Automatic Inference of Cross-modal Connection Topologies for X-CNNs

Laurynas Karazija, Petar Veličković, and Pietro Liò

Computer Laboratory, University of Cambridge, Cambridge, United Kingdom.
laurynas.karazija@cantab.net, {pv273, pl219}@cam.ac.uk

Abstract. This paper introduces a way to learn cross-modal convolutional neural network (X-CNN) architectures from a base convolutional network (CNN) and the training data to reduce the design cost and enable applying cross-modal networks in sparse data environments. Two approaches for building X-CNNs are presented. The base approach learns the topology in a data-driven manner, by using measurements performed on the base CNN and supplied data. The iterative approach performs further optimisation of the topology through a combined learning procedure, simultaneously learning the topology and training the network. The approaches were evaluated agains examples of hand-designed X-CNNs and their base variants, showing superior performance and, in some cases, gaining an additional 9% of accuracy. From further considerations, we conclude that the presented methodology takes less time than any manual approach would, whilst also significantly reducing the design complexity. The application of the methods is fully automated and implemented in Xsertion library.[1]

Keywords: deep learning, model selection and structure learning, optimisation algorithms, evolutionary neural networks.

1 Introduction

In recent years, deep learning has become a popular approach, revolutionising various fields, including computer vision [1], agent control [2] and natural language processing [3]. However, training such deep neural nets requires vast amounts of labeled data, a limiting factor in many fields. An example of this is biomedical research, where few data examples can be obtained because the number of patients taking part in clinical studies is limited.

A proposal by Veličković et al. [4] has introduced cross-modal convolutional networks (X-CNNs) that use several constituent CNNs to process parts of wide data. This architecture has shown performance improvements in sparse data environments [5], offering a new way to handle different types of data present (different *modalities*). In X-CNNs, separate CNN super-layers are utilised to process each modality individually, producing specialised feature detectors. The

[1] Code is publicly available at https://github.com/karazijal/xsertion.

classifier part of the network is shared between super-layers. Additionally, *cross-modal connections* are established between super-layers to share information between modalities. However, this architecture carries a significant overhead – the number of design decisions concerning the network composition grows at least quadratically with the number of different types of data available. This means that such a technique is not well suited for widespread application.

This paper proposes a solution to this problem by introducing two methods to construct cross-modal architectures automatically, learning the required architecture from a base untrained convolutional network and a portion of the training data. This serves to both reduce the design effort and facilitate easy application by non-expert practitioners. Furthermore, the evaluation of the approaches showed that not only are the automatically constructed models better than hand-constructed ones, but also the additional time and design effort requirements are superior to those of the manual approaches. The next section discusses related work, followed by explanation of the methods, evaluation and conclusions.

2 Related Work

The idea to learn neural network architectures is not new and there has been substantial previous work on this. Initially, this was primarily achieved through the use of evolutionary algorithms and meta-heuristic optimisation approaches such as particle swarm optimisation [6], evolutionary programming net by Yao et al. [7], neuroevolution of augmenting topologies by Stanley et al. [8] as well as variants of neural trees by Zhang et al. [9] and Chen et al. [10] to name a few. However, all these approaches were aimed at a single MLP problem, and involved constructing ensembles of models through time-consuming runs of cross-validation whilst still operating in low-dimensional problem spaces.

Initial attempts at adjusting deep neural nets algorithmically were primarily focused on reducing the computational complexity. Molchanov et al. [11] used train-prune iterations to remove connections from CNN models. Similarly, Hu et al. [12] utilised a more data-driven approach to remove whole neurons. Wen et al. [13] proposed structured sparsity learning as a way to regularise CNNs to produce a more compact form. However, all these approaches relied on having a trained CNN and aimed to simplify its structure by removing elements for computational- and energy- efficiency with a minimal reduction in performance. This project, instead, aims to adjust and add topological elements to the CNNs algorithmically to increase performance.

Work by Zoph et al. [14], Real et al. [15] and Baker et al. [16] revisited ideas of evolutionary algorithms, applying them to deep learning by training an agent to design NNs using reinforcement learning. The approaches showed competitive results but search space could not include cutting-edge topological modifications, and came with extensive computational, data and time costs. Work presented here concentrates on introducing one such novel topology automatically and could be used in conjunction with their work to achieve state-of-the-art results in low-data availability environments.

3 Methodology

Two methods, the base and iterative approach, take as input an untrained CNN to use as a blueprint for the sort of network that would be appropriate for the dataset. Then, a portion of training data is used to infer and introduce two topological aspects of X-CNNs: separation into super-layers and introduction of cross-modal connections. The methods produce networks with similar number of parameters to the input CNN, maintaining similar computational complexity.

It should be noted that the separation of data into different modalities falls outside the scope of these approaches, and could be handled either through unsupervised methods such as clustering or done manually through domain knowledge. To carry out work here, we utilised known modalities for visual data of *luminance* and *chroma difference*, drawing inspiration from the human visual system.

3.1 Base Approach

We conducted a series of ablation experiments to investigate important aspects of X-CNN architecture. This allowed decomposing the problem of CNN→X-CNN transform into several steps that can be carried out algorithmically. In short, the base approach constructs the cross-modal architecture using measures of generalisation accuracy of each modality, to calculate hyper-parameters for each super-layer and connections between them, whilst a heuristically guided search is used to find connection positions. This reduces the design complexity from a quadratic of a number of modalities to two hyper-parameters α, β. The following details the key steps (Fig. 1) in the base approach with relevant findings from the experiments.

Modality Informativeness (Steps 1 and 2). To perform CNN $\rightarrow$ X-CNN transformation some notion of how informative modality is was required. In deep learning, near-raw data is used in the models, so existing methodologies, such as feature selection and ranking, are not suited to examine the significance of modalities. Here, we found accuracy measures performed on the base CNN using only one modality to be suitable for the task, giving a modality informativeness measure n_{l_i} for modality i.

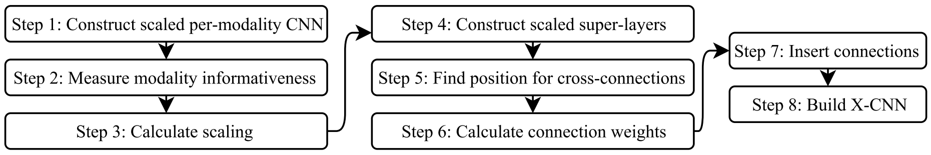
Step 1: Construct scaled per-modality CNN
Step 2: Measure modality informativeness
Step 3: Calculate scaling
Step 4: Construct scaled super-layers
Step 5: Find position for cross-connections
Step 6: Calculate connection weights
Step 7: Insert connections
Step 8: Build X-CNN

Fig. 1. The steps of the base automatic approach to creating X-CNN topology.

Super-layers (Steps 3 and 4). Super-layers are instantiated as copies of the base CNN, which take as input only one modality and share the classifier part of the network. However, experiments have shown that giving more feature maps to a more informative data split allows appropriate feature representation to be constructed whilst keeping the overall complexity of the network low. To support a variety of architectures and not be bound by dimensional constraints required by certain operations, a scaling multiplier, parametrised as follows, is used:

$$s_{l_1} = \frac{n_{l_1}^{\alpha}}{\sum_i n_{l_i}^{\alpha}}. \tag{1}$$

where α is a hyper-parameter used to tune how much higher informativeness is prioritised. This multiplier scales appropriate hyper-parameters of layers within the super-layers, such as controlling number of kernels.

Cross-modal Connections. The experiments have shown that cross-modal connections are key to capturing cross-modal interactions, without which model performance is severely degraded. The algorithm approaches building connections in three steps: firstly, finding a position where the connections could be placed, secondly, determining how super-layers should connect and, finally, determining the composition of the connection.

Position (Step 5). The place where cross-modal connections should be established in the network is more dependent on the CNN topology rather than data. We introduced a heuristic that places cross-modal connections at the ends of blocks or modules, such as commonly used downsampling operations following convolution [1,17] or various merge points used in other architectures [18,19,20]. This approach was verified using linear probe[2] methodology in [21]. Such points were shown to have a large ratio of accuracy to feature volume suggesting that a suitably dense representation exists that could be used as extra context by other modalities.

Connectivity (Step 7). The problem of placing cross-modal connections is modelled as a directed graph where nodes are points from super-layers and edges are connections (Fig. 2). It is reasonable to assume that at the same depth feature extractors of similar complexities are formed.[3] We therefore always connect points at the same depth. This also allows circumventing the problems of projection such as those encountered by the authors of [19,20], which limited their ability to optimise the computational efficiency of the network. Additionally, since the connections are concatenated rather than summed, the network maintains the ability to utilise the information passed through the connections at

[2] A linear classifier is trained using outputs of frozen intermediate layers as inputs, measuring generalisation performance.

[3] For example, it is known that nearly always the initial convolutional layers in CNNs learn to be edge extractors [22].

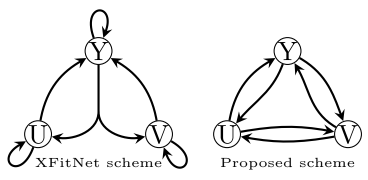

Fig. 2. Problem of placing cross-modal connections shown as a directed graph.

lower depths if that is optimal. Experiments have shown that a fully-connected graph allows learning appropriate connections between each pair of modalities and sharing information between all. This leads to better performance than a more restricted variant used in XKerasNet and XFitNet [4].

Composition (Step 6). To abstract from specifics of operations performed on the connection, we introduced a concept of a connection *weight*[4], which controls hyper-parameters of these operations. The experiments showed that connections with equal weights did not perform as well as connections that were weighted more heavily for originating in a more informative node. This is because the connections implement a mapping accomplishing three things. Firstly, they *compress* the information transferred along the connection through weighted combinations of outputs from the origin super-layer, reducing the parameter cost of destination super-layer. Secondly, they implement an *affine transformation* of features. Finally, they provide *gating* during training, which prevents gradients of a highly informative super-layer from propagating too much into a less informative one. Intermediate computations on the cross-connection soak up much of the transferred gradient to train the connection itself, enabling each super-layer to learn to be an optimal feature extractor for its respective modality.

The connection weight controls this behaviour. Thus, a desired weight $w_{l_1,l_2} \in [0,1]$ of a connection between super-layers $l_1 \rightsquigarrow l_2$ is such a number $w_{l_1,l_2} > 0.5$ for connecting a node from a more informative position to a less informative one. A sensible parametrisation of this is

$$w_{l_1,l_2} = \frac{n_{l_1}^{\beta}}{n_{l_1}^{\beta} + n_{l_2}^{\beta}}, \tag{2}$$

where β is a hyper-parameter controlling the discounting of lower informativeness. With $\beta \to 0$ all nodes are treated equally. When $\beta \to \infty$, most of the weight is assigned to the features transferred from the informative super-layer. If the weight is set to zero, the connection is dropped.

3.2 Iterative Approach

The iterative approach (Fig. 3) extends the previous methodology by adopting a learning procedure for the connection weights. It works by constructing

[4] This should not be confused with parameters of the layers themselves, in other literature sometimes referred to as weights as well.

successive generations of X-CNN models, where each generation contains models for each pair of modalities. The first generation uses equal-weighted connections, whilst second-generation uses the base approach calculations. Afterwards, a gradient ascent procedure on connection weights is performed. We adapt ideas behind Nasterov accelation for Adam optimiser [23,24] to construct an update procedure based on the generalisation measures for the connection weights. We also add weight-decay regularisation to ensure that resulting X-CNN does not grow too complex.

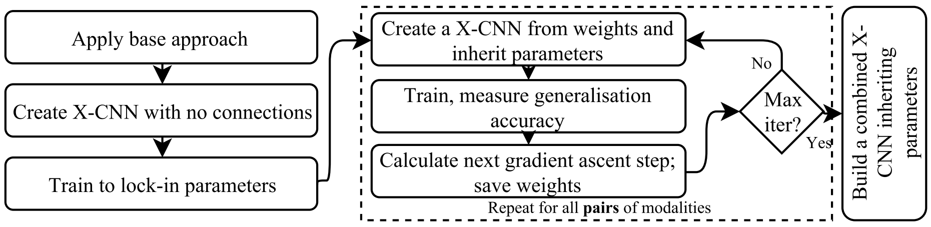
Apply base approach
Create X-CNN with no connections
Train to lock-in parameters
Create a X-CNN from weights and inherit parameters
Train, measure generalisation accuracy
Calculate next gradient ascent step; save weights
Max iter?
No
Yes
Repeat for all **pairs** of modalities
Build a combined X-CNN inheriting parameters

Fig. 3. Iterative approach for producing a trained X-CNN

Parameter Inheritance. To make this approach viable in practice, additional technique is needed to reduce the time requirements. An initial pre-training step is conducted where X-CNN without connections is trained for several epochs to lock parameter positions. Alternatively, the same random seed value can used for initialisation across models. This enables inheriting layer parameters between successive generations, making training of each generation require only a small number of updates to adapt to new connections, greatly reducing time requirement.

Combined Learning. The parameter inheritance transforms the gradient ascent procedure for connection-weights into a combined learning procedure for a full X-CNN, where both the connection-weights and model parameters are learned simultaneously. Effectively, the search is performed not in the parameter space of the resulting X-CNN, but across a combined parameter space of all potential X-CNNs with different connection weights. This is only possible because the search performed is inherently *greedy.* It assumes that a better minima can be found using better connections weights rather than by continuing to optimise parameters. In other words, combined learning only takes steps on a subset of axes at once. This could lead to a problem where parameters become too finely tuned for a particular connection configuration, "poisoning" future generations. To counteract this, we introduce a slight perturbation to the parameters between generations, by averaging across a couple of generations, which acts an additional regularisation against this type of overfitting.

Table 1. Comparison of accuracies of KerasNet based models

CIFAR-10

Model\\$^{p\%}$	20% (%)	40% (%)	60% (%)	80% (%)	100% (%)
KerasNet	70.02 ± 0.14	76.57 ± 0.16	79.28 ± 0.17	81.40 ± 0.10	82.55 ± 0.11
XKerasNet	71.00 ± 0.23	76.92 ± 0.10	79.62 ± 0.16	81.32 ± 0.10	82.68 ± 0.15
Xsertion	$\mathbf{72.02 \pm 0.70}$	$\mathbf{77.29 \pm 0.16}$	$\mathbf{79.92 \pm 0.07}$	$\mathbf{81.54 \pm 0.10}$	$\mathbf{82.93 \pm 0.09}$

CIFAR-100

Model\\$^{p\%}$	20% (%)	40% (%)	60% (%)	80% (%)	100% (%)
KerasNet	28.20 ± 0.13	36.28 ± 0.24	42.14 ± 0.56	45.40 ± 0.33	48.53 ± 0.35
XKerasNet	30.41 ± 0.32	39.32 ± 0.39	43.95 ± 0.40	47.08 ± 0.23	48.96 ± 0.22
Xsertion	$\mathbf{31.31 \pm 0.49}$	$\mathbf{40.01 \pm 0.11}$	$\mathbf{44.74 \pm 0.20}$	$\mathbf{47.75 \pm 0.27}$	$\mathbf{50.29 \pm 0.53}$

4 Evaluation

The approaches were evaluated against hand-optimised X-CNNs from [4]: XKerasNet based on KerasNet [25] and XFitNet based on Fitnet4 [17]. The datasets used were CIFAR-10/100 [26]. All models were trained using a learning rate of 10^{-3} utilising Adam optimiser [23]. Training was done using batch size of 32/128 for 200/230 epochs for Keras/Fitnet based models. Xsertion library built X-CNN topologies using KerasNet and FitNet4 as blueprints, using 80% of the training set for internal training for 80 epochs and 20% of the training set as a validation set for internal metrics. Hyper-parameters α of 1 and 2 and β of 2 and 4 were used, respectively. At each datapoint, $p\%$ of *per-class* samples were retained in the training set.

4.1 Results

Base Approach. Tables 1 and 2 detail the results. The evaluation shows that models constructed using the base approach outperform their baseline and hand-constructed counterparts on all data availability points. A significant margin is maintained between 95% confidence intervals. It is important to consider time requirements as well. In theory, the time requirement for the base approach is $O(nk)$, where n is the number of modalities and k is a function showing the time taken to train the base model. However, all models used to take measurements are scaled down by $1/n$, thus the time commitment in practice is closer to that of training the base CNN. However, if the topology were to be constructed by hand, either through a grid search or through trial and error, it would take more than a single try to arrive at a peak performing topology. Similarly, the number of hyper-parameters is reduced from $O(n^2)$ of all pair-wise connections to only two.

Table 2. Comparison of accuracies of FitNet based models

CIFAR-10

Model\\$^{p\%}$	20% (%)	40% (%)	60% (%)	80% (%)	100% (%)
FitNet	75.47 ± 0.32	82.02 ± 0.18	84.98 ± 0.20	86.22 ± 0.19	87.42 ± 0.05
XFitNet	76.56 ± 0.24	82.43 ± 0.07	85.11 ± 0.19	86.23 ± 0.18	87.42 ± 0.08
Xsertion	$\mathbf{77.35 \pm 0.15}$	$\mathbf{82.66 \pm 0.09}$	$\mathbf{85.43 \pm 0.12}$	$\mathbf{86.78 \pm 0.16}$	$\mathbf{87.77 \pm 0.22}$

CIFAR-100

Model\\$^{p\%}$	20% (%)	40% (%)	60% (%)	80% (%)	100% (%)
FitNet	29.29 ± 1.69	40.91 ± 2.48	50.94 ± 0.51	55.47 ± 0.96	58.92 ± 0.60
XFitNet	36.17 ± 0.27	48.02 ± 0.72	54.18 ± 0.36	57.98 ± 0.33	60.32 ± 0.29
Xsertion	$\mathbf{38.59 \pm 0.37}$	$\mathbf{50.11 \pm 0.30}$	$\mathbf{55.48 \pm 0.41}$	$\mathbf{59.06 \pm 0.63}$	$\mathbf{61.67 \pm 0.31}$

Iterative Approach. The iterative approach was further applied to optimise the automatically constructed networks. 15 iterations were used, training each model for a maximum of 30 epochs. On CIFAR-100 with KerasNet base CNN, the accuracy was observed to jump to 51.07% roughly 0.3% above the upper 95% confidence bar. Similarly, on CIFAR-10, the accuracy jumped to 83.36%, again roughly 0.3% above the upper 95% confidence bar. In the base approach, luminance modality Y was deemed significantly more important than V, which in turn was slightly more important than U. The iterative approach strengthened $Y \rightsquigarrow U$ connection and weakened $U \rightsquigarrow Y$. However, the converse was true for $Y \rightsquigarrow V$. It seems that it is important to transfer information from Y to U and from V to Y. Whilst connections between U and V did not disappear, they became significantly weakened both ways. For FitNet on CIFAR-10/100, the iterative approach did not yield any significant improvements. The learned connection weights were very close to those of the base approach.

Application to Residual Learning. The base approach was further applied to networks utilising residual learning [19], to see how it performs with state-of-art architectures. A variant of residual in residual network [27] was used here, which utilised pre-activations and contained 12 residual blocks. It was trained for 200 epochs, with a batch size of 64, using Adam optimiser [23], with learning rate schedule of $10^{-3}, 10^{-4}, 10^{-5}$ transitioning at 50 and 75 epochs. The layers were initialised as in [28] with 10^{-4} L_2 regularisation applied to all parameters. The base network achieved 85.72% on CIFAR-10 and 55.43% on CIFAR-100 datasets. When the base approach was applied to architecture, with $\alpha = 2, \beta = 2$, 80 epochs, the final networks achieved 88.81% and 61.33% on CIFAR-10/100, respectively, showing value and validity of methodology even for the latest networks.

5 Conclusion

We presented an way to apply the ideas of cross-modality and a library that helps to facilitate that. The results show that the automatically constructed topologies perform better than the baseline networks and hand-optimised X-CNNs, without adding additional parameters and reducing the time required to produce such topologies. The base approach successfully introduces difficult modification of CNN architecture using a data-driven procedure, which removes vast amounts of hyper-parameters that need to be considered. In that respect, the presented work reduces the complexity of applying a state-of-the-art advance in CNN design to a library call. Ideas behind combined learning procedure can be transferred to other work focused on building networks automatically from scratch, speeding up traversal of the vast search space. Hopefully, this shows that the bleeding-edge results in deep learning need not to exist in a vacuum. The ideas behind the work here can be transferred and applied to other research, resulting in a more automated field, which invites further adoption.

References

1. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems. pp. 1097–1105 (2012)
2. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., et al.: Human-level control through deep reinforcement learning. Nature 518(7540), 529–533 (2015)
3. Hinton, G., Deng, L., Yu, D., Dahl, G.E., Mohamed, A.r., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T.N., et al.: Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. IEEE Signal Processing Magazine 29(6), 82–97 (2012)
4. Veličković, P., Wang, D., Laney, N.D., Liò, P.: X-cnn: Cross-modal convolutional neural networks for sparse datasets. In: Computational Intelligence (SSCI), 2016 IEEE Symposium Series on. pp. 1–8. IEEE (2016)
5. Velickovic, P., Karazija, L., Lane, N.D., Bhattacharya, S., Liberis, E., Lio, P., Chieh, A., Bellahsen, O., Vegreville, M.: Cross-modal recurrent models for weight objective prediction from multimodal time-series data. ArXiv e-prints (2017)
6. Yao, X.: Evolving artificial neural networks. Proceedings of the IEEE 87(9), 1423–1447 (1999)
7. Yao, X., Liu, Y.: Epnet for chaotic time-series prediction. Simulated Evolution and Learning pp. 146–156 (1997)
8. Stanley, K.O., Bryant, B.D., Miikkulainen, R.: Real-time neuroevolution in the nero video game. IEEE transactions on evolutionary computation 9(6), 653–668 (2005)
9. Zhang, B.T., Ohm, P., Mühlenbein, H.: Evolutionary induction of sparse neural trees. Evolutionary Computation 5(2), 213–236 (1997)
10. Chen, Y., Yang, B., Dong, J., Abraham, A.: Time-series forecasting using flexible neural tree model. Information sciences 174(3), 219–235 (2005)
11. Molchanov, P., Tyree, S., Karras, T., Aila, T., Kautz, J.: Pruning convolutional neural networks for resource efficient transfer learning. CoRR abs/1611.06440 (2016)

12. Hu, H., Peng, R., Tai, Y., Tang, C.: Network trimming: A data-driven neuron pruning approach towards efficient deep architectures. CoRR abs/1607.03250 (2016)
13. Wen, W., Wu, C., Wang, Y., Chen, Y., Li, H.: Learning structured sparsity in deep neural networks. In: Advances in Neural Information Processing Systems. pp. 2074–2082 (2016)
14. Zoph, B., Vasudevan, V., Shlens, J., Le, Q.V.: Learning transferable architectures for scalable image recognition. CoRR abs/1707.07012 (2017)
15. Real, E., Moore, S., Selle, A., Saxena, S., Suematsu, Y.L., Le, Q.V., Kurakin, A.: Large-scale evolution of image classifiers. CoRR abs/1703.01041 (2017)
16. Baker, B., Gupta, O., Naik, N., Raskar, R.: Designing neural network architectures using reinforcement learning. CoRR abs/1611.02167 (2016)
17. Romero, A., Ballas, N., Kahou, S.E., Chassang, A., Gatta, C., Bengio, Y.: Fitnets: Hints for thin deep nets. CoRR abs/1412.6550 (2014)
18. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2015)
19. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 770–778 (2016)
20. Srivastava, R.K., Greff, K., Schmidhuber, J.: Highway networks. CoRR abs/1505.00387 (2015)
21. Alain, G., Bengio, Y.: Understanding intermediate layers using linear classifier probes. arXiv preprint arXiv:1610.01644 (2016)
22. Zeiler, M.D., Fergus, R.: Visualizing and understanding convolutional networks. In: European conference on computer vision. pp. 818–833. Springer (2014)
23. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. CoRR abs/1412.6980 (2014)
24. Dozat, T.: Incorporating nesterov momentum into adam. `http://cs229.stanford.edu/proj2015/054_report.pdf` (2016)
25. Chollet, F., et al.: Train a simple deep cnn on the cifar10 small images dataset. `https://github.com/keras-team/keras/blob/master/examples/cifar10_cnn.py` (2015)
26. Krizhevsky, A., Hinton, G.: Learning multiple layers of features from tiny images. `https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf` (2009)
27. Targ, S., Almeida, D., Lyman, K.: Resnet in resnet: Generalizing residual architectures. CoRR abs/1603.08029 (2016)
28. He, K., Zhang, X., Ren, S., Sun, J.: Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In: Proceedings of the IEEE international conference on computer vision. pp. 1026–1034 (2015)