Supplementary Material: Online Incremental Feature Learning with Denoising Autoencoders

Guanyu Zhou
Department of EECS
University of Michigan
Ann Arbor, MI 48109
guanyuz@umich.edu

Kihyuk Sohn
Department of EECS
University of Michigan
Ann Arbor, MI 48109
kihyuks@umich.edu

Honglak Lee
Department of EECS
University of Michigan
Ann Arbor, MI 48109
honglak@eecs.umich.edu

1 Discussion of the Update Rules

We introduce the update rules that were considered in this work and discuss the robustness of each update rule in relation to its hyperparameters.

1.1 Update rule based on heuristics

Roughly speaking, this update rule is based on the following idea: increase the number of feature increments when the performance improves (i.e., the model is not at optimum), and decrease the number of feature increments when there is minimal or no performance improvement (i.e., the model has converged). From this intuition, we consider the following update rule (referred to as "update rule I"):

$$\Delta N_{t+1} = \begin{cases} \Delta N_t + 1, & \frac{e_t}{e_{t-1}} < (1-\epsilon_1) \\ \lfloor \Delta N_t / 2 \rfloor, & \frac{e_t}{e_{t-1}} > (1-\epsilon_2) \\ \Delta N_t, & \text{otherwise} \end{cases}, \quad (1)$$

$$\Delta M_t = \lceil \gamma \Delta N_t \rceil, \quad (2)$$

where ΔN_t is the number of feature increments, ΔM_t is the number of merged features, and e_t is the objective function value for recent 10,000 training examples at time t.

We have four hyperparameters $\epsilon_1, \epsilon_2, \gamma, \Delta N_0$ in this rule. It is quite evident from equation (1) that ϵ_1 and ϵ_2 adjust the pace of feature increments, where we accelerate by decreasing ϵ_1 or increasing ϵ_2, and vice versa. Further, we can control the rate of convergence by tuning γ and ΔN_0. Ideally, these hyperparameters should be determined through cross-validation, which makes the algorithm quite complicated. However, we found through several controlled experiments that the proposed update rule is fairly insensitive to the selection of hyperparameters. For example, given the reasonable value of $\gamma = 0.5$ and $\Delta N_0 = 20$, the incremental feature learning with update rule I resulted in comparable classification accuracies for $\epsilon_1 \in [0.005, 0.05]$ and $\epsilon_2 \in [0, 0.02]$ (e.g., within 1% difference on bg-img-1M dataset).

Algorithm 1 Update rule II

Given a batch of data (e.g., 10,000 examples),
repeat
  Add k new features and compute the validation performance.
until the validation performance decreases
Merge $\Delta M_t = \lceil \gamma \Delta N_t \rceil$ features.

1.2 Alternative update rules

Although the update rule I has shown robustness to its hyperparameters, the learning algorithm with many hyperparameters is generally difficult to use since it requires additional efforts in finding their correct values. In this section, we consider two alternatives that involve fewer hyperparameters.

Before we discuss about the update rules, we first describe how we performed *validation* in online setting. At each iteration, we take the current batch of online data for training and the next two batches of data for validation and testing, where the first one is used for validation, and the second one is used for testing. Specifically, after updating the parameters with the current "training" batch, then these three batches are *shifted* by one (i.e., in the next iteration, "testing" batch becomes "validation" batch, "validation" batch becomes "training" batch, and we fetch the next batch for "testing"). In other words, each batch is used first for testing, then for validation, and finally for training. This ensures that our training and validation procedure does not give any unfair advantage to the online testing performance.

The first approach is to select ΔN_t that minimizes the objective function value (i.e., classification error) given a current data batch. For example, at each iteration, we keep adding k features until the validation classi-

Algorithm 2 Update rule III

Given a batch of data, add one feature and compute the validation performance.
if the validation performance decreases **then**
  Remove the newly added feature.
end if

fication accuracy begins to decrease. We denote this method as an update rule II. We also evaluated with $\gamma = 0.5$ for the update rule II.

The update rule III is motivated by the forward stagewise additive models [2, 3], which shares a similar flavor with AdaBoost [1]. In update rule III, we train with a single new feature at each iteration and determine whether to keep the feature based on the performance without merging process.

In our experiments, the update rules II and III showed similar classification performance to that of the update rule I (e.g., classification errors within 1% difference for bg-img-1M dataset), which assures that the proposed incremental feature learning algorithm is fairly robust to several variants of update rules.

1.3 Update rule based on information criteria

Finally, we introduce an update rule using an approximate Bayesian method (e.g., regularizing with Akaike Information Criterion (AIC) or Bayesian Information Criterion (BIC)) that penalize on the model complexity. To be more specific, we describe the updating procedures as follows:

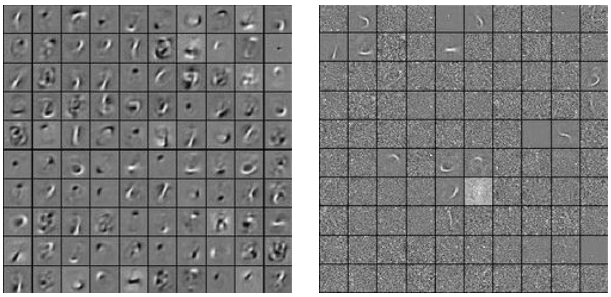
1. At each batch of online data, propose q different models for the number of feature increments $\{(\Delta N, \Delta M)_i\}_{i=1}^q$ and initialize them.
2. Evaluate the objective function penalized by either of the following information criterion penalty for each proposed model (described in more detail below):

$$\text{AIC}: \quad \mathcal{L}_{hybrid}(\mathbf{x}, \mathbf{y}) + M \tag{3}$$

$$\text{BIC}: \quad \mathcal{L}_{hybrid}(\mathbf{x}, \mathbf{y}) + \frac{1}{2} M \log N, \tag{4}$$

   where M is the number of adjustable parameters (proportional to the number of features) and N is the number of training examples in each batch.
3. Accept the model with the best validation performance.

We used $q = 3$ and set $\{(\Delta N, \Delta M)_i\}_{i=1}^q$ as constant values for our experiments. In step 2, we adaptively switch between the AIC and BIC penalties to accelerate the convergence based on the following idea: depending on our decision in the previous iteration, if we have chosen to increase the number of features, we select the AIC penalty, which is weaker than the BIC penalty since it doesn't depend on the number of training examples; in a similar manner, we select the BIC penalty if we have picked the model that decreases the number of features in the previous step.



(a) (b)

Figure 1: Visualizations of the learned filters using (a) IncMDAE and (b) DAE on bg-rand-1M dataset

As seen from the Figure 3 in the main paper, the incremental learning algorithm with this update rule converges to a similar number of features regardless of the initial number of features. Moreover, they also result in similar classification accuracies within 1% difference.

2 Visualization of the Filters

For qualitative evaluation, we visualize sets of filters learned using incremental and non-incremental DAEs on bg-rand-1M datasets. As Figure 1(a) shows, most of the filters learned using IncMDAE captured meaningful "pen-stroke" bases, whereas the filters learned using the baseline DAE are typically noisy (Figure 1(b)). They are similar to the background patterns of the training examples and contain only a few pen-stroke bases. We believe that these visualized filters partially reveal the reason our model achieved less significant improvement in incremental learning over the non-incremental counterpart in the generative performance measure (reconstruction error) than in the discriminative criteria (classification error). In fact, the reconstruction error can be easily reduced by simply adding more hidden units in the DAE (i.e., by learning more background patterns), as suggested by Figure 1(b). However, our incremental feature learning method learns new features from the most difficult subset of the data; in classification tasks, this subset consists of misclassified examples. Therefore, the new features learned from these difficult examples help learning a better decision boundary, and our model can outperform the baseline model in classification.

	Extended datasets	MNIST variation datasets
Foreground digits	randomly sampled from MNIST-8M (with deformations)	selected from MNIST
Background images	randomly sampled from a subset of 2,000 images from CIFAR-10	randomly sampled from a set of 20 images downloaded from the internet
Foreground rectangles (rect-img-1M)	randomly extracted (with random widths and heights) from a disjoint subset of another 2,000 images from CIFAR-10	one of the same 20 images downloaded from the internet except the one used for background

Table 1: Comparison between extended and original dataset

Model	DBN-3			SAE-3			SDAE-3		
dataset	Ext.	Orig.	Ref. [6]	Ext.	Orig.	Ref. [6]	Ext.	Orig.	Ref. [6]
rot-bg-img	60.90	48.19	47.39	66.47	52.97	51.93	61.09	46.12	44.49
rot	25.29	10.81	10.30	29.08	11.06	10.30	27.02	10.66	10.29
bg-rand	15.51	7.30	6.73	20.11	13.19	11.28	17.07	10.39	10.38
bg-img	24.97	18.24	16.31	32.20	24.07	23.00	24.07	17.14	16.68
rect-img	33.25	23.16	22.50	34.16	25.11	24.05	32.07	21.68	21.59

Table 2: Classification errors of baseline methods on several datasets

* Extended (Ext.): The performance of the models implemented on the datasets with 10,000 training examples and 50,000 test examples (same number of examples in MNIST variation dataset) sampled from our extended datasets.
* Original (Orig.): The performance of the models implemented on MNIST variation and rect-img datasets.
* Reference (Ref.): The performance of the models reported in [6] on MNIST variation and rect-img datasets.

3 Extended MNIST Variation Datasets

The large-scale dataset used in the paper was extended from the original benchmark[1] introduced in [6] based on a similar process. However, the details are not identical to the previously published datasets. Here, we clarify the differences in Table 1.

To generate bg-img-1M, bg-rand-1M, rot-1M and rot-bg-img-1M datasets, we randomly selected 1 million digit images from MNIST-8M dataset [5] and applied corresponding variations; for bg-img-1M and rot-bg-img-1M, we randomly selected 2,000 images from CIFAR-10 dataset [4] and set them as a background; for bg-rand-1M, we used uniform random noise as a background; for rot-1M and rot-bg-img-1M, each digit was rotated at a random angle.

For rect-img-1M, we generated 1 million rectangular shapes with random width and height on top of randomly selected 2,000 images (background) and then filled the rectangular shapes with another randomly selected 2,000 images (foreground). All the foreground and background images were from CIFAR-10 dataset.

As Table 1 suggests, the extended MNIST variation dataset involves more diverse background images and deformed digits. To assess the datasets' difficulty, we test DBN-3, SAE-3 and SDAE-3 on both the original MNIST variation benchmark and the extend dataset with the limited number of training and testing examples as the original dataset (i.e., 10,000 for training and 50,000 for testing). The summary results are shown in Table 2. Our implementations of the existing baseline models are comparable to [6] with a small performance difference. Moreover, given the same amount of training and testing data, we can see that the extended dataset is more challenging than the original benchmark datasets.

[1] `http://www.iro.umontreal.ca/~lisa/twiki/bin/view.cgi/Public/MnistVariations`

References

[1] Y. Freund and R. E. Schapire. Experiments with a new boosting algorithm. *ICML*, 1996.

[2] J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: a statistical view of boosting. *The annals of statistics*, 28(2):337–407, 2000.

[3] T. Hastie, R. Tibshirani, and J. H. Friedman. *The elements of statistical learning*. Springer, 2009.

[4] A. Krizhevsky. Learning multiple layers of features from Tiny Images. Master's thesis, University of Toronto, 2009.

[5] G. Loosli, S. Canu, and L. Bottou. Training invariant support vector machines using selective sampling. In *Large Scale Kernel Machines*, pages 301–320. MIT Press, 2007.

[6] P. Vincent, H. Larochelle, Y. Bengio, and P. Manzagol. Extracting and composing robust features with denoising autoencoders. In *ICML*, 2008.