HYPER-PARAMETER OPTIMIZATION FOR CONVOLUTIONAL NEURAL NETWORK COMMITTEES BASED ON EVOLUTIONARY ALGORITHMS

Erik Bochinski, Tobias Senst, Thomas Sikora

Technische Universität Berlin
Communication Systems Group
{*bochinski,senst,sikora*}@nue.tu-berlin.de

ABSTRACT

In a broad range of computer vision tasks, convolutional neural networks (CNNs) are one of the most prominent techniques due to their outstanding performance. Yet it is not trivial to find the best performing network structure for a specific application because it is often unclear how the network structure relates to the network accuracy. We propose an evolutionary algorithm-based framework to automatically optimize the CNN structure by means of hyper-parameters. Further, we extend our framework towards a joint optimization of a committee of CNNs to leverage specialization and cooperation among the individual networks. Experimental results show a significant improvement over the state-of-the-art on the well-established MNIST dataset for hand-written digits recognition.

Index Terms— Image Classification, Convolutional Neural Network, Evolutionary Algorithm, MNIST, Hyper-parameter Optimization

1. INTRODUCTION

Convolutional Neural Networks (CNNs) have demonstrated superior performance on a variety of computer vision tasks in recent years. The success of CNNs is not only caused by the outstanding performance, e.g. in automated image processing or video-based classification and recognition systems. CNN-based methods impose a shift of design and engineering efforts from hand-crafted features, e.g. HoG [1] for image processing or LaSIFT [2] for video processing, towards the design of network connectivity structures, extensive training phases with considerable amounts of data and suitable optimization strategies.

The optimal choice of hyper-parameters is one of the major challenges when applying CNN-based methods. To our knowledge, there are no reliable approaches to identify specific network structures that may lead to significant performance improvements. In practice, this means that the network structures are often chosen by an "educated guess" rather than a reasoned decision. Since a broad range of CNN network structures yield good results, it stays unclear if a current solution obtains optimal or near optimal configurations or if different structures can improve previous results.

In this paper we define hyper-parameters as the configuration of the network structure and thus pose the problem of finding the optimal configuration of a CNN as an optimization problem of hyper-parameters. In the literature, the optimization of hyper-parameters has been addressed e.g. by Bergstra and Yoshua [3] who proposed to apply a grid or random fashion search. Since the hyper-parameter space is in general large and testing is computationally expensive, the need for a more structured way of searching this space arises. More sophisticated methods for hyper-parameter optimization have been proposed by Snoek *et al.* [4] with a Bayesian optimization framework and by Miguel *et al.* [5] and Hutter *et al.* [6] who applied evolutionary algorithms (EAs).

In recent times, the performance of CNN-based systems was further improved by building committees of multiple CNNs [7, 8, 9]. In [8], Lui *et al.* proposed negative correlation learning to train multiple neural networks simultaneously and interactively by adding a penalty term for correlation to the error function and thus promote cooperation and specialization among the networks. Chen *et al.* [9] applied this technique to Radial Basis Function Networks and introduced an additional regularization term. To obtain the optimal trade-off between empirical error, correlation and regularization, they integrated a multi-objective EA. In [7], Ciresan *et al.* proposed to average the output of multiple CNNs. This simple, yet effective approach is based on the idea that if the errors between networks are uncorrelated, they can be reduced by averaging. To reduce correlation of the prediction errors, they proposed different methods of pre-processing and normalizing the input data.

The objective of this paper is twofold: 1) we propose an efficient hyper-parameter optimization strategy for the CNN network structure based on an evolutionary algorithm and 2) we extend the EA-based hyper-parameter optimization towards a committee of multiple CNNs. We describe the proposed EA-based hyper-parameter optimization scheme in

Section 2. In this work we focus on hyper-parameters that describe the structure of a CNN such as layer and kernel sizes. Other training-related hyper-parameters will not be considered as modern optimizers, e.g. [10], are not sensitive against variations and consequently need no extensive tuning. In Section 3 we present our approach to generate reliable committees of CNNs. The performance of the proposed system will be evaluated on the MNIST dataset [11] for hand-written digits recognition in Section 4.

2. HYPER-PARAMETER OPTIMIZATION USING EVOLUTIONARY ALGORITHM

The goal of this section is to find an optimal CNN network structure which will be encoded as a set of hyper-parameters. Therefore, we define the hyper-parameter $\mathbf{h} = \{\mathbf{h}_{conv}, \mathbf{h}_{fc}\}$ as the set of structural parameters composed by $\mathbf{h}_{conv}$, the parameters of the convolutional layers, and $\mathbf{h}_{fc}$, the parameters of the fully connected layers. The convolutional layer parameter set is given by $\mathbf{h}_{conv} = \{\mathbf{l}_0, \ldots, \mathbf{l}_{n-1},\}$ where $\mathbf{l}_i = (k_{count}, k_{size})$ denotes the configuration tuple of the i^{th} layer, i.e. number of kernels k_{count} and the kernel size k_{size}. The fully connected layers are defined by $\mathbf{h}_{fc} = \{s_0, s_{n-1}\}$ with s_i being the layer size. Let $\mathcal{H}$ be the set of possible CNN configurations. Then the goal of the hyper-parameter optimization is to find a configuration $\mathbf{h} \in \mathcal{H}$ that minimizes the application-specific error of the respective CNN.

Since the solution space is discrete and thus the error function is not continuous nor differentiable, traditional optimization approaches such as gradient-decent are not applicable. Therefore we propose to use evolutionary algorithms (EAs) which are able to deal with these challenging conditions. EAs are biologically inspired by Darwin's theory of evolution. The core aspect of EAs is based on the concept of survival of the fittest in a population P which is defined as a set of individuals. The individuals contain the parameters to be optimized. Thus the population is a set of samples in the solution space. During optimization, the population P is evolved in a two-step manner: at first individuals are modified by crossover and/or mutation. Then a fitness-based selection is applied to produce the next generation. This means the EA is an iterative process and the optimum is sought from different positions simultaneously. Hence trapping in local minimum can be avoided [12].

In this work we define an individual by the hyper-parameters $\mathbf{h}$. The layers $\mathbf{l}_i$ are sorted from convolutional to fully connected. The convolutional layers itself are sorted in descending order first based on k_{size} and then based on k_{count}. The fully connected layers are sorted in descending order based on s. In the context of EAs, the items of the layer tuples $\mathbf{l}_i$ as well as the fully connected layer sizes s_i are denoted as genes. This strict sorting scheme reduces the amount of possible network configurations drastically and thus limits the solution space. We assume that optimal networks are within these boundaries and that most implausible configurations can be excluded, such as alternating convolution and fully connected layers. The population at a given time step t consists of μ individuals so that $P^t = \{\mathbf{h}_0, \ldots, \mathbf{h}_{\mu-1}\}$.

We use the $(\mu + \lambda)$ evolutionary algorithm where λ denotes the number of offspring individuals evolved by mutation and crossover. The λ offsprings are equally generated by applying crossover and mutation to a set of randomly sampled individuals from the parent population. We apply a one-point crossover method which exchanges all genes after a randomly selected position in the genome of two individuals. The mutation contains two operations: the variation and the elimination or creation of a gene. The variation is implemented by the polynomial bounded mutation function of the NSGA-II algorithm [13]. The value x of a gene is updated to x' with a certain probability p_m as follows:

$$x' = \lfloor \min(\max(x + \delta_q(x_u - x_l), x_l), x_u) \rfloor \quad (1)$$

with

$$\delta_q = \begin{cases} 2\delta_1 - 1 & , u < 0.5 \\ 1 - 2\delta_2 & , u \geq 0.5 \end{cases}$$

$$\delta_1 = \left(u + (0.5 - u) \left(1 - \frac{x_i - x_l}{x_u - x_l} \right)^{\eta+1} \right)^{\frac{1}{\eta+1}}$$

$$\delta_2 = \left(1 - u + (u - 0.5) \left(1 - \frac{x_u - x_i}{x_u - x_l} \right)^{\eta+1} \right)^{\frac{1}{\eta+1}}$$

and $x, x' \in [x_l, x_u]$ where x_l and x_u are the lower and upper bounds of the respective genes, η denotes the distribution index and $u \in [0, 1]$ being a value sampled from a uniform distribution. The elimination or creation operators are applied with the same probability p_m and are restricted by the boundary conditions of maximal and minimal layer sizes for the convolutional and fully connected layers. If the hyper-parameter $\mathbf{h}$ of an individual has not been changed, the weight-parameters of a CNN will not be re-trained.

The first population P^0 is generated by a random sampling of the solution space in predefined boundaries. In the next step an interim population $P^t_{\mu+\lambda}$ is constructed with μ individuals of the parent population P^t and λ offspring individuals. The next generation P^{t+1} is then created by selecting the μ fittest individuals of the interim population $P^t_{\mu+\lambda}$. The fitness function $f(\mathbf{h})$ equals the classification error $e(\mathbf{h})$ of the respective CNN structure $\mathbf{h}$ trained with the training data and evaluated with validation data of the respective dataset. This iterative process is terminated after a predefined number of γ generations.

3. JOINT OPTIMIZATION FOR COMMITTEES OF MULTIPLE CONVOLUTIONAL NETWORKS

In the previous section we have proposed an optimization method of the CNN network structure that finds the best

Parameter	Value
γ	50
μ	30
λ	10
p_m	0.3
η	5

(a) EA Parameters

Parameter	Value
Conv Layers	0-6
Kernel Size	1-8
Conv Layer Size	1-128
Initial Conv Layers	2-4
FC Layers	0-4
FC Layer Size	16-2048
Initial FC Layers	1-2

(b) Boundaries & Initialization

Parameter	Value
Optimizer	Adam
α	0.0001
Dropout (FC)	50%
Act. Function	ReLU
Batchsize	100
Train Epochs	30
Cost Function	MSE

(c) CNN training parameters

Table 1: Parameters used in all experiments

performance of a single CNN. In order to further improve the classification performance, committees of multiple CNNs will be used. A committee is a set of K trained CNNs where the classification is performed by fusing the CNN scores. We use the average of the decision values over the K CNNs. In theory, the lowest average error is obtained if the classification errors of the K CNNs are uncorrelated [7]. For that reason we propose a novel fitness function that takes the global classification error of the population into account:

$$f(\mathbf{h}_m) = \sum_{j}^{N} e_j(\mathbf{h}_m) \cdot \left(\sum_{i}^{\mu+\lambda} e_j(\mathbf{h}_i) \right)^k, \ \mathbf{h}_m, \mathbf{h}_i \in P^t_{\mu+\lambda} \tag{2}$$

where $\mathbf{h}_m$ is the m^{th} hyper-parameter associated with the m^{th} CNN of the current interim population, $e_j(\mathbf{h}) \in \{0, 1\}$ denotes the classification error of the j^{th} sample of the validation dataset of size N and k being a penalty exponent. If the current CNN misclassifies a sample j, the penalization depends on the classification error of the whole population. With the penalization $k > 1$ this effect can be enhanced.

To build a committee, we first perform the hyper-parameter optimization as in Section 2, based on the modified fitness function in Eq. 2. The committee then consists of the K fittest CNNs when considering all evaluated hyper-parameter configurations. An alternative approach could be an exhaustive search for better committees as analyzed in [7], however due to the large amount of possible combinations this is not feasible. We believe a fixed strategy of selecting the committee is a good trade-off between accuracy and complexity.

4. EXPERIMENTS

In this section we evaluate the proposed evolutionary based hyper-parameter optimization for CNN committees. The experiments were conducted on the well-known MNIST [11] classification dataset. It consists of 28×28 pixel sized grayscale images of handwritten digits (0-9) and is divided into 60,000 training and 10,000 test samples. We further split the training set into 50,000 images for CNN training and 10,000 images for validation. The pixel range of the images is scaled from $[0, 255] \in \mathbb{Z}$ to $[0, 1] \in \mathbb{R}$. No further preprocessing or augmentation is done on the data.

All CNNs are trained using the tensorflow library [14]. Optimization is done using the Adam algorithm [10] with the configuration shown in Table 1c). No padding is used at the convolutional layers and no pooling layers are included. The training samples are permuted after each epoch. If the training loss falls below the validation loss, early stopping is performed and the training is terminated. Otherwise it is continued until a maximum count of training epochs is reached. This is done to prevent overfitting and to reduce the training time.

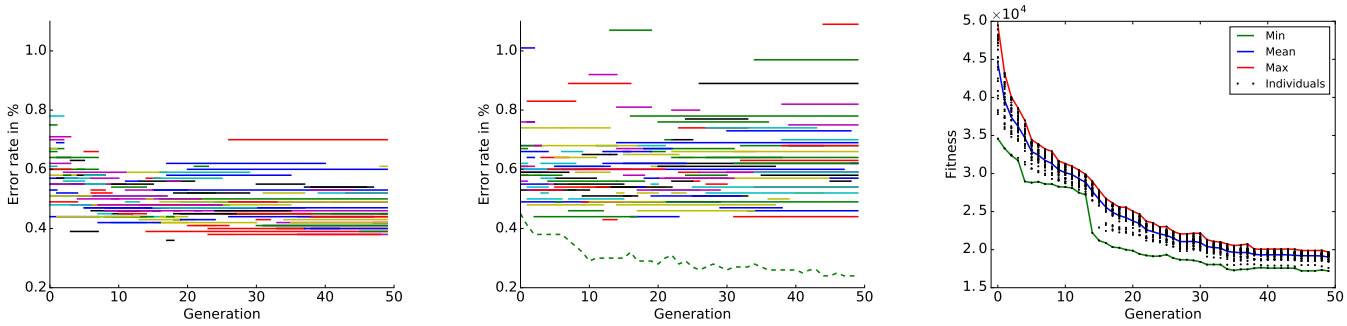
The implementation of the EA algorithm is based on the DEAP evolutionary computation framework [15]. The parameters of the EA used for all experiments have been chosen empirically and are shown in Table 1 a) and b) with a committee size of $K = 34$ CNNs.

In our experiments we compare the performances of the evolutionary based hyper-parameter optimization for independent CNNs (IEA-CNN) and the proposed joint optimization for a committee of multiple CNNs (CEA-CNN). Figure 1 shows the evolution of individuals for evolving generations related to the error rate. Each line of a specific color denotes a single individual and its classification error. In both scenarios the initial generation achieved accuracies below 0.8%, except for some outliers not included in the graphs. This shows that many network structures yield good results when being initialized randomly. The results show a higher rate of convergence for the independently optimized CNNs than for the joint optimized population. The CNNs obtained by joint optimization have a higher variance in terms of error rate. It appears that some well-performing networks have been discarded while some worse performing networks have been maintained due to separation of the dataset set into validation and test data for the hyper-parameter optimization and the final evaluation. However, no significant overfitting to the validation data has been noticed.

At a first glance, IEA-CNN tends to be more accurate then the CEA-CNN due to the higher rate of convergence considering the individual error rates. The evolution of the CEA-CNN accuracy (green doted line in Figure 1b) shows that regardless of the higher variance in the population and the lower convergence rate, the total error rate is converging to a lower error level than for the individual case. We assume that the higher variance indicates a higher ability of specialization and cooperation between the individual CNNs.

Figure 1 b) (right) shows the distribution of the fitness values referring to Eq. 2 for each individual during the joint optimization process, as well as the minimal and maximal bounds and the average curve for a better comparison. The experimental results show that when an individual with a particularly low fitness is found, the mean fitness of all individuals in the following generations drops significantly.

The mean number of training epochs ranges from 16 to 20 in the first and last generation for CEA-CNN which is significantly less than compared to recent approaches [7]. As a result, the training of a single CNN is less computationally expensive and allows evaluating more individuals to improve the



(a) Independent CNN optimization

(b) Joint committee optimization for k=2

Fig. 1: Distribution of individuals during the hyper-parameter optimization for (a) independent CNN optimization and (b) joint committee optimization. The error rate of each individual is represented as a line. The dotted line in (b) shows the accuracy of the committee. The right plot (b) shows the distribution and development of the fitness values referring to Eq. 2 of the individuals in the population.

sampling of the solution space. The CNNs in the final population consists of 4-5 convolutional layers and 1-3 (mostly 2) fully connected layers. These structures are considerably deeper than in [11, 16, 17] but not as deep as in [18]. The kernel sizes range from $8\times8-7\times7$ to $4\times4-3\times3$, where some networks have also a 1×1 layer and/or start with a 6×6 layer. This surprises since normally smaller kernel sizes are used for this input size. The last population of the single network optimization is less diverse. The most typical structure can be described as $\mathbf{h}_{conv} = \{(107,7),(88,7),(123,6),(102,3)\}$ and $\mathbf{h}_{fc} = \{104\}$ with only minor deviations, mostly in the last layers.

Table 2 concludes the results of our experiments for the independent EA-based optimized CNN (IEA-CNN) and the joint EA-based committee optimization (CEA-CNN) with varying penalization k. We compare these results with a subset of representative state-of-the-art methods not using augmentation on the training data. It shows that the EA-based optimized IEA-CNN network, which applies a very baseline CNN, is competitive to the recent state-of-the-art. Moreover, the joint based hyper-parameter optimization of the committees significantly outperforms the IEA-CNN as well as recent approaches. It can be shown that CEA-CNN performs best with $k = 2$. Figure 2 shows all misclassified samples of the MNIST dataset.

Method	Test Error in %
LeNet-5 [11]	0.95
Deeply Supervised Net [16]	0.39
Schallow CNN [17]	0.37
Recurrent CNN [19]	0.31
Gated Pooling CNN [18]	0.29
IEA-CNN	0.34
CEA-CNN, k=1	0.26
CEA-CNN, k=2	**0.24**
CEA-CNN, k=3	0.28

Table 2: MNIST classification error: comparison of state-of-the-art methods without using any augmentation on the training data.

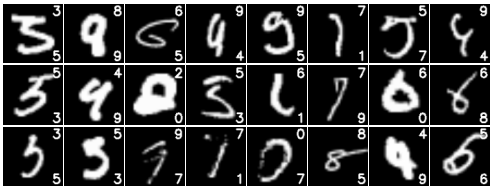


Fig. 2: All 24 classification errors for k=2 of the 10,000 test images. The upper and bottom left numbers indicate the ground truth and classification results respectively.

5. CONCLUSION

We proposed an evolutionary algorithm-based hyper-parameter optimization approach for committees of multiple CNNs. The objective is to optimize the structure of a CNN in relation to the performance of a committee rather than the individual accuracy. Therefore, we proposed an evolutionary algorithm-based hyper-parameter optimization scheme and a novel fitness function supporting this objective. The experiments on the MNIST dataset demonstrate significant improvements to the state-of-the-art. This is mainly due to the ability of the joint optimized committee to leverage specialization and cooperation among the individual CNNs. Since the proposed optimization framework is generic to any type of networks and tasks, we plan for future work to apply this technique to more sophisticated CNN approaches such as R-CNNs [19].

6. ACKNOWLEDGEMENTS

The research leading to these results has received funding from the European Community's FP7 and BMBF-VIP+ under grant agreement number 607480 (LASIE) and 03VP01940 (SiGroViD).

7. REFERENCES

[1] Navneet Dalal and Bill Triggs, "Histograms of Oriented Gradients for Human Detection," in *IEEE Computer Vision and Pattern Recognition*, 2005, pp. 886–893.

[2] Tobias Senst, Volker Eiselein, and Thomas Sikora, "A local feature based on lagrangian measures for violent video classification," in *IET International Conference on Imaging for Crime Detection and Prevention*, 2015, pp. 1–6.

[3] James Bergstra and Yoshua Bengio, "Random search for hyper-parameter optimization," *Journal of Machine Learning Research*, vol. 13, pp. 281–305, 2012.

[4] Jasper Snoek, Hugo Larochelle, and Ryan P. Adams, "Practical bayesian optimization of machine learning algorithms," in *Advances in neural information processing systems*, 2012, pp. 2951–2959.

[5] David Orive Miguel, Gorka Sorrosal Yarritu, Cruz Enrique Borges Hernández, Cristina Martín Andonegui, and Ainhoa Alonso Vicario, "Evolutionary algorithms for hyperparameter tuning on neural networks models," in *European Modeling and Simulation Symposium*, 2014.

[6] Ilya Loshchilov and Frank Hutter, "CMA-ES for hyperparameter optimization of deep neural networks," in *International Conference on Learning Representations*, 2016.

[7] Dan Claudiu Ciresan, Ueli Meier, Luca Maria Gambardella, and Jurgen Schmidhuber, "Convolutional neural network committees for handwritten character classification," in *International Conference on Document Analysis and Recognition*, 2011, pp. 1135–1139.

[8] Yong Liu and Xin Yao, "Ensemble learning via negative correlation," *Neural Networks*, vol. 12, no. 10, pp. 1399–1404, 1999.

[9] Huanhuan Chen and Xin Yao, "Multiobjective neural network ensembles based on regularized negative correlation learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 12, pp. 1738–1751, 2010.

[10] Diederik Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," in *International Conference for Learning Representations*, 2015.

[11] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[12] Raul Rojas, "Genetic algorithms," in *Neural Networks: A Systematic Introduction*, chapter 17, pp. 427–448. Springer-Verlag Berlin Heidelberg, 1996.

[13] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and T. A. M. T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-ii," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.

[14] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, Software available from tensorflow.org.

[15] Félix-Antoine Fortin, François-Michel De Rainville, Marc-André Gardner, Marc Parizeau, and Christian Gagné, "DEAP: Evolutionary algorithms made easy," *Journal of Machine Learning Research*, vol. 13, pp. 2171–2175, 2012.

[16] Chen-Yu Lee, Saining Xie, Patrick Gallagher, Zhengyou Zhang, and Zhuowen Tu, "Deeply-supervised nets," in *International Conference on Artificial Intelligence and Statistics*, 2015, pp. 562–570.

[17] Mark D. McDonnell and Tony Vladusich, "Enhanced image classification with a fast-learning shallow convolutional neural network," in *International Joint Conference on Neural Networks*. IEEE, 2015, pp. 1–7.

[18] Chen-Yu Lee, Patrick W. Gallagher, and Zhuowen Tu, "Generalizing pooling functions in convolutional neural networks: Mixed, gated, and tree," in *International Conference on Artificial Intelligence and Statistics*, 2016, pp. 464–472.

[19] Ming Liang and Xiaolin Hu, "Recurrent convolutional neural network for object recognition," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 3367–3375.