources are limited, the maximum population size is restricted to 90.

2.4. Parent selection mechanism

The procedure for selecting parents is to distinguish between individuals on the basis of their quality. An individual is a parent when it has been chosen to produce offsprings by variation. As described by Bäck and Eiben [4] the parents are determined by a tournament selection procedure. Tournament selection [3, Chap. 5.2.4] is conducted by running several "tournaments" among a few individuals chosen at random from the population. The winner of each tournament (the one with the best fitness) is selected as a parent. The tournament size determines the number of participants per tournament and depends on the population of the current cycle. This also applies to the number of population members who are allowed to participate in the tournament. To calculate said number the Formula (2) was used, which takes into account the maximum permitted population size.

$$\mathbf{participants} = popsize_{limit} - popsize_{current} \quad (2)$$

The tournament size is determined by the formula (3), where $toursize_{limit}$ always corresponds to one tenth of the $popsize_{limit}$.

$$\mathbf{toursize} = toursize_{limit} * \frac{popsize_{current}}{popsize_{limit}} \quad (3)$$

If the population limit is reached, the number of participants is limited to **two** until the threshold value is undershot again.

2.5. Variation operators

New population members are generated by applying variation operators to existing individuals. This introduces the necessary diversity in the population making new innovations possible. The variation operator can be either of the mutation- or of the recombination-type.

Mutation

A variation operator, which affects only one member of the population, is called mutation. If a genotype is mutated, a (slightly) altered mutant is formed, which can be described as a child or offspring. The mutation chance specifies the probability with which a mutation takes place.

Genes of the category symbolic are mutated by replacing the original allele with a randomly selected. However, the **current** allele has a higher chance of being selected again than the other possible alleles. This type of mutation is also called sampling.

An allele of the integer type is mutated by a creep mutation [3, Chap. 4.3.1]. The original value is added to a randomly selected value from a Gaussian normal distribution with a median of 0, and a sigma value, which depends on a maximum limit. Therefore, the sigma value always corresponds to 0.025 (2.5%) times the limit. For example, if the maximum limit were 1000, the corresponding sigma value would be 25. There is also a 5% chance that the original value will be reset.

Nonuniform mutation [3, Chap. 4.4.1] is used to mutate an allele of the float type. This mutation procedure is similar to the

creep mutation, but a different sigma value is selected here, as it is equated with the individual's chance of mutation.

Each population member has its own chance of mutation, which is co-evolved according to the method described in [5]. Before all other genes, the **Mutation-Chance-Gene** is mutated using the non-uniform mutation method. The resulting new mutation chance is the probability with which the remaining genes are mutated.

Recombination

Recombination, or the so called crossover, is a variation operator that combines the information of two parent genotypes in one or two progeny. The crossover chance determines the probability with which a crossover occurs. If no recombination takes place, a clone of the respective individual is generated.

Since the crossover chance is part of the evolutionary process, it is represented by the **Crossover-Chance-Gene**. Thus, each population member has an individual crossover chance. The recombination process is based on the procedure described in [3, Chap. 8.4.7]. The individual crossover chance p_c of a parent is compared with a random number r ($r \in [0, 1]$). One parent is "ready to mate" if $p_c > r$ applies. This opens up the following possibilities:

1. When both parents are ready to mate, a crossover takes place.
2. If both parents are not ready to mate, they are cloned.
3. If only one parent is willing to mate, a clone of the unwilling parent is created. For the remaining individual a new partner is chosen randomly from the pool of parents, who is also checked for his willingness to mate.

The recombination itself takes place in the style of a uniform crossover [3, Chap. 4.2.2]. Taking into account the respective parent individual crossover chance, a descendant receives one chromosome of its parents. The exact procedure is depicted in Figure 3.

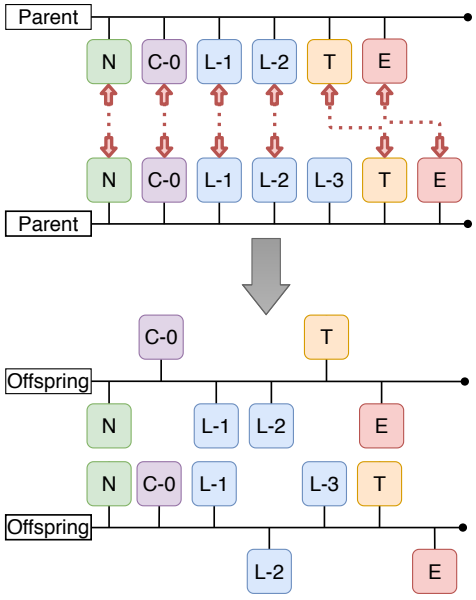

Figure 3: Illustrated procedure of our uniform crossover

2.6. Survivor selection mechanism

Similar to the parent selection, the survivor selection procedure also distinguishes between individuals on the basis of their fitness values. However, the method is used at a different stage of the evolutionary cycle (after the generation and evaluation of offspring). Due to the limited resources (e.g. the maximum number of individuals in a population) only certain individuals (the survivors) become members of the next population. Our selection procedure follows an age-based [3, Cap. 5.3.1] replacement strategy. Thus, each newly created individual is assigned a value (remaining lifetime, in short **RLT**) using the formula (4) as described by Bäck [4]. The RLT is reduced by **1** after each cycle, thus determining how long a population member remains alive. However, the lifetime of the individual with the highest fitness remains unchanged. Where $MinLT\ (\alpha)$ and $MaxLT\ (\omega)$ stand for the permissible minimum and maximum lifetime of an individual. The other variables are linked to the current status of the population. These variables are $fitness\ (i)$, $AvgFit\ (AF)$, $BestFit\ (BF)$ and $WorstFit\ (WF)$. They stand for the fitness of the individual i, the average fitness, the best fitness and the worst fitness of the current population. The prefactor calculation is $\eta = \frac{1}{2} \cdot (\omega - \alpha)$.

$$\mathbf{RLT}\ (i) = \begin{cases} \alpha + \eta \cdot \frac{WF - fitness(i)}{WF - AF} & \text{if } fitness\ (i) \geq AF \\ \frac{1}{2}\ (\alpha + \omega) + \eta \cdot \frac{AF - fitness(i)}{AF - BF} & \text{if } fitness\ (i) < AF \end{cases} \tag{4}$$

The authorized minimum and maximum lifetime of an individual has been set to **1** and **7**. If the fitness value of a newly created individual i is better than the average fitness, it receives a lifetime from **5** to **7**, otherwise a lifetime from **1** to **4**. Within these subareas, the better individuals have a longer life span than the individuals with a lower fitness. Assigning a lifetime results in several overlapping populations.

2.7. Initialization and termination

The individuals of the first population are generated randomly. Since the available resources are limited, the expressions of the respective genes are bounded. These limits must not be exceeded by variation operators either.

Since the optimum is not known in advance, the termination criterion is the completion of the **40th** cycle.

3. EXPERIMENTS

3.1. Setup

To evaluate the proposed genetic algorithm we use the TUT Urban Acoustic Scenes 2018 dataset from subtask A provided by the DCASE 2018 Challenge [6]. The dataset consists of 10-seconds audio segments from 10 different acoustic scenes. For every acoustic scene, audio was captured in different cities and multiple locations. To train and measure the performance of the generated models we use the development dataset with the suggested partioning for training and testing.

To generate the input features for the neural networks the stereo audio recordings were first converted into mono channels. Thereafter the librosa library (v0.6.1) [7] was used to extract log mel spectrograms with 100 mel bands. For the Short-Time Fourier

Transform (STFT) a Hamming window with a size of 2048 samples (43ms) and a hop size of 1024 samples (21ms) was used. The resulting spectrograms were then divided into sequences with a certain number of frames that define the sequence length. For the creation of the sequences an overlap of 50 % was used. The sequence length can vary depending on the different models generated by the genetic algorithm.

In order to speed up the process as a whole, several computers were connected via a self-written network module, which uses a client-server concept. The server distributes the genotypes from the current cycle population to all available clients, on which side the decoding takes place. Altogether 15 clients equipped with an NVIDIA GTX 1060 were available for the neural network training process. Therefore depending on the current population size and complexity of the genomes a cycle took around 2 to 3 hours.

To find a good compromise between exploitation and exploration, two independent sessions of 10 cycles each were initially completed. Afterwards, the best 30 models from each of these sessions were added to the initial population of a new session.

The best neural network of the final session was used for classification. In addition, a different strategy was pursued. From a cycle the **10** best individuals cloud also be selected to vote together on the class of an audio sample. Where individuals in higher ranks have more votes to weight them higher. Finally the class with the most votes wins. This type of classification is referred to in this paper as **population vote**.

3.2. Results

Table 3.2 illustrates the final results. At the end our best cnn (named "Rank1") reached an average accuracy of 72.8 % on the development dataset. For the population vote strategy, on the other hand, an average accuracy of 74.7 % was reached.

Scene label	Baseline CNN	Rank1 CNN	Population Vote
Airport	72.9 %	84.9 %	85.7 %
Bus	62.9 %	63.2 %	67.4 %
Metro	51.2 %	71.3 %	71.6 %
Metro station	55.4 %	75.3 %	81.9 %
Park	79.1 %	81.0 %	82.2 %
Public square	40.4 %	53.2 %	56.0 %
Shopping mall	49.6 %	75.3 %	73.8 %
Street, pedestrian	50.0 %	67.2 %	69.6 %
Street, traffic	80.5 %	85.0 %	86.2 %
Tram	55.1 %	72.0 %	72.4 %
Average	**59.7 %**	**72.8 %**	**74.7 %**

Table 1: The class-wise accuracy for task 1 Subtask A evaluated on the development dataset.

The genome of the "Rank1" CNN can be seen in Figure 4.

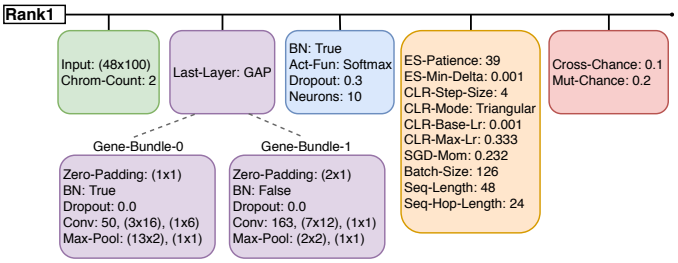

Figure 4: Best genome of the session. GAP stands for Global-Average-Pooling and BN for Batch-Normalization. The numbers in the brackets are the filter size and the filter stride for the convolutional and max-pooling layers and the first number for the convolutional layer stands for the number of filters.

4. CONCLUSION

In this paper, we described how we developed and evaluated a genetic algorithm called "DeepSAGA" to automatically generate CNNs optimized for the classification of the ten acoustic scenes of the DCASE 2018 Challenge. With an accuracy of 74.7 % on the development dataset the algorithm showed promising results with this specific dataset. Nevertheless, the approach is also applicable for other classification problems which could be tested in the future.

Throughout the sessions, the approach of population vote resulted in a higher accuracy than that of the best model of the corresponding cycle. However, there were fluctuations in the accuracy difference, which should be further researched. Additionally, the extension to generate neural networks including recurrent layer could further improve classification results but also introduce a vast of new parameters that have to be tested by the algorithm.

5. REFERENCES

[1] A. Mesaros, T. Heittola, and T. Virtanen, "TUT database for acoustic scene classification and sound event detection." IEEE, 2016, pp. 1128–1132. [Online]. Available: http://ieeexplore.ieee.org/document/7760424/

[2] 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 *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2017 Workshop (DCASE2017)*, 2017, pp. 85–92.

[3] A. Eiben and J. Smith, *Introduction to Evolutionary Computing*, ser. Natural Computing Series. Springer Berlin Heidelberg, 2015. [Online]. Available: http://link.springer.com/10.1007/978-3-662-44874-8

[4] T. Bäck and A. E. Eiben, "An emperical study on GAs without parameters," in *International Conference on Parallel Problem Solving from Nature*. Springer, 2000, pp. 315–324. [Online]. Available: https://link.springer.com/chapter/10.1007/3-540-45356-3_31

[5] T. Bäck, "The interaction of mutation rate, selection, and self-adaptation within a genetic algorithm." in *PPSN*, 1992, pp. 87–96.

[6] A. Mesaros, T. Heittola, and T. Virtanen, "A multi-device dataset for urban acoustic scene classification," 2018. [Online]. Available: http://arxiv.org/abs/1807.09840

[7] B. McFee, C. Raffel, D. Liang, D. P. Ellis, M. McVicar, E. Battenberg, and O. Nieto, "librosa: Audio and music signal analysis in python," in *Proceedings of the 14th python in science conference*, 2015, pp. 18–25. [Online]. Available: http://www.academia.edu/download/40296500/librosa.pdf