REPUBLIC OF TURKEY
YILDIZ TECHNICAL UNIVERSITY
GRADUATE SCHOOL OF SCIENCE AND ENGINEERING

CYCLICAL CURRICULUM LEARNING

Himmet Toprak KESGİN

MASTER OF SCIENCE THESIS
Department of Computer Engineering
Program of Computer Engineering

Supervisor
Assoc. Prof. Dr. Mehmet Fatih AMASYALI

December, 2021

REPUBLIC OF TURKEY
YILDIZ TECHNICAL UNIVERSITY
GRADUATE SCHOOL OF SCIENCE AND ENGINEERING

CYCLICAL CURRICULUM LEARNING

A thesis submitted by Himmet Toprak KESGİN in partial fulfillment of the requirements for the degree of **MASTER OF SCIENCE** is approved by the committee on 27.12.2021 in Department of Computer Engineering, Program of Computer Engineering.

Assoc. Prof. Dr. Mehmet Fatih AMASYALI
Yildiz Technical University
Supervisor

Approved By the Examining Committee

Assoc. Prof. Dr. Mehmet Fatih AMASYALI, Supervisor
Yildiz Technical University ______________________

Prof. Dr. Sırma YAVUZ, Member
Yildiz Technical University ______________________

Assoc. Prof. Dr. Hakan AYRAL, Member
Istanbul Bilgi University ______________________

I hereby declare that I have obtained the required legal permissions during data collection and exploitation procedures, that I have made the in-text citations and cited the references properly, that I haven't falsified and/or fabricated research data and results of the study and that I have abided by the principles of the scientific research and ethics during my Thesis Study under the title of Cyclical Curriculum Learning supervised by my supervisor, Assoc. Prof. Dr. Mehmet Fatih AMASYALI. In the case of a discovery of false statement, I am to acknowledge any legal consequence.

Himmet Toprak KESGİN

Signature

This study was supported by the Scientific and Technological Research Council of Turkey (TUBITAK) Grant No: 120E100.

Dedicated to my family

ACKNOWLEDGEMENTS

I would like to thank my advisor, Assoc. Prof. Mehmet Fatih AMASYALI, who guided me academically and the thesis study with his valuable ideas and guidance.

I would like to thank my family for their understanding, patience, love and support throughout my life and thesis study.

Himmet Toprak KESGİN

TABLE OF CONTENTS

LIST OF SYMBOLS

∇	Derivative
ep	End percentage of data set size
sp	Initial percentage of data set size
γ	Learning rate
l	Loss function
μ	Mean
M	Model
W	Model weights
g_t	Pacing function
p	Probability
r	Score values
f	Scoring function
α	Speed of cycle
σ	Standard deviation
D	Training data

LIST OF ABBREVIATIONS

ANN	Artificial Neural Network
CCL	Cyclical Curriculum Learning
CL	Curriculum Learning
DIH	Dynamic Instance Hardness
GCL	Greedy Curriculum Learning
NER	Named Entity Recognition
PCL	Probabilistic Curriculum Learning
PMF	Probability Mass Function
ROGS	Random Ordered Growing Sets
SPCL	Self Paced Curriculum Learning
ST	Self-taught

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Cyclical Curriculum Learning

Himmet Toprak KESGİN

Department of Computer Engineering
Master of Science Thesis

Supervisor: Assoc. Prof. Dr. Mehmet Fatih AMASYALI

Human education system is based on curriculum. The curriculum sets the subjects to be learned, the order and priority of the subjects. Artificial Neural Networks, which is a machine learning algorithm, was created with inspiration from human learning. Optimization of artificial neural networks is the process of minimizing the error in the training data set according to the determined loss function. Since neural network optimization is a non-convex optimization, the training may end with saddle points or bad local minimums. However, unlike in human learning, samples are not prioritized in classical neural network training.

The use of examples in a meaningful order in neural network training is called Curriculum Learning. There are different versions of Curriculum Learning as CL (prioritize easy examples), Anti-CL (prioritize difficult examples), Rand-CL (random selection). CL methods can accelerate the neural network optimization as well as increase the final performance of the model by obtaining a better local minimum. CL methods achieve more successful results by increasing or decreasing the dataset size throughout the training. However, a generally efficient CL method for various architectures and data sets is not found.

In this thesis, cyclical curriculum learning (CCL) method ,in which the data size used during training changes cyclically rather than simply increasing or decreasing, is proposed. The CCL method was tested for 14 text, 4 image classification datasets. In these tests, CCL significantly increases success in the top-1 accuracy metric over classical and current CL methods.

Keywords: Curriculum learning, artificial neural networks, deep learning, optimization

YILDIZ TECHNICAL UNIVERSITY
GRADUATE SCHOOL OF SCIENCE AND ENGINEERING

ÖZET

Döngüsel Planlı Öğrenme

Himmet Toprak KESGİN

Bilgisayar Mühendisliği Anabilim Dalı
Yüksek Lisans Tezi

Danışman: Doç.Dr. Mehmet Fatih AMASYALI

İnsan eğitim sistemi müfredata dayanır. Müfredat, öğrenilecek konuları, konuların sıralamasını ve önceliğini belirler. Bir makine öğrenmesi olan yapay sinir ağları insan öğreniminden ilham alır. Yapay sinir ağları optimizasyonu belirlenen kayıp fonksiyonuna göre eğitim setindeki hatayı en aza indirme işlemidir. Yapay sinir ağlarında optimizasyon dış bükey olmayan optimizasyon olduğundan eğitim eğer noktalarında veya kötü yerel minimum noktalarında sona erebilir. Ancak insan öğrenmesinden farklı olarak klasik yapay sinir ağları eğitiminde örnekler önceliklendirilmez.

Yapay sinir ağı eğitiminde örneklerin anlamlı bir sırada kullanılması planlı öğrenme olarak adlandırılır. Planlı öğrenmenin kolay örnekleri önceliklendiren, zor örnekleri önceliklendiren, örnekleri rastgele önceliklendiren versiyonları mevcuttur. Planlı öğrenme modelin eğitimini hızlandırabileceği gibi daha iyi bir local minimuma ulaşmasını sağlayarak modelin nihai performansını da iyileştirebilir. Planlı öğrenme eğitim esnasında eğitim veriseti boyutunu giderek attırırarak veya giderek azaltarak daha başarılı sonuçlar elde eder. Ancak tüm verisetleri ve modeller için genel geçer başarılı bir planlı öğrenme algoritması bulunamamıştır.

Bu tezde veri seti boyutlarının sadece arttığı veya sadece azaltığı değil belirli bir aralıkta döngüsel olarak değiştiği döngüsel planlı öğrenme yöntemi önerilmiştir. Önerilen yöntem 14 metin 4 görüntü sınıflandırma verisetinde test edilmiştir. Döngüsel planlı öğrenme mevcut yöntemlere göre oldukça başarılı sonuçlar göstermektedir.

Anahtar Kelimeler: Planlı öğrenme, derin Öğrenme, optimizasyon, yapay sinir ağları

1
INTRODUCTION

1.1 Literature Review

Our education system is structured around curricula. Curriculum determines which subjects will be learned, their order from basic to complex, and the duration of education. Using a certain meaningful subject order can facilitate learning. Curriculum is very often used in both human and animal learning [1, 2] .

Artificial neural networks (ANNs) are among the most popular and successful machine learning algorithms. ANNs are made up of interconnected nodes, inspired by the simplified form of neurons in the brain [3] . In Ann model training, training samples are presented to model in random order and without any priority throughout the entire training. The use of curriculum in machine learning algorithms is called Curriculum Learning (CL) [4] .

In neural network training, a target loss function is determined. The error made by the model in the training set is tried to be minimized according to this loss function. This process is known as optimization. When the model to be trained consists of too many parameters, it takes a long time to train the model. The use of CL can improve the ultimate performance of the deep learning model and enable faster convergence.

While performing the optimization in the vanilla method, the entire training dataset is separated into specific batches. The number of samples in each batch is called batch size. Updates are performed with these batches. An epoch is completed when all batches have been used. In contrast to the vanilla technique, CL often begins training using a subset of the training set. This subset consists of presumed easy samples. Initially epochs are performed using only this subset. Then the size of this subset is increased and the training continues. This situation is maintained until the training is completed using the entire data set.

According to the study, using CL in ANNs training can improve generalization, enhance the speed of non-convex optimization, and find a better local minimum for convex optimization [4] . The research was supported by artificially generated image classification datasets and language modeling tasks.

In order to apply curriculum to deep learning models, it is necessary to rank the samples and determine the speed of their presentation to the model. CL's scoring function and pacing function were formally defined in the study. [5] . The scoring function determines the difficulty order of the samples, while the pacing function specifies the pace at which these samples are presented to the model. They used the trained model and another model based on transfer learning to determine the difficulty of the samples. CL has been shown to be more successful then vanilla for top-1 accuracy metrics in the test set on multiple image classification datasets.

CL is divided into three algorithms: curriculum, anti-curriculum, and random-curriculum. Curriculum refers to a condition in which the training samples are arranged from easy to difficult, anti-curriculum refers to a situation in which the training samples are ordered from difficult to easy, and random-curriculum refers to a situation in which the training samples are randomly ordered. Random-curriculum training is not the same as vanilla training. Vanilla training uses the entire dataset in each epoch. Random-curriculum divides the training set into smaller chunks and focuses on this small subset at the beginning of the training. In other words, it prioritises the samples in the training set, like curriculum and anti-curriculum method. However, unlike them, it assigns this priority randomly, rather than according to a set of rules.

This thesis is divided into the following sections. Section 2 includes basic information about Curriculum Learning, algorithms, and previous studies. In the 3th section, the proposed method, the cyclical curriculum algorithm, is introduced. The datasets and models used in section 4 are included. The experimental results are explained in the 5th section, and finally, the results and discussions are included in the 6th section.

1.2 Objective of the Thesis

There are studies in the literature that have better results with curriculum [4–7] , anti-curriculum [8–10] and random-curriculum [11, 12]. However, in general, a successful CL algorithm has not been found in all datasets and models. The research compared curriculum methods for a wide range of dataset models and hyperparameters [11] . As a result, it showed that the curriculum performed better by a very small margin. They stated that the conditions where the curriculum provides significant improvement are limited training time and noisy datasets.

The purpose of this thesis is to compare existing curriculum design and training algorithms and to propose a new and generally successful curriculum training algorithm.

1.3 Hypothesis

Within the scope of this thesis, the existing curriculum methods and vanilla method in the literature were applied and compared. In contrast to these methods, a new and more successful training algorithm has been proposed.

The hypothesis of the thesis is that with the proposed Cyclical Curriculum Learning algorithm, in deep learning training, significantly better results can be obtained compared to classical training that does not use a curriculum, using the same dataset, and same training hyperparameters.

2
CURRICULUM LEARNING

CL is a training algorithm in which training samples are given to the deep learning model in a certain order. CL is based on the assumption that introducing simple examples to the model earlier can speed up training and increase the classifier's overall performance. Two functions must be defined in CL training. The scoring function f, which calculates the difficulty levels and ranks of all samples in the training set, The second function is the $g(t)$ pacing function, which determines the rate at which samples will be presented to the model.

2.1 Scoring Function

All samples must have a certain score value to be able to apply CL. These scores are determined by the f function. The key problem for CL is to specify the f function. f function defined as follows. Let $D = (x_i, y_i)$ where D represents for training data, $x_i \in R^d$ single training point and y_i its label. We defined the difficulty values of the training samples with the scoring function f. Scoring function can be any function, (x_i, y_i) is more difficult than sample (x_j, y_j) if f $(x_i, y_i) < f(x_j, y_j)$.

Deep learning algorithms perform better as the number of data increases. However, creating a training set with a large amount of data can be costly. Experts usually determine the curriculum in education, but on top of that, determining the difficulty values of all examples by experts will increase the cost even more. For this reason, algorithms that automatically generate difficulty values have been proposed.

2.1.1 Self-taught Scoring Function

One of the most widely used methods to automatically determine the f function is to use another machine learning algorithm. In this method, the training dataset is trained with a model other than the model to be trained. After the training is completed, the model's predictions for the training set are taken. The smaller the difference between the actual values and the predicted values, the easier the sample. The self-taught algorithm, on the other hand, uses the model itself to be trained to determine scores. In this method, after the training set is trained with the model, the f function is determined by using the model's predictions on the training set. Then the model is trained from scratch with different initial weights using the obtained scores. In this method, the model is trained for a given number of epochs. Then, loss values are calculated for each training sample in the training set. These loss values are then inverted and converted to probability values. The following is a formal definition of a self-taught scoring function.

$$k_i = \frac{1}{l(y_i, F_1(X_i, W)))} \tag{2.1}$$

$$r_i = \frac{k_i}{\sum k_i} \tag{2.2}$$

where l is loss function, F_1 is trained model, W is final model parameters for F_1 and r is score values for each sample, X_i is single training point y_i its label. If we used the model itself to calculate scores, this method is called *self-taught* (ST) scoring function. Following the calculation of scores, the model is trained from scratch with different initial weights.

2.1.2 Transfer Learning Scoring Function

In the transfer learning scoring function, a different model from the model to be trained is used to find the order of the training samples. Generally, a more complex and successful model is used. The point of the transfer learning scoring function is that using a more successful model can generate better rankings. In this way, CL will be able to work more effectively. Transfer Learning Scoring function is calculated as in the self-taught algorithm. Only the model from which the scores are obtained is not the same as the model to be trained, but a different model. For this, first of all, the model for which the scores are to be obtained must be trained. Then, the loss values are calculated for each sample in the training set. Lastly, these loss values are reversed and used for CL training.

2.1.3 Text Based Scoring Functions

In Curriculum Learning, automatic ordering does not have to be by training a model. For example, natural language processing datasets consist of sentences. Characteristics of sentences can be used to identify the difficulties of the samples. Two different criteria are specified for determining the difficulty using sentence characteristics. These are sentence length and n-gram based methods.

2.1.3.1 Sentence Length Scoring Function

One could argue that long sentences are more difficult because long sentences are made up of many short sentences or phrases. In order to understand a long sentence, its small parts must be understood first. Or vice versa, long sentences can be interpreted more easily as they give more clues about the meaning of the sentence. In either case, sentence length can be used as an automatic difficulty determination criterion. The sentence length scoring function can be defined as follows. Let s_i sequences of words $s_i = \{w^i_0, ..., w^i(N_i)\}$. Assuming that long sentences are easy scores calculated as follows $k_i = length(s_i)$ and $r_i = \frac{k_i}{\sum k_i}$. For using short sentences first, k_i is reversed as $1/k_i$ before normalized.

2.1.3.2 N-gram Based Scoring Functions

We use some words quite often and others rarely. Due to the structure of languages, some words are used more frequently than other words. The assumption in n gram-based methods is that sentences with frequent words are easier. Introducing rare words to the model early during training may result in poor word representations. In n gram based methods, unigram, bigram, and trigram scoring functions can be calculated. The calculation of these score functions is given as follows.

$$
\begin{aligned}
p(w_n) &= \frac{\sum_{x \in C} c(w_n)}{uc} \\
p(w_n, w_m) &= \frac{\sum_{x \in C} c(w_n, w_m)}{bc} \\
p(w_n, w_m, w_j) &= \frac{\sum_{x \in C} c(w_n, w_m, w_j)}{tc} \\
unigram - \epsilon(s_i) &= -\sum_{n=0}^{l(s_i)} p(w_n) * \log(p(w_n)) \\
bigram - \epsilon(s_i) &= -\sum_{n=0}^{l(s_i)-1} p(w_n, w_{n-1}) * \log(p(w_n, w_{n-1})) \\
trigram - \epsilon(s_i) &= -\sum_{n=0}^{l(s_i)-2} p(w_n, w_{n-1}, w_{n-2}) * \log(p(w_n, w_{n-1}, w_{n-2}))
\end{aligned}
\tag{2.3}
$$

where uc, bc, tc are the total counts of unigram, bigram and trigram in the corpus. C is corpus, $c(y)$ represents count of y, $x \in C$ sample in the corpus $w_i \in x$, $l(x)$ are word count and n-gram count in a sentence.

2.2 Pacing Functions

After the ranking and scoring of the samples are completed, it is necessary to determine at what rate they will be presented to the model. The pacing function sets the pace in which samples are presented. It is usually defined as a monotonically non-decreasing function. Each value of the pacing function contains the number of elements of subsets of the training set. Pacing function, g_t. g_t is used to determine the size of the training set's subsets $D'_1, ...D'_n \subseteq D$

There are numerous pacing functions that can be defined, including exponential, parabolic, linear, and single step. The term linear pacing function refers to a situation in which the size of the subset increases linearly. For example, let the total number of elements in the training set is 3n and the training is to be completed in three stages. In this case, after the 3n examples are ordered from easy to difficult, the easiest n samples form the 1st stage. After these samples are trained as much as the determined epoch number, n more samples are included in the training and the training continues with 2n samples as the determined epoch number. Finally, the training is completed using the entire training set.

The exponential pacing function, on the other hand, expresses that the rate at which samples are presented to the model is exponential. The starting percentage of the training set and the coefficient of increase of the exponential function should be defined. Formally, the exponential pacing function is defined as follows.

$$g_t(i) = min(sp.inc^{\lfloor \frac{i}{sl} \rfloor}, 1).N \tag{2.4}$$

where sp is starting percent, sl epoch count for each stage, inc increase amount of exponential function and N is Training set sample count.

In the parabolic pacing function, the rate at which samples are presented to the model increases as the parabolic function. There are power and starting percent parameters that need to be defined. The definition of a parabolic pacing function can be given formally as follows.

$$g_t(i) = min((sp.\lfloor \frac{i}{sl} \rfloor)^{inc}, 1).N \tag{2.5}$$

where sp is starting percent, sl epoch count for each stage, inc increase amount of parabolic function and N is training set sample count.

The single step pacing function has only two distinct size values. The training set is divided into two parts: easy and difficult. The first stage of the training is carried out with as many examples as the number of samples determined as easy. Then the training is completed using the entire data set, as in the classical method.

2.3 Curriculum Learning Training Algorithms

After determining the difficulty values and speed of the training set, namely the scoring function and pacing function, the training can be completed with one of the following algorithms.

- Greedy Curriculum Learning (GCL)
- Probabilistic Curriculum Learning (PCL)
- Self Paced Curriculum Learning (SPCL)
- Random Ordered Growing Sets (ROGS)

2.3.1 Greedy Curriculum Learning

In Greedy Curriculum Learning, all samples are ranked from easy to hard using the scoring function. Then, the training starts with the easiest n samples according to the pacing function and continues until the determined epoch number. Then, the size of the subset trained is incrementally increased until the entire dataset is used, and finally the training is completed using the entire dataset. GCL training is given in Algorithm-1.

Algorithm 1 GCL Training

Require:

M : Model to train.
D : Training dataset, consists of X: Input and y: Output.
f : Scoring function.
g : Pacing function.
T : Epoch count for training.

procedure TRAIN
    Sort D according to f, in descending order
    for $i \in \{1...T\}$ **do**
        k $= g_t(i)$
        $D_i' = D[1...k]$
        TRAIN_MODEL(M,D_i') (Perform one epoch optimization for D_i')
    end for
end procedure

2.3.2 Probabilistic Curriculum Learning

Probabilistic Curriculum Learning randomizes in such a way that easy examples are more likely to be selected, rather than greedily choosing the n easiest examples. For this, it converts the values in the scoring function into probability values to be selected. Unlike GCL, the subsets samples are not the same each epoch, as each epoch is selected according to the values in the scoring function. PCL training pseudocode is given in Algorithm-2.

Algorithm 2 PCL Training

Require:

M : Model to train.
D : Training dataset, consists of X: Input and y: Output.
f : Scoring function.
g : Pacing function.
T : Epoch count for training.

procedure TRAIN
    for $i \in \{1...T\}$ **do**
        k $= g_t(i)$
        $n \longleftarrow$ randomly pick k samples indices from distribution f
        TRAIN_MODEL(M,$D(n)$) (Perform one epoch optimization for $D(n)$)
    end for
end procedure

2.3.3 Self Paced Curriculum Learning

Everyone does not learn the same subject at the same rate. Likewise, the order of the subjects learned may vary according to everyone. Self-Paced Learning (SPL) dynamically determines the curriculum during training, rather than predetermining a fixed curriculum like other algorithms. The difficulty values and rank of the training examples change and are recalculated at the end of each epoch. Therefore, it does not require a predetermined scoring function. The training algorithm for SPL is given in Algorithm-3.

Algorithm 3 SPL Training

Require:

M : Model to train.
D : Training dataset, consists of X: Input and y: Output.
g : Pacing function.
T : Epoch count for training.

procedure TRAIN
  for $i \in \{1...T\}$ **do**
    k $= g_t(i)$
    f = CALCULATE_LOSS_VALUES(M,D) (For each sample calculate loss values and f function.)
    Sort D according to f, in descending order
    $D'_i = D[1...k]$
    TRAIN_MODEL(M,D'_i) (Perform one epoch optimization for D'_i)
  end for
end procedure

2.3.4 Random Ordered Growing Sets

In the literature, there are studies that achieve better results with both curriculum learning and anti-curriculum learning. This can implies that the improvement may result solely from increasing the size of the datasets, rather than from the use of any sorting. Unlike other algorithms, Random Ordered Growing Sets are not sorted according to any criteria, sorting is performed randomly. The difference between the classical method and training can be explained as follows. In the classical method, the entire data set is used in each epoch. In ROGS, a random subset is determined and training is performed with this subset. The model does not see other training instances until the specified epoch number is passed. ROGS can be said to be GCL where the ranking is determined randomly.

Algorithm 4 ROGS Training

Require:
M : Model to train.
D : Training dataset, consists of X: Input and y: Output.
g : Pacing function.
T : Epoch count for training.

procedure TRAIN
    Randomly shuffle dataset D
    for $i \in \{1...T\}$ **do**
        k $= g_t(i)$
        $D_i' = D[1...k]$
        TRAIN_MODEL(M,D_i') (Perform one epoch optimization for D_i')
    end for
end procedure

2.4 Previous Work

In the first study where Curriculum Learning was introduced [4] ; three different experimental studies were carried out. In the first study, it was carried out on a completely randomly generated 2-dimensional data set. The data set was primarily classified with naive bayes and the samples that the classifier predicted incorrectly were considered as noise. Then, the desired SVM to be trained was run only on the samples correctly classified by the naive bayes. Statistically significantly better results were obtained than SVM, in which all samples were used. In the second experiment, geometric shapes were created artificially. Two different data sets were created. These are BasicShapes and GeomShapes. According to GeomShapes, BasicShapes has less variability in properties such as object position, size, orientation. Curriculum is designed as follows. The model was first started with the easy training set, and after the swich epoch was reached, the training continued with GeomShapes. Better results were obtained than training without a curriculum. Finally, studies were carried out on the natural language modeling task. Curriculum design was made according to the frequency of the words. The training was primarily carried out with sentences containing only the most frequent 5000 words. Later, 5000 more words were added, and sentences in which the first 10,000 words were used. The training continued until all the sentences in the training set were used. Curriculum has reached a better minimum than all sentences were used without any curriculum.

In the study, scoring and pacing functions are defined [5] . As the scoring function, the model itself was self-taught, and the transfer scoring function, which used a more successful model than the current model, was used. They demonstrated the effectiveness of their models in test set performance at training speed.

Rather than pre-deciding on the difficulty values of the samples, SPL [13, 14] dynamically adjusts the difficulty values. It uses the loss values on the training set as the difficulty determination criterion. Samples smaller than a certain threshold value are used in each epoch in training. This threshold value is gradually increased. The training continues until the entire data set is used. In four different data sets, they experimentally demonstrated the success of SPL.

MentorNet [15] uses another strategy for curriculum. It trains two models simultaneously, mentorNet and studentNet. MentorNet determines the weights of the samples and studentNet performs the training according to these weights. In the study, extensive experiments were performed on datasets containing controlled noise and read-world noise. They showed that the performance of Deep CNNs on corrupted labels can be effectively improved.

Dynamic Instance Hardness (DIH) is another recommended curriculum strategy [16] . In this method, the difficulty values of the training samples are calculated dynamically as in SPL. However, unlike the SPL, only the instantaneous loss values of the model are not used. Loss values in previous iterations are also included. For this, a moving average is calculated over the loss values produced during the training. In addition to the losses, difficulty determination methods such as the loss difference from the previous iteration and whether the predicted label has changed are also used. Furthermore, DIH training begins with the entire dataset and gradually reduces it. In the later stages of the training, examples with high DIH values, that is, difficult ones, are focused on. DIHCL outperforms SGD and recent CL methods in terms of efficiency and final performance on 11 image classification datasets.

In his study [11] , it has been examined in which situations the curriculum works. For this purpose, extensive experiments were carried out with various scoring and pacing functions. As a result, it has been stated that the effect of curriculum learning on the final performance is very small. In addition, it showed a similar performance with random-curriculum, anti-curriculum and curriculum training algorithms. Curriculum performed quite well in tag noise and limited training time scenarios.

Rather than optimizing many hyperparameters, competence-based curriculum learning proposes a simple one-parameter curriculum method [17] . This parameter is the number of stages the curriculum will consist of. Samples below a certain difficulty value are used randomly at each stage in the training. Sentence length and word rarity are used for difficulty determination criteria. The proposed method significantly reduced the training time and improved the blue score up to 2.2.

There are also some studies reporting that the curriculum has no effect. For example, in this study in which language modeling was done with deep learning [18] ; curriculum design was linguistically motivated. The samples were sorted according to the sample length and sentence entropy values. For certain tasks, evaluations were made on glue benchmarks. There is no compelling evidence that the curriculum improves model training.

3
CYCLICAL CURRICULUM LEARNING

There are studies with curriculum, anti-curriculum, and random-curriculum that have obtained more successful results than the classical way in the studies on curriculum learning described in the previous section. These basic methods, however, cannot achieve the same level of success for every dataset and model. Also according to certain studies, CL does not improve success. Therefore, it can be said that there is no widely approved CL algorithm that works efficiently for every model and data set.

When the previous studies are examined, in some studies the data set starts from a small subset and increases gradually; in some studies, the training starts with the entire dataset and gradually decreases. This implies that changes in the size of the dataset during training may have an impact on success. In this thesis, this situation in the literature is examined, and Cyclical Curriculum Learning (CCL) is proposed, where the size of the training dataset not only increases or decreases gradually, but changes the size of the dataset cyclically.

In addition, when the training dataset subset containing a certain number of elements is used, not the entire training dataset, it is necessary to determine according to which criteria the selected samples for the subset will be selected. These examples can be chosen randomly, giving priority to the easy ones or giving priority to the hard ones. Another challenge is identifying easy and hard samples.

There are other cyclical approaches to deep learning. For learning rate, which is an important hyperparameter in deep learning; cyclical learning rates have been proposed [19]. With this method, much better results have been obtained with this method than with fixed or decreasing learning rates. The training time has been shortened considerably. In addition, the method greatly reduces the cost of hyperparameter search for the learning rate.

3.1 Cyclical Training Dataset Sizes

In classical deep learning training, the entire dataset is used in each epoch. In Curriculum approaches, the dataset sizes used in epochs increase or decrease. In the proposed CCL method, the data set size used both increases and decreases throughout the training. The size of the training set size to be used at which stage of the training is determined by hyperparameters. In Cyclical Training Dataset Sizes, the hyperparameters are starting percent, end percent, alpha, and epoch count. Starting percent specifies what fraction of the dataset the training will begin with. End percent sets the fraction of the training to go up to. The alpha parameter sets the speed at which the start fraction goes to the end percentage. Epoch count indicates the total number of epochs the training will take. The Cyclical Training Dataset Sizes algorithm is given in Algorithm-5. Figure-3.1 shows how the size of the training dataset changes

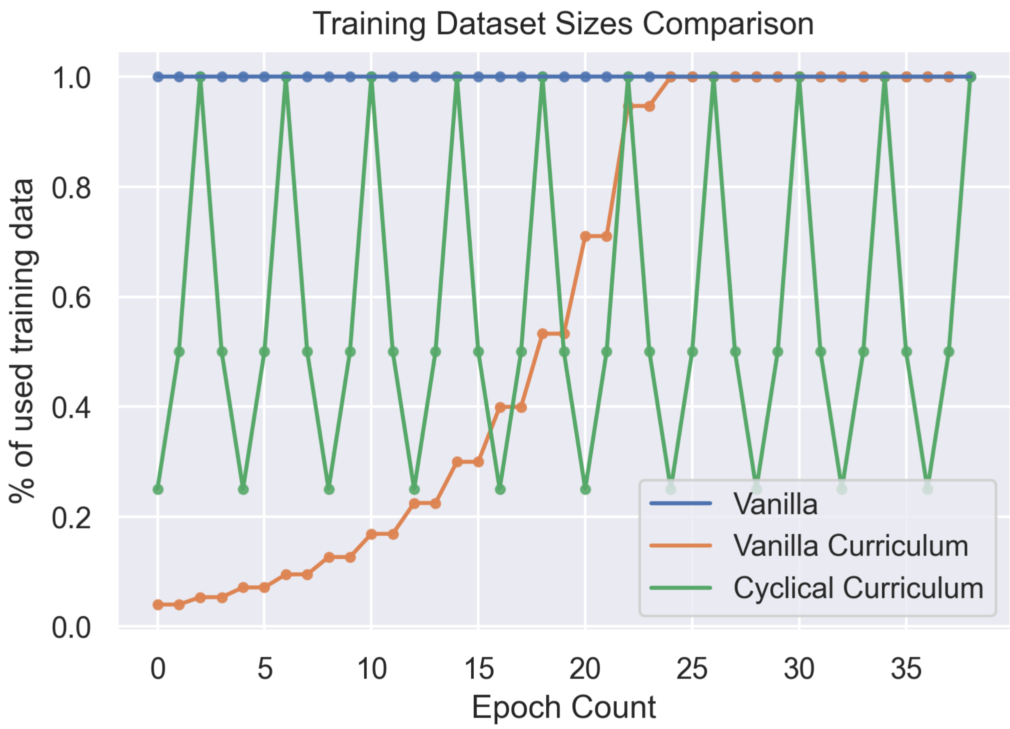

Figure 3.1 Training dataset sizes comparison

throughout the training. For Cyclical Dataset Sizes, start percent = 0.25 finish percent 1.0 alpha = 0.5 was used.

Algorithm 5 Get Dataset Sizes

Require:
T : epoch count for training.
sp : initial percentage of data set size.
ep : end percentage of data set size.
α : speed of cycle.

procedure GET_DATASET_SIZES(sp, ep, α, T)
    $s \leftarrow [\ ]$ (initialize empty list)
    $n \leftarrow sp$
    $s.append(n)$
    for $t \in \{1...(T-1)\}$ **do**
        if (n == sp) or (($S_{t-1} < S_t$) and (n!= ep)) **then**
            $n \leftarrow min((n * (1/\alpha)), ep)$
        else
            $n \leftarrow max((n * (\alpha)), sp)$
        end if
        $s.append(n)$
    end for
    return S
end procedure

3.2 Selecting Samples

Since a subset of the training dataset is used at certain stages of the training, it is necessary to determine how to select elements as much as this subset size. This selection can be made in three different ways:

- Prioritizing easy examples,
- Prioritizing difficult examples,
- Random selection.

In addition to this selection method, it should be determined whether the selection is a greedy approach as in the GCL algorithm or a probabilistic approach as in the PCL algorithm.

CCL prioritizes the easy samples in subset selection from these algorithms and makes its selection probabilistically. In order to make the probabilistic selection, the values in the scoring function are converted to the probability mass function (pmf) with the following equation.

$$r_i = \frac{f_i}{\sum f_i} \tag{3.1}$$

where f_i is scoring function, r_i pmf function to be used for random selection. The following sampling is used for probabilistic selection. $p_i \sim r(i)$ where $r(i)$ is probability scores. The training dataset sample indexes are selected according to the pmf function. Algorithm-5 (Section 3.1) determines what proportion of sample indices will be selected in each epoch. Sampling is done without replacement.

3.3 Determining Scores

All samples must have a certain score value in order to make a probabilistic random selection. In the CCL, this score is calculated as follows. First of all, the classical method is trained according to the determined hyperparameters and epoch number. Then the loss values on the model's training set are calculated. Since small loss values represent easy samples, loss values are inverted with respect to multiplication. Afterwards, all samples are divided by their sums and normalized to their probability values.

Formally, the score (probability) values for each sample is calculated using the equations below.

$$k_i = \frac{1}{l(y_i, M_1(X_i, W)))} \tag{3.2}$$

$$r_i = \frac{k_i}{\sum k_i} \tag{3.3}$$

where l is loss function, M_1 is trained model, W is final model parameters for M_1 and r is score values for each sample. Then, samples are selected with the probabilistic selection algorithm. (See section 3.2)

3.4 Training

CCL training can be started after the selection probability of each sample and the size of the cycle dataset are determined. The training is performed with the following Algorithm-6.

Vanilla training uses the entire dataset at each step in training. This corresponds to the situation where S(i) = 1 and all scores are equal in the CCL Training algorithm. When all scores are equal but dataset sizes change, it is referred to as Random CCL. CCL Training indicates that the dataset sizes in the epochs change and the scores of the samples are different.

The get_sizes function is explained in section 3.1. It returns a list that containing values such as $[0.25, 0.5, 1, 0.5, 0.25, 0.5, 1]$ which represents what fraction of dataset

Algorithm 6 Training

Require:

D : Training dataset, consists of X: Input and y: Output.
M_1 : the model by which the scores will be obtained.
M_2 : the model to train.
T : epoch count for training.
sp : initial percentage of data set size.
ep : end percentage of data set size.
α : speed of cycle.

procedure TRAIN
    TRAIN_MODEL(M_1,D) (Train the scoring model with entire training data.)
    $s \leftarrow$ GET_SIZES(ep, sp, α, T) (Section 3.1)
    $r \leftarrow$ GET_SCORES(M_1, D) (Section 4.3)
    $F_2 \leftarrow$ INITILIZE_MODEL (Initilize the model with random weights)
    for $t \in \{1...T\}$ **do**
        $n \leftarrow$ SELECT_BY_SCORES($D, s[t], r$) (Section 3.2)
        UPDATE_MODEL(M_2,$D(n)$)
    end for
end procedure

will be used corresponding epoch. The *get_scores* and *select_by_scores* functions are explained in sections 3.2 and 3.3. The process done here is the process of converting these loss values to probability values after training the model and calculating the loss values for all samples. Then, the selection of the samples is performed randomly according to these probability values.

4
DATASETS AND MODELS

To empirically measure the performance of the proposed CCL algorithm, many dataset-model pairs were collected from sources on the Internet. This section describes the relevant datasets and models. For this, four image classification and fourteen text classification dataset-model pairs were used. While choosing datasets and models, task and model diversity were prioritized.

4.1 Image Classification Datasets and Models

Four image classification data sets were used in the experiments. These are the cifar10 [20], cifar100 [20], fashion mnist (fmnist) [21] and stl_10 [22] datasets.

4.1.1 Cifar10 Dataset and Model

The cifar10 dataset consists of 10 different classes. It contains 50000 training samples and 10000 test samples in total. Each sample is 32x32 color images. The number of instances in the classes is equal. Each class contains 5000 samples for training and 1000 for testing. CNN model [23] is used for cifar-10 dataset. Summary of the cifar10 model is in Figure-4.1.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_3 (Conv2D)            (None, 30, 30, 32)        896
_________________________________________________________________
batch_normalization_3 (Batch (None, 30, 30, 32)        128
_________________________________________________________________
activation_3 (Activation)    (None, 30, 30, 32)        0
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 15, 15, 32)        0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 13, 13, 64)        18496
_________________________________________________________________
batch_normalization_4 (Batch (None, 13, 13, 64)        256
_________________________________________________________________
activation_4 (Activation)    (None, 13, 13, 64)        0
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 6, 6, 64)          0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 4, 4, 64)          36928
_________________________________________________________________
batch_normalization_5 (Batch (None, 4, 4, 64)          256
_________________________________________________________________
activation_5 (Activation)    (None, 4, 4, 64)          0
_________________________________________________________________
flatten_1 (Flatten)          (None, 1024)              0
_________________________________________________________________
dense_1 (Dense)              (None, 10)                10250
=================================================================
Total params: 67,210
Trainable params: 66,890
Non-trainable params: 320
_________________________________________________________________
```

Figure 4.1 Cifar10 dataset model summary

4.1.2 Cifar100 Dataset and Model

The other dataset used is the cifar100 dataset, which is quite similar to the cifar10. Unlike Cifar10, it includes 100 different classes. Cifar100 contains 50000 training samples and 10000 test samples in total. Each sample is 32x32 color images. Each class contains 500 samples for training and 100 for testing. Summary of the cifar100 model is in Figure-4.2.

Layer (type)	Output Shape	Param #
conv2d_27 (Conv2D)	(None, 30, 30, 32)	896
batch_normalization_31 (Batc	(None, 30, 30, 32)	128
activation_10 (Activation)	(None, 30, 30, 32)	0
max_pooling2d_15 (MaxPooling	(None, 15, 15, 32)	0
conv2d_28 (Conv2D)	(None, 13, 13, 64)	18496
batch_normalization_32 (Batc	(None, 13, 13, 64)	256
activation_11 (Activation)	(None, 13, 13, 64)	0
max_pooling2d_16 (MaxPooling	(None, 6, 6, 64)	0
conv2d_29 (Conv2D)	(None, 4, 4, 64)	36928
batch_normalization_33 (Batc	(None, 4, 4, 64)	256
activation_12 (Activation)	(None, 4, 4, 64)	0
flatten_6 (Flatten)	(None, 1024)	0
dense_10 (Dense)	(None, 64)	65600
batch_normalization_34 (Batc	(None, 64)	256
activation_13 (Activation)	(None, 64)	0
dense_11 (Dense)	(None, 100)	6500

Total params: 129,316
Trainable params: 128,868
Non-trainable params: 448

Figure 4.2 Cifar100 dataset model summary

4.1.3 Fmnist Dataset and Model

The third dataset used is fmnist. The dataset contains 10 different classes. Classes consist of 28x28 total 784 pixels. Images are greyscale. While the total number of training samples is 60000, the number of test samples is 10000. The number of samples in classes is equal in both training and testing. Summary of the fmnist model is in Figure-4.3.

Layer (type)	Output Shape	Param #
conv2d_30 (Conv2D)	(None, 26, 26, 32)	320
batch_normalization_35 (Batc	(None, 26, 26, 32)	128
activation_14 (Activation)	(None, 26, 26, 32)	0
max_pooling2d_17 (MaxPooling	(None, 13, 13, 32)	0
conv2d_31 (Conv2D)	(None, 11, 11, 64)	18496
batch_normalization_36 (Batc	(None, 11, 11, 64)	256
activation_15 (Activation)	(None, 11, 11, 64)	0
max_pooling2d_18 (MaxPooling	(None, 5, 5, 64)	0
conv2d_32 (Conv2D)	(None, 3, 3, 64)	36928
batch_normalization_37 (Batc	(None, 3, 3, 64)	256
activation_16 (Activation)	(None, 3, 3, 64)	0
flatten_7 (Flatten)	(None, 576)	0
dense_12 (Dense)	(None, 64)	36928
batch_normalization_38 (Batc	(None, 64)	256
activation_17 (Activation)	(None, 64)	0
dense_13 (Dense)	(None, 10)	650

Total params: 94,218
Trainable params: 93,770
Non-trainable params: 448

Figure 4.3 Fmnist dataset model summary

4.1.4 Stl10 Dataset and Model

The final data used for image classification tasks is stl10. Only labeled examples were used in the training. There are 5000 training and 8000 test samples labeled in the dataset. Each sample is 96x96 color images. Summary of the stl10 model in figure 4.4.

```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_33 (Conv2D)           (None, 94, 94, 32)        896
_________________________________________________________________
batch_normalization_39 (Batc (None, 94, 94, 32)        128
_________________________________________________________________
activation_18 (Activation)   (None, 94, 94, 32)        0
_________________________________________________________________
max_pooling2d_19 (MaxPooling (None, 47, 47, 32)        0
_________________________________________________________________
conv2d_34 (Conv2D)           (None, 45, 45, 64)        18496
_________________________________________________________________
batch_normalization_40 (Batc (None, 45, 45, 64)        256
_________________________________________________________________
activation_19 (Activation)   (None, 45, 45, 64)        0
_________________________________________________________________
max_pooling2d_20 (MaxPooling (None, 22, 22, 64)        0
_________________________________________________________________
conv2d_35 (Conv2D)           (None, 20, 20, 64)        36928
_________________________________________________________________
batch_normalization_41 (Batc (None, 20, 20, 64)        256
_________________________________________________________________
activation_20 (Activation)   (None, 20, 20, 64)        0
_________________________________________________________________
flatten_8 (Flatten)          (None, 25600)             0
_________________________________________________________________
dense_14 (Dense)             (None, 10)                256010
=================================================================
Total params: 312,970
Trainable params: 312,650
Non-trainable params: 320
_________________________________________________________________
```

Figure 4.4 Stl10 dataset model summary

Adam [24] optimizer is used in all image classification datasets. The loss function determined in the training is the categorical cross entropy loss function.

4.2 Text Classification Datasets and Models

In the experiments, 14 text classification data sets, different tasks and models were used. These are 20news [25] , sarcasm [26] , reuters [27] , hotel [28] , stweet [29] , ctweet [30] , qpair [31] , food [32] , sof [33] , toxic [34] , reddit [35] , squad [36] , imdb [37] , ner [38] datasets.

4.2.1 20news Dataset and Model

The 20news is a data set consisting of news documents. This dataset can be used to classify texts. The purpose of classification is to find the topic of the news document. Documents in the dataset are labeled for 20 different news topics. Pre-trained glove vectors were used to train the model. Pre-trained glove vectors have 100 dimensions. The summary of the model used in the training is as follows [39] . The 20news dataset contains 16000 training samples and 4000 test samples in total.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, None, 100)         2000200
_________________________________________________________________
conv1d (Conv1D)              (None, None, 128)         64128
_________________________________________________________________
max_pooling1d (MaxPooling1D) (None, None, 128)         0
_________________________________________________________________
conv1d_1 (Conv1D)            (None, None, 128)         82048
_________________________________________________________________
max_pooling1d_1 (MaxPooling1 (None, None, 128)         0
_________________________________________________________________
conv1d_2 (Conv1D)            (None, None, 128)         82048
_________________________________________________________________
global_max_pooling1d (Global (None, 128)               0
_________________________________________________________________
dense (Dense)                (None, 128)               16512
_________________________________________________________________
dropout (Dropout)            (None, 128)               0
_________________________________________________________________
dense_1 (Dense)              (None, 20)                2580
=================================================================
Total params: 2,247,516
Trainable params: 247,316
Non-trainable params: 2,000,200
_________________________________________________________________
```

Figure 4.5 20news dataset model summary

4.2.2 Sarcasm Dataset and Model

The sarcasm dataset consists of news headlines. What you want to classify is whether there is a mockery in these titles. So it is a two-class classification problem. The basic layer used by the classifier model is bi-directional gru [40] . The sarcasm dataset contains 25757 training samples and 2862 test samples in total.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, 100, 10)           100000
_________________________________________________________________
bidirectional (Bidirectional (None, 100, 64)           8448
_________________________________________________________________
global_average_pooling1d (Gl (None, 64)                0
_________________________________________________________________
dense (Dense)                (None, 500)               32500
_________________________________________________________________
dropout (Dropout)            (None, 500)               0
_________________________________________________________________
dense_1 (Dense)              (None, 2)                 1002
=================================================================
Total params: 141,950
Trainable params: 141,950
Non-trainable params: 0
_________________________________________________________________
```

Figure 4.6 Sarcasm dataset model summary

4.2.3 Reuters Dataset and Model

The Reuters dataset also consists of news texts. There are news belonging to 46 different categories in total. Although the number of samples in some categories is higher than the others, there are at least 10 samples in each category. Dense layers consisting of 2 hidden layers and the relu activation function were used for classification [41] . The Reuters dataset contains 8982 training samples and 2246 test samples in total.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 64)                640064
_________________________________________________________________
dense_1 (Dense)              (None, 64)                4160
_________________________________________________________________
dense_2 (Dense)              (None, 46)                2990
=================================================================
Total params: 647,214
Trainable params: 647,214
Non-trainable params: 0
```

Figure 4.7 Reuters dataset model summary

4.2.4 Hotel Dataset and Model

The hotel dataset consists of reviews written about hotels. Ratings of these comments are also available. These ratings range from 1 to 5. The score of the evaluation made with the model is tried to be estimated. The basic layer used by the classifier model is a bi-directional lstm [42] . The hotel dataset contains 13128 training samples and 6763 test samples in total.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, 350, 16)           792576
_________________________________________________________________
bidirectional (Bidirectional (None, 350, 152)          56544
_________________________________________________________________
conv1d (Conv1D)              (None, 348, 64)           29248
_________________________________________________________________
max_pooling1d (MaxPooling1D) (None, 174, 64)           0
_________________________________________________________________
flatten (Flatten)            (None, 11136)             0
_________________________________________________________________
dropout (Dropout)            (None, 11136)             0
_________________________________________________________________
dense (Dense)                (None, 128)               1425536
_________________________________________________________________
dropout_1 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_1 (Dense)              (None, 64)                8256
_________________________________________________________________
dropout_2 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_2 (Dense)              (None, 5)                 325
=================================================================
Total params: 2,312,485
Trainable params: 2,312,485
Non-trainable params: 0
_________________________________________________________________
```

Figure 4.8 Hotel dataset model summary

4.2.5 Stweet Dataset and Model

The stweet is a sentiment analysis dataset. The tweets were collected and labeled as positive and negative. The dataset, which contains a total of 1,600,000 tweets, contains 1280000 training examples and 320000 test examples. In the trained model, 300 dimensional pre-trained glove vectors were used [43] .

```
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 30)]              0
_________________________________________________________________
embedding (Embedding)        (None, 30, 300)           87172500
_________________________________________________________________
spatial_dropout1d (SpatialDr (None, 30, 300)           0
_________________________________________________________________
conv1d (Conv1D)              (None, 26, 64)            96064
_________________________________________________________________
bidirectional (Bidirectional (None, 128)               66048
_________________________________________________________________
dense (Dense)                (None, 512)               66048
_________________________________________________________________
dropout (Dropout)            (None, 512)               0
_________________________________________________________________
dense_1 (Dense)              (None, 512)               262656
_________________________________________________________________
dense_2 (Dense)              (None, 2)                 1026
=================================================================
Total params: 87,664,342
Trainable params: 491,842
Non-trainable params: 87,172,500
_________________________________________________________________
```

Figure 4.9 Stweet dataset model summary

4.2.6 Ctweet Dataset and Model

The ctweet dataset contains tweets related to the covid-19 outbreak. The labels of the samples are positive, negative and neutral. The trained model tries to predict the emotional state of the tweet as positive, negative or neutral. Bidirectional-lstm is used in the model [44] . The ctweet dataset contains 41157 training samples and 3798 test samples in total.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, 54, 16)            577872
_________________________________________________________________
bidirectional (Bidirectional (None, 54, 512)           559104
_________________________________________________________________
global_max_pooling1d (Global (None, 512)               0
_________________________________________________________________
dropout (Dropout)            (None, 512)               0
_________________________________________________________________
dense (Dense)                (None, 64)                32832
_________________________________________________________________
dropout_1 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_1 (Dense)              (None, 3)                 195
=================================================================
Total params: 1,170,003
Trainable params: 1,170,003
Non-trainable params: 0
_________________________________________________________________
```

Figure 4.10 Ctweet dataset model summary

4.2.7 Qpair Dataset and Model

The qpair dataset consists of question-sentence pairs. The purpose of the task here is to understand whether these two questions mean the same thing. In other words, it is a paraphrase detection task. In the dataset Labels; one, the sentences mean the same thing, or zero, they mean different. An lstm-based model was used for classification [45] . The qpair dataset contains 270915 training samples and 133436 test samples in total.

```
Model: "model"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_1 (InputLayer)            [(None, 100)]        0
__________________________________________________________________________________________________
input_2 (InputLayer)            [(None, 100)]        0
__________________________________________________________________________________________________
embedding (Embedding)           (None, 100, 128)     768000      input_1[0][0]
__________________________________________________________________________________________________
embedding_1 (Embedding)         (None, 100, 128)     768000      input_2[0][0]
__________________________________________________________________________________________________
bidirectional (Bidirectional)   (None, 100, 64)      41216       embedding[0][0]
__________________________________________________________________________________________________
bidirectional_1 (Bidirectional) (None, 100, 64)      41216       embedding_1[0][0]
__________________________________________________________________________________________________
global_max_pooling1d (GlobalMax (None, 64)           0           bidirectional[0][0]
__________________________________________________________________________________________________
global_max_pooling1d_1 (GlobalM (None, 64)           0           bidirectional_1[0][0]
__________________________________________________________________________________________________
dot (Dot)                       (None, 1)            0           global_max_pooling1d[0][0]
                                                                 global_max_pooling1d_1[0][0]
__________________________________________________________________________________________________
dense (Dense)                   (None, 40)           80          dot[0][0]
__________________________________________________________________________________________________
dropout (Dropout)               (None, 40)           0           dense[0][0]
__________________________________________________________________________________________________
dense_1 (Dense)                 (None, 10)           410         dropout[0][0]
__________________________________________________________________________________________________
dense_2 (Dense)                 (None, 2)            22          dense_1[0][0]
==================================================================================================
Total params: 1,618,944
Trainable params: 1,618,944
Non-trainable params: 0
```

Figure 4.11 Qpair dataset model summary

4.2.8 Food Dataset and Model

The food dataset consists of comments about meals. It is a sentiment analysis dataset. There are 2 labels, positive and negative. The food dataset contains 218503 training samples and 145670 test samples in total.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, 130, 32)           192032
_________________________________________________________________
lstm (LSTM)                  (None, 100)               53200
_________________________________________________________________
dense (Dense)                (None, 2)                 202
=================================================================
Total params: 245,434
Trainable params: 245,434
Non-trainable params: 0
_________________________________________________________________
```

Figure 4.12 Food dataset model summary

4.2.9 Sof Dataset and Model

The sof dataset contains 60000 questions asked on stackoverflow. The task in this dataset is to understand whether the question asked is of good quality. The questions are classified into 3 different categories.

1. No quality edits have been made.
2. The low quality question scored negative but the question remained open after those changes.
3. Questions closed without any modifications.

A model with three hidden layers is used for classification [46] . The sof dataset contains 45000 training samples and 15000 test samples in total.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 128)               1280128
_________________________________________________________________
dense_1 (Dense)              (None, 128)               16512
_________________________________________________________________
dense_2 (Dense)              (None, 64)                8256
_________________________________________________________________
dense_3 (Dense)              (None, 3)                 195
=================================================================
Total params: 1,305,091
Trainable params: 1,305,091
Non-trainable params: 0
_________________________________________________________________
```

Figure 4.13 Sof dataset model summary

4.2.10 Toxic Dataset and Model

The toxic dataset contains a large amount of Wikipedia comments. These reviews have been labeled by humans as to whether it is toxic or not. Pre-trained glove word embeddings and one-dimensional convolution layers are used in the training [47] . The toxic dataset contains 159571 training samples and 63978 test samples in total.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, None, 100)         2000000
_________________________________________________________________
dropout (Dropout)            (None, None, 100)         0
_________________________________________________________________
conv1d (Conv1D)              (None, None, 250)         75250
_________________________________________________________________
max_pooling1d (MaxPooling1D) (None, None, 250)         0
_________________________________________________________________
conv1d_1 (Conv1D)            (None, None, 250)         312750
_________________________________________________________________
global_max_pooling1d (Global (None, 250)               0
_________________________________________________________________
dense (Dense)                (None, 250)               62750
_________________________________________________________________
dropout_1 (Dropout)          (None, 250)               0
_________________________________________________________________
dense_1 (Dense)              (None, 2)                 502
=================================================================
Total params: 2,451,252
Trainable params: 451,252
Non-trainable params: 2,000,000
_________________________________________________________________
```

Figure 4.14 Toxic dataset model summary

4.2.11 Reddit Dataset and Model

The reddit dataset consists of 1.3 million comments on the reddit website. Comments are labelled as sarcastic or not sarcastic. An lstm-based model is used for classification [48] . The reddit dataset contains 808616 training samples and 202157 test samples in total.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, 100, 16)           128000
_________________________________________________________________
bidirectional (Bidirectional (None, 128)               41472
_________________________________________________________________
dropout (Dropout)            (None, 128)               0
_________________________________________________________________
dense (Dense)                (None, 2)                 258
=================================================================
Total params: 169,730
Trainable params: 169,730
Non-trainable params: 0
_________________________________________________________________
```

Figure 4.15 Reddit dataset model summary

4.2.12 Squad Dataset and Model

The squad dataset is different from other datasets. It is a question answer dataset, not a classification dataset. The dataset consists of Wikipedia articles and answers. Answer to each question is a segment of text or a span from the relevant reading passage. A bert model is trained for the squad dataset [49] . The toxic dataset contains 86136 training samples and 10331 test samples in total.

```
Model: "model"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_1 (InputLayer)            [(None, 384)]        0
__________________________________________________________________________________________________
input_3 (InputLayer)            [(None, 384)]        0
__________________________________________________________________________________________________
input_2 (InputLayer)            [(None, 384)]        0
__________________________________________________________________________________________________
tf_bert_model (TFBertModel)     TFBaseModelOutputWit 109482240   input_1[0][0]
                                                                 input_3[0][0]
                                                                 input_2[0][0]
__________________________________________________________________________________________________
start_logit (Dense)             (None, 384, 1)       768         tf_bert_model[0][0]
__________________________________________________________________________________________________
end_logit (Dense)               (None, 384, 1)       768         tf_bert_model[0][0]
__________________________________________________________________________________________________
flatten (Flatten)               (None, 384)          0           start_logit[0][0]
__________________________________________________________________________________________________
flatten_1 (Flatten)             (None, 384)          0           end_logit[0][0]
__________________________________________________________________________________________________
activation (Activation)         (None, 384)          0           flatten[0][0]
__________________________________________________________________________________________________
activation_1 (Activation)       (None, 384)          0           flatten_1[0][0]
==================================================================================================
Total params: 109,483,776
Trainable params: 109,483,776
Non-trainable params: 0
__________________________________________________________________________________________________
```

Figure 4.16 Squad dataset model summary

4.2.13 Imdb Dataset and Model

The imdb dataset is a two-class dataset consisting of comments about movies. Dataset contains 25000 examples for 25000 tests for training. The purpose of the dataset is to determine whether the comments are positive or negative. The Bert model was used in the training [50] .

```
  "The `lr` argument is deprecated, use `learning_rate` instead.")
Model: "model"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_1 (InputLayer)            [(None, 512)]        0
__________________________________________________________________________________________________
input_3 (InputLayer)            [(None, 512)]        0
__________________________________________________________________________________________________
input_2 (InputLayer)            [(None, 512)]        0
__________________________________________________________________________________________________
tf_bert_model (TFBertModel)     TFBaseModelOutputWit 108310272   input_1[0][0]
                                                                 input_3[0][0]
                                                                 input_2[0][0]
__________________________________________________________________________________________________
global_average_pooling1d (Globa (None, 768)          0           tf_bert_model[0][0]
__________________________________________________________________________________________________
dense (Dense)                   (None, 128)          98432       global_average_pooling1d[0][0]
__________________________________________________________________________________________________
dense_1 (Dense)                 (None, 2)            258         dense[0][0]
==================================================================================================
Total params: 108,408,962
Trainable params: 98,690
Non-trainable params: 108,310,272
__________________________________________________________________________________________________
```

Figure 4.17 Imdb dataset model summary

4.2.14 Ner Dataset and Model

In the Named Entity Recognition (ner) dataset, the purpose is to identify named entities in the text. Examples of named entities are: "Organization", "Person", "Event", "Natural Phenomenon" etc. A simple transformer-based model was used in the training [51] . The ner dataset contains 38367 training samples and 9592 test samples in total.

```
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 104)]             0
_________________________________________________________________
token_and_position_embedding (None, 104, 32)           1129056
_________________________________________________________________
transformer_block (Transform (None, 104, 32)           10656
_________________________________________________________________
time_distributed (TimeDistri (None, 104, 17)           561
=================================================================
Total params: 1,140,273
Trainable params: 1,140,273
Non-trainable params: 0
```

Figure 4.18 Ner dataset model summary

5
EXPERIMENTAL RESULTS

In this section, the proposed CCL method is compared with the classical method, which does not use any sorting, and other curriculum methods. Deep learning algorithms are stochastic algorithms. In other words, some processes in training determined randomly. For example, deep learning algorithms are initialized with random initial weights, and according to this initialization, the training may end at different local minimums. Or, batch sizes count, the samples in them, and the order of the batches affect the outcome of the training. In other words, the classical method gives different results in each trial. For this reason, in order to make a fair evaluation, methods were trained starting from the same initial weights. In total, the methods were trained five times with 5 different initial weights and the averages of the success values were taken. Evaluations were made on student-t test.

Numerous metrics can be used to evaluate machine learning algorithms. Since the datasets used in this study are class-balanced, the most commonly used top-1 accuracy metric was used for evaluation. The top-1 accuracy metric is calculated as the ratio of correctly classified samples to all samples in the test dataset. The top-1 accuracy metric was used for all datasets except for 2 datasets in this study. Since the squad dataset is not a classification dataset, but a question-answer dataset, a metric other than top-1 accuracy should be used. Therefore, the exact match between the predicted and the actual response was used as the percantege evaluation criterion. The other dataset that does not use Top-1 accuracy for evaluation is the ner dataset. There are 17 tags in total in the ner dataset used for the Named entity recognition task. However, most of these tags are 'O' tags. Even if the model predicts all tags as that tag, it reaches very high accuracy values. For this reason, the accuracy for the ner dataset is calculated without the o tag.

Hyperparameters affect the success of deep learning algorithms. The hyperparameter choices used in the experiments are explained below. For image classification datasets, 128 batch sizes were used, except for the stl10 dataset. Since stl10 contains fewer samples than other datasets, 32 batch sizes are used. In all image classification datasets, the Adam algorithm was used as the optimizer. The same hyperparameters were used for vanilla, CCL, and other curriculum algorithms throughout all trainings. For the training of text classification datasets, the hyperparameters taken from the source were used for all dataset-model pairs. For vanilla training before CCL, the vanilla epoch number in the source was used.

Besides the hyperparameters in classical training, some parameters need to be determined in order to change the training dataset sizes in CCL. In the experiments in all datasets, parameters start percent = 0.25, end percent = 1, alpha = 0.5 were used for cyclical dataset sizes. The cycle is completed by using the quarter, half, whole, half and again quarter of the data. In this way, the training continues until the total number of epochs is reached.

5.1 Training Process

After all the hyperparameters are determined, the models can start to be trained. While training the models, their performance on the test set was measured in linear intervals. The maximum accuracy achieved in these linear intervals measurements was determined as the success of the model. Some methods achieved their highest accuracy earlier in the training period, while some methods reached it later. The reason why only the final versions of the models are not used is that some models start to overfit the data after they reach their highest success, and their success on the test set starts to decrease. For fair evaluation, since all models are trained with an equal number of epochs, the evaluation criterion is used as the maximum accuracy reached.

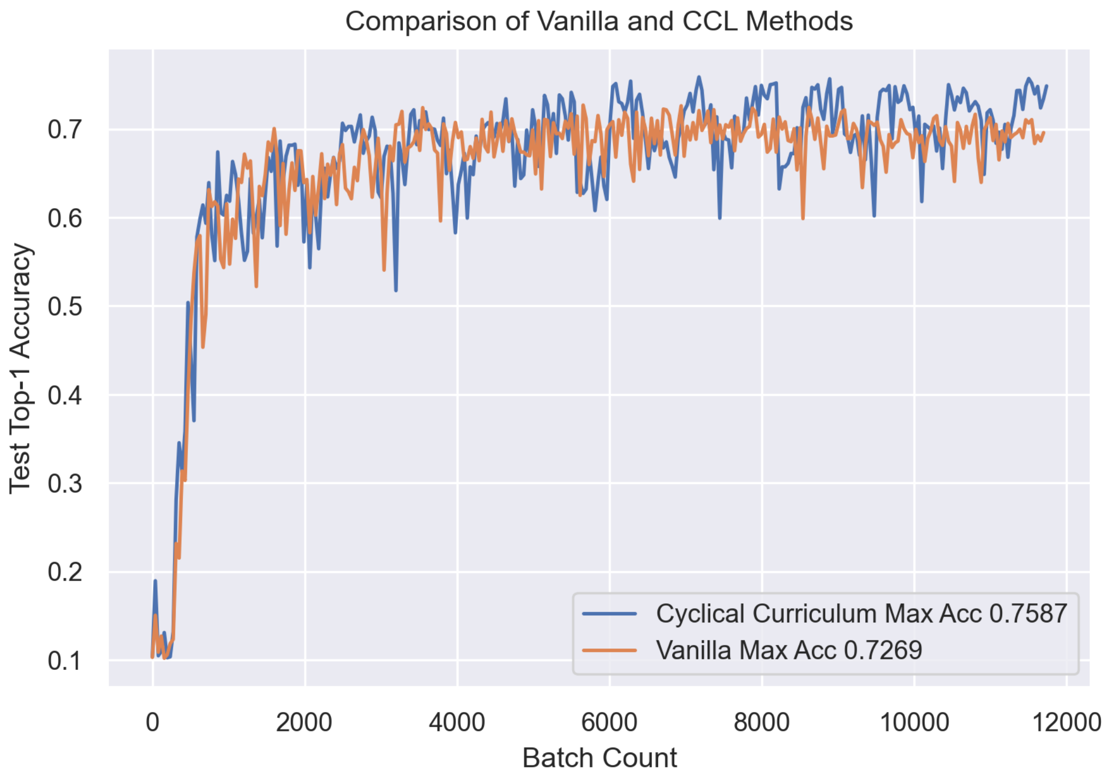

Figure 5.1 Comparison of vanilla and cyclical curriculum method for $\alpha = 0.5$, $sp = 0.25$, $ep = 1$ parameters in cifar 10 data set.

Compared to the vanilla method for CCL, the accuracy rate oscillates over a wide range, with many ups and downs. This enables it to produce better local minimums. The fluctuation seen in the accuracy value can be observed similarly in the study [19] where cyclical learning rates are suggested. In addition, after a certain epoch in classical training, while the model overfits the training set, CCL prevents or delays overfitting.

5.2 Summary of Results

In the experiments, all models were initialized with the same weights. Models were trained five times from scratch, and accuracy averages were taken. Student-test was used to compare the methods. The p value was taken as 0.05. All methods have been compared with the vanilla method. CCL's results was statistically significant than vanilla's in 10 of 18 datasets. And only once has it yielded bad results. There was no significant difference for the other 7 datasets. With comparison to vanilla, in the table statistically better ones are shown in green, worse ones are shown in red.

Different versions of CCL were also included in the experiments. The different versions are named according to the strategy of selecting the samples. These are

Table 5.1 Average maximum accuracy table

	Vanilla	CCL	Anti-CCL	Rand-CCL	CL	Anti-CL	Rand-CL
cifar10	72.09	75.51	69.83	70.01	72.98	66.46	69.26
cifar100	38.77	42.45	35.40	36.54	37.05	31.70	33.16
fmnist	90.74	91.32	90.66	90.64	91.10	89.83	90.23
stl_10	59.53	56.94	56.78	55.58	59.06	58.49	57.90
20_news	71.65	72.43	69.34	69.61	72.36	68.80	71.05
sarcasm	82.23	82.65	81.33	81.36	81.74	82.10	82.18
reuters	79.82	80.30	79.60	79.68	80.63	79.26	79.46
hotel	60.51	60.42	59.97	59.77	60.67	59.35	60.22
stweet	78.02	78.03	77.82	78.03	78.03	77.96	77.96
ctweet	85.88	85.73	85.71	85.49	85.46	83.18	84.49
qpair	78.44	78.28	78.52	78.42	78.07	78.39	78.21
food	94.04	94.20	93.92	93.99	94.18	93.81	93.96
sof	88.13	88.31	87.93	88.32	88.16	87.10	87.22
toxic	92.75	93.31	92.71	92.76	93.35	92.74	93.29
reddit	72.82	72.81	72.81	72.78	72.64	72.80	72.77
squad	78.11	78.26	78.18	77.86	78.66	77.70	78.66
imdb	92.11	92.24	91.27	92.14	91.16	91.65	92.10
ner	73.40	74.25	72.96	72.93	72.77	72.82	73.38

CCL (prioritize easy samples), Anti-CCL (prioritize difficult samples) and rand-CCL (randomize samples). Of these methods, CCL was clearly more successful than the other methods. Not only did changing the size of the dataset over time, but also using easy examples when determining the selected subsets increased the success. While CL, which is a method that only increases the dataset size, cannot outperform the vanilla method, it cannot outperform the vanilla method in methods that change the dataset size cyclically but do not use easy samples.

Other curriculum methods used are CL, Anti-CL, and Rand-CL. In these methods, scoring was done by self-taught as in CCL. The examples are divided into 3 different subsets: easy, medium, and difficult. Training at CL first started using only easy examples. Afterwards, the training continues by combining easy and medium examples, and at the end, the training is completed using the entire data set. In Anti-CL, the same process is done, starting with the difficult samples. Rand-CL randomly allocates subsets and trains the model with the same procedure.

5.3 Why does Cyclical Curriculum Works?

Some machine learning algorithms specifically deep learning models try to minimize a function throughout training. Let the F function be minimized for the D data set ($D = \{x_i, y_i\}_{i=1}^{N}$). Let D dataset consist of N samples and x_i represent the i^{th} training example and y_i its label. The function F to be minimized can be written as follows.

$$argmin(F_w) \tag{5.1}$$

$$F(w) = \frac{1}{N}\sum_{i=1}^{N} f_i(w) \tag{5.2}$$

$$f_i = l(y_i, M(x_i, w)) \tag{5.3}$$

M is the prediction for x_i of a model with w parameters, y_i is the actual label of the sample i and l is the loss function.

This optimization process can be performed with the gradient descent (GD) algorithm $w_{t+1} = w_t - \gamma \nabla_w F(w_t)$ where w all parameters of the model (weights of deep learning model) ∇_w derivative of loss function with w parameters and γ is learning rate. However, the GD algorithm is not suitable for large scale problems. Instead, in practice, the optimization is performed using the stochastic gradient descent (SGD) algorithm with the equation $w_{t+1} = w_t - \gamma \nabla_w f_i(w_t)$. The difference of the SGD algorithm from the GD algorithm is that it updates the weights at each step, not according to all samples in the training set, but according to a small number of randomly selected samples at each step. Since SGD does not use all training examples at each step, it makes some errors compared to GD. the error of updating with sample i at time t can be shown with $| Error_t = \nabla_w f_i(w_t) - \nabla_w F(w_t) |$.

If the i^{th} sample is selected from a uniform distribution it represents SGD and calculated by equation 5.4 where N is the number of samples and r_i is i^{th} sample's probability of being selected.

$$r_i = \frac{1}{N} \tag{5.4}$$

If samples are selected from the exponential distribution, samples with low loss will be more likely to be selected. We name it as exponential distributed stochastic gradient descent (ESG).

The probability of being selected in ESG is calculated by equation 5.5.

$$r_i = \frac{\lambda.exp(-\lambda.f_i)}{\sum_{n=1}^{N} \lambda.exp(-\lambda.f_i)} \tag{5.5}$$

Selecting r_i using ESG makes an error with respect to GD where all examples are used instead of the i^{th} example. The expected error made can be described by equation 5.6.

$$E[error(w_t)] = E[(\Delta_w f_i(w_t) - \Delta_w F(w_t))^2] \tag{5.6}$$

We can write expected value of the error for SGD with equation 5.7 and ESG equation 5.8 using variance bias decomposition. For $A = \Delta_w f_i(w_t)$ and $B = \Delta_w F(w_t)$

$$E_{sgd}[error(w_t)] = (E_{sgd}[A] - B)^2 + E_{sgd}[A^2] - E^2_{sgd}[A] \tag{5.7}$$

$$E_{esg}[error(w_t)] = (E_{esg}[A] - B)^2 + E_{esg}[A^2] - E^2_{esg}[A] \tag{5.8}$$

Theorem 5.1. *If the loss values come from a normal distribution $N(\mu, \sigma^2)$ at time t, the expexted error of SGD will be lower than the ESG error.*

$f_i(w_t) \sim N(\mu, \sigma^2) \implies E_{sgd}[error(w_t)] < E_{esg}[error(w_t)]$.

Proof. For $A = \Delta_w f_i(w_t)$ and $B = \Delta_w F(w_t)$, the expected value of SGD and ESG errors are given in equation 5.7 and 5.8 and can be calculated as follows.

Since $f_i(w_t) \sim N(\mu, \sigma^2)$

$$E_{sgd}[A] = \mu \tag{5.9}$$

From definition of $F(w_t)$

$$B = \mu \tag{5.10}$$

From definition of variance

$$E_{sgd}[A^2] - E^2_{sgd}[A] = \sigma^2 \tag{5.11}$$

Therefore

$$E_{sgd}[error(w_t)] = 0 + \sigma^2 \\ = \sigma^2 \tag{5.12}$$

$E_{esg}[A]$ and $E_{esg}[A^2]$ can be calculated respectively.

$$E_{esg}[A] = \frac{E[\lambda.f_i(w_t).exp(-.\lambda.f_i(w_t))]}{E[\lambda.exp(-\lambda.f_i(w_t))]} \\ = \frac{\lambda.exp(\frac{\lambda(\lambda.\sigma^2-2.\mu)}{2}).(\mu-\lambda.\sigma^2)}{\lambda.exp(\frac{\lambda(\lambda.\sigma^2-2.\mu)}{2})} \\ = \mu - \lambda\sigma^2 \tag{5.13}$$

$$E_{esg}[A^2] = \frac{E[\lambda.f_i^2(w_t).exp(-\lambda.f_i(w_t))}{E[\lambda.(w_t).exp(-\lambda.f_i(w_t))} \\ = \lambda^2.\sigma^4 - 2.\lambda.\mu.\sigma^2 + \mu^2 + \sigma^2 \tag{5.14}$$

$$E_{esg}[A^2] - E_{esg}^2[A] = \lambda^2.\sigma^4 - 2.\lambda.\mu.\sigma^2 + \mu^2 + \sigma^2 - (\mu - \lambda.\sigma^2)^2 \\ = \sigma^2 \tag{5.15}$$

Therefore

$$E_{esg}[error(w_t)] = (\mu - \lambda.\sigma^2 - \mu)^2 + \sigma^2 \\ = \lambda^2\sigma^4 + \sigma^2 \tag{5.16}$$

We calculated the expected value of the SGD error as $E_{sgd}[Error(w_t)] = \sigma^2$ and the expected value of the ESG error as $E_{esg}[error(w_t)] = \lambda^2\sigma^4 + \sigma^2$. Therefore we found that the expected value of the SGD error is lower than the expected value of the ESG error.

$$\sigma^2 < \lambda^2\sigma^4 + \sigma^2 \implies E_{sgd}[Error(w_t)] < E_{esg}[error(w_t)]$$

■

Theorem 5.2. *If the loss values come from a half-normal distribution at time t, and $\sigma\lambda < \pi$ the expected value of ESG error will be lower than the expected value of SGD error.*

$f_i(w_t) \sim HalfNorm(\mu, \sigma^2) \wedge \sigma\lambda < \pi \implies E_{esg}[error(w_t)] < E_{sgd}[error(w_t)]$.

Proof. From equation 5.7, we can calculate the expected error of SGD as follows;

Since $f_i(w_t) \sim HalfNorm(\mu, \sigma^2)$

$$\begin{aligned} E_{sgd}[A] &= \mu + \sigma\sqrt{\frac{2}{\pi}} \\ B &= \mu + \sigma\sqrt{\frac{2}{\pi}} \end{aligned} \tag{5.17}$$

$$E_{sgd}[A^2] - E_{sgd}^2[A] = \sigma^2(1 - \frac{2}{\pi}) \tag{5.18}$$

Therefore

$$\begin{aligned} E_{sgd}[Error(w_t)] &= 0^2 + \sigma^2(1 - \frac{2}{\pi}) \\ &= \sigma^2(1 - \frac{2}{\pi}) \end{aligned} \tag{5.19}$$

From equation 5.8, we can calculate the expected error of ESG as follows;

$$E_{esg}[A] = \frac{E[\lambda \Delta_w f_i(w_t).\exp(-\lambda \Delta_w f_i(w_t)]}{E[\lambda.\exp(-\lambda \Delta_w f_i(w_t)]} \tag{5.20}$$
$$= \frac{\lambda(\mu - \sigma^2\lambda)(\exp(\frac{\sigma^2\lambda^2}{2} - \lambda\mu)).\operatorname{erfc}(\frac{\sigma.\lambda.\sqrt{2}}{2}) + \frac{\sigma\sqrt{2}\exp(-\mu.\lambda)}{\sqrt{\pi}}}{\lambda(\exp(\frac{\sigma^2\lambda^2}{2} - \lambda\mu))\operatorname{erfc}(\frac{\sigma.\lambda.\sqrt{2}}{2})}$$
$$= \mu - \sigma^2\lambda + \frac{\sigma\sqrt{2}.\exp(\frac{-\sigma^2\lambda^2}{2})}{\operatorname{erfc}(\frac{\sigma\lambda\sqrt{2}}{2})\sqrt{\pi}}$$

We can call last part of this expression as C for ease of representation.

$$C = \frac{\sigma\sqrt{2}.\exp(\frac{-\sigma^2\lambda^2}{2})}{\operatorname{erfc}(\frac{\sigma\lambda\sqrt{2}}{2})\sqrt{\pi}} \tag{5.21}$$

$$E_{esg}[A^2] = \frac{E[\lambda \Delta_w f_i^2(w_t).\exp(-\lambda \Delta_w f_i(w_t)]}{E[\lambda.\exp(-\lambda \Delta_w f_i(w_t)]} \tag{5.22}$$
$$= \sigma^4\lambda^2 - 2\sigma^2\lambda\mu + \mu^2 + \sigma^2 - C(\sigma^2\lambda - 2\mu)$$

$$E_{esg}[A^2] - E_{esg}^2[A] = \sigma^4\lambda^2 - 2\sigma^2\lambda\mu + \mu^2 + \sigma^2 - C(\sigma^2\lambda - 2\mu) - (\mu - \sigma^2\lambda + C)^2 \tag{5.23}$$
$$= \sigma^2 + C\sigma^2\lambda - C^2$$

$$E_{esg}[A] - B = \mu + \sigma\sqrt{\frac{2}{\pi}} - (\mu - \sigma^2\lambda + C) \tag{5.24}$$
$$= \sigma\sqrt{\frac{2}{\pi}} + \sigma^2\lambda - C$$

Therefore

$$E_{esg}[Error(w_t)] = (\sigma\sqrt{\frac{2}{\pi}} + \sigma^2\lambda - C)^2 + \sigma^2 + C\sigma^2\lambda - C^2 \tag{5.25}$$
$$= (\sigma^2\lambda - C)(\sigma^2\lambda + 2\sigma\frac{2}{\pi}) + \sigma^2(\frac{2}{\pi} + 1)$$

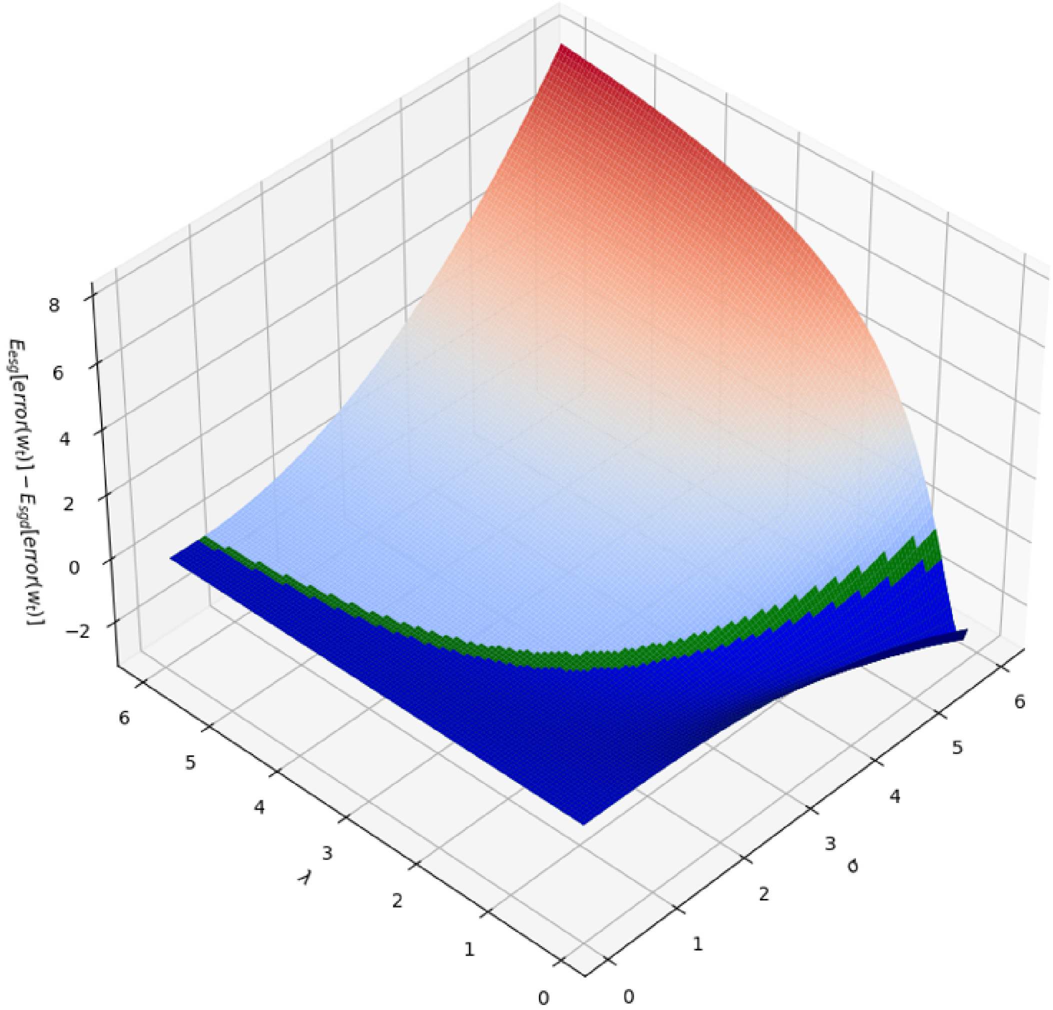

Figure 5.2 $E_{esg}[Error(w_t)] - E_{sgd}[Error(w_t)]$ for σ and λ

Since the C term contains the erfc function, its analytic equivalent is not written. For this reason, we evaluate these expressions numerically. We plotted $E_{esg}[error(w_t)] - E_{sgd}[error(w_t)]$ according to σ and λ in Figure-5.2. Figure-5.2 shows the region where ESG is lower than SGD. The blue area shows that expected value of ESG error is lower than expected value of SGD error. And the green line shows the line where $\lambda\sigma = \pi$.

As seen in Figure-5.2, $\sigma\lambda <= \pi \implies E_{esg}[Error(w_t)] < E_{sgd}[Error(w_t)]$.

■

Observation — 1. *In deep learning models, the losses of the samples come from the normal distribution before the training starts. As the training continues, the losses of the samples change to the half normal distribution.*

Observation — 2. *During training, when the model is optimized with SGD, the training sample's losses changes normal distribution to half normal distribution, when the model is optimized with ESG, the training sample's losses changes half normal distribution to normal distribution. See figure-5.3.*

Figure-5.3 shows the variation of the losses of the samples during the training of the deep learning model. In the 1st graph, while the training has not started, the losses are distributed according to the normal distribution. In this case, after SGD is applied to the losses, the 2nd graph is formed. In 2nd graph, the losses turn into a half normal distribution. If SGD is continued to be applied at this stage, the losses will remain in the half normal distribution. However, at this stage, if ESG, which makes less error than SGD, is applied to the losses distributed with the half normal distribution, the loss values of the samples will again resemble the normal distribution. In the CCL algorithm, this situation continues throughout the training.

Theorem 5.3. *The expected value of the error of applying SGD and ESG cyclically (CCL algorithm) is smaller than SGD.*

$$MSE(CCL) < MSE(SGD) \tag{5.26}$$

Proof. From observation-1, the losses come from the normal distribution before the training starts.

From observation-2, if the losses come from the normal distribution, when SGD is applied, the losses are distributed according to the half normal distribution, and when the losses are distributed according to the half normal distribution, when ESG is applied, the losses turn into the normal distribution.

According to Theorem-5.1, if the losses come from a normal distribution, the expected value of the error of applying SGD is lower than ESG.

According to Theorem-5.2, if the losses come from a half normal distribution, the expected value of the error of applying ESG is lower than SGD.

CCL uses SGD when losses are normally distributed, and ESG when losses are half normally distributed. Because of this CCL makes fewer errors than SGD.

$$MSE(CCL) < MSE(SGD) \tag{5.27}$$

Therefore CCL is a better estimator than SGD for F.

∎

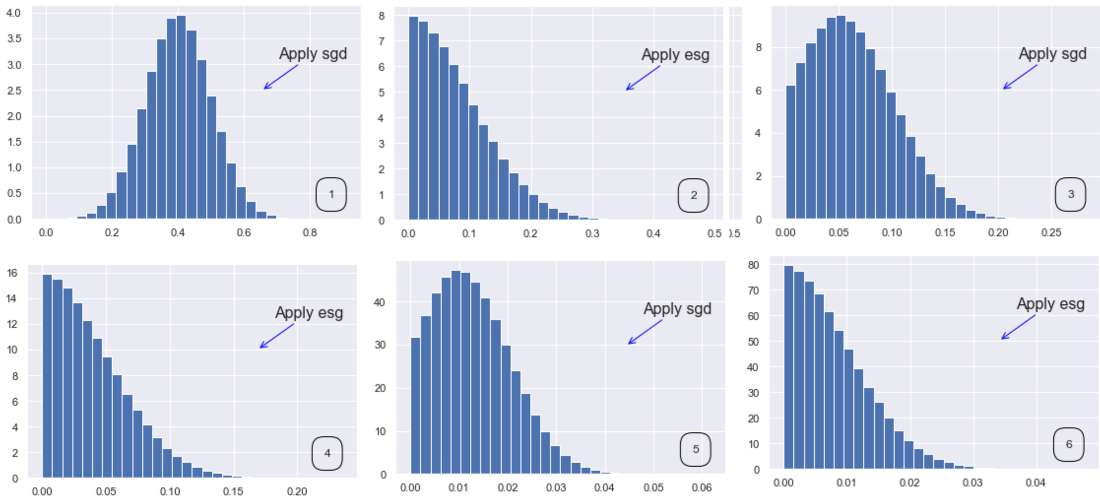

Figure 5.3 Distribution of loss values according to different selection possibilities

Theorem 5.4. *If $r_i = \frac{1}{f_i \sum \frac{1}{f_i}}$ is defined for ESG, the expected value of CCL error is lower than SGD.*

Proof. When we solve equation-5.28 for λ, we get $\lambda = \frac{ln(x)}{x}$. This function's maximum value is $\frac{1}{e}$ for $X = e$. From teorem-5.2 ESG makes less error than SGD where $\lambda\sigma < \pi$ When we replace lamda with its maximum value we get

$$exp(-\lambda.x) = \frac{1}{x} \tag{5.28}$$

$$\frac{1}{e}\sigma < \pi. \tag{5.29}$$

$$\sigma < \pi.e$$

For values of $\sigma < \pi.e$, using $\frac{1}{x}$ instead of $exp(-\lambda.x)$ the expected value of CCL error is lower than SGD.

∎

6
RESULTS AND DISCUSSIONS

Optimization of deep learning networks is performed using derivative and backpropagation. It is tried to minimize the error made by the model in the training dataset for the determined loss function. This is a non-convex optimization process. Therefore, training may end at bad local minimum or saddle points.

Curriculum Learning (CL) is proposed to improve optimization in deep networks. Training in CL starts with a subset of the dataset, this subset consists of easy examples. Then the used dataset size is gradually increased. There are studies in the literature showing that success can be increased with CL, Anti-CL, Rand-CL methods. These studies generally increase the size of the dataset and there are studies that decrease the size of the dataset. There are also studies in the literature reporting that CL has no effect on success.

In this study, existing CL methods were examined and a literature review was carried out. CCL, which is a method that both increases and decreases the dataset size during the training, has been proposed, inspired by the existing methods that reduce or increase the dataset size and cyclical learning rates. CCL determines the cycle of the dataset size with the start percent, end percent, and alpha parameter. In training, subset samples are selected based on predetermined scores calculated by the model itself. In addition, studies can be carried out to determine these scores during training in order to avoid the cost of training another model in determining the difficulties in the training set. The proposed CCL method has been compared with classical training and other curriculum methods for different tasks, architectures and datasets. In these comparisons, significantly successful results were obtained with CCL compared to the vanilla method.

REFERENCES

[1] K. A. Krueger, P. Dayan, "Flexible shaping: How learning in small steps helps," *Cognition*, vol. 110, no. 3, pp. 380–394, 2009.

[2] B. F. Skinner, "Reinforcement today.," *American Psychologist*, vol. 13, no. 3, p. 94, 1958.

[3] "Artificial neural network - wikipedia." (Nov. 11, 2021), [Online]. Available: `https://en.wikipedia.org/wiki/Artificial_neural_network` (visited on 11/12/2021).

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

[5] G. Hacohen, D. Weinshall, "On the power of curriculum learning in training deep networks," in *International Conference on Machine Learning*, PMLR, 2019, pp. 2535–2544.

[6] B. Xu, L. Zhang, Z. Mao, Q. Wang, H. Xie, Y. Zhang, "Curriculum learning for natural language understanding," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020, pp. 6095–6104.

[7] D. Weinshall, G. Cohen, D. Amir, "Curriculum learning by transfer learning: Theory and experiments with deep networks," in *International Conference on Machine Learning*, PMLR, 2018, pp. 5238–5246.

[8] T. Kocmi, O. Bojar, "Curriculum learning and minibatch bucketing in neural machine translation," *arXiv preprint arXiv:1707.09533*, 2017.

[9] X. Zhang *et al.*, "An empirical exploration of curriculum learning for neural machine translation," *arXiv preprint arXiv:1811.00739*, 2018.

[10] X. Zhang, P. Shapiro, G. Kumar, P. McNamee, M. Carpuat, K. Duh, "Curriculum learning for domain adaptation in neural machine translation," *arXiv preprint arXiv:1905.05816*, 2019.

[11] X. Wu, E. Dyer, B. Neyshabur, "When do curricula work?" *arXiv preprint arXiv:2012.03107*, 2020.

[12] M. N. Mermer, M. F. Amasyali, "Scalable curriculum learning for artificial neural networks," *IPSI BGD TRANSACTIONS ON INTERNET RESEARCH*, vol. 13, no. 2, 2017.

[13] M. Kumar, B. Packer, D. Koller, "Self-paced learning for latent variable models," *Advances in neural information processing systems*, vol. 23, 2010.

[14] L. Jiang, D. Meng, Q. Zhao, S. Shan, A. G. Hauptmann, "Self-paced curriculum learning," in *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[15] L. Jiang, Z. Zhou, T. Leung, L.-J. Li, L. Fei-Fei, "Mentornet: Learning data-driven curriculum for very deep neural networks on corrupted labels," in *International Conference on Machine Learning*, PMLR, 2018, pp. 2304–2313.

[16] T. Zhou, S. Wang, J. A. Bilmes, "Curriculum learning by dynamic instance hardness," *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[17] E. A. Platanios, O. Stretcu, G. Neubig, B. Poczos, T. M. Mitchell, "Competence-based curriculum learning for neural machine translation," *arXiv preprint arXiv:1903.09848*, 2019.

[18] D. Campos, "Curriculum learning for language modeling," *arXiv preprint arXiv:2108.02170*, 2021.

[19] L. N. Smith, "Cyclical learning rates for training neural networks," in *2017 IEEE winter conference on applications of computer vision (WACV)*, IEEE, 2017, pp. 464–472.

[20] A. Krizhevsky, G. Hinton, *et al.*, "Learning multiple layers of features from tiny images," 2009.

[21] H. Xiao, K. Rasul, R. Vollgraf, "Fashion-mnist: A novel image dataset for benchmarking machine learning algorithms," *arXiv preprint arXiv:1708.07747*, 2017.

[22] A. Coates, A. Ng, H. Lee, "An analysis of single-layer networks in unsupervised feature learning," in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, JMLR Workshop and Conference Proceedings, 2011, pp. 215–223.

[23] "Convolutional neural network (cnn) | tensorflow core." (Nov. 6, 2021), [Online]. Available: `https://www.tensorflow.org/tutorials/images/cnn?hl=en` (visited on 11/06/2021).

[24] D. P. Kingma, J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[25] "Home page for 20 newsgroups data set." (Sep. 22, 2014), [Online]. Available: `http://qwone.com/~jason/20Newsgroups/` (visited on 11/11/2021).

[26] R. Misra, P. Arora, "Sarcasm detection using hybrid neural network," *arXiv preprint arXiv:1908.07414*, 2019.

[27] "Uci machine learning repository: Reuters-21578 text categorization collection data set." (Nov. 11, 2021), [Online]. Available: `https://archive.ics.uci.edu/ml/datasets/reuters-21578+text+categorization+collection` (visited on 11/11/2021).

[28] M. H. Alam, W.-J. Ryu, S. Lee, "Joint multi-grain topic sentiment: Modeling semantic aspects for online reviews," *Information Sciences*, vol. 339, pp. 206–223, 2016.

[29] A. Go, R. Bhayani, L. Huang, "Twitter sentiment classification using distant supervision," *CS224N project report, Stanford*, vol. 1, no. 12, p. 2009, 2009.

[30] "Coronavirus tweets nlp - text classification | kaggle." (Sep. 8, 2020), [Online]. Available: `https://www.kaggle.com/datatattle/covid-19-nlp-text-classification` (visited on 11/19/2021).

[31] "Question pairs dataset | kaggle." (Feb. 2, 2017), [Online]. Available: https://www.kaggle.com/quora/question-pairs-dataset (visited on 11/19/2021).

[32] J. J. McAuley, J. Leskovec, "From amateurs to connoisseurs: Modeling the evolution of user expertise through online reviews," in *Proceedings of the 22nd international conference on World Wide Web*, 2013, pp. 897–908.

[33] I. Annamoradnejad, J. Habibi, M. Fazli, "Multi-view approach to suggest moderation actions in community question answering sites," 2020.

[34] "Toxic comment classification challenge | kaggle." (Nov. 19, 2021), [Online]. Available: https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge (visited on 11/19/2021).

[35] M. Khodak, N. Saunshi, K. Vodrahalli, "A large self-annotated corpus for sarcasm," *arXiv preprint arXiv:1704.05579*, 2017.

[36] P. Rajpurkar, J. Zhang, K. Lopyrev, P. Liang, "Squad: 100,000+ questions for machine comprehension of text," *arXiv preprint arXiv:1606.05250*, 2016.

[37] A. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, C. Potts, "Learning word vectors for sentiment analysis," in *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies*, 2011, pp. 142–150.

[38] "Annotated corpus for named entity recognition | kaggle." (Sep. 21, 2017), [Online]. Available: https://www.kaggle.com/abhinavwalia95/entity-annotated-corpus (visited on 11/19/2021).

[39] K. Team. "Using pre-trained word embeddings." (Nov. 15, 2021), [Online]. Available: https://keras.io/examples/nlp/pretrained_word_embeddings/ (visited on 11/19/2021).

[40] "Sarcasm detection:a guide for ml and dl approach | kaggle." (Nov. 11, 2021), [Online]. Available: https://www.kaggle.com/subbhashit/sarcasm-detection-a-guide-for-ml-and-dl-approach (visited on 11/11/2021).

[41] "Github - yogeshchandra12345/reuters-newswire-topics-classification: Reuters newswire topics classification." (Nov. 11, 2021), [Online]. Available: https://github.com/yogeshchandra12345/Reuters-newswire-topics-classification (visited on 11/11/2021).

[42] "Hotel reviews sentiment prediction | kaggle." (Nov. 11, 2021), [Online]. Available: https://www.kaggle.com/shahraizanwar/hotel-reviews-sentiment-prediction (visited on 11/11/2021).

[43] "Nlp beginner - text classification using lstm | kaggle." (Nov. 19, 2021), [Online]. Available: https://www.kaggle.com/arunrk7/nlp-beginner-text-classification-using-lstm (visited on 11/19/2021).

[44] "Covid19 tweets- sentiment prediction rnn [85% acc] | kaggle." (Nov. 19, 2021), [Online]. Available: https://www.kaggle.com/shahraizanwar/covid19-tweets-sentiment-prediction-rnn-85-acc (visited on 11/19/2021).

[45] "Quora similarity task wide neural network | kaggle." (Nov. 19, 2021), [Online]. Available: https://www.kaggle.com/nilanml/quora-similarity-task-wide-neural-network/comments (visited on 11/19/2021).

[46] "Stack overflow qa classification - baseline models | kaggle." (Nov. 19, 2021), [Online]. Available: https://www.kaggle.com/imoore/stack-overflow-qa-classification-baseline-models (visited on 11/19/2021).

[47] "Glove : Toxic comments classification | kaggle." (Nov. 19, 2021), [Online]. Available: https://www.kaggle.com/vortexkol/glove-toxic-comments-classification (visited on 11/19/2021).

[48] "Lstm text classification on reddit sarcasm | kaggle." (Nov. 19, 2021), [Online]. Available: https://www.kaggle.com/yudhagalang/lstm-text-classification-on-reddit-sarcasm (visited on 11/19/2021).

[49] "Dredwardhyde/bert-examples: Fine tuning bert for text classification and question answering using tensorflow pytorch frameworks." (Nov. 19, 2021), [Online]. Available: https://github.com/dredwardhyde/bert-examples (visited on 11/19/2021).

[50] "Fine-tuning with custom datasets — transformers 4.12.5 documentation." (Nov. 19, 2021), [Online]. Available: https://huggingface.co/transformers/custom_datasets.html (visited on 11/19/2021).

[51] "Named entity recognition using transformers." (Nov. 15, 2021), [Online]. Available: https://keras.io/examples/nlp/ner_transformers/ (visited on 11/19/2021).

PUBLICATIONS FROM THE THESIS

Projects

1. Doç.Dr. Mehmet Fatih AMASYALI, 120E100, "Yapay Öğrenmede Eğitim Örneklerini Sıralama Optimizasyonu", TÜBİTAK 1001, Bursiyer, 2020.