Improving Convolutional Neural Network Design via Variable Neighborhood Search

Teresa Araújo[1,2], Guilherme Aresta[1,2], Bernardo Almada-Lobo[1,2], Ana Maria Mendonça[1,2], and Aurélio Campilho[1,2]

[1] INESC TEC - Institute for Systems and Computer Engineering, Technology and Science, Porto, Portugal,
{tfaraujo,gmaresta}@inesctec.pt,
[2] Faculdade de Engenharia da Universidade do Porto, Porto, Portugal

Abstract. An unsupervised method for convolutional neural network (CNN) architecture design is proposed. The method relies on a variable neighborhood search-based approach for finding CNN architectures and hyperparameter values that improve classification performance. For this purpose, t-Distributed Stochastic Neighbor Embedding (t-SNE) is applied to effectively represent the solution in a 2D space. k-Means clustering divides this representation space having in account the relative distance between neighbors. The algorithm is tested in the CIFAR-10 image dataset. The obtained solution improves the CNN loss by over 15% and the respective accuracy by 5%. Moreover, the network shows higher predictive power and robustness, validating our method for the optimization of the CNN design.

Keywords: machine learning, convolutional neural network, parameter optimization, variable neighborhood search

1 Introduction

Convolutional Neural Networks (CNNs) [8] are a machine learning technique capable of automatically learn relevant features from multi-dimensional data, such as images. These methods have shown to achieve state-of-the-art performance in several different tasks being successfully applied on object recognition [6] and biomedical image analysis [1], among others.

Machine Learning algorithms often require careful tuning of their hyperparameters [11]. In particular, deep learning methods, such as CNNs, require thorough tuning of numerous parameters [11]. The classification accuracy of a CNN is influenced by multiple factors, such as the number of neurons and organization of layers of each type, the regularization strength and the dropout percentage. Parameter search is particularly difficult for these networks since the evaluation of the goal function is very expensive due to the high training time.

CNNs' architecture and hyperparameters are usually manually tuned for a particular dastaset [5], which hardly retrieves a close-to-optimal model and can

be time-consuming. Thus, automatic methods that perform the optimization of the CNN design are desirable. Some works in the literature aim at the development of automatic methods for optimizing the CNN architecture and/or hyperparameters. Jin *et al.* [5] proposed a submodularity and supermodularity method for optimizing neural network architectures, while other hyperparameters were optimized using grid-search or, alternatively, set to recommended values. Snoek *et al.* [11] presented methods for Bayesian optimization of machine learning algorithms' hyperparameters. The method was tested in the CIFAR-10 dataset [7], using the network from [6]. A total of nine hyperparameters were optimized, among which are the number of epochs, the learning rate and the pooling size. They report an improvement of 3% in the validation accuracy relative to the initial parameter setting.

Heuristic and metaheuristics techniques are commonly used for solving optimization problems. However, their implementation for CNN improvement is still little explored. Further, most optimization methods do not focus on the architecture design but rather on the optimization of hyperparameters such as learning rate, weight initialization and the number of epochs to run the model. We propose a Variable Neighborhood Search-like (VNS) approach for CNN architecture and hyperparameter optimizing. The VNS algorithm is composed of two steps. The first one is a diversification of the search in which a new neighborhood is generated. Then, an intensification step allows to achieve a new local optima. This process is repeated for increasing neighborhoods, allowing to obtain different local optima and thus, most likely, improve the incumbent solution [4].

2 Materials and Methods

2.1 Solution representation and neighborhood definition

Let S be a $m \times n$ matrix, where m is the number of layers and n the number of parameters of a network. S is characterized by a set of parameter indices s:

$$S_{m,n} = \begin{pmatrix} s_{1,1} & s_{1,2} & \cdots & s_{1,n} \\ s_{2,1} & s_{2,2} & \cdots & s_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ s_{m,1} & s_{m,2} & \cdots & s_{m,n} \end{pmatrix} \tag{1}$$

where s has correspondence to a given layer parameter as function of its column coordinate in S. For instance, the type of layer is mapped by:

$$s_{i,1} \in \{1, 2, 3, 4\} \mapsto \{\mathbf{C}, \mathbf{P}, \mathbf{D}, \mathbf{FC}\} \tag{2}$$

where **C** stands for convolutional, **P** for pooling, **D** for dropout, and **FC** for fully-connected layers. Both **C** and **FC** are followed by rectified linear units (ReLU), except for the last **FC** layer, which is activated with a softmax function to perform the final classification. The remaining columns of the matrix contain the values of the parameters needed for the network design. The fifth column

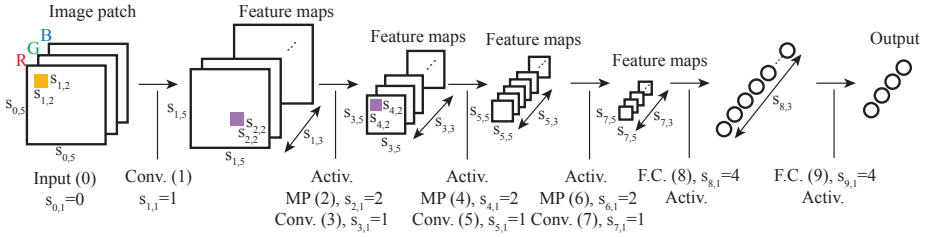


Fig. 1: Example of a Convolutional Neural Network architecture according to the proposed solution representation. ■ convolutional kernel; ■ max-pooling kernel.

stores the output size of each layer, allowing to assess the validity of the network architecture. The second column of S defines the size of the filters of **C** (3, 5, 7 or 9 in our experiments) and **MP** layers (2, 3, 4), the dropout percentage of **D** (0, 0.25, 0.50) layers and the number of neurons of **FC** (4, 16, 32, 64, 128). Like-wise, the third column indicates the number of maps of **C** layers (8, 16, 32, 64) and the fourth the L2 regularization (0.01) of **C**, **M** and **FC** layers. A CNN architecture illustrating the proposed solution representation is shown in Fig. 1

Neighbors of the current solution S are networks that result from movements of *insert*, *remove* and *swap* applied to S. A neighbor is considered valid if it respects the following rules: 1) it does not have 2 consecutive **MP** layers or **D** layers; 2) the output size of the layer before **MP** must be divisible by the **MP** filter size; 3) **MP** cannot come after **D** layers, following the traditional CNN design; 4) **FC** must appear only at the end of the CNN;5) the first layer must be **C**; *Remove* movements tend to be easier to perform. To avoid shallow architectures, these movements are penalized relatively to the other two. Similarly, *swap* movements are penalized relative to *insert* movements. For this purpose, an exponential function is used for biasing the operation decision.

2.2 Network performance evaluation

In this study, the networks' performance is evaluated on the independent test set in terms of i) accuracy (acc), corresponding to the ratio of correctly predicted image classes (Eq. 3) and ii) loss (L), which is related to how confident the CNN is in the predicted label (Eq. 4):

$$acc(u,g) = \frac{\sum_{i=1}^{N}\sum_{j=1}^{M} y_{ij}\hat{y}_{ij}}{N} \quad (3) \qquad L(u,g) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{ij} log(p_{ij}) \quad (4)$$

where u is the model in study, g the current training epoch, N is the number of test samples, M is the number of classes, $y_{ij} \in \{0,1\}$ is a binary variable with value 1 if observation i belongs to class j, $\hat{y}_{ij} \in \{0,1\}$ a binary variable with value 1 if the predicted class is j and p_{ij} the respective prediction probability.

Minimizing L is the same as maximizing the predictive power of the system, i.e, of maximizing the certainty of correct labeling over the independent test set.

Let $U = \{u_1, u_2, ...u_p\}$ be the neighborhood of the incumbent solution u_0, generated from u_0 by randomly performing r operations. New incumbent solutions are found by minimizing the cost function $C(u)$:

$$C(u) = \left(\frac{1}{v}\sum_{g=V-v}^{V} acc_{tr}(u,g)\right)^{-1}\left(\frac{1}{v}\sum_{g=V-v}^{V} acc_{tt}(u,g)\right)^{2}\left(\frac{1}{v}\sum_{g=V-v}^{V} L_{tt}(u,g)\right)^{-1} \tag{5}$$

where V is the number of training epochs, v the number of epochs to analyze and u the current network design. The first two terms of Eq. 5 state that the model should have similar accuracy performance for both training (tr) and test samples (tt), while prioritizing test accuracy. The third term minimizes the test loss, thus increasing the generalization capability of the network. A new incumbent solution u_0 is considered every time that $C(u)^{-1} > C(u_0)^{-1}$. This strategy is known as first neighbor search.

The training process has a near-constant duration that depends on the complexity of the network. Since predicting the behavior of the network based on few training epochs is a complex task [2], it is important to select V such that the obtained performance results are representative without substantially increasing the tunning time. In this work, V is manually selected by analyzing the behavior of the initial solutions.

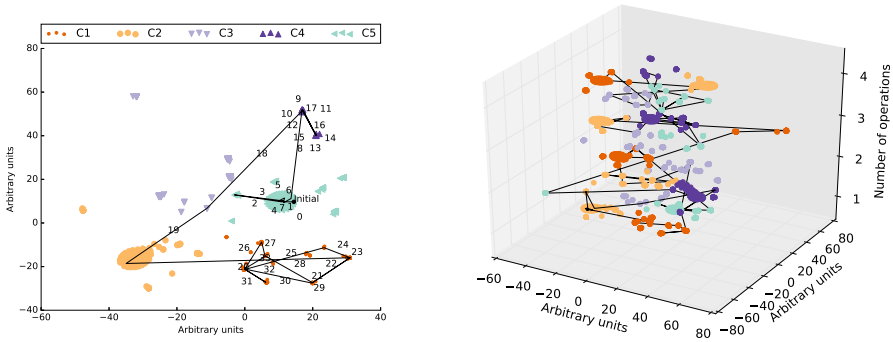
2.3 Solution space exploration

The incumbent solution's neighborhood is defined as the set of neighbors obtained from u_0 by performing r randomly selected viable operations. These neighbors are characterized by a set of features related to the architecture of the network which are used for describing the solution space: i) number of **C**, **MP** and **FC** layers; ii) average number of maps of **C** layers; iii) average filter size of **C** layers and iv) average pooling size of **MP**. Note that although these features do not fully describe the solution, their network characterization is more complete than a measurement of similarity such as the edit distance.

The dimension of the feature space is then reduced to 2D by applying a t-Distributed Stochastic Neighbor Embedding (t-SNE) [9]. Based on the natural aggregation of the neighbors, a K-means [10] approach is used for dividing the solution space. The discussed pre-processing steps allow to obtain a well defined solution space. This solution space is now searched through a variable neighborhood search (VNS) approach, as detailed in Algorithm 1.

Algorithm 1 Variable Neighborhood Search for Convolutional Neural Network design improvement. t_r: overall elapsed time; t_c: time elapsed in each cluster; $t_{r,max}$: maximum running time; $t_{c,max}$: cluster-search maximum running time; u_0: initial solution; U: set of neighbors of u_0; n_{op}: number of operations (insertion, deletion, swap) to perform; k: number of clusters Clu to form from U.

Require: u_0, $t_{c,max}$, $t_{r,max}$, n_{op}, k
  $t_r \leftarrow 0$, $loss(u_0) \leftarrow \text{train}(u_0)$
  while $t_r < t_{r,max}$ **do**
    $U \leftarrow \text{generateNeighbors}(u_0, n_{op})$, $C \leftarrow \text{cluster}(U, k)$
    for c **in** Clu **do**
      $t_c \leftarrow 0$
      while $t_c < t_{c,max}$ **do**
        $u \leftarrow \text{randomSelection}(c)$, $loss(u) \leftarrow \text{train}(u)$
        if $loss(u)$ **better than** $loss(u_0)$ **then**
          $u_0 \leftarrow u$, **break**
        end if
        update(t_c)
      end while
    end for
    update(t_r), $n_{op} \leftarrow n_{op} + 1$
  end while



(a) (b)

Fig. 2: 2D and 3D schemes of the proposed Variable Neighborhood Search method for Convolutional Neural Network design improvement. The solution space is reduced via t-SNE and clustering is performed using k-means. 2a: example of visited solutions of a neighborhood generated from the incumbent solution by performing one operation; 2b: visited neighbors with one to five operations.

The proposed method performs a search inside each of the k groups of the solution space. Each time a network with higher performance is found, as defined by Eq. 5, the incumbent solution u_0 is updated. For that purpose, the algorithm

starts by evaluating randomly selected neighbors in the same cluster of u_0. A preliminary accuracy evaluation after 3 training epochs guarantees that the current solution is viable. Otherwise, a different neighbor is explored. Similarly, redundant neighbors are skipped. When a better solution is found, or alternatively if the pre-established cluster running time is exceeded, the algorithm moves to a different cluster. After visiting all clusters, the number of operations r is incremented, a new solution space is generated and the process is repeated until the overall running time is exceeded. Fig. 2 illustrates the proposed scheme.

3 Results

The proposed VNS-based approach for CNN architecture optimization is evaluated on the CIFAR-10 dataset [7]. This dataset is composed of 50 000 training images and 10 000 test RGB images of 10 different classes. The small size of the images lowers the training time by reducing the number of parameters to learn and memory requirements, thus allowing a simpler assessment of the proposed methodology. The incumbent solution is the network from [3] (Eq. 6). The achieved architecture is shown in Eq. 7. The influence of the total number of training epochs, V, and the maximum search time per cluster, $t_{c,max}$ is assessed. Experiments were performed using a CPU Intel i7-5960X workstation, 32GB RAM and GPU Nvidia GTX1080. Python3.5 and Keras framework were used for experiment design and evaluation. Experiments were performed in GPU.

$$S_{init} = \begin{pmatrix} 0 & 0 & 0 & 0 & 32 \\ 1 & 3 & 32 & 0 & 32 \\ 1 & 3 & 32 & 0 & 32 \\ 2 & 2 & 0 & 0 & 16 \\ 3 & 0.25 & 0 & 0 & 16 \\ 1 & 3 & 64 & 0 & 16 \\ 1 & 3 & 64 & 0 & 16 \\ 2 & 2 & 0 & 0 & 8 \\ 3 & 0.25 & 0 & 0 & 8 \\ 4 & 512 & 0 & 0 & 512 \\ 3 & 0.5 & 0 & 0 & 512 \\ 4 & 10 & 0 & 0.05 & 10 \end{pmatrix} \quad (6) \qquad S_f = \begin{pmatrix} 0 & 0 & 0 & 0 & 32 \\ 1 & 3 & 32 & 0 & 32 \\ 1 & 3 & 32 & 0 & 32 \\ 2 & 2 & 0 & 0 & 16 \\ 3 & 0.25 & 0 & 0 & 16 \\ 1 & 3 & 64 & 0 & 16 \\ 1 & 3 & 64 & 0 & 16 \\ 1 & 9 & 16 & 0.1 & 16 \\ 3 & 0.25 & 0 & 0 & 16 \\ 4 & 512 & 0 & 0 & 512 \\ 3 & 0.5 & 0 & 0 & 512 \\ 4 & 10 & 0 & 0.05 & 10 \end{pmatrix} \quad (7)$$

The validation accuracy and loss metrics (which in our study corresponds to the performance in the test set) of the obtained models are compared with the initial model after 200 training epochs. For the solution search process, the influence of $t_{c,max}$ is studied for $t_{c,max} \in \{2, 4\}$ (hours) with $V = 50$. Similarly, $V \in \{20, 50\}$ (training epochs) is studied for $t_{c,max} = 2$. The total solution search time is 70 hours for each of the assessed combinations and each training epoch has an approximate duration of 30 seconds. The average number of better solutions found is 4.3 ± 0.4 over all the studies.

The obtained loss and accuracy curves for the found solutions are shown in Fig. 3a and Fig. 3b, respectively, for 120 training epochs. Fig. 3a shows that the

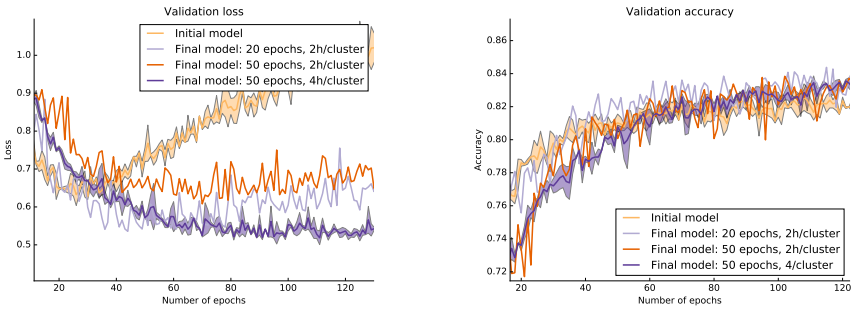
Table 1: Validation accuracy and loss of the obtained solutions for key training epochs. **S0** - initial model; **S1** - 2h/cluster with 20 training epochs; **S2** - 2h/cluster with 50 training epochs; **S3** - 4h/cluster with 50 training epochs;

(a) Accuracy

Epoch	S0	S1	S2	S3
30	0.63	**0.60**	0.73	0.67
65	0.81	**0.53**	0.63	0.54
70	0.81	0.56	0.61	**0.52**
111	0.98	0.61	0.68	**0.52**

(b) Loss

Epoch	S0	S1	S2	S3
30	**0.80**	0.79	0.78	0.77
65	0.82	**0.84**	0.82	0.82
70	0.82	**0.84**	0.83	0.83
111	0.82	**0.84**	0.83	0.83



(a) (b)

Fig. 3: Validation accuracy and loss curves of the obtained solutions. Two repetitions are performed for the initial model and for the 50 epochs with 4 hours per cluster study. 3a: validation loss of the best found solutions; 3b: validation accuracy of the best found solutions.

found solutions yield a lower and more constant loss than the initial model. The best architectures provide a higher degree of confidence in their predictions and are less prone to over-fit to the training data. Accuracy-wise, the new models show to either perform similarly or slightly better than the initial model. Overall, the found solutions are more robust than the initial model without compromising the accuracy. This shows that the proposed VNS-based heuristic is successfully capable of improving the design of the CNN.

The key results achieved for the studied V and $t_{c,max}$ values are summarized in Table 1a and Table 1b. During the 70 hours of search, in the **S1** and **S3** approaches (refer to Table 1a) approximately 240 different solutions are visited while in **S2** the number decreases to 100. The minimum achieved loss is 0.52, corresponding to an improvement of 17% relatively to the original solution loss. For 30 training epochs, number for which the lowest loss value of the original solution is obtained, the improvement is 5%. Accuracy-wise, the proposed system allows a maximum 5% accuracy improvement. For 30 epochs the obtained

accuracy is lower only because the increased complexity of the obtained solution tends to slow down the training process in terms of epochs. Furthermore, although not directly comparable because of the difference in the initial network and implementation, this value is greater than the 3% achieved by [11] for the same running time. Note that, despite the success of the proposed methodology, further improvements could be achieved by increasing the solution space to include different network optimizers, regularization values, etc. For instance, it would be easy to vary the stride of **C** layers, which has shown to be an alternative to adding **MP** layers [12]. Furthermore, the selection of the V and $t_{c,max}$ parameters seems not to be crucial since in the performed study, all the obtained models achieve high performance given enough number of training epochs.

The comparison between the studied approaches indicates that **S1** shows the best compromise between loss and accuracy improvement as function of the number of training epochs. For this model, shown in Eq. 7, a significant performance boost is achieved by doubling the network training time in relation to the initial solution. Note that in terms of CNN architecture the obtained model is uncommon and thus would not probably be studied during manual network tuning. The added 9×9 convolution layer in the end of the **C-MP** sequence is aggregating the previously learned filters into a smaller feature space, which may contribute to the improvement of the performance. However, further studies need to be done to confirm this hypothesis. Because of this, automatic design methods such as this may prove to be advantageous. The lower $t_{c,max}$ and V values, reinforced by cluster-wise division achieved via t-SNE and k-means methods, allow the VNS-based approach to visit neighbors with lower similarity, which contributes to the identification of better solutions.

4 Conclusions

An unsupervised method for CNN architecture design is proposed. A VNS-based approach is used for finding solutions that improve a problem-oriented cost function. For that purpose, t-SNE allows to effectively represent the solution space in a 2D space, which is easier to interpret and k-Means clustering divides the solution space having in consideration the distance between neighbors. The obtained solution improves the CNN loss by over 15% and the respective accuracy by 5%. The obtained solution shows higher predictive power and robustness.

Based on the achieved results, further improvements to the system can be performed. Generically, the initial solution could be randomly generated instead of being user provided. This could diversify the solution space and thus allow for further performance improvement. Ultimately, the entire network design process could be automated. It would be of interest to study classification tasks of higher complexity, such as medical image analysis. For that purpose, other relevant parameters, such as the learning rate, weight initialization, loss function, among others, could be optimized. A more complete set of features to describe the solution space could be used for describing the solution space. Similarly, the number of epochs needed to evaluate to network could be adapted to the complexity of

the network i.e., reducing the number of epochs to a minimum should allow to increase the number of neighbors visited for the same period of time.

Acknowledgements T. Araújo and G. Aresta equally contributed to this work. Project "NanoSTIMA: Macro-to-Nano Human Sensing: Towards Integrated Multimodal Health Monitoring and Analytics/NORTE-01-0145-FEDER-000016" is financed by the North Portugal Regional Operational Programme (NORTE 2020), under the PORTUGAL 2020 Partnership Agreement, and through the European Regional Development Fund (ERDF).

References

1. Ciresan, D.C., Giusti, A., Gambardella, L.M., Schmidhuber, J.: Mitosis detection in breast cancer histology images with deep neural networks. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 8150 LNCS(PART 2), 411–418 (2013)
2. Domhan, T., Springenberg, J.T., Hutter, F.: Speeding up automatic hyperparameter optimization of deep neural networks by extrapolation of learning curves. IJCAI International Joint Conference on Artificial Intelligence 2015-Janua, 3460–3468 (2015)
3. Github: Cifar 10 CNN, `https://github.com/fchollet/keras/blob/master/examples/cifar10{_}cnn.py`
4. Hansen, P., Mladenovi, N.: Variable neighborhood search: Principles and applications 130 (2001)
5. Jin, J., Yan, Z., Fu, K., Jiang, N., Zhang, C.: Neural Network Architecture Optimization through Submodularity and Supermodularity pp. 1–10 (2016)
6. Krizhevsky, A., Sutskever, I., Hinton, G.E.: ImageNet Classification with Deep Convolutional Neural Networks. Advances in Neural Information Processing Systems 25, 11061114 (2012)
7. Krizhevsky, A.: Learning Multiple Layers of Features from Tiny Images (2009)
8. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient Based Learning Applied to Document Recognition. Proceedings of the IEEE 86(11), 2278–2324 (1998)
9. Maaten, L.J.P.V.D., Hinton, G.E.: Visualizing high-dimensional data using t-sne. Journal of Machine Learning Research 9, 2579–2605 (2008)
10. Macqueen, J.: Some methods for classification and analysis of multivariate observations. Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability 1(233), 281–297 (1967)
11. Snoek, J., Larochelle, H., Adams, R.: Practical Bayesian optimization of machine learning algorithms. Advances in Neural Information ... pp. 1–9 (2012)
12. Springenberg, J.T., Dosovitskiy, A., Brox, T., Riedmiller, M.: Striving for Simplicity: The All Convolutional Net. In: ICLR 2015. pp. 1–14 (2015)