
Deep Active Learning with a Neural Architecture Search - Supplementary Material

Anonymous Author(s)

Affiliation

Address

email

1 Related Work

Active learning has attracted considerable attention since the early days of machine learning. The literature on active learning in the context of classical models such as SVMs is extensive [3, 6, 22, 2, 1, 11], and clearly beyond the scope of this paper. Active learning of deep neural models, as we consider here, has hardly been considered to date. Among the prominent related results, we note Gal et al. [8], who presented active learning algorithms for deep models based on a Bayesian Monte-Carlo dropout (MC-dropout) technique for estimating uncertainty. Wang et al. [23] applied the well-known softmax response (SR) idea supplemented with pseudo-labeling (self-labeling of highly confident points) for active learning. Sener and Savarese [20] and Geifman and El-Yaniv [9] proposed using coresets on the neural embedding space and then exploiting the coreset loss of unlabeled points as a proxy for their uncertainty. A major deficiency of most of these results is that the active learning algorithms were applied with a neural architecture that is already known to work well for the learning problem at hand. This hindsight knowledge is, of course, unavailable in a true active learning setting. To mitigate this problematic aspect, in [9] it was suggested that the active learning be applied only over the “long tail”; namely, to initially utilize a large labeled training sample to optimize the neural architecture, and only then to start the active learning process. This partial remedy suffers from two deficiencies. First, it cannot be implemented in small learning problems where the number of labeled instances is small (e.g., smaller than the “long tail”). Secondly, in Geifman and El-Yaniv’s solution, the architecture is fixed after it has been initially optimized. This means that the final model, which may require a larger architecture, is likely to be sub-optimal.

Here, we initiate the discussion of architecture optimization in active learning within the context of deep neural models. Surprisingly, the problem of hyperparameter selection in classical models (such as SVMs) has not been discussed for the most part. One exception is the work of Huang et al. [11] who briefly considered this problem in the context of linear models and showed that active learning performance curves can be significantly enhanced using a proper choice of (fixed) hyperparameters. Huang et al. however, chose the hyperparameters in hindsight. In contrast, we consider a dynamic optimization of neural architectures during the active learning session.

In *neural architecture search* (NAS), the goal is to devise algorithms that automatically optimize the neural architecture for a given problem. Several NAS papers have recently proposed a number of approaches. In [24], a reinforcement learning algorithm was used to optimize the architecture of a neural network. In [25], a genetic algorithm is used to optimize the structure of two types of “blocks” (a combination of neural network layers and building components) that have been used for constructing architectures. The number of blocks comprising the full architecture was manually optimized. It was observed that the optimal number of blocks is mostly dependent on the size of the training set. More efficient optimization techniques were proposed in [14, 18, 19, 16]. In all these works, the architecture search algorithms were focused on optimizing the structure of one (or two) blocks that were manually connected together to span the full architecture. The algorithm proposed in [15] optimizes both the block structure and the number of blocks simultaneously.

39 When considering NAS for fully-connected networks, [4] proposed an algorithm that iteratively
40 adds neurons to an existing layer or to initiate a new layer. Their algorithm iteratively optimizes the
41 width and depth of a network. For a comprehensive survey on NAS techniques, see [5]. To the best
42 of our knowledge, no work has been done on architecture searches for active learning.

43 2 Experimental Design and Details

44 2.1 Datasets

45 **CIFAR-10.** The CIFAR-10 dataset [13] is an image classification dataset containing 50,000 training
46 images and 10,000 test images that are classified into 10 categories. The image size is $32 \times 32 \times 3$
47 pixels (RGB images).

48 **CIFAR-100.** The CIFAR-100 dataset [13] is an image classification dataset containing 50,000 train-
49 ing images and 10,000 test images that are classified into 100 categories. The image size is $32 \times 32 \times 3$
50 pixels (RGB images).

51 **Street View House Numbers (SVHN).** The SVHN dataset [17] is an image classification dataset
52 containing 73,257 training images and 26,032 test images classified into 10 classes representing
53 digits. The images are digits of house numbers cropped and aligned, taken from the Google Street
54 View service. Image size is $32 \times 32 \times 3$ pixels.

55 2.2 Architectures and Hyperparameters

56 We used an architecture search space that is based on the Resnet architecture [10]. The *initial block*
57 contains a convolutional layer with filter size of 3×3 and depth of 64, followed by a max-pooling
58 layer having a spatial size of 3×3 and strides of 2. The *basic block* contains two convolutional
59 layers of size 3×3 followed by a ReLU activation. A residual connection is added before the
60 activation of the second convolutional layer, and a batch normalization [12] is used after each layer.
61 The classification block contains an average pooling layer that reduces the spatial dimension to
62 1×1 , and a fully connected classification layer followed by softmax. The search space is defined
63 according to the formulation in Section 3, and spans all architectures in the range $A(B_r, 1, 1)$ to
64 $A(B_r, 12, 5)$.

65 As a baseline, we chose two fixed architectures. The first architecture was the one optimized for
66 the first active round (optimized over the initial seed of labeled points), and which coincidentally
67 happened to be $A(B_r, 1, 2)$ on all tested datasets. The second architecture was the well-known
68 Resnet-18, denoted as $A(B_r, 2, 4)$, which is some middle point in our search grid.

69 We trained all models using *stochastic gradient descent* (SGD) with a batch size of 128 and momen-
70 tum of 0.9 for 200 epochs. We used a learning rate of 0.1, with a learning rate multiplicative decay
71 of 0.1 after epochs 100 and 150. Since we were dealing with different sizes of training sets along
72 the active learning process, the epoch size kept changing. We fixed the size of an epoch to be 50,000
73 instances (by oversampling), regardless of the current size of the training set S_t . A weight decay of
74 $5e-4$ was used, and standard data augmentation was applied containing horizontal flips, four pixel
75 shifts and up to 15-degree rotations.

76 The active learning was implemented with an initial labeled training seed (k) of 2000 instances. The
77 active mini-batch size (b) was initialized to 2000 instances and updated to 5000 after reaching 10000
78 labeled instances. The maximal budget was set to 50,000 for all datasets¹. For time efficiency rea-
79 sons, the iNAS algorithm was implemented with $T_{iNAS} = 1$, and the training of new architectures
80 in iNAS was early-stopped after 50 epochs, similar to what was done in [21].

81 2.3 Query Functions

82 We applied the following three well known query functions.

83 **Softmax Response.** The softmax response method (SR) simply estimates prediction confidence
84 (the inverse of uncertainty) by the maximum softmax value of the instance. In the batch pool-based
85 active learning setting that we consider here, we simply query labels for the least confident b points.

¹SVHN contains 73,256 instances and was, therefore, trimmed to 50000.

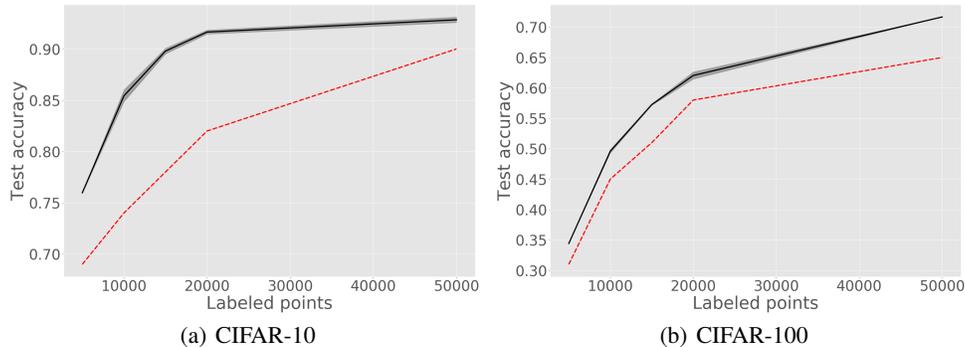


Figure 1: Comparison of active-iNAS to coreset of Sener and Savarese [20] across two datasets, (a) CIFAR-10, (b) CIFAR-100. In black (solid) – active-iNAS, red (dashed) – coreset (Sener and Savarese).

86 **MC-dropout [8, 7].** The points in the pool are ordered based on their prediction uncertainty esti-
 87 mated by MC-dropout and queried in that order. The MC-dropout was implemented with $p = 0.5$
 88 for the dropout rate and 100 feed-forward iterations for each sample.

89 **Coreset [20, 9].** The coreset method was implemented as follows. For a trained model f , we denote
 90 the output of the representation layer (the layer before the last) as ϕ_f . For every sample $x \in U$,
 91 its coreset loss is measured as $\min_{x' \in S} (d(\phi_f(x'), \phi_f(x)))$, where d is the l_2 euclidean distance. We
 92 iteratively sampled the point with the highest coreset loss with respect to the latest set S , b times.

93 2.4 Code for Active-iNAS

94 The code for all the experiments performed in the paper can be found in the following (anonymized)
 95 link. https://github.com/anonygit32/active_inas

96 3 Direct Comparison to Other Work

97 We compare performance of active-iNAS to two other deep active learning papers. We begin by
 98 considering the algorithm proposed by Sener and Savarese [20] in the the last ICLR . To compare
 99 apples to apples we use the setting proposed by these authors and apply their active training schedule
 100 consisting of five active rounds applied with 5,000, 10,000, 15,000, 20,000, and 50,000 labeled
 101 points; see [20] for details. Their algorithm is based on the coreset query function (that has also
 102 been discussed in our paper) to train a deep model with the VGG-16 architecture whose selection
 103 considerations were not discussed. In our comparison, we use the results reported in [20] for the
 104 CIFAR-10 and CIFAR-100 datasets. The learning curves in Figure 1 show their reported results
 105 (dashed, red) and ours (solid, black) applied with the softmax response. Obviously, active-iNAS is a
 106 clear winner. It is clearly evident that the active-iNAS outperforms the coreset of [20] on all budgets.

107 Motivated by similar prior-knowledge considerations as discussed here, Geifman and El-Yaniv [9]
 108 proposed the “long tail” active learning setting in which a (relatively large) labeled set is initially
 109 queried at random (i.e., passively) and used for acquiring “prior knowledge” (e.g., to select an archi-
 110 tecture). As Sener and Savarese, these authors also proposed to utilize the the coreset idea to
 111 form a querying function. Here we compare their results to our approach, in which active learning
 112 is operated from the start of the active session with iNAS. Is it the case that a large labeled training
 113 set can be used to acquire sufficient knowledge to select a useful architecture? Can our approach
 114 compete with such knowledge?

115 The results are presented in Figure 2 showing active-iNAS applied with softmax response (solid
 116 black), with coreset (dashed green) and the long-tail algorithm (dotted red). Here again, winners are
 117 clearly identified. It is striking that the first 25,000 was that were passively sampled by “long-tail”
 118 , achieved similar accuracy to 10000 points actively queried by active-iNAS in CIFAR-10 (and less
 119 than 15000 points for CIFAR-100). Concluding that the active-iNAS achieves better accuracy for
 120 any budget compared to the long tail setting.

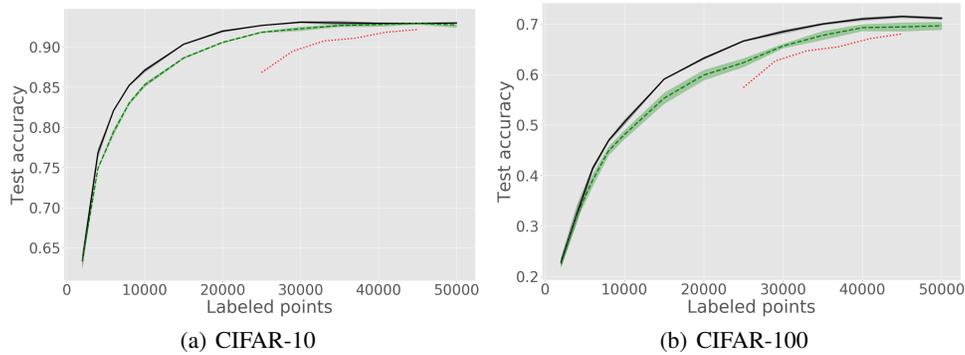


Figure 2: Comparison of active-iNAS to coreset of Geifman and El-Yaniv [9] across two datasets, (a) CIFAR-10, (b) CIFAR-100. In black (solid) – active-iNAS over softmax response, green (dashed) – active-iNAS over coreset and red (dotted) – coreset (Geifman and El-Yaniv).

121 References

- 122 [1] Maria-Florina Balcan and Phil Long. Active and passive learning of linear separators under log-concave
123 distributions. In *Conference on Learning Theory*, pages 288–316, 2013.
- 124 [2] Yoram Baram, Ran El Yaniv, and Kobi Luz. Online choice of active learning algorithms. *Journal of*
125 *Machine Learning Research*, 5(Mar):255–291, 2004.
- 126 [3] D. Cohn, L. Atlas, and R. Ladner. Improving generalization with active learning. *Machine Learning*,
127 15(2):201–221, 1994.
- 128 [4] Corinna Cortes, Xavi Gonzalvo, Vitaly Kuznetsov, Mehryar Mohri, and Scott Yang. Adanet: Adaptive
129 structural learning of artificial neural networks. *arXiv preprint arXiv:1607.01097*, 2016.
- 130 [5] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Neural architecture search: A survey. *arXiv*
131 *preprint arXiv:1808.05377*, 2018.
- 132 [6] Y. Freund, H.S. Seung, E. Shamir, and N. Tishby. Information, prediction, and Query by Committee. In
133 *Advances in Neural Information Processing Systems (NIPS) 5*, pages 483–490, 1993.
- 134 [7] Yarín Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncer-
135 tainty in deep learning. In *international conference on machine learning*, pages 1050–1059, 2016.
- 136 [8] Yarín Gal, Riashat Islam, and Zoubin Ghahramani. Deep bayesian active learning with image data. *arXiv*
137 *preprint arXiv:1703.02910*, 2017.
- 138 [9] Yonatan Geifman and Ran El-Yaniv. Deep active learning over the long tail. *arXiv preprint*
139 *arXiv:1711.00941*, 2017.
- 140 [10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition.
141 In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.
- 142 [11] Tzu-Kuo Huang, Alekh Agarwal, Daniel J Hsu, John Langford, and Robert E Schapire. Efficient and
143 parsimonious agnostic active learning. In *Advances in Neural Information Processing Systems*, pages
144 2755–2763, 2015.
- 145 [12] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing
146 internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.
- 147 [13] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. 2009.
- 148 [14] Chenxi Liu, Barret Zoph, Jonathon Shlens, Wei Hua, Li-Jia Li, Li Fei-Fei, Alan Yuille, Jonathan Huang,
149 and Kevin Murphy. Progressive neural architecture search. *arXiv preprint arXiv:1712.00559*, 2017.
- 150 [15] Hanxiao Liu, Karen Simonyan, Oriol Vinyals, Chrisantha Fernando, and Koray Kavukcuoglu. Hierarchi-
151 cal representations for efficient architecture search. *arXiv preprint arXiv:1711.00436*, 2017.

- 152 [16] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*, 2018.
153
- 154 [17] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading
155 digits in natural images with unsupervised feature learning. In *NIPS workshop on deep learning and*
156 *unsupervised feature learning*, volume 2011, page 5, 2011.
- 157 [18] Hieu Pham, Melody Y Guan, Barret Zoph, Quoc V Le, and Jeff Dean. Efficient neural architecture search
158 via parameter sharing. *arXiv preprint arXiv:1802.03268*, 2018.
- 159 [19] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. Regularized evolution for image classifier
160 architecture search. *arXiv preprint arXiv:1802.01548*, 2018.
- 161 [20] Ozan Sener and Silvio Savarese. Active learning for convolutional neural networks: A core-set approach.
162 In *International Conference on Learning Representations*, 2018.
- 163 [21] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, and Quoc V Le. Mnasnet: Platform-aware
164 neural architecture search for mobile. *arXiv preprint arXiv:1807.11626*, 2018.
- 165 [22] Simon Tong and Daphne Koller. Support vector machine active learning with applications to text classi-
166 fication. *Journal of machine learning research*, 2(Nov):45–66, 2001.
- 167 [23] Keze Wang, Dongyu Zhang, Ya Li, Ruimao Zhang, and Liang Lin. Cost-effective active learning for deep
168 image classification. *IEEE Transactions on Circuits and Systems for Video Technology*, 2016.
- 169 [24] Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. *arXiv preprint*
170 *arXiv:1611.01578*, 2016.
- 171 [25] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning transferable architectures for
172 scalable image recognition. *arXiv preprint arXiv:1707.07012*, 2(6), 2017.