Data Sampling Algorithms for Active Learning

Pedro Mantica, Josh Hellerstein

***Abstract*— One of the greatest challenges of using modern deep learning techniques is shortening the time required to train a model to convergence. Many solutions have emerged trying to accelerate model convergence – from creating specific hardware such as Google's Tensor Processing Unit (TPU) to different model architectures which more efficiently utilize computation time. One class of learning algorithms called active learning (and similarly curriculum learning) seeks to accelerate convergence through selective data sampling. In this paper we explore the effectiveness of different active learning algorithms, present small modifications to existing algorithms, and attempt to intuitively explain our results. We found that the best active learning method we implemented consistently achieved an improvement of over 2% over our baseline attempts, with little additional cost.**

I. INTRODUCTION

A. Doodle Challenge: Motivation

The Doodle Recognition Challenge is a machine learning competition hosted by Kaggle which requires participants to classify 50 Million hand-sketched doodles of common items (planes, stars, hands, etc...) that span 340 classes. Our initial trials were focused on achieving the highest top-3 accuracy possible for this data set, in order to win the competition. However, after 30+ experiments trained to convergence with various models architectures and hyper-parameters, we happened upon the interesting problem of active learning. Each of our experiments would take over 12 hours to train on an Nvidia Tesla V100 GPU. Attempting to accelerate our iteration process, we began training on subsamples of data, but noticed variable results. This begged the question: how could we optimally sample data such that we reach a higher accuracy within a shorter amount of time?

B. The Active Learning Problem

Active learning describes a selection process used to determine which data should be fed into the model during training. From our previous competition-trials, we determined that active learning would be especially applicable because our data had a lot of interclass and intraclass variance, and bad drawings. By using simple I.I.D random sampling, we are throwing away the information of which samples are "expert" drawings, and which are erroneous drawings. We found this information was particularly difficult to discern for all 50 Million examples from simply inspecting the data set, or using traditional outlier removal techniques.

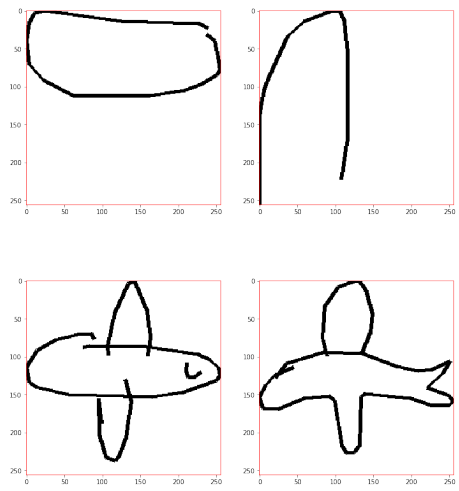

Fig. 1. The top two images are uninformative examples of airplanes, while the bottom two images are representative examples of airplanes.

C. Active Learning Algorithms

We examine 4 main active learning algorithm variants, and benchmark each of their results against I.I.D sampling.

1) Worst Class Re-weighting
2) Simple Curriculum Learning
3) Bucketed Curriculum Learning [1]
4) Active Bandit Learning [2]

The first two algorithms were formalized by the researchers after seeing variants of their methods used on several online resources including Kaggle and Fast.Ai. The second two algorithms are more formal and have been documented in the literature.

For each of the algorithms, we allow the model to make 1 pass through 500k images (spanning 30 classes). We run 3 trials of each setup, and report the final validation top3 accuracies and corresponding sample variances.

II. DOODLE CHALLENGE: COMPETITION

As mentioned above, we found our motivation for the active learning research while competing in the Kaggle Doodle Challenge. In this section, we describe our experimentation procedure and how it motivated us to work in curriculum learning.

A. Model Selection

We found Benchmark Analysis of Representative Deep Neural Network Architectures [3] to be a useful resource that outlined the speed, size and accuracy of popular ImageNet models. Below we comment on some of the models we considered.

1) MobileNet: We tried MobileNet, for its reported high performance on the competition data by other competition participants.

2) NasNet: Tried because it achieved state of the art results in the ImageNet challenge. We quickly abandoned it since it could barely fit in our GPU's memory, and took too long for it to train over 50 million images (motivation for active learning).

3) InceptionResNetV2: We discarded InceptionResNetV2 for a similar reason.

4) NasNetMobile: We tried NasNetMobile next, but it processed batches slower than MobileNet (as it performed architecture search).

5) ResNet50, SeResNext: We found that ResNet50 was the model of choice for the Stanford DawnBench Competition [4], which ranked models according to the time it took for them to achieve 93 percent accuracy on the ImageNet dataset. However, neither SeResNext or ResNet50 showed significant improvement over MobileNet baseline in the first 8 hours of training and we discarded these models.

Since we later found that top competitors used some of the models we discarded, this experience showed us how selecting the right model can be a difficult, time consuming and imprecise of a process.

B. Hyperparameter Tuning

Another highly time consuming area of the competition was hyper parameter tuning. This task was especially difficult since hyper parameters are often linked to each other such as the learning rate and the batch size. Furthermore, parameters such the image size presented a trade-off between accuracy and time to convergence. We guided ourselves throughout by following literature on the topic. For instance, Goyal et. al. [5] motivated us to keep a large batch size and Smith et. al. [6] correctly guided us to choose a high batch size (512 samples). Furthermore, we plotted accuracy vs. increasing learning rates to find the optimal learning rate as suggested in Smith et. al. [7].

C. Other modifications

Inspired by "Sketch-a-Net that Beats Humans" [8] we attempted to encode more information into the model. We linearly decayed the color of the doodle strokes according to order in which they were drawn. Intuitively, we wanted to add more weights to the first strokes of the doodle which tended to be more central components of the image. Unfortunately, this method again showed little benefit.

Finally, we attempted to use pre-trained ImageNet weights, but even though this helped accelerate initial training, it provided a minimal reduction in overall training time. These results

were consistent with "Rethinking ImageNet Pre-training" [9].

D. Takeaways

After 3 weeks of optimizing our model we reached a top 15 percent score in the Kaggle competition. We found that the 3 most successful factors contributing to the performance were a large batch size of 512, image sizes of 128x128 and letting our model train for 2 straight days. At this point, we were dissatisfied with the process – particularly with the long experimentation feedback loop. We pivoted to working on active learning and curriculum learning because we've come to realize the importance of reducing the convergence time of neural networks to speed up the tuning process.

III. Active Learning

For the next part of the project, we formed a smaller data set by uniformly sampling elements across 30 classes until we obtained a combined total of $500,000$ samples (we similarly formed a validation set of 1000 samples). We performed this sub-sampling in order to reduce our iteration time and be able to try different methods. Furthermore, for fair evaluation we set a budget of $500,000$ iterations for each of these methods, (just enough to iterate through each sample once). As a baseline we shuffled the dataset and feed each instance once into our model.

A. Computing the difficulty of samples

Inspired by previous work [1], we decided to define sample difficulty as the loss on a previously trained model. Note that even though other approaches in the literature use simple models such as logistic regression to classify the difficulties of the training examples, we simply used one of the previously partially trained neural networks to compute the difficulties. We see this as justified, because researchers often train models more than once, and would generally seek to improve overall experimentation time. Therefore, we first train our MobileNet with the entire dataset of $500,000$ samples and then rated the sample's difficulty by the cross entropy loss according to each.

IV. Worst Class Re-Weighting

Worst class re-weighting seeks to train the model by sampling batches from classes which the model performs poorly on. After each backwards pass on a batch sized 512, we evaluate the number of mistakes the model makes on a validation batch from each class. We then adjust the probability of a given class being sampled proportional to the number of mistakes the model makes on the validation class set.

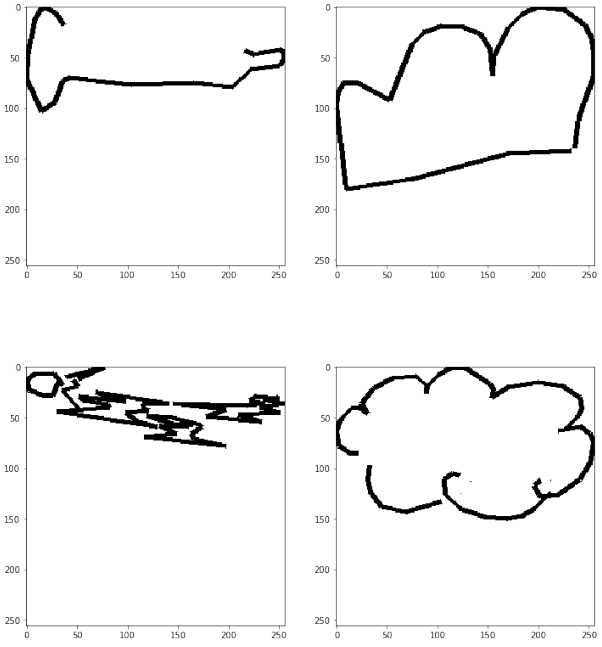

Fig. 2. Trumpets Vs Cloud Images. (Trumpets Left, Clouds Right)

A. Process:

1) Initialize class probabilities p_k
2) After each backwards pass, count the number of mistakes the model makes on a batch from each class's validation set.
3) Normalize number of mistakes on each class and update sample probabilities p_i to be this value.

4) Sample batch according to probabilities p_k

Above (Figure 2) we show two examples of how the trumpets class is inherently harder for people to draw, and thus for us to recognize, compared to the clouds class. This method would select trumpets as a class to train on over the clouds class. As we will see, this may present a training issue if the trumpets class is in fact, not learnable (noise).

B. *Intuition:*

This method seeks to train the model on difficult classes. The motivation is that training on these examples might make the model focus on learning what it doesn't know to increase overall accuracy.

V. SIMPLE CURRICULUM LEARNING

Simple Curriculum Learning seeks to train the model by training on "easy" examples before "hard" examples.

A. *Process:*

1) Let D_i be the difficulty, (cross entropy loss), of sample s_i using our pretrained network.
2) Sort samples by increasing D_i.
3) Train target network by looping through sorted samples.

B. *Intuition:*

Unlike Worst Class Re-Weighting, this method seeks to train the model on the "easiest" samples first. The motivation, formally described in curriculum learning literature, is that easy examples are a predecessor to hard examples from a learning perspective. This should allow the model to capture the low-hanging fruit of easy examples, and gradually update parameters to accommodate more difficult examples.

VI. BUCKETED CURRICULUM LEARNING

We see that the difficulty over each of the classes varies significantly. In Bucketed Curriculum learning we ensure we correctly classify easy examples first.

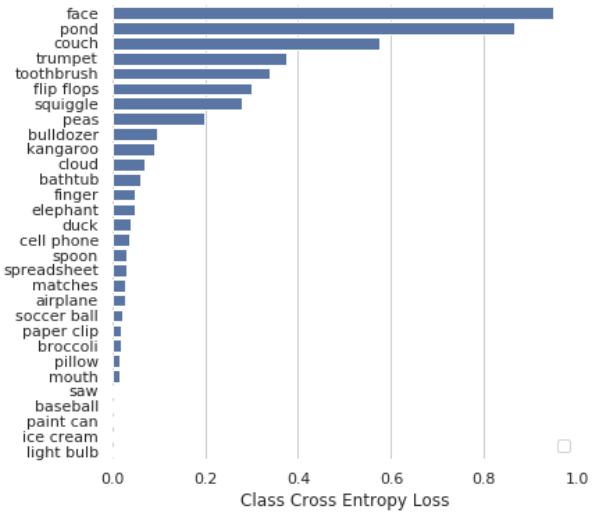

Fig. 3. Classes vary widely in their difficulty (measured through cross-entropy loss)

A. *Process:*

1) Compute D_i the difficulty, (cross entropy loss), of sample s_i using our pretrained network.
2) Let k be the number of buckets, (we used $k = 3$ in our experimentations), and n be the number of training samples. We construct difficulty buckets as follows:

   $$B_j = [s_i | D_{q(j/k)} \leq D_i < D_{q(j/k)}, i \in 0, ..., N]$$

   Where $D_q(x)$ is the difficulty at the $x-th$ percentile.
3) We divide the training process into k equally sized periods. At period t, we construct $B^* = \bigcup_{j=1}^{t} B_j$, randomly shuffle it and then sequentially sample from it.

B. *Intuition:*

We based this method on the final project of students from Stanford. Intuitively, we divided

the data points into k buckets according to their difficulty and divide the our training time into k periods. In the first period, we start from sampling solely from the bucket of easiest examples. In the second we period, we sample training points from both the easiest and next easiest buckets. We continue this until at the final period we only sample uniformly from the entire training set. Intuitively, this model both trains on increasing level of difficulty as well and spends a larger amount of time training on easier examples that might be viewed as low hanging fruits.

VII. ACTIVE BANDIT LEARNING

Active Bandit Learning investigated by Graves et. al. [2] presents the data selection problem as a multi-arm bandit problem. The reward in this scenario (as defined in the Deep Mind Paper) is the Prediction Gain of training on a given sample. Prediction gain is defined in Graves et. al. as the instantaneous change in loss for a sample i, before and after training on sample i:

$$R_i := \textit{Loss}(i, \theta) - \textit{Loss}(i, \theta')$$

Where θ are the model parameters before training and θ' are the model parameters after training on sample i.

Further, the "arms" in our experiments are 10 clusters composed of samples based on the percentile of their "difficulty" within their class, D_k, measured as cross-entropy loss (same as the other active learning methods). Cluster k corresponds to the kth difficulty percentile a sample lies in (i.e. cluster 4 contains all the 30th to 40th difficulty percentile samples, where the percentile is within each of the classes). We update probabilities of sampling from these difficulty clusters based on the Prediction Gain of the latest batch.

The algorithm also contains an exploitation parameter E. With probability E we do the active bandit learning method described, and with probability $1 - E$ we randomly choose a cluster to sample a batch. This allows us to not get stuck in a feedback loop of only exploiting the Prediction Gain from choosing a single class to train from.

A. Modifications:

1) A general Framework: One important difference between our implementation is that in the paper, the authors built clusters using an intrinsic property of the data, while we propose a more general methodology (clustering on difficulty). The authors built the active bandit learning framework for the "Repeat Copy task", where a neural network is given a sequence of random bits and is then asked to output that sequence a given number of times. In their approach, they made clusters based on specific data attributes. However, in our framework, we built a more general curriculum solely based on the difficulty of the samples.

B. Process:

1) Similarly to Bucketed Curriculum Learning, we divide the data into difficulty buckets. However, now we add the constraint that the class frequencies remain the same as in the entire dataset. For example, if there are m instances of clouds, then the m/k easiest clouds must be in the first bucket, where k is the number of buckets.
2) We initialize weights w_1,...,w_k to 0
3) We sample a cluster according to the following probability distribution: $p(j) = E * e^{w_1} / \sum_i^k e^{w_1} + (1 - E) * 1/k$
4) We sample a batch X from the cluster and compute the self prediction gain, $R(x) = L(X, \theta) - L(X, \theta')$. Then we compute it's rank, $r(X)$ relative to all the other self prediction gains we've computed so far.
5) Finally we update the weights as follows. If $t > H$, $w_j \leftarrow w_j + \eta * (2 * r(X) - t)/t$, where η can be seen as the learning rate, t is the number of batch we've computed so far and H is the size our startup period. The default values that we used for η, and H were $1/5$ and 100.

C. Intuition:

We try to

VIII. ANALYSIS

As mentioned above, we ran 3 trials for each of the algorithms. The learning curve plots have a shaded region, which represents the standard deviation of the curve according to the trials. The mean curve is plotted. Unless otherwise specified, the accuracy means test accuracy.

A. Initial Comparison

Fig. 4. Learning curves for the different algorithms

The learning curves above reflect 3 trials for each algorithm with a batch size of 512. As will be mentioned below, we did an additional 3 experiments for the active bandit algorithm with a smaller batch size of 32 (after analyzing some of our results) and it outperformed the simple uniform I.I.D method after these modifications.

B. Worst Class Re-weighting

Surprisingly, the Worst Class Re-weighting method performed worse than the original uniform sampling. We hypothesize that this is due to the fact that some classes tend to be much noisier and difficult to learn than others. Therefore, by concentrating in the class with the largest number of mistakes, we also focus on the noisiest and most difficult examples. By computing the average cross entropy training loss, we noticed that the cross entropy loss was much higher for several class. We then compared the trumpet examples, (which had a high average cross entropy loss), vs the cloud examples, (which had a low loss) (See Figure 2). We noticed that even though most clouds were well drawn, trumpets were often scribbles and regularly included obscene unrelated images.

TABLE I

TRIALS (BATCH SIZE 512)

	Top 3 Accuracy
Uniform Sampling	**0.8983333271741868**
Active Bandit Learning	0.8946666666666667
Active Bandit Learning (Exp Decay)	0.8926666666666666
Worst Class Re-weighting	0.8809999877214431
Bucketed Curriculum Learning	0.8503333404858907
Simple Curriculum Learning	0.8449999815225602
Full Exploration	0.8366666666666668

TABLE II

TRIALS (BATCH SIZE 32)

	Top 3 Accuracy
Active Bandit Learning	0.9366666666666665
Uniform Sampling	0.9139999936024349

C. Simple Curriculum Learning

After discouraging results in Worst Class Re-weighting, we were inspired by ”Curriculum Learning” [10] to attempt to feed the model samples according to their difficulty. However, our Simple Curriculum Learning method also showed discouraging results (Figure 5). We noticed that the training loss remained significantly higher than the validation loss for this method. We believe that examples of the same class that have similar difficulties tend to be quite similar. Therefore, as the model progresses through the training examples it overfits to the local ”cluster” of similar examples.

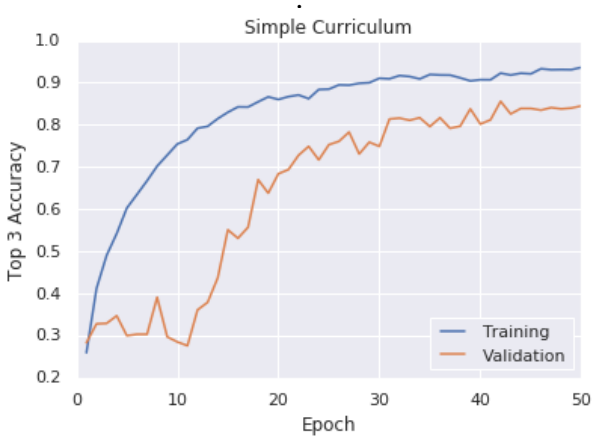

Fig. 5. Simple curriculum learning appears to overfit the data as training and validation sets have significant discrepancies

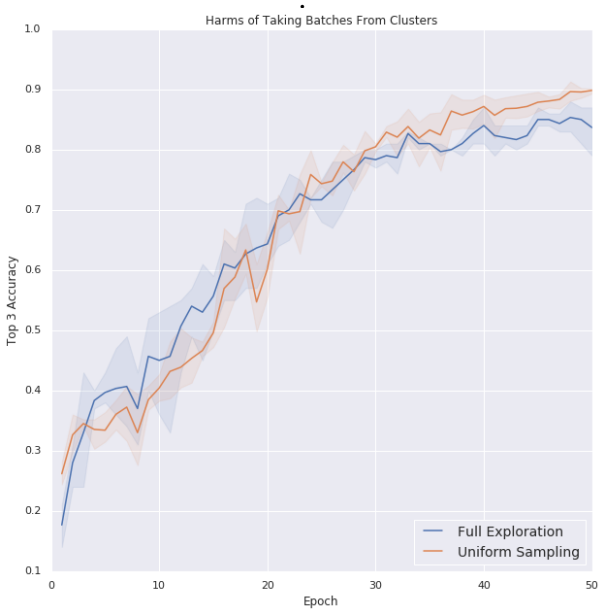

Fig. 6. Describe Plot Above

D. Bucketed Curriculum Learning

We believe that we got discouraging results for bucketed curriculum learning because of errors similar to the ones we had in simple curriculum learning. Specifically, the classes were not evenly distributed across the "difficulty" buckets. Furthermore, it was unclear at each point when we should switch buckets, and how many buckets we should have. This excess of hyperparameters motivated us to work on a more automated approach.

E. Active Bandit Learning

1) Initial Lack of Improvement: Note that at first, based on figure 4, Active Bandit Learning did not pose any improvement over Uniform Sampling. We will next talk about a result that helped explain why this was the case and how to deal with this issue.

2) Full Exploration: We got a curious result when we set the exploitation parameter E to 0. In this "full exploration" mode, we always just uniformly sample an entire batch from one of the 10 clusters. We expected it's performance to be similar to that of uniform sampling, but it performed worse than Uniform Sampling. We hypothesized the following potential reason for the fall in accuracy:

As we mentioned before, we have empirically noticed that samples of the same class with the same level of difficulty tend to be quite similar. Therefore, when we sample a entire batch from a difficulty cluster we obtain a highly similar set of points and intuitively this has highly negative consequences. Within a single batch, we use similar samples, which is bad for regularization – especially when the batch size is large. We also noticed that when we set the exploitation parameter E to 0.8 our model performed at similar levels to Uniform Sampling. This indicated that Active Bandit Learning had an innate sampling "drawback", but had some learning capabilities when we gave it "room to learn" ($E = 0.8$). We believed that if we reduced the batch size we would help relieve this drawback and help our model outperform uniform sampling.

3) Success with Reduced Batch Size: After we reduced the batch size, (while keeping the total number of training samples constant), for both Uniform Sampling and for Active Bandit Learning we noticed a dramatic improvement in accuracy. More importantly however, we noticed that Active Bandit Learning significantly outperformed uniform by more than a 2 percent difference in accuracy, (while the standard

deviation in performance of both methods was less than 0.5 percent accuracy). We noticed that even the worst accuracies of **Active Bandit Learning was better than the best Uniform Sampling accuracy** after our batch-size modification.

F. Analysis of Active Bandit Learning

In figure 8, we plot the probability distribution of the Active Bandit Learning algorithm across the different difficulty clusters. As expected the algorithm keeps away from assigning weight to the more difficult cluster, (which corresponds to cluster 10). However, interestingly enough, the algorithm also doesn't focus too much on the easiest examples either and instead tends to focus on the clusters with mid level of difficulty. This indicates that curriculum learning and data selection in general is not about focusing on either the hardest or easiest, but instead finding the sweetspot in the middle.

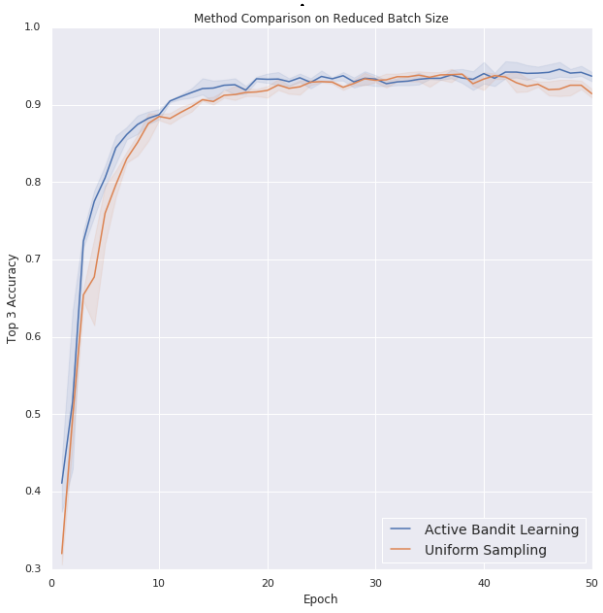

Fig. 7. With reduced batch size the Active Bandit Learning algorithm is able to outperform Uniform Sampling, especially in the later parts of training when the models have nearly converged.

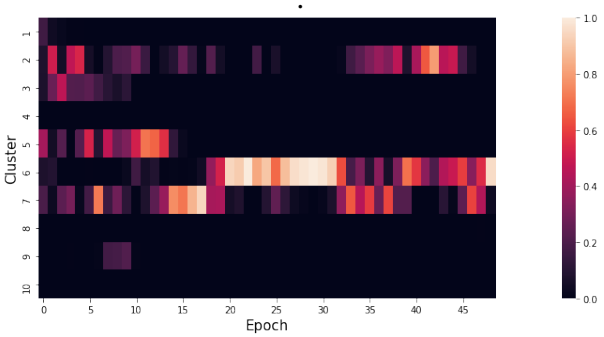

Fig. 8. Heatmap of the probability distribution across difficulty clusters

IX. CONCLUSION

Even though most of the curriculum/active learning methods we have attempted yielded worse than I.I.D performance, we find that Active Bandit Learning can lead to a significant model improvement. For future work we would like to explore computing the self prediction gain instead by using the magnitude of the learning descent step. Further, we note medium difficulty examples present the best learning opportunity for models. Based on these results, we believe that curriculum and active learning hold a key place in accelerating extremely long machine learning workloads.

A. DIVISION OF LABOR

Pedro wrote a large portion of the implementation for the initial Doodle Kaggle Challenge, and active learning algorithms. He also took part in acquiring the 4 servers used in the competition. He also researched active learning.

Josh researched about active and curriculum learning and digested several of the prior work on which this paper is based upon. He formalized the content scope and approaches of the project. He setup and maintained experimentation pipelines on the 4 servers that we used throughout the competition. He integrated Omniboard, an incredibly useful framework to visualize and log experiments.

Finally, both of us equally divided the work over each of the sections for the presentation and writing the paper.

REFERENCES

[1] R. Palamuttam, N. Sohoni, and H. Mohammad, "Cs 229 final project: Automated curriculum learning."

[2] A. Graves, M. G. Bellemare, J. Menick, R. Munos, and K. Kavukcuoglu, "Automated curriculum learning for neural networks, 2017," *URL https://arxiv.org/abs/1704.03003*, vol. 3003.

[3] S. Bianco, R. Cadene, L. Celona, and P. Napoletano, "Benchmark analysis of representative deep neural network architectures," *IEEE Access*, vol. 6, pp. 64 270–64 277, 2018.

[4] C. Coleman, D. Narayanan, D. Kang, T. Zhao, J. Zhang, L. Nardi, P. Bailis, K. Olukotun, C. Ré, and M. Zaharia, "Dawnbench: An end-to-end deep learning benchmark and competition," *Training*, vol. 100, no. 101, p. 102, 2017.

[5] P. Goyal, P. Dollár, R. Girshick, P. Noordhuis, L. Wesolowski, A. Kyrola, A. Tulloch, Y. Jia, and K. He, "Accurate, large minibatch sgd: training imagenet in 1 hour," *arXiv preprint arXiv:1706.02677*, 2017.

[6] S. L. Smith, P.-J. Kindermans, C. Ying, and Q. V. Le, "Don't decay the learning rate, increase the batch size," *arXiv preprint arXiv:1711.00489*, 2017.

[7] L. N. Smith, "Cyclical learning rates for training neural networks," in *Applications of Computer Vision (WACV), 2017 IEEE Winter Conference on*. IEEE, 2017, pp. 464–472.

[8] Q. Yu, Y. Yang, Y.-Z. Song, T. Xiang, and T. Hospedales, "Sketch-a-net that beats humans," *arXiv preprint arXiv:1501.07873*, 2015.

[9] K. He, R. Girshick, and P. Dollár, "Rethinking imagenet pre-training," *arXiv preprint arXiv:1811.08883*, 2018.

[10] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning," in *Proceedings of the 26th annual international conference on machine learning*. ACM, 2009, pp. 41–48.