TRINITY COLLEGE DUBLIN

ML1819 Research Assignment 2

302. Curriculum Learning: Does it work?

Team 16

Edward BERGMAN
15335633

Patrick LEE
15325692

Seng LEUNG
15322499

source code
https://github.com/sengleung/curriculum-learning

activity
https://github.com/sengleung/curriculum-learning/graphs/contributors

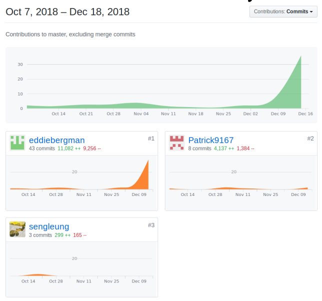
Contributions

BERGMAN
EMNIST/Methodology

LEE
Language Translation/Research

LEUNG
Related Work/Research

Commit Activity



Oct 7, 2018 – Dec 18, 2018

Contributions to master, excluding merge commits

eddiebergman #1
43 commits 11,082 ++ 9,256 --

Patrick9167 #2
8 commits 4,137 ++ 1,384 --

sengleung #3
3 commits 299 ++ 165 --

December 18, 2018

The Effectiveness of Curriculum Learning

Edward Bergman
Trinity College Dublin
Dublin, Ireland
ebergman@tcd.ie

Patrick Lee
Trinity College Dublin
Dublin, Ireland
leep2@tcd.ie

Seng Leung
Trinity College Dublin
Dublin, Ireland
sleung@tcd.ie

1 INTRODUCTION

Curriculum learning is an approach to machine learning that is inspired by human education. Learning objectives are classified by an expert to create a gradual progression from easy to hard material in a linear fashion. This seems an intuitive approach to learning for humans but how does this compare when introduced to a machine learning algorithm?

There are two categories of curriculum based learning, 'Active' and the contrasting 'Inactive' kind. In 'Active' training, the networks predictions are used to influence what data it will be exposed to next.

While investigating curriculum learning, we found ample literature on 'Active' learning using statistical and information diversity approaches [4] [6] [1]. A trade off for using 'Active' learning is the increased computational cost to evaluate the next phase of the curriculum. In settings where there are multiple models to train or the dataset is relatively large compared to time, this can be a serious drawback.

We focus on 'Inactive' learning where the data is turned into a curriculum prior to training. A problem we encountered was conflicting results, the effectiveness of 'Inactive' curriculum seemed to either be minimal or else exceeded baselines [5]. This is to be expected as a human must impose some heuristic specific for each data set which may or may not be a hindrance to the overall learning rate of a network.

Our research is focused at understanding how imposing our own heuristics can impact the learning rate of neural networks. What are the effects of having a curriculum based distribution over the data as opposed to a random uniform distribution? How do certain distributions fair against each other?

We hope to investigate these questions with two different models, a Convolution Neural Network (CNN) tasked with identifying handwritten characters and a Long Short Term Memory (LSTM) model tasked with translating phrases from French to English.

2 RELATED WORK

The concept of curriculum learning was discussed by Bengio et al. in 2009 [2]. Results from the study indicated that curriculum learning showed significant improvements in generalisation and the speed of convergence of the training process over their baselines. They supplied their own heuristics for two toy problems and each showed an improvement over their baseline counterparts.

Zaremba and Sutskever [7] explored the use of the naive curriculum in Bengio et al. [2] for training an LSTM in a sequence to sequence learning problem. They found the naive approach did not perform well but a mix between the naive approach and random sampling outperformed their baseline. This would imply the distribution of samples in a curriculum can be more important than presenting samples 'easy' to 'hard', as defined by a heuristic.

Collier and Beel discovered that the the optimal syllabus for each task is task dependant. They explored various different distributions over difficulty sorted samples and found that it had limited effect on the generalisation capabilities of an LSTM. They further investigated an automated curriculum learning technique *Predictive Gain*, which performs competitively against hand-crafted syllabuses [5]. We hope to extend an aspect of their work and explore the effectiveness with different distributions.

3 METHODOLOGY

For our data sets, we need a heuristic that we can use to design a curriculum. We score all our samples according to this heuristic. Finally, we create a collection of tasks, a **task** being a subset of the of samples. These tasks are created by sampling from our data set according to different distributions. These tasks contents are based on our heuristic scores to create a **curriculum**.

3.1 Data Sets

For our character recognition CNN, we use the EMNIST data set, specifically the **balanced** set. This is a 137,000 sample data set with 47 classes, each with 3000 samples belonging to that class.

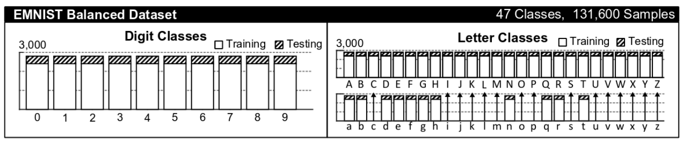


Figure 1: EMNIST balanced image classification data set [3]

For our second dataset we used a set of 150,000 English-French bilingual pairs, that vary in length and complexity, to create a multi-class classification model that can translate sentences. The data to be processed is cleaned initially of any punctuation or letter-case.

3.2 Heuristics

For the EMNIST dataset, we classify our samples from being 'easy' to 'hard'. To do this, we obtain a mean image of all samples in a class and sum over the per pixel difference between an image and its mean. The closer an image is to its mean, the lower its score and the 'easier' we consider that sample. As we do not handle small transformations to our images, for example a slightly rotated 'A', we feel this heuristic is applicable for our model.To prevent any data leakage, our mean image is calculated only from our training data.

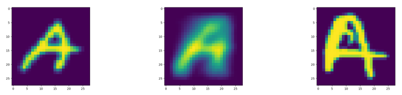


Figure 2: A visual comparison of easy vs hard heuristic 'easy' (left) , mean (centre) , 'hard' (right)

For our language translation dataset, the classification of difficulty was mapped to the average length of the bilingual pairs. We felt this was a sufficient method as shorter and simpler words are normally used to make up larger more complicated sentences, so a sensible classification would be to first teach the building blocks that make up harder tasks.

3.3 Creating a Curriculum

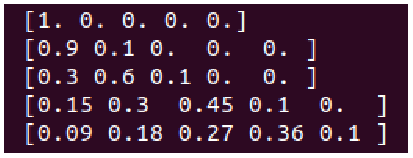

```
[1.  0.  0.  0.  0.]
[0.9 0.1 0.   0.   0.  ]
[0.3 0.6 0.1 0.   0.  ]
[0.15 0.3  0.45 0.1  0.  ]
[0.09 0.18 0.27 0.36 0.1 ]
```

Figure 3: A curriculum with 5 tasks (vertical) created with a distribution D = { 90% , 10% , 0%}
Each line represents what percentage of the task we sampled from which chunk, line two represents that task 2 is 90% from chunk0 and 10% from chunk1

To create a curriculum, we sort our data from 'easiest' to 'hardest'. We divide this into **T** chunks, ordered by difficulty. To create our **T** tasks, we sample from these chunks according to the distribution we wish to investigate. Our distributions are specified by **D** = { p, c, f } , p, c, f representing the percentage to sample from previous, current and future chunks respectively.

As we go through and create each task **i**, each chunk **c** gets a weighted part of the distribution:

$$c < i ? \frac{c}{\frac{p*(p+1)}{2}} * (c+1) * Dp$$

$$c > i ? \frac{c}{\frac{f*(f+1)}{2}} * (T-c) * Df$$

$$c == i ? 1 * Dc$$

i: index of task
c: index of chunk
p: count of chunks 0 -> i
f: count of chunks i -> T
T: amount of tasks/chunks
D: distribution = {Dp, Dc, Df}

This effectively gives more weight to chunks which are closer to the current task.

3.4 Baselines

We consider two different baselines:

- **Unsorted:** We feed in the data unsorted. We split it into T tasks which have random samples to make comparisons easier
- **Sorted:** All our data fed to our model in difficulty sorted order. Again we split this into T tasks to make comparisons, where task 0 will be all the 'easiest' data. This is equivalent to **D** = { 0% , 100% , 0%}

3.5 Investigating Distributions

We wanted to investigate various trade offs in **D** and the results we would see. For character recognition, we take note of the models accuracy, loss and top-k accuracy with k = 2. This would allow us to see how many samples a model must see until it reached its optimum, given a certain distribution. As many characters are similar (1:l , s:5), we expect our model to fall victim to uncertainties in this regard, hence we also track top-2 accuracy to see if its top 2 outputs in the softmax layer correctly predicted the character class.

Tradeoff	p	c	f
Previous & Current			
	70%	30%	0%
	50%	50%	0%
	30%	70%	0%
Future & Current			
	0%	30%	70%
	0%	50%	50%
	0%	70%	30%
Current & (Future + Previous)			
	10%	80%	10%
	30%	40%	30%
	50%	0%	50%
Previous & Current (some Future)			
	20%	70%	10%
	60%	30%	10%
	10%	60%	30%
	50%	20%	30%
Future & Current (some Previous)			
	10%	70%	20%
	10%	30%	60%
	30%	60%	10%
	30%	20%	50%

As the amount of tasks has an effect on the samples, we also try each of these distributions with T = 5, 10, 25. As chunks closer to the task receive more weight, as T increases, we would expect this to be similar to a smaller T with a higher sampling from the current chunk.

Lastly, to have a more controlled test, we create 3 untrained models and initialise their weights. Each configuration of T and D is done on 3 models with their results averaged out.

4 RESULTS AND DISCUSSION

4.1 Character Recognition

4.1.1 Baseline Comparison. We take notice of two major points here. One is that the model trained on unsorted random data, similar to a uniform distribution, performed much better than our model trained on sorted data. The unsorted model converged to its optimum in less samples and seems to have converged to an optimum with less loss and higher accuracy.

The **spike** in loss and reduction in accuracy that occurs for the sorted model coincides with the last few tasks of an epoch which represents the most 'difficult' data. We estimate that the model may be losing some general patterns used for each class in an attempt to learn the more 'difficult' data, relearning the basic general patterns as we enter the new epoch.

4.1.2 Trade off Comparisons. An immediate pattern that emerges in our experiments is that all distributions generally fall between our two baselines in all metrics except for when we have no future task consideration, in which case they perform equally or worse than our sorted baseline. This seems to reinforce the previous estimate of what causes the spike to occur at the end of an epoch. Models with these distributions learn general patterns that work for the majority of the data but fail to work on difficult examples that make up the missing portion of accuracy. The models that have the highest spikes are those which have emphasis on Dc over Df and Dp. If we look at distributions with no Dc, those models had the smallest spike.

4.2 Language Translation

4.2.1 Baseline Comparison. For the language translation task we trained three models and compared their BLEU scores. The metrics we used for evaluating the language translation models are applied after the models have been trained. The three models we trained are:

- **Unsorted:** The baseline where the model was trained on all the data at once.
- **Sorted:** A model was generated for each distribution creating 18 models in total.

The BLEU 1 scores of the sorted models outperformed the baseline but not by much, (fig. 10). This suggests that by implementing more tasks and sampling from future/previous or current within that task gives the model the reinforcement to output better results. The most striking part of the results is that the models that had a higher rate of sampling from previous tasks achieved a higher score. Models 0-3 were completely biased towards current and previous tasks when choosing the data to use and model 10 had negligible weighting for future tasks. This gives a good indication that curriculum learning is a viable technique for more effective programs.

4.3 Conclusions

4.3.1 Character Recognition Curriculum. We estimate the spike to be a result of the model biasing towards the data is it currently exposed to. This is an interesting result as it suggests that a curriculum based off heuristics must be careful and identify if the most 'difficult' samples contain similarities to the patterns of 'easier' samples. As we have seen, when they do not, this can lead to the model attempting to learn one set of patterns and then have its attention shifted to an unrelated set of patterns. This **spike** can be an identifier for these kinds of heuristics. A way to combat this if no other heuristics can be identified is to attempt include some uniform sampling over the entire dataset as done in [5].

We had hoped the curriculum would help the model converge to an optimum sooner than the unsorted baseline but as we can see it did not. This is likely due to our heuristic only accounting for samples in the training set.

4.3.2 Language Translation Curriculum. Our results for this test were promising, they seemed to suggest curriculum learning could perform well when training a model for this certain task, we would, however, require a lot more thorough investigation before we could come to a conclusive answer. We would have liked to experiment with our method of classification and syllabus but with time restraints and difficulties implementing this model with the syllabus and restricted processing power, we were limited in the scope of our research. It is however a good base to proceed with more rigorous testing and investigation with confidence that there is some credence to the underlying ideas behind curriculum learning.

5 LIMITATIONS AND OUTLOOK

A limitation we faced was the time it took to train the models. We attempted to leverage VM instances on Google Cloud to perform training on multiple configurations but we had to discrete distributions and select a sample that we thought would cover a wide basis. Further work could be done on trying continuous distributions over the data, clustering data according to heuristics and sampling from a distributions centred on these clusters.

REFERENCES

[1] Vanya Avramova. 2015. *Curriculum Learning with Deep Convolutional Neural Networks.* Master's thesis. KTH, School of Computer Science and Communication (CSC).

[2] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum Learning. In *Proceedings of the 26th Annual International Conference on Machine Learning (ICML '09).* ACM, New York, NY, USA, 41–48. https://doi.org/10.1145/1553374.1553380

[3] Gregory Cohen, Saeed Afshar, Jonathan Tapson, and André van Schaik. 2017. EMNIST: an extension of MNIST to handwritten letters. *CoRR* abs/1702.05373 (2017). arXiv:1702.05373 http://arxiv.org/abs/1702.05373

[4] David A. Cohn, Zoubin Ghahramani, and Michael I. Jordan. 1996. Active Learning with Statistical Models. *CoRR* cs.AI/9603104 (1996). http://arxiv.org/abs/cs.AI/9603104

[5] Mark Collier and Jöran Beel. 2018. An Empirical Comparison of Syllabuses for Curriculum Learning.

[6] Alex Graves, Marc G. Bellemare, Jacob Menick, Rémi Munos, and Koray Kavukcuoglu. 2017. Automated Curriculum Learning for Neural Networks. *CoRR* abs/1704.03003 (2017). arXiv:1704.03003 http://arxiv.org/abs/1704.03003

[7] Wojciech Zaremba and Ilya Sutskever. 2015. Reinforcement Learning Neural Turing Machines. *CoRR* abs/1505.00521 (2015). arXiv:1505.00521 http://arxiv.org/abs/1505.00521

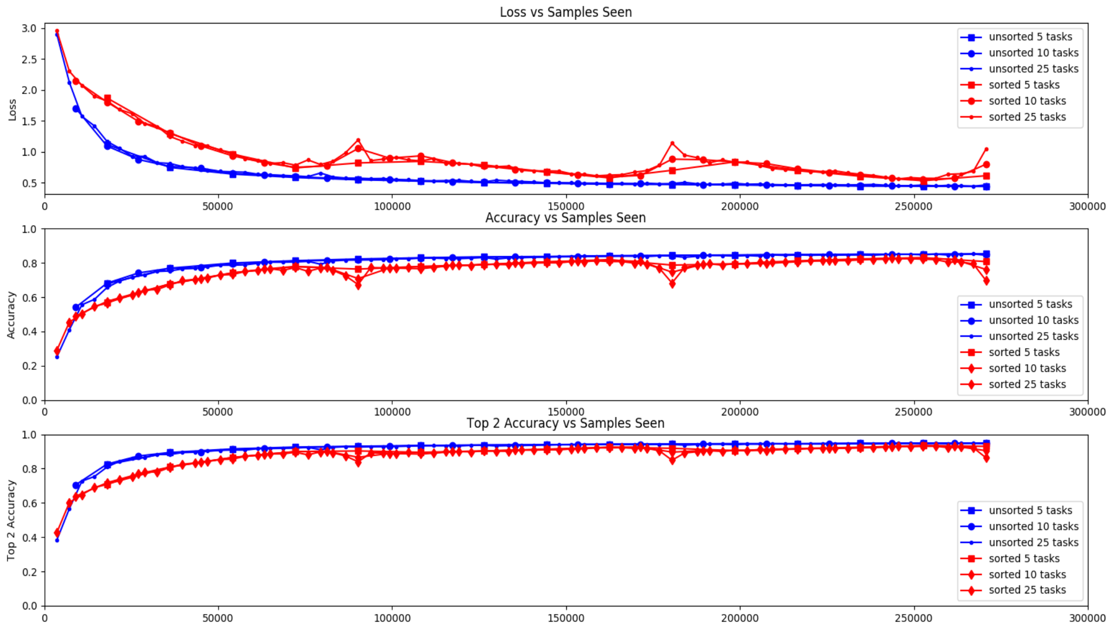


Figure 4: A comparison between our two baselines for character recognition

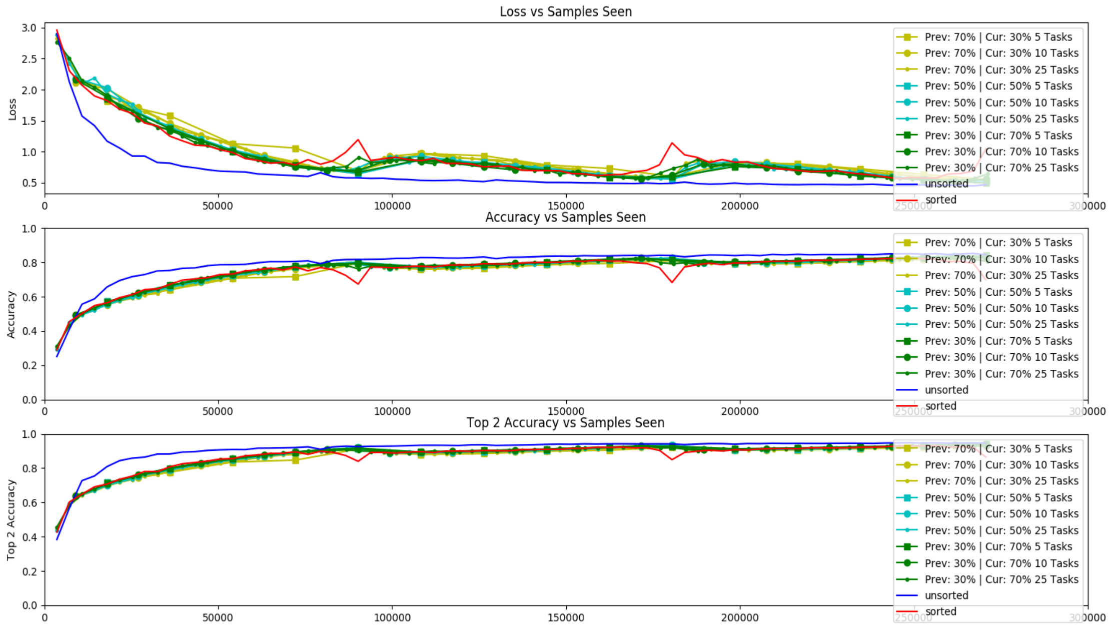


Figure 5: Here we investigated trading distribution weighting between only Dp and Dc in the character recognition task

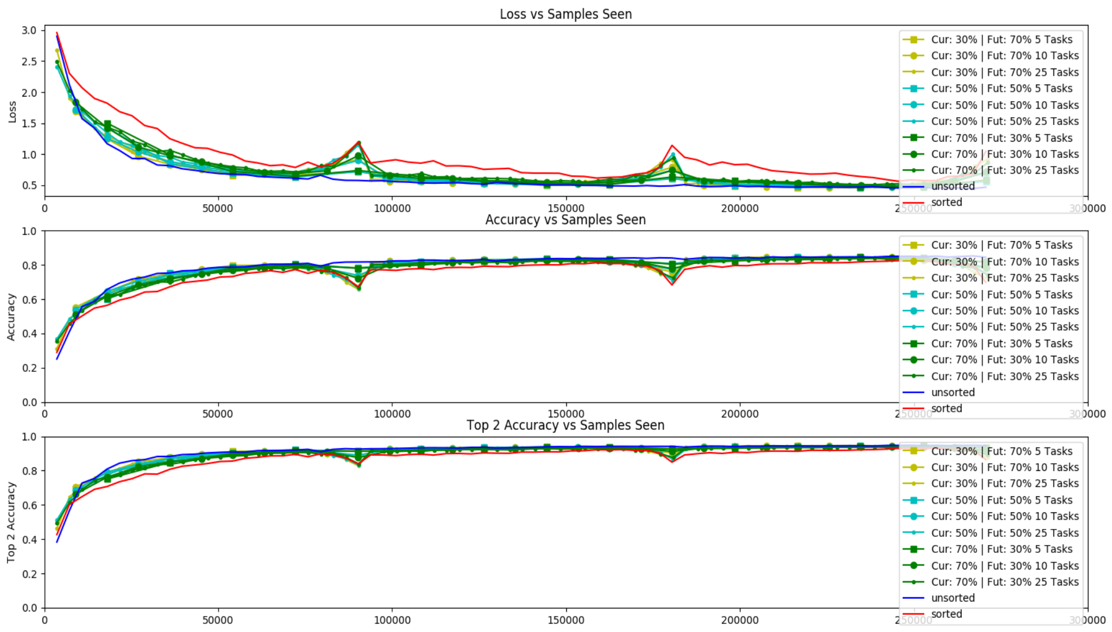


Figure 6: Here we investigated trading distribution weighting between only Dc and Df in the character recognition task

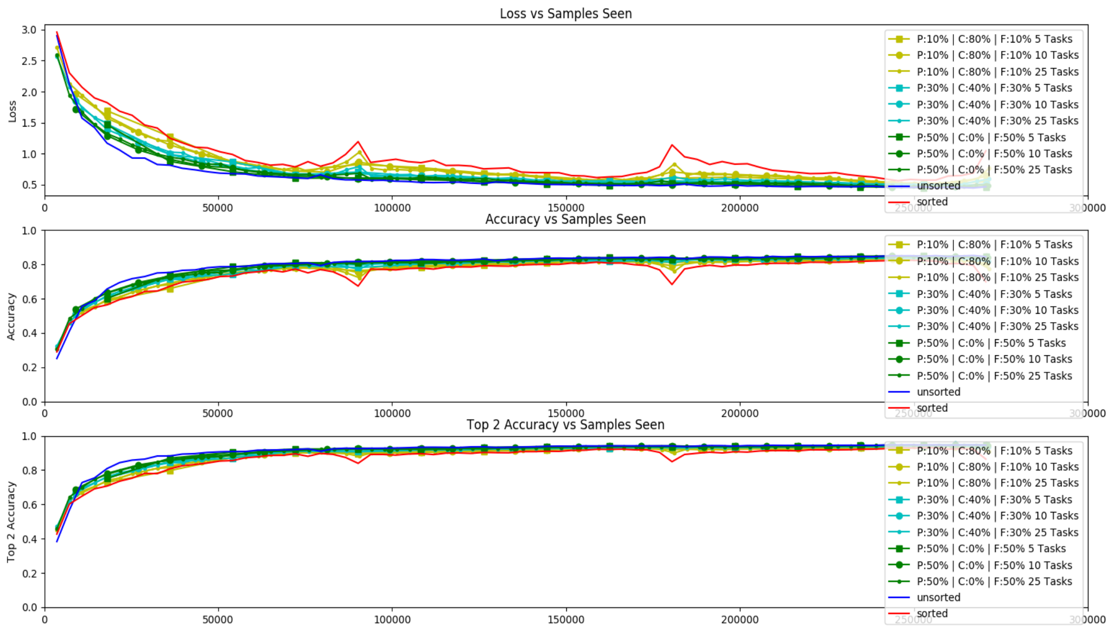


Figure 7: Here we investigated trading distribution weighting between Dp, Df and Dc in the character recognition task

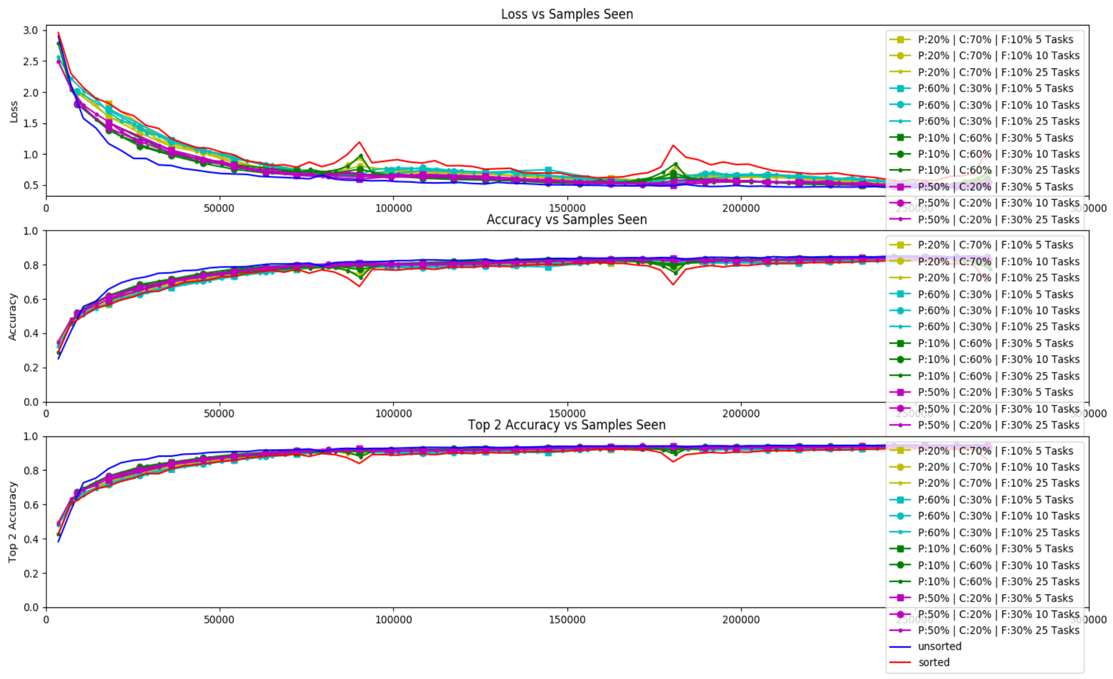
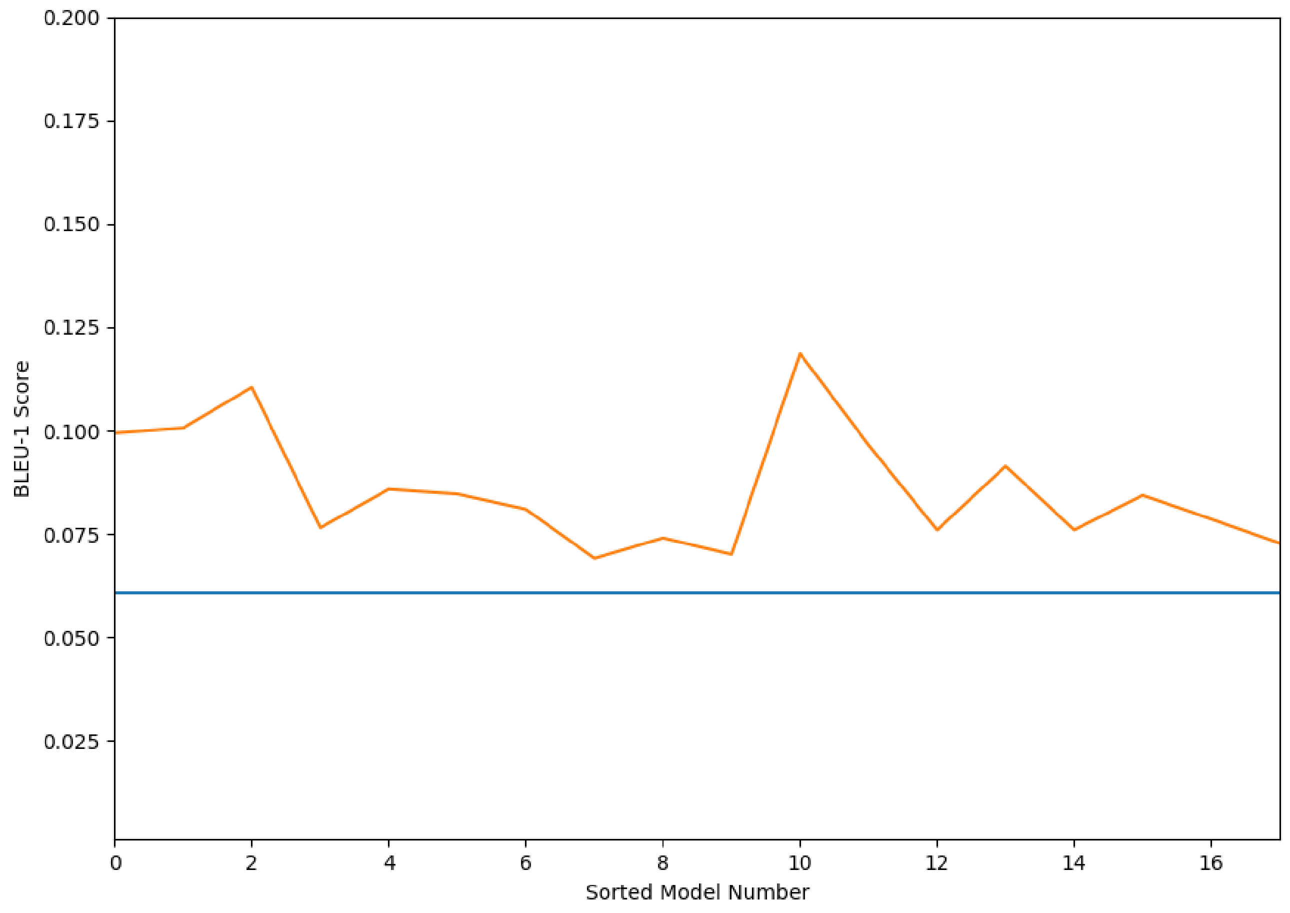


Figure 8: Here we investigated trading distribution weighting between only Dc and Dp with some fixed *Df* in the character recognition task



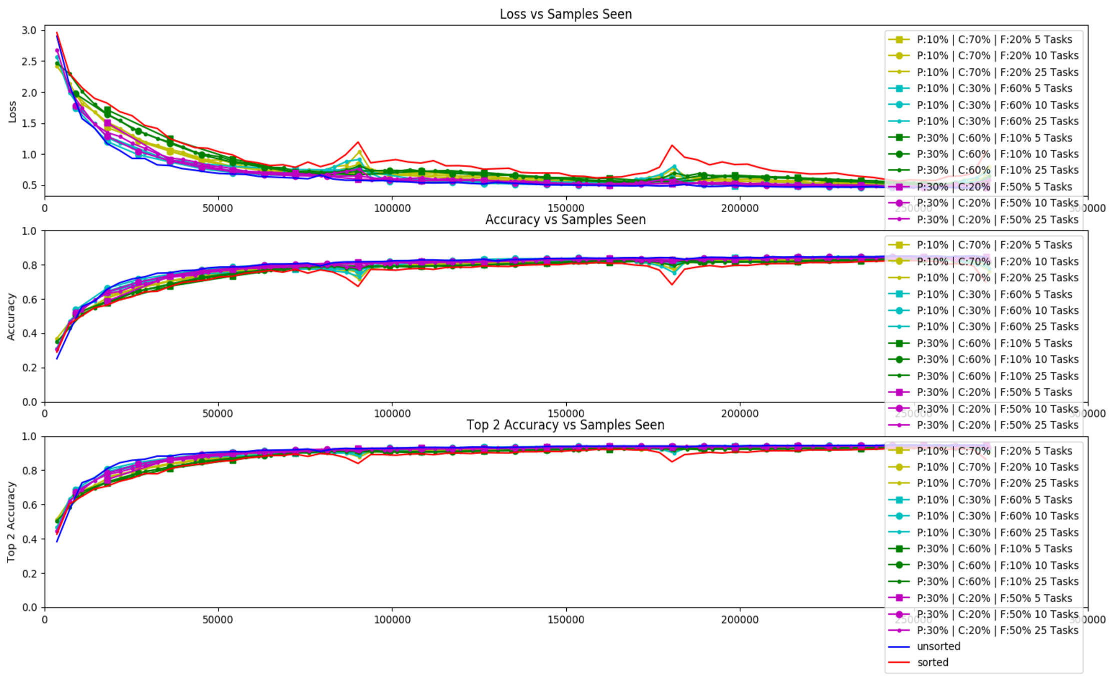


Figure 9: Here we investigated trading distribution weighting between only Dc and Df with some fixed *Dp* in the character recognition task