Starting Big: Exploring the effects of curriculum learning on generalization

Michael Laielli Giscard Biamby David Fair

1 Introduction

Inspired by recent studies in cognitive development, we design a curriculum specialized for visual object classification. We explore whether starting the training with objects that occupy a large percentage of image pixels might result in better generalization, particularly when the number of training examples is very low, since the model will be forced to discover object-based features from the outset (as opposed to context-based features). We posit that object-based features are better for generalizing to novel categories with only a few examples because object appearance varies less than all likely background appearances. We create a new benchmark to better study the effects of our curriculum on convergence and generalization on visual object recognition.

Curriculum learning can be generalized as an attempt to provide structure to the challenge of classifying data in a training set such that the order in which samples appear during training follows some logical sequence from easy to difficult. This training procedure was derived from applying modern pedagogical strategies (where problems are assigned in an increasingly difficult manner) to the field of machine learning with the goal of reducing model convergence time and improving the quality of the loss minima found during gradient descent.

What characterizes a curriculum
One of the more critical design choices for implementing this training strategy is determining how the difficulty metric (which determines ordering) is defined. This definition can either be highly specific to the data that you are working with or generalized to a broader array of classification problems. We approached this learning strategy from both angles, defining a curriculum based upon the ratio of an object's size relative to the contextual field surrounding it (specific to image classification) and by defining difficulty in real time based upon the individual loss contributions of each image from the previous epoch (which can be broadly applied to different classification problems). Once a curriculum has been defined, how it is applied to sampling the dataset during training can then be determined. To this end we experimented with defining discrete difficulty buckets (either based upon object size ratio or loss contribution) and by a more continuous method of sampling from the training dataset via a normal distribution where we scaled difficulty based upon an image's deviation (in difficulty) from the sample mean.

Object-to-context pixel ratio introduction
With this curriculum strategy we first needed to attain a dataset which had object positions within an image pre-classified. To resolve this constraint we utilized the COCO dataset which comes pre-labeled with object boundary boxes for every image in the dataset. We then calculated the ratio of the pixels comprising the object within the image and the remaining (context) pixels surrounding it. Intuitively, an object with a smaller ratio would be harder to identify as it comprises a much smaller

portion of the image and would therefore be harder to match to the filters within our neural network. This, at least, was our initial assumption heading into the project.

Real-time loss curriculum introduction
Our results from the previous curriculum experiments provided motivation for us to move away from an intuitive definition of a difficulty metric and more towards one that is dictated by the model itself. Our first attempt at letting the model dictate difficulty involved finishing an entire model run and then ordering images based on individual loss contributions during the final epoch of training. After the difficulty order had been determined, a second training cycle was initiated which weighted sampling based on this ordering of loss contributions. We quickly found that our difficulty metric wasn't relevant to starting an entirely new training cycle as it was determined in the later epochs of the previous model run. To combat this, we tried averaging the loss contributions of each image over every epoch, but again this resulted in lackluster performance. We finally settled upon a constantly evolving difficulty metric, one that is determined by the previous epochs loss values. This avoided the requirement of finishing an entire cycle of training to determine a curriculum and proved to yield higher testing accuracies than the aforementioned strategies.

2 Related work

Cognitive development
Egocentric vision in cognitive development} Recent work in Cognitive Development by Bambach et al. (2017) suggests that, visually speaking, toddlers prefer to start big. More so than their parents, toddlers actively manipulate objects to get close-up views of new objects. They also show that convolutional neural networks perform better when trained on a set of images collected from a child's point of view than from a parent's during collaborative play with novel objects.

Curriculum learning
Elman (1993) showed that when training neural networks "starting small" can help i.e. starting with easy training exemplars before progressing to difficult ones can have a positive effect on the performance of neural networks. Bengio et al. (2009) explored this idea further on vision and language tasks, proposing that curriculum learning can be viewed as a continuation method and confirming that curriculum learning can result in faster convergence and better generalization.

Low-shot learning
Harihan and Girshick (2016) proposed a new benchmark for evaluating the ability of vision algorithms to generalize in the case low amounts of training images. In contrast to most other "few-shot learning" research where the model learns from scratch on a dataset with a low number of samples across many categories. Harihan and Girshick model the more realistic scenario where the model can be initialized on a base set with many examples before being applied to the task of generalizing with few exemplars.

3 Creating CBAS-34: A benchmark for size-based curriculum learning

Lin et al. (2014) created the COCO dataset to pose a difficult computer vision task that involves multiple object detection on non-iconic images. COCO features full object segmentations which we use to crop out images for 68 object categories and compute the ratio of objects pixels to background pixels. This ratio allows us to build a curriculum by ordering training images according to object size.

We scale the images cropped from the COCO dataset to a size of 32 by 32 pixels for ease-of-computation and for comparability to the popular CIFAR-10 image dataset which features 50,000 images (32 by 32px) across 10 categories.

We split the 68 categories (34/34) into a base set and (low-shot) novel set, similar to the benchmark proposed by Harihan and Girshick (2016). The CBAS-34 base set has 34 categories with 1,500 training images and 150 validation images per category. The CBAS-LS set has 34 novel categories with 100 training images and 150 validation images per category.

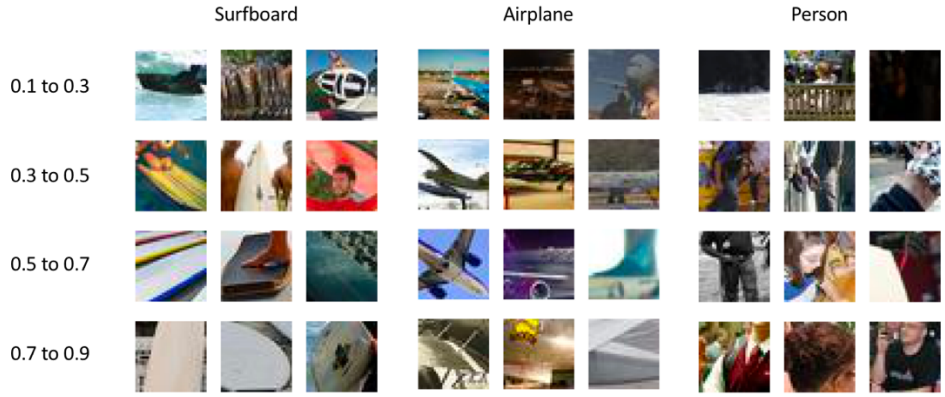

Figure 1: Samples from CBAS-34 arranged according to size ratio (left side)

Our baseline experiments show that CBAS-34 is relatively difficult task (Table 1). We show common architectures that do very well on CIFAR-10, see a significant drop in performance on CBAS-34. We chose LeNet, AlexNet, and simple 3-layer convolutional neural network for baseline experiments. For the 3-layer convnet, we decreased the kernel size in the first layer and added dilation to the last two layers to increase the receptive field, leverage context, and preserve signals from small object getting lost. Surprisingly, the 3-layer convolutional neural network does significantly better than AlexNet with many more parameters when both are trained to convergence.

Table 1: Baseline performance on CBAS-34

Model	Params (M)	CBAS-34 Top-1	CBAS-34 Top-5	CIFAR-10 Top-1	CIFAR-10 Top-5
LeNet	0.06	26.14	59.54	69.60	97.49
3-layer conv	0.07	36.16	70.48	82.92	99.18
AlexNet	2.47	30.31	63.77	77.22	98.47

4 Results and discussion

Convergence and generalization

While we were not able to show improvement in generalization over training regularly with no curriculum, we do see the *Middle-out* and *Start big i.e. the easy-to-hard curricula*, generalizing better than *Random* and *Start small*. (Table 2)

Table 2: Curriculum learning on base classes with large n

Curriculum	Model	Iterations	Top-1 Accuracy	Top-5 Accuracy
None	3-layer conv	255,00	36.16	70.15
	AlexNet	1,338,750	30.31	58.20
Random	3-layer conv	153,000	28.13	60.13
	AlexNet	306,000	22.98	52.65
Middle-out	3-layer conv	153,000	31.66	63.25
	AlexNet	306,000	24.04	56.55
Start small	3-layer conv	153,000	26.58	58.42
	AlexNet	306,000	23.46	53.49
Start big	3-layer conv	153,000	29.03	60.55
	AlexNet	306,000	23.55	53.03

Real-time loss experiment results

Once settling upon our new strategy we then devised a method of applying the curriculum to training in variety of ways. After experimenting with iterating through the training set in a purely sequential manner (dictated by loss values from the previous epoch in ascending order), we found that randomized sampling was still necessary. This is where we implemented our strategy of defining a set of sampling weights to be applied to the entire dataset based on 'difficulty'. We sampled with replacement so that the set of images utilized in each epoch was different, and we defined the weights using the probability density function of a normal distribution with a sample mean and variance determined by loss values. Throughout training, we shifted the mean of this sampling distribution from lower loss values to higher loss values progressively over the course of all epochs. We accomplished this shift by altering the mean of the sampling distributions to be +/- a certain number of standard deviations based on which epoch of training was currently underway. By starting the sampling distribution out a few (typically 2) standard deviations below the sample mean, we were able to scale the difficulty of training images up as intended. This resulted in faster convergence and higher testing accuracy (typically 1-2% more than traditional training methods).

Real time curriculum	No Curriculum
Alexnet: 0.21	AlexNet: 0.22
3-layer conv: 0.30	3-layer conv: 0.29
LeNet: 0.21	LeNet: 0.20

Low-shot Generalization

We ran experiments on the few-shot learning task using the cbas34 dataset as the base set, and the cbasLS as the novel set. We trained several convolutional networks end-to-end on the task of classifying images from the 34 categories within the base set. Once the CNN models were trained we sent images through a forward pass of the models and extracted vectors in the final later after the convolutions, before the classification layers. The choice of where exactly to extract the vectors was arbitrary, but we used the last layer to get features that give a higher overview of the image. For AlexNet the vector embeddings were 256-dimensional, and for DileNet the embeddings were 4096-dimensional. We would like to further experiment with results extracting the vectors from different layers in the network, and with more network architectures.

After training the CNN's on the cbas34 train set, we trained a k-Nearest Neighbors Classifier with k=3. We used k=3 so that the models trained with N=1 or 2 would have a roughly similar number neighbors in the embedded vector space (where N is the number of few-shot examples per category that the CNN's were trained with) as the models with N>2.

We trained a total of 7 CNN's, each one either being either AlexNet or DileNet, and each using one of the following curriculum methods: None, Starting Big, Starting Small, or Middle Out, and then. For each of these seven models we trained a knn classifier with N=(1, 2, 5, 10, 20) training samples from each category in the novel set, and then evaluated prediction accuracy against the cbasLS (novel set) training split. Our results are listed in table

Table 3: Top-1 accuracy for low training data (n)

Curriculum	Model	n=1	n=2	n=5	n=10	n=20
None	3-layer conv	5.22	7.03	8.61	9.98	12.10
	AlexNet	9.08	6.83	9.40	11.34	13.09
Start big	3-layer conv	3.84	3.40	3.71	4.43	5.92
	AlexNet	4.57	4.73	6.54	7.59	9.23
Middle-out	3-layer conv	4.69	4.50	6.05	4.63	7.20
	AlexNet	5.09	6.55	7.26	9.57	10.43

REFERENCES

In developing this project, we consulted with the following people at Berkeley: Tom Griffiths, Alison Gopnik, Trevor Darrell, Fisher Yu, and Anja Rohrbach.

Jeffrey L. Elman. Learning and development in neural networks: the importance of starting small. *Cognition*, 1993

Yoshua Bengio, Jerome Louradour, Ronan Collobert and Jason Weston. Curriculum Learning. *ICML*, 2009

Sven Bambach, David J. Crandall, and Linda B. Smith and Chen Yu. An egocentric perspective on active vision and visual object learning in toddlers. *Seventh Joint IEEE Conference on Development and Learning and on Epigenetic Robotics*, 2017

Tsung-yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan and C. Lawrence Zitnick Microsoft COCO: Common Objects in Context. 2014

Krizhevsky, Alex and Sutskever, Ilya and Hinton, Geoffrey E. ImageNet Classification with Deep Convolutional Neural Networks 2012

Barath Harihan and Ross Girshick. Low-shot Visual Recognition by Shrinking and Hallucinating Features. 2016

Yang, W. (2018) "pytorch-classification" [software]. https://github.com/bearpaw/pytorch-classification.git

Code Appendix

Description of our code repository. Our code repository is located at https://github.com/mlaico/cbas/. There are instructions there on the main page on how to set up and running with the code base. But we would like to also point out some key files in the code, especially our Jupyter notebooks, some of which have a tutorial aspect to them.

Notebooks:

coco/cbas/cbas34_real_time_loss_curriculum.ipynb
(Tutorial aspect)

coco/cbas/Real_time_curriculum.ipynb
(Tutorial aspect)

coco/few_shot_embedding.ipynb
(Tutorial aspect) Walks the reader through the few-shot learning process. covers embedding, and knn classification for few-shot learning.

coco/create_cbas-80_-34_and_-LS.ipynb
This notebook was used to create the cbas34 and cbasLS datasets

Scripts:

coco/pytorch-classification/few_shot_testing.py
Script to generate few-shot experiment results

After you have set up the coco/cbas/ folder (see the readme.md in our repo), you should have the following structure: