Automated Curriculum Learning

Nimit Sohoni, Halwest Mohammad, Rahul Palamuttam
{nims,halwestm,rpalamut}@stanford.edu

PROBLEM

It is well known that the human brain learns best by perceiving progressively more complex examples of a concept. Similarly, it has been observed a curriculum of progressively more complex examples can accelerate training in neural networks as well. **Curriculum Learning** attempts to train deep networks in phases of progressively more difficult objectives. We propose a methodology for automating curriculum learning using simple classifiers to estimate the difficulty of training examples and thereby automatically design a curriculum, and present our current results.

METHODS

We want to train a classifier f_W for some task. To help us train this model, we will have a simple classifier g_θ that takes an input and calculates a confidence score for each of the classes. We will define the *difficulty* of a training example (X,Y) [where X is the input and Y is the correct class label] with respect to g_θ as $-g_\theta(X)_Y$. So, an example is *difficult* with respect to g_θ if g_θ gives a low score to the correct class and conversely, *easy* if it gets a high score for the correct class.

The first step of the framework is then to train the model g_θ on the training data. After we have done this, we sort the data into k bins by their difficulty with respect to g_θ: $B_1, \ldots, B_k$, where all the B_i's have (roughly) the same number of training examples and the examples in B_j are more difficult with respect to g_θ than those in B_i if $j > i$. We then split the training into k stages, where at stage j we train f_W on all examples in $\bigcup_{i \le j}$ using batch stochastic gradient descent.

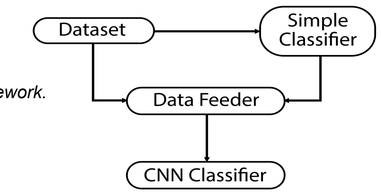


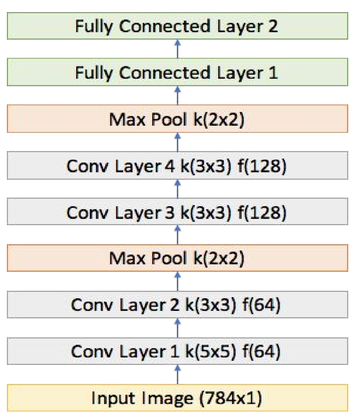
Fig. 1. *High-level framework.*

Main CNN Model

We first pass the image through **two convolutional layers**, with 64 output channels each, and a kernel size of 5x5 and 3x3 respectively. The output of 2nd convolution layer is fed into a max pool layer with a 2x2 kernel. From here we the output is passed through two more convolutional layers each with 128 channels and 3x3 kernels.

We then pass the signal through a final max pooling layer with a 2x2 kernel and feed it to **two fully-connected (FC) layers**. The first fully connected layer mapped the embedding to a 1024 vector, and the second mapped the 1024 size vector to the class mappings, i.e. a size 10 vector. We used 50% **dropout** while training as well as L_2 weight **regularization**. We initialized our weights using Xavier initialization. We used the leaky ReLU (LReLU) activation function with parameter 0.01: *LReLU(x) = max(x, 0.01x)*. The final activation function is a softmax (as is typical for multiclass prediction tasks).

Fig. 2. *CNN Architecture.*



Simple Models

We experimented with the following simple classifiers to compute difficulty scores: **Logistic Regression**, **Multi-class SVM**, **Kernelized Multi-class SVM*** (Gaussian RBF)**, k-Nearest Neighbors*** (majority vote prediction), and **Shallow CNN** (1 conv layer, 2 FC layers).

Note: Due to the high (quadratic+) computational cost, the Kernelized SVM and KNN "basic" classifiers did not finish training in time.

DATASETS

MNIST: 70,000 28px *x* 28px single-channel images, each in 1 of 10 classes. [55k train / 5k dev / 10k test]
[784-dimensional input data]
CIFAR-10: 60,000 32px *x* 32px tri-channel images, each in 1 of 10 classes [50k train / 3k dev / 7k test]
[3,072-dimensional input data]
All attributes are normalized to lie in [0,1].

Noise: We added varying levels of noise to our training and validation examples, in order to artificially make the (fairly basic) task "harder" and to investigate the robustness of the networks trained with curriculum learning to those without. *Procedure*: For each example X_i, an value K_i is independently drawn uniformly from the interval [0,S], where S is a fixed pre-specified number. IID noise values drawn from $N(0, K_i)$ are then added to each attribute of X_i (i.e. each channel of each pixel). Each attribute is clipped to remain in [0,1]. See **Fig. 4** for example noised MNIST images.

RESULTS

For CIFAR, the maximum noise scale S was fixed at 5. For MNIST, it was fixed at 20 [due to the easiness of the task].

The logistic regression (LR) and shallow CNN (SCNN) models show similar performance boosts on both tasks. SVM, however, performs no better than random.

We compute validation accuracies on both noised and un-noised validation data.

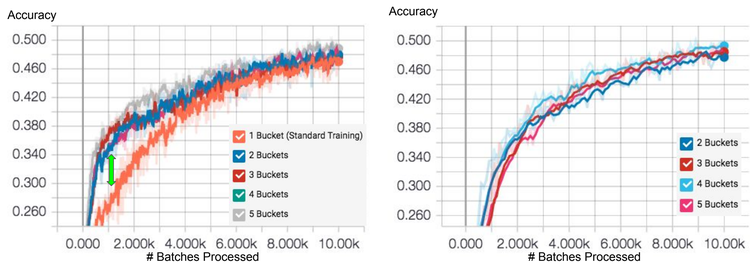


Fig. 5. *Un-noised* validation accuracy on CIFAR data for CNN with LR (left) and a SCNN (right) as simple classifiers

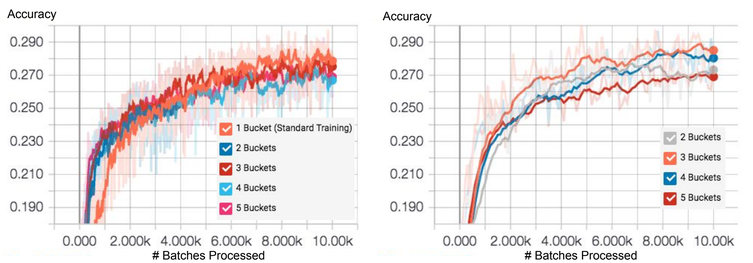


Fig. 6. *Noised* validation accuracy on CIFAR data for CNN with LR (left) and a SCNN (right) as simple classifiers

Fig. 3. *MNIST (left, occluded) and CIFAR (right)*

Fig. 4. *MNIST noised example with low random noise variance K_i (top) and example with high K_i (bottom). The extreme variation in noise between these two examples leads to the bottom one having a much higher difficulty score with respect to all tested basic classifiers.*

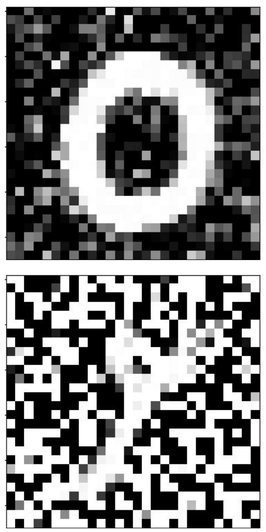


Table 1. Model performance on noised MNIST task, compared to simple classifier training accuracy.

	Simple Classifier Training Accuracy (MNIST)	(Final CNN Val Accuracy, # Batches to 80%, to 90%): 2 Buckets	(Final CNN Val Accuracy, # Batches to 80%, to 90%): 3 Buckets
LR	0.2534	(0.93, 2000, 2500)	(0.94, 500, 800)
SVM	0.1121	(0.11, –, –)	(0.82, 2900, –)
ShallowCNN	0.4068	(0.93, 900, 1700)	(0.94, 500, 900)

CONCLUSIONS & FUTURE WORK

This automated approach to curriculum learning shows the potential to decrease network training time. However, the approach requires additional hyperparameters to consider when tuning models, especially the choice of basic classifier and number of buckets. Our experiments have shown that we can make a learning problem more challenging by adding random noise with differing levels of variance to the dataset, and correspondingly show a greater benefit to curriculum learning, though this is admittedly somewhat contrived and may not even work for all datasets (particularly those with discrete-valued features). Despite these drawbacks, this approach is promising because it may allow deep learning practitioners to expend less effort on data selection in the future, being able to simply apply the curriculum approach instead to filter bad examples. Moreover, we believe that this adaptive curriculum learning approach could be a promising avenue for future research. Our next goal is to try sampling from examples at a frequency inversely proportional to their difficulty level, with the sampling frequency for hard examples increasing as training proceeds [rather than splitting examples into discrete buckets as we currently do].

REFERENCES

[1] Y. Bengio, N. Boulanger-Lewandowski, and R. Pascanu. Advances in optimizing recurrent networks. In ICASSP, pages 8624–8628. IEEE, 2013.
[2] L. Deng. The mnist database of handwritten digit images for machine learning research. IEEE Signal Processing Magazine, November 2012.
[3] G. Hinton. Neural networks for machine learning - lecture 6a - overview of mini-batch gradient descent. 2012. [7] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. CoRR, abs/1412.6980, 2014.
[4] T. Matiisen, A. Oliver, T. Cohen, and J. Schulman. Teacher-student curriculum learning. CoRR, abs/1707.00183, 2017.
[5] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. S. Bernstein, A. C. Berg, and F. Li. ImageNet large scale visual recognition challenge. CoRR, abs/1409.0575, 2014.