Curriculum Learning

Koren Levenbrown, Naor Dalal
SISE Department at Ben Gurion University of the Negev

August 15, 2021

Abstract

Training neural networks is traditionally done by providing a sequence of random mini-batches sampled uniformly from the entire training data. The selected article proposed a new method named curriculum learning which is motivated by the idea of since neural network architectures are inspired by the human brain, it seems reasonable to consider that the learning process should also be inspired by how humans learn.

The method attempts to train the model with increasing levels of difficulty of the training examples in order to facilitate and optimize the training process. To employ curriculum learning, the training algorithm must resolve 2 problems: (i) sort the training examples by difficulty (ii) select a series of mini-batches that exhibit an increasing level of difficulty. The article presented an empirical evaluation of the Curriculum learning method and show improvements in learning speed and final performance.

Introduction

The following paper proposes an improvement for the neural network model training process based on the curriculum learning method that proposed in the article [1].

Our code is in the Github link.

KEYWORDS

neural networks, deep learning, curriculum learning, stratified.

1 Curriculum learning method

To simplify the idea behind the proposed CL method to real-life we can take an example of the establishment of a curriculum for human students. Teachers need to address two challenges: (i) Arrange the material in a way that reflects difficulty or complexity. (ii) Select the pace by which the material is presented – going over the simple ideas too fast may lead to more confusion than benefit, while moving along too slowly may lead to unproductive learning.

Curriculum learning as investigated here [1] deals with the question of how to use prior knowledge about the difficulty of the training examples, in order to sample each mini-batch non-uniformly and thus boost the rate of learning and the accuracy of the final classifier. The paradigm of CL is based on the intuition that it helps the learning process when the learner is presented with simple concepts first.

In the article [1] they composing CL into two separate, but closely related, sub-tasks and their corresponding functions. The first, termed scoring function, determines the "difficulty" or "complexity" of each example in the data. The scoring function makes it possible to sort the training examples by difficulty, and present to the network the easier (and presumably simpler) examples first. Scoring is done based on transfer learning or bootstrapping. The second function, termed pacing function, determines the pace by which data is presented to the network. The pace may depend on both the data itself and the learner.

We evaluate Transfer scoring function that computed as follows: First, we take the Inception network [1] pre-trained on ImageNet dataset [2] and run each training image through it, using the activation levels of its penultimate layer as a feature layer. Second, we use the features to train SVM classifier and use its confidence score at the scoring function for each image.

The pacing function that we evaluate is fixed exponential pacing. Fixed exponential pacing has a fixed step length, and exponentially increasing data size in each step. Formally, it is given by (N represent the size of data):

$$g(i) = min(starting\_percent \cdot inc^{\lfloor \frac{i}{step\_length} \rfloor}, 1) \cdot N$$

[1] https://pytorch.org/hub/pytorch_vision_inception_v3/
[2] https://image-net.org/

We chose these functions because they yield better performances than others functions.

Pros

- SGD enables to apply curriculum learning on neural networks in a straightforward manner.
- More significant when the task is more difficult (i.e. lower vanilla test accuracy). The reason may be that in easier problems there is a sufficient number of easy examples in each mini batch even without CL.
- Creating uniformly batches causes balanced batches so that each class has a similar number of examples and that ensures minimal and balanced diversity each batch.
- Reduce the convergence time and improves the training accuracy during all stages of learning.

Cons

- Since curriculum learning usually implies restricting the set of samples to a subset of easy samples in the preliminary training stages, it might constrain SGD to converge to a local minimum which it is hard to escape as increasingly difficult samples are gradually added.
- A mismatch between the chosen pre trained model for transfer scoring and the compared datasets requires enlargement of the images 10 times might leads to low difficulty-scoring quality and reducing the potential of the approach.
- The possible values variety of for the pacing function leads to non-optimal hyper-parameters optimization in most of the combinations that offered in the article - too fast/slow increasing data amount.
- According to our experiment the CL approach was not significant better than the vanilla approaches.

Algorithm 1: Curriculum learning method

Input: pacing function g, scoring function f, data X.

Output: sequence of mini-batches $[B'_1,...,B'_M]$

sort X according to f, in ascending order

result $\leftarrow$ []

forall *i=1,...,M* **do**

size $\leftarrow$ g(i)

$X'_i \leftarrow X$[1,...,size]

uniformly sample B'_i from X'

append B'_i to result

end forall

return *result*

2 Stratified batching learning method

We selected Stratified method as well-know sampling algorithm. Stratified sampling aims at splitting a data set so that each split is similar with respect to something. In a classification setting, it is often chosen to ensure that the train and test sets have approximately the same percentage of samples of each target class as the complete set. we've implemented stratified method that ensure that each batch has also the same number of samples of each target class in order to avoid imbalanced batches. The idea behind the method is to encourage stable learning and eliminate sampling bias.

Algorithm 2: Stratified method

Input: data X.

Output: stratified sequence of mini-batches $[B'_1,...,B'_M]$

sort X according to target class

result $\leftarrow$ []

while X *is not empty* **do**

pick a batch B_i from X in size of target classes in stratified way

append B_i to *result*

remove B_i from X

end while

return *result*

3 Linear curriculum learning method

In the following section we represent our improvement for the original Curriculum learning method and we name it as Linear curriculum learning method.
There are two main process that we could to improve in the specific suggested approach and we decided to focus on the pacing function.

Our improvement
The pacing function that we suggest focuses on linear data amount increasing with step length = 1 instead of exponential and fixed as suggested in the CL article. The linear pacing function increases the amount of data each step of training, we set the slope as a hyperparameter to control the amount of data we increase each step.

Algorithm 3: Linear Curriculum learning method

Input: linear pacing function g, scoring function f, data X.
Output: stratified sequence of mini-batches $[B'_1,...,B'_M]$
sort X according to f, in ascending order
$result \leftarrow []$
forall *i=1,...,M* **do**
    size $\leftarrow$ g(i)
    $X'_i \leftarrow X[1,...,\text{size}]$
    uniformly sample B'_i from X'
    append B'_i to result
end forall
return *result*

The linear pacing function is:

$$g(i) = min(starting\_percent \cdot N + inc \cdot i, N)$$

4 Training

In our experiment we selected the best algorithm that presented in the CL article which use transfer scoring function based on Inception-V3 model. we've examine 4 algorithms: (i) **Case 1**: Vanilla method (ii) **Case 2**: Stratified method (iii) **Case 3**: Best curriculum method (iv) **Case 4**: Linear Curriculum method.

Hyperparameters

We've focused on three main parameters that derived from the curriculum algorithm: **step length** - the number of iterations in each step, **increase amount** - an exponential factor used to increase the size of the data used for sampling mini-batches in each step, **starting percent** - the fraction of the data in the initial step considered at the beginning of the training. Additional parameters were the learning rate and the learning rate schedule parameters.

The hyperparameters examined each case as follows: **Case1**: Batch size: 100, Epochs:150, Lr step length: [200, 400, 600, 800], Initial lr: [0.035, 0.05, 0.01, 0.001, 0.0035], Decay lr: [1.5, 1.3, 1.1] **Case2**: Batch size: 100, Epochs: 150, Lr step length: [200, 400, 600, 800], Initial lr: [0.035, 0.05, 0.01, 0.001, 0.0035], Decay lr: [1.5, 1.3, 1.1] **Case3**: Batch size: 100, Epochs: 150, Pacing function parameters: Starting percent: [0.05, 0.1, 0.15, 0.2], Increase amount: [1.5, 2, 3], Step length: [50, 100, 200, 400, 800], Lr step length: [200, 400, 600, 800], Initial lr: [0.035, 0.05, 0.01], Decay lr: [1.5, 1.3, 1.1] **Case4**: Batch size: 100, Epochs: 150, Pacing function parameters: Starting percent: [0.05, 0.1, 0.15, 0.2], Increase amount: [1.2, 1.1, 1.3, 1.5], Lr step length: [200, 400], Initial lr: [0.035, 0.05, 0.01], Decay lr: [1.5, 1.3, 1.1]

Preprocessing

The preprocess for the three algorithms has the same steps as follows:

- Image normalization
- Image resize to fit to transfer scoring model
- Image resize to fit to learner model

The preprocessing is applied on each training/validation/test set separately to avoid data leakage.

Model architecture

CNN model containing 8 convolutional layers with 32, 32, 64, 64, 128, 128, 256, 256 filters respectively. Each layer goes through Batch normalization layer and ELU activation function. The first 6 layers have filters of size 3×3, and the last 2 layers have filters of size 2×2. Every second layer there is a 2×2 max-pooling layer and a 0.25 dropout layer. After the convolutional layers, the units are flattened, and there is a fully-connected layer with 512 units followed by 0.5 dropout layer. The batch size was 100. The output layer

is a fully connected layer with output units matching the number of classes in the data set, followed by a Softmax layer. We trained the network using the SGD optimizer, with cross-entropy loss.

Randomized cross validation

Nested cross validation, external loop with 10 folds and internal loop with 3 folds. we've applied the internal for the hyper parameters optimization 50 times while the parameters selected randomly.

5 Results

To examine our method, first we compared to the relative data sets, learner model and number of epochs that examined in the article. We decided to use only 5K samples from the CIFAR10 data sets due the long runtime.

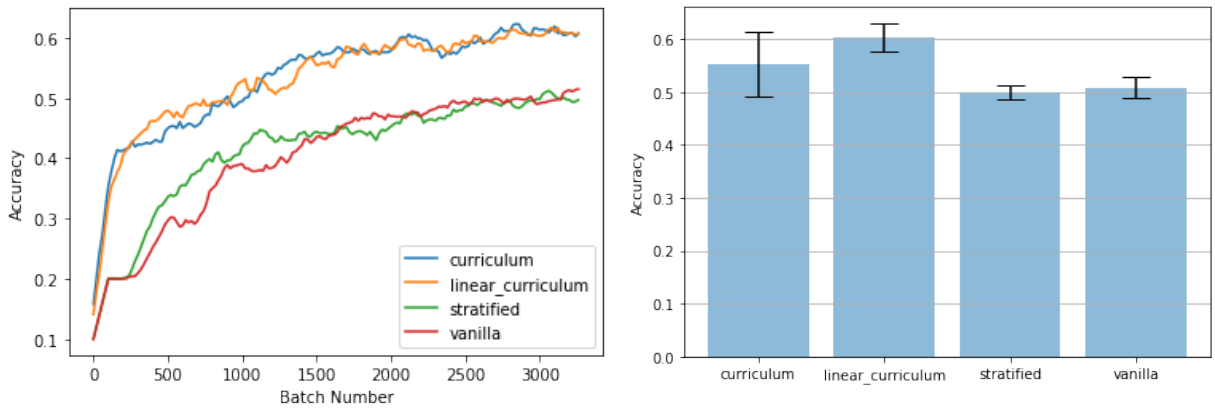

Figure 1: CIFAR100 small mammals

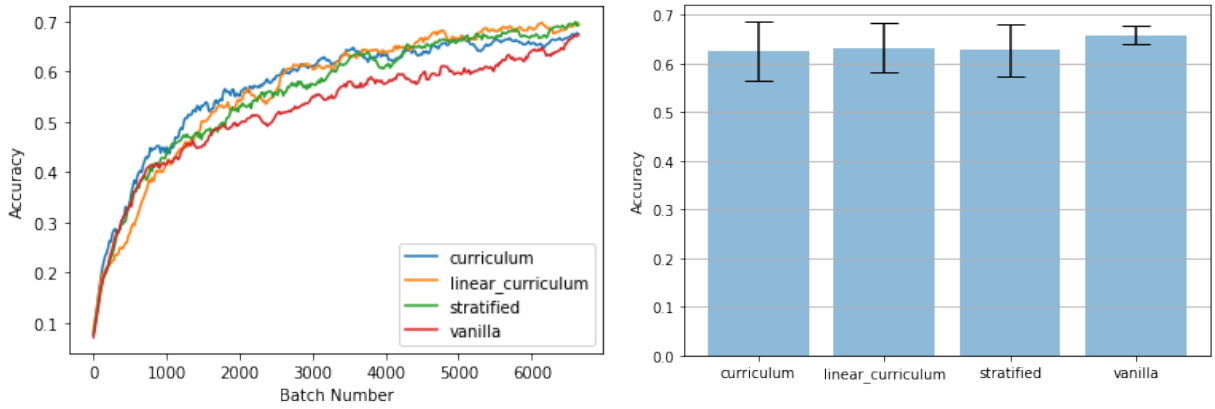

Figure 2: CIFAR10

As we can see above and according to the empirical experiments the proposed method has better results and low variance than the suggested curriculum method. We've perform our experiment on 19 various data sets as detailed at the Appendix A.

6 Result summary

Results summary for the entire datasets and approaches (algorithms) that examined:

Dataset Name	Algorithm Name	Accuracy	FPR	Precision	AUC
CIFAR100_small_mammals	curriculum	0.608	0.081	0.613	0.844
CIFAR100_small_mammals	**linear_curriculum**	**0.610**	**0.090**	**0.622**	**0.851**
CIFAR100_small_mammals	stratified	0.516	0.121	0.529	0.780
CIFAR100_small_mammals	vanilla	0.548	0.113	0.551	0.815
CIFAR100_household_furniture	curriculum	0.756	0.061	0.755	0.926
CIFAR100_household_furniture	**linear_curriculum**	**0.772**	**0.057**	**0.788**	**0.942**
CIFAR100_household_furniture	stratified	0.688	0.078	0.690	0.899
CIFAR100_household_furniture	vanilla	0.708	0.073	0.700	0.903
CIFAR100_people	**curriculum**	**0.444**	**0.139**	**0.459**	**0.732**
CIFAR100_people	linear_curriculum	0.440	0.140	0.481	0.716
CIFAR100_people	stratified	0.428	0.143	0.433	0.686
CIFAR100_people	vanilla	0.396	0.151	0.428	0.686
CIFAR100_food_containers	curriculum	0.764	0.059	0.781	0.911
CIFAR100_food_containers	**linear_curriculum**	**0.788**	**0.053**	**0.805**	**0.943**
CIFAR100_food_containers	stratified	0.720	0.070	0.728	0.903
CIFAR100_food_containers	vanilla	0.744	0.064	0.750	0.913
CIFAR100_fish	curriculum	0.744	0.064	0.747	0.909
CIFAR100_fish	**linear_curriculum**	**0.748**	**0.063**	**0.759**	**0.918**
CIFAR100_fish	stratified	0.704	0.074	0.703	0.902
CIFAR100_fish	vanilla	0.716	0.071	0.722	0.893
CIFAR100_large_natural_outdoor_scenes	curriculum	0.840	0.040	0.845	0.965
CIFAR100_large_natural_outdoor_scenes	**linear_curriculum**	**0.844**	**0.039**	**0.850**	**0.974**
CIFAR100_large_natural_outdoor_scenes	stratified	0.824	0.044	0.826	0.958
CIFAR100_large_natural_outdoor_scenes	vanilla	0.780	0.055	0.782	0.943
CIFAR100_large_man_made_outdoor_things	curriculum	0.804	0.049	0.807	0.958
CIFAR100_large_man_made_outdoor_things	**linear_curriculum**	**0.856**	**0.036**	**0.855**	**0.963**
CIFAR100_large_man_made_outdoor_things	stratified	0.776	0.056	0.780	0.938
CIFAR100_large_man_made_outdoor_things	vanilla	0.804	0.049	0.799	0.943
CIFAR100_vehicles_1	curriculum	0.800	0.050	0.802	0.947
CIFAR100_vehicles_1	**linear_curriculum**	**0.820**	**0.045**	**0.827**	**0.973**
CIFAR100_vehicles_1	stratified	0.732	0.067	0.733	0.902
CIFAR100_vehicles_1	vanilla	0.740	0.065	0.752	0.913
CIFAR10_all	curriculum	0.676	0.036	0.691	0.941
CIFAR10_all	**linear_curriculum**	**0.718**	**0.031**	**0.729**	**0.958**
CIFAR10_all	stratified	0.710	0.032	0.731	0.950
CIFAR10_all	vanilla	0.680	0.036	0.695	0.949
STL10_first	**curriculum**	**0.772**	**0.057**	**0.773**	**0.943**
STL10_first	linear_curriculum	0.768	0.058	0.781	0.948
STL10_first	stratified	0.734	0.054	0.788	0.938
STL10_first	vanilla	0.741	0.056	0.821	0.943
STL10_second	curriculum	0.724	0.069	0.737	0.926
STL10_second	**linear_curriculum**	**0.740**	**0.065**	**0.759**	**0.936**
STL10_second	stratified	0.732	0.067	0.736	0.936
STL10_second	vanilla	0.712	0.072	0.729	0.921

Dataset Name	Algorithm Name	Accuracy	FPR	Precision	AUC
EMNIST_Letters_first	curriculum	0.985	0.003	0.986	1.000
EMNIST_Letters_first	linear_curriculum	0.985	0.003	0.986	1.000
EMNIST_Letters_first	stratified	0.992	0.001	0.992	0.999
EMNIST_Letters_first	vanilla	0.985	0.003	0.985	1.000
EMNIST_Digits_all	curriculum	0.986	0.002	0.986	1.000
EMNIST_Digits_all	linear_curriculum	0.988	0.001	0.988	1.000
EMNIST_Digits_all	stratified	0.996	0.000	0.996	1.000
EMNIST_Digits_all	vanilla	0.998	0.000	0.998	1.000
EMNIST_Merged	**curriculum**	**1.000**	**0.000**	**1.000**	**1.000**
EMNIST_Merged	linear_curriculum	0.978	0.003	0.924	0.999
EMNIST_Merged	stratified	0.935	0.011	0.645	0.970
EMNIST_Merged	vanilla	0.935	0.011	0.654	0.976
FMNIST_first	curriculum	0.936	0.016	0.935	0.993
FMNIST_first	linear_curriculum	0.942	0.017	0.931	0.991
FMNIST_first	stratified	0.948	0.013	0.950	0.994
FMNIST_first	**vanilla**	**0.952**	**0.012**	**0.951**	**0.995**
FMNIST_second	curriculum	0.968	0.008	0.969	0.998
FMNIST_second	**linear_curriculum**	**0.980**	**0.006**	**0.977**	**0.998**
FMNIST_second	stratified	0.976	0.006	0.976	0.997
FMNIST_second	vanilla	0.972	0.007	0.972	0.998
ImageNet_first	curriculum	0.740	0.087	0.739	0.918
ImageNet_first	linear_curriculum	0.738	0.087	0.737	0.915
ImageNet_first	**stratified**	**0.796**	**0.068**	**0.804**	**0.935**
ImageNet_first	vanilla	0.775	0.075	0.780	0.924
ImageNet_second	curriculum	0.786	0.053	0.792	0.941
ImageNet_second	linear_curriculum	0.800	0.050	0.805	0.951
ImageNet_second	**stratified**	**0.822**	**0.045**	**0.825**	**0.958**
ImageNet_second	vanilla	0.794	0.052	0.812	0.956
ImageNet_third	curriculum	0.850	0.037	0.853	0.969
ImageNet_third	linear_curriculum	0.850	0.037	0.852	0.969
ImageNet_third	stratified	0.855	0.036	0.855	0.974
ImageNet_third	**vanilla**	**0.858**	**0.035**	**0.858**	**0.976**
ImageNet_fourth	curriculum	0.847	0.038	0.851	0.969
ImageNet_fourth	**linear_curriculum**	**0.865**	**0.034**	**0.868**	**0.980**
ImageNet_fourth	stratified	0.855	0.036	0.864	0.979
ImageNet_fourth	vanilla	0.861	0.035	0.866	0.978
ImageNet_fifth	curriculum	0.868	0.033	0.872	0.980
ImageNet_fifth	**linear_curriculum**	**0.883**	**0.029**	**0.889**	**0.983**
ImageNet_fifth	stratified	0.863	0.034	0.878	0.985
ImageNet_fifth	vanilla	0.875	0.031	0.884	0.985

7 files description

Main.ipynb - The main notebook which will run the entire training process for all the various datasets and algorithms that will tested.
Net.py - Includes the network architecture, training (fit) and eval (predict) methods.
utils.py - Includes the pacing and order functions, pre-processing methods and Friedman test function.
visualization_utils.py - Includes the graphs plotting, metrics evaluation and results exportation functions.
config_file - Datasets and sub datasets list that will examine.

8 Statistical significance testing

We chose Accuracy performance metric and performed Friedman test to determine whether the differences are statistically significant. We selected our significance level (α) to be 0.05. The Friedman test rejected the null hypothesis with 1.346117627090377e-08 pvalue. Because we rejected the null hypothesis we do a Post-Hoc test to test the differences between the linear curriculum algorithm to the other algorithms.
We calculated the average ranks:
curriculum average rank: 2.7142857142857144
linear curriculum average rank: 1.7380952380952381
stratified average rank: 2.8333333333333335
vanilla average rank: 2.7142857142857144
As we can see linear curriculum has the best average rank (best average accuracy). After that we plot the graph ranks with critical difference value of CD = 0.9537924302488462.

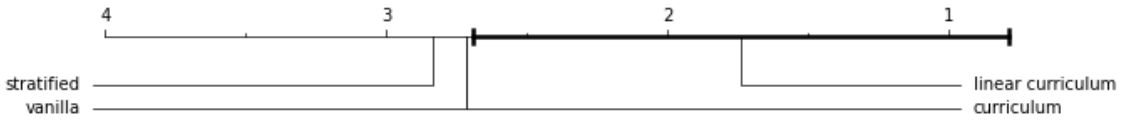

Figure 3: Post-Hoc test

The Friedman and post-Hoc methods implementation took from here.
As we can see in figure 3, linear curriculum algorithm is statistically significant from the other algorithms because all the other algorithms are outside the CD line around linear

curriculum algorithm. It means that linear curriculum improvement is not random, it is statistically significant that it is an improvement over the rest of the algorithms.

9 Conclusions

- The Linear approach allows for an intermediate approach between the extreme approaches which on the one hand trains all the data from the beginning and on the other hand trains gradually, aggressively, and continuously on the same data.
- Despite the improvement that the proposed approach introduced most of the pros and cons of the curriculum approach remained for the proposed approach as well.
- Beyond the rapid convergence time of the proposed approach (that equal to the time presented by the curriculum approach) it can be seen that in most cases the fluctuations of the results by the metric on the test set were more smooth and consistent and we estimate that happened due to the small, fixed, and frequent gradual addition that caused the non-formation gap between the learning steps as presented by the curriculum approach.

10 Appendix

A. Additional data sets results Below described 7 subsets from CIFAR100. Each data set includes 5 classes and 2500 samples, 500 samples each class. As we can see our proposed approach was more accurate in the most of the cases.

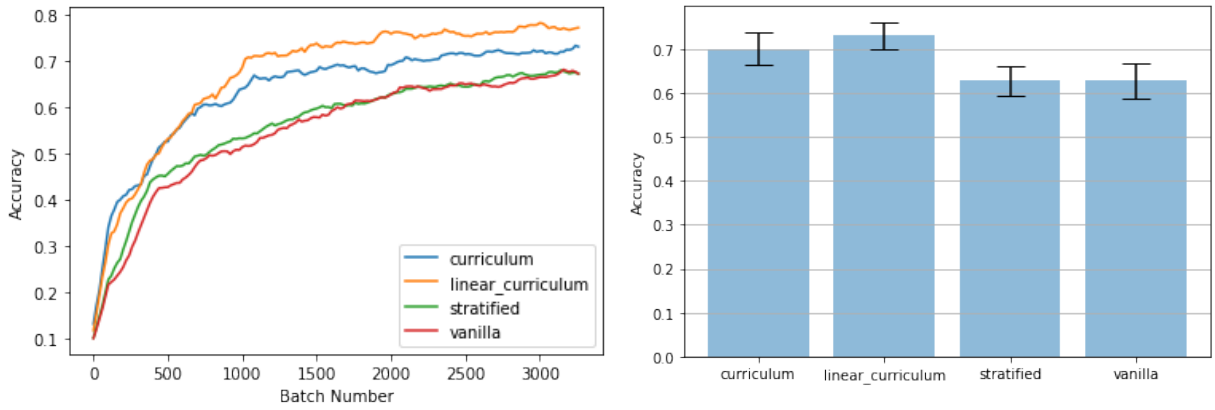

Figure 4: CIFAR100 - house hold furniture

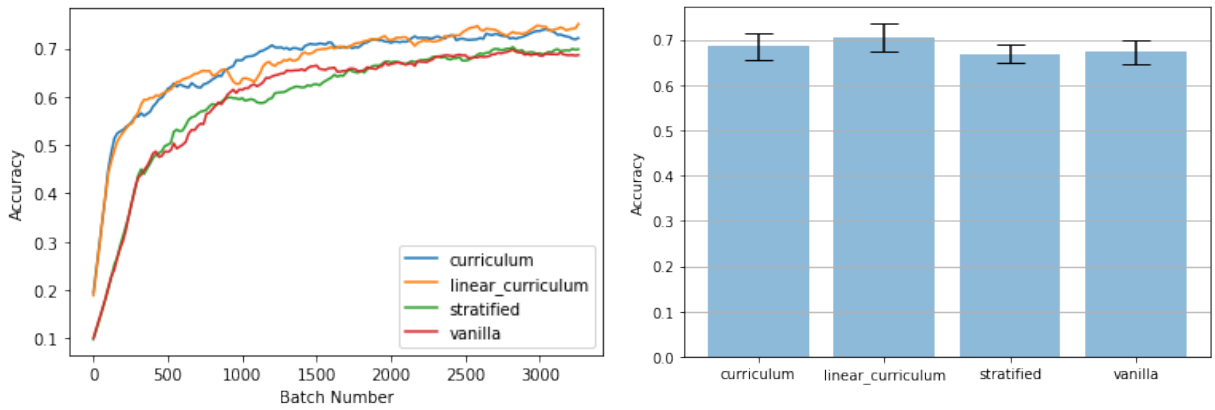

Figure 5: CIFAR100 - Fish

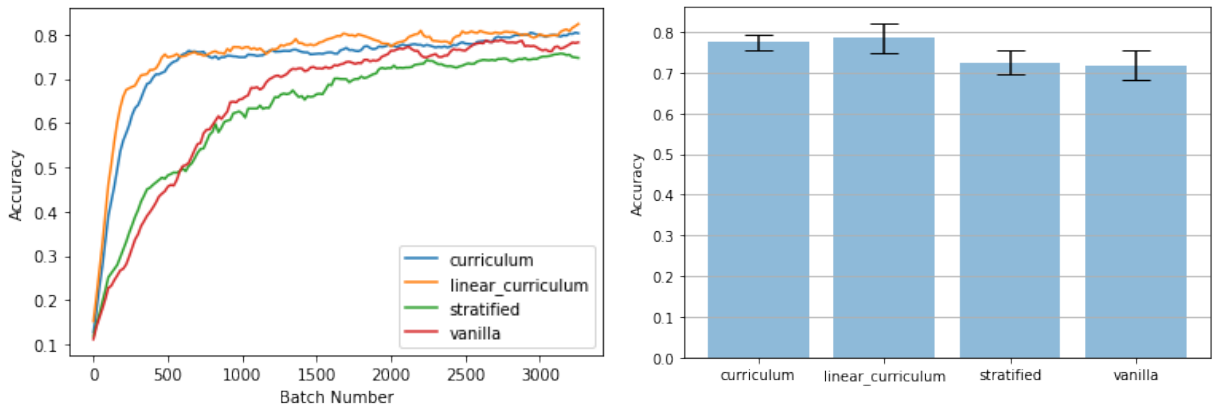

Figure 6: CIFAR100 - Large man made outdoor thing

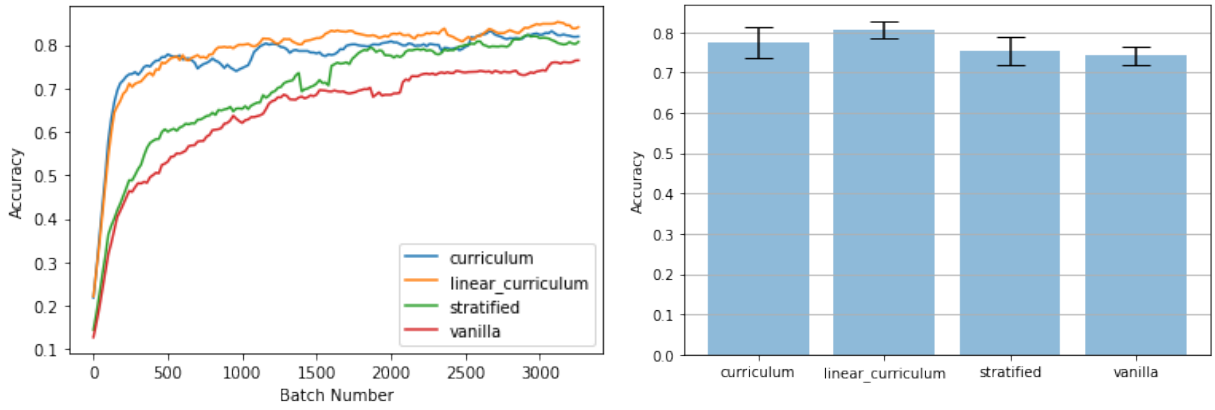

Figure 7: CIFAR100 - Large natural outdoor scenes

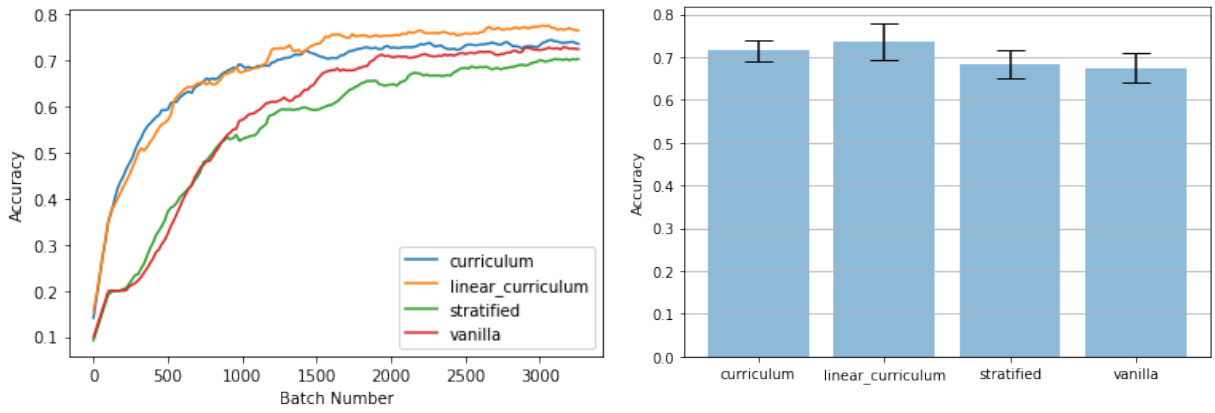

Figure 8: CIFAR100 - Food container

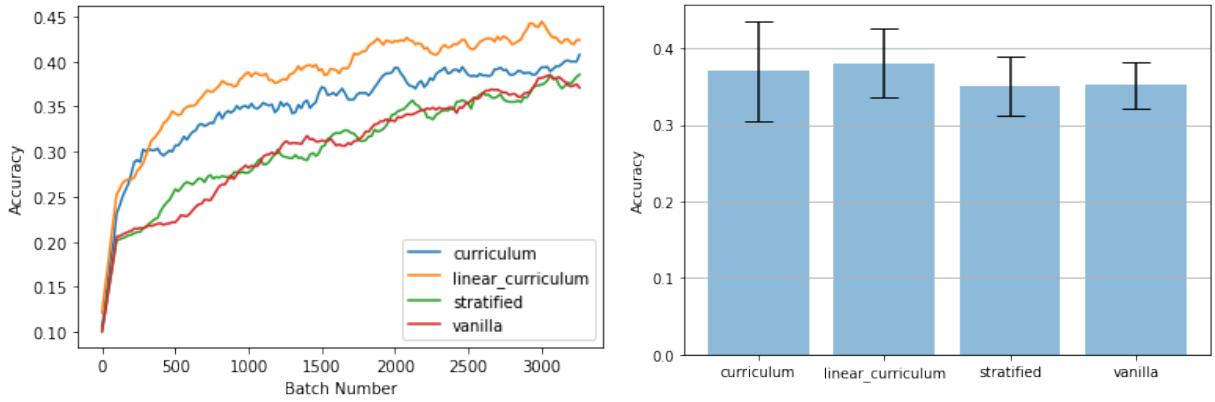

Figure 9: CIFAR100 - People

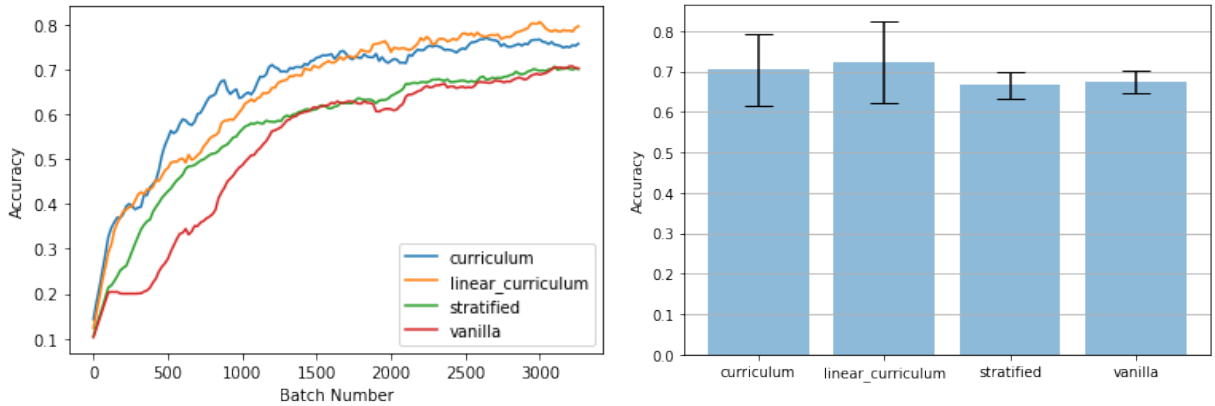

Figure 10: CIFAR100 - Vehicles1

Below described two STL10 subsets. Each subset includes 5 classes and 2500 samples, 500 samples each class. As we can see our new method succeed to produce the most accurate

model but when we ran the empirical experiment 10 times we can see that our method has the lower average accuracy and high variance.

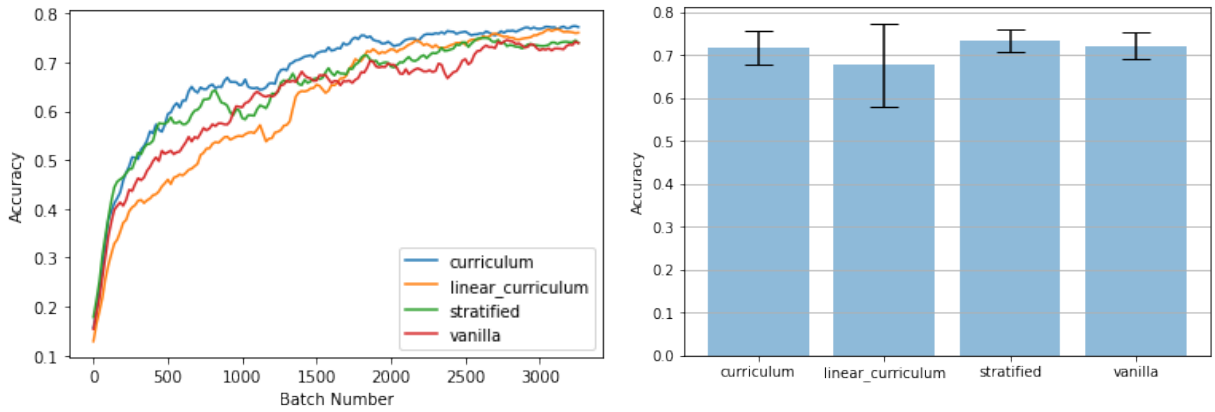

Figure 11: STL10 - Airplane, Bird, Car, Cat, deer

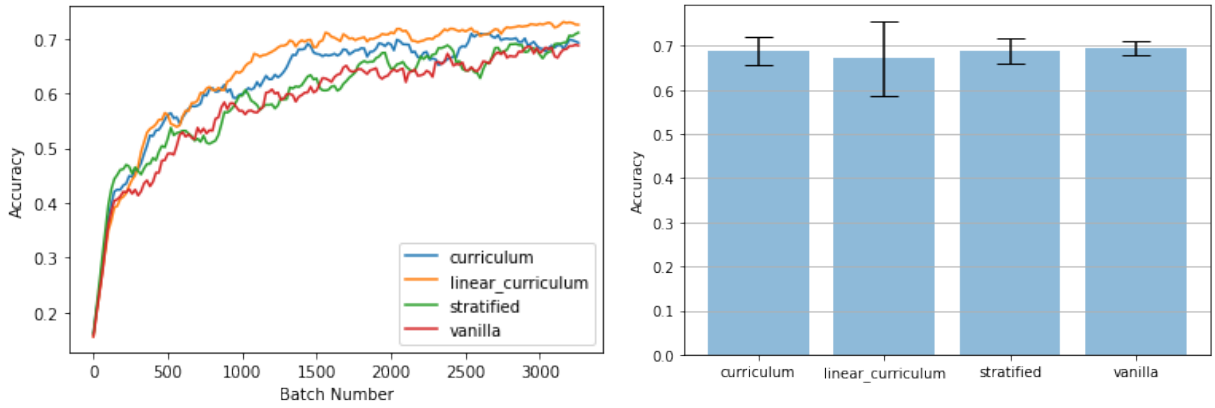

Figure 12: STL10 - Dog, Horse, Monkey, Ship, Truck

Below described three Emnist subsets. Each subset includes 5K samples. As we can see the two first subsets turn out to be very easy tasks thus the curriculum methods have no benefit over the vanilla and the stratified. The third subset was more accurate than the traditional methods but not than the CL method:

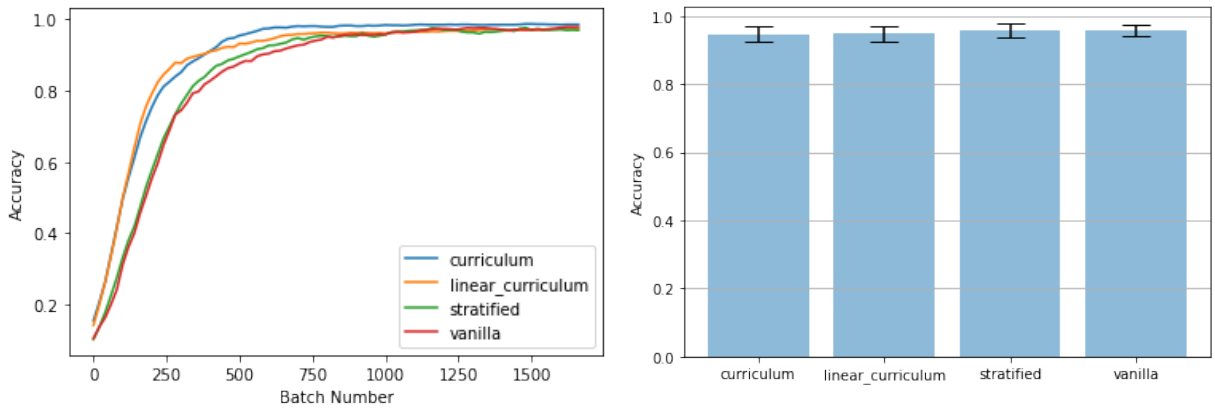

Figure 13: Emnist Letters - a, b, d, e, g, z, t

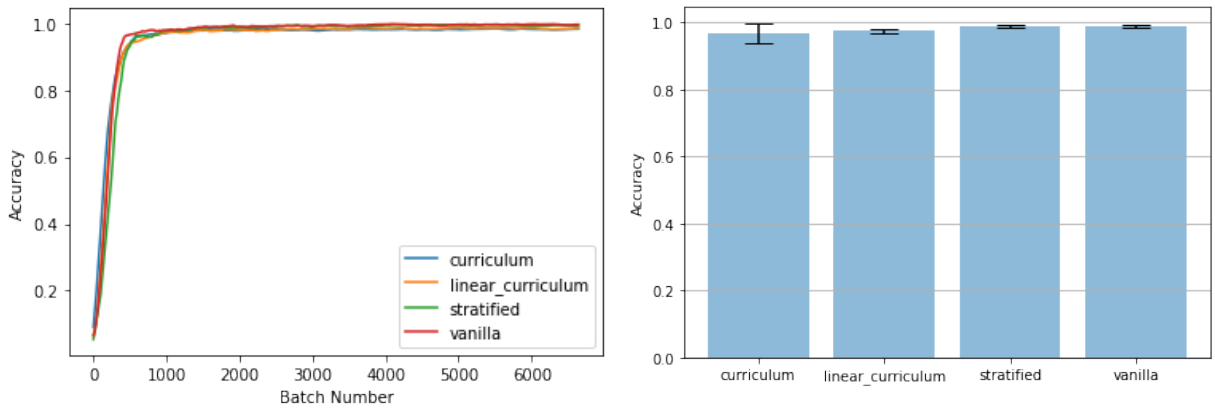

Figure 14: Emnist Digits - 0-9

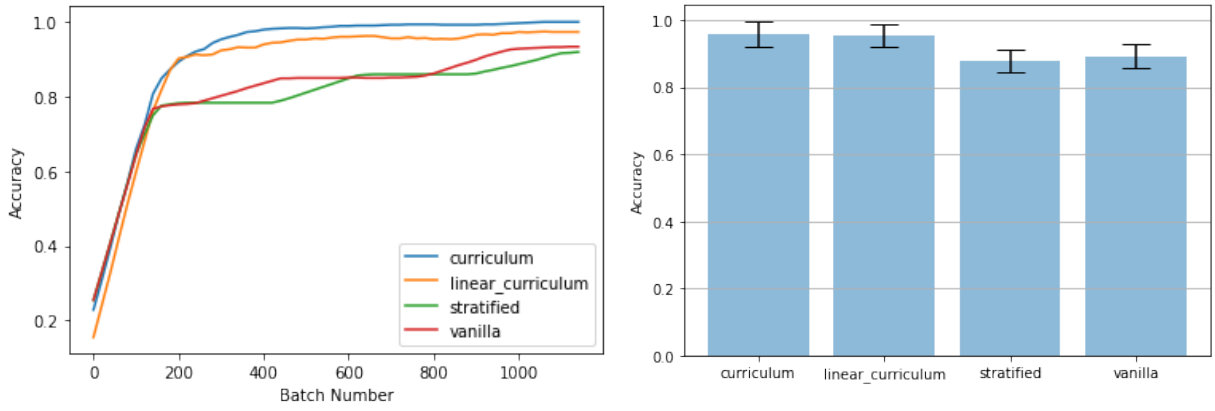

Figure 15: Emnist Merged - A, T, a, j, 4, 6, 0

Below described the Imagenet subsets, each subset includes 5K samples:

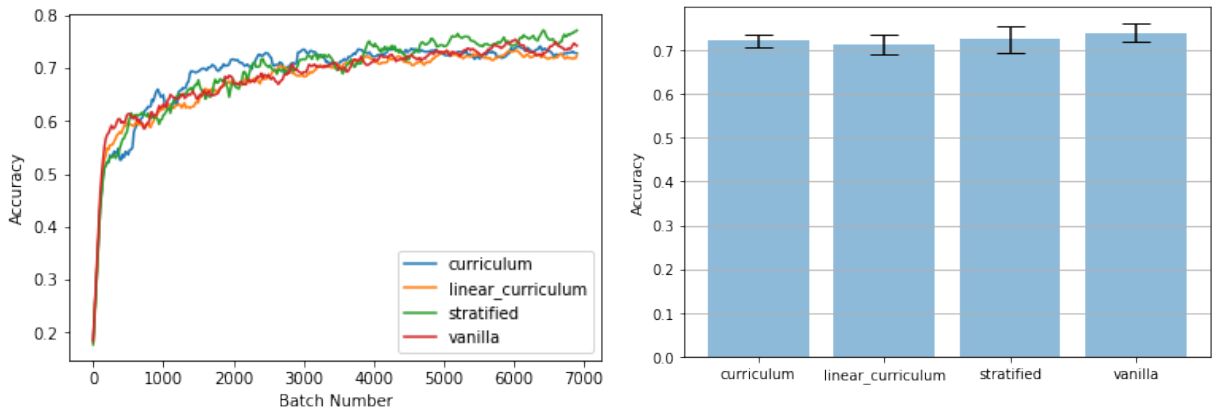

Figure 16: Imagenet first - Orange, Banana, Cup, Red wine, Fretzel

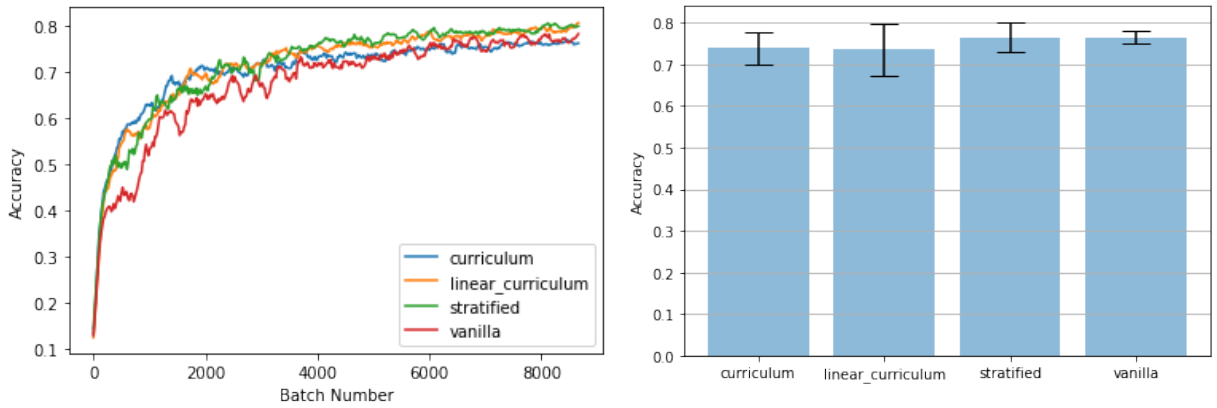

Figure 17: Imagenet second - Stove, Printer, Pillow, Pyjama, Missile

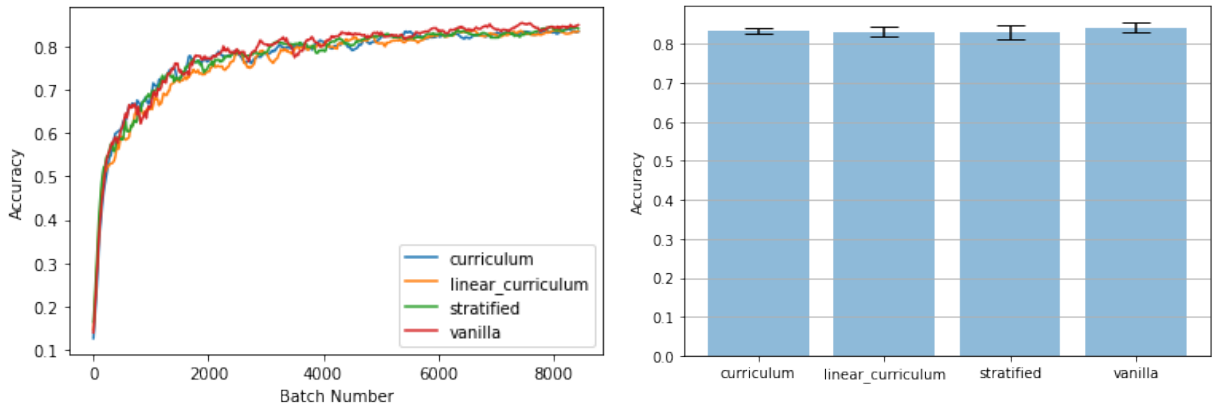

Figure 18: Imagenet third - Hammer, Dome, Balloon, Nail, Mask

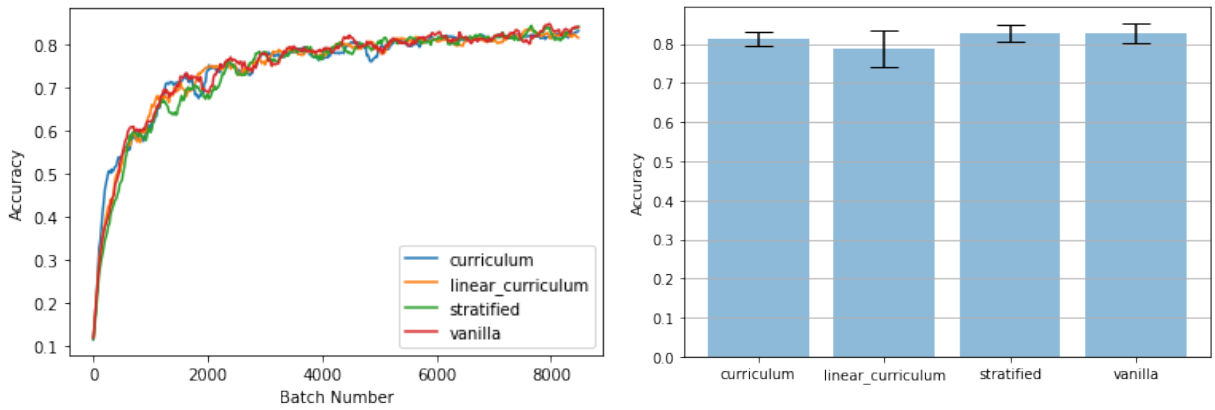

Figure 19: Imagenet fourth - Golf ball, Envelope, Zebra, Poodle, Flamingo

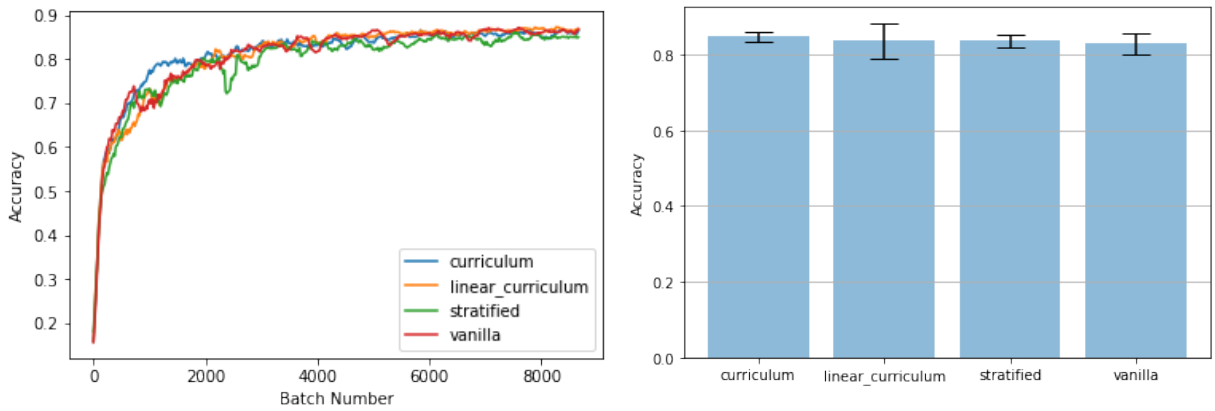

Figure 20: Imagenet fifth - Snail, Jelly fish, Rugby ball, Mushroom, Corn

Below described the FMNIST subsets, each subset includes 5K samples:

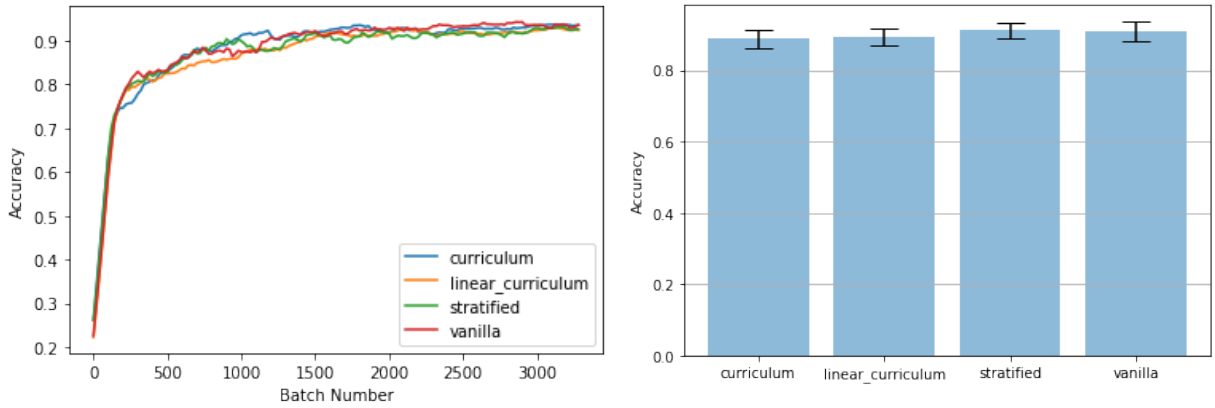

Figure 21: FMNIST first - T-shirt/top, Trouser, Pullover, Dress, Coat

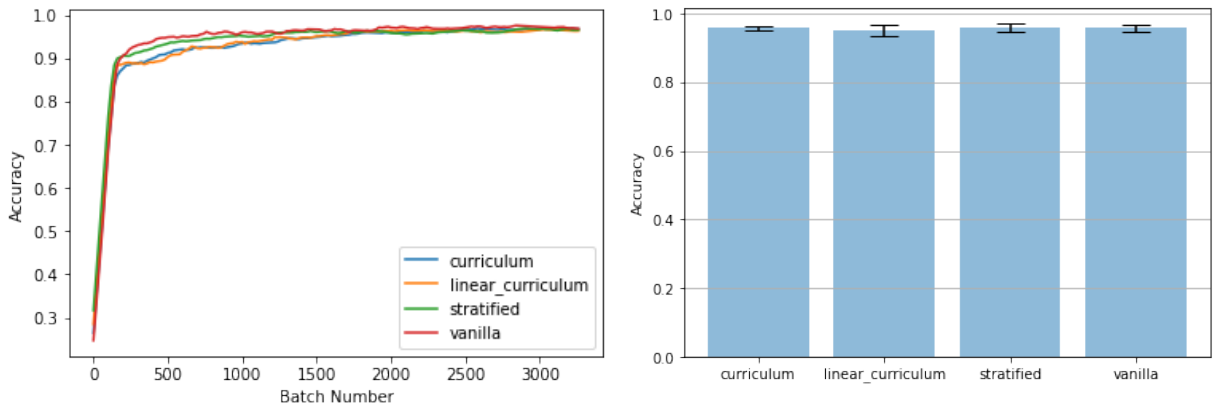

Figure 22: FMNIST second - Sandal, Shirt, Sneaker, Bag, Ankle boot

References

[1] Guy Hacohen and Daphna Weinshall. On the power of curriculum learning in training deep networks. In *International Conference on Machine Learning*, pages 2535–2544. PMLR, 2019.