Learning Theory & Advance Machine Learning Project

Being Friends Instead of Adversaries: Deep Networks Learn from Data Simplified by Other Networks

Ihsan ULLAH IHSAN2131@GMAIL.COM
Ousmane CISSE OUSMANE.CISSE@UNIVERSITE-PARIS-SACLAY.FR
Mohammed LANSARI MOHAMMED.LANSARI@UNIVERSITE-PARIS-SACLAY.FR
Imad BOUHOU IMAD.BOUHOU@UNIVERSITE-PARIS-SACLAY.FR
M2 - Artificial Intelligence, Université Paris-Saclay

1. Introduction

In the research community, researchers have experimented in various ways training Neural Networks to increase the efficiency and quality of training. This paper specifically focuses on the approaches that deals with the training data. **Curriculum Learning** (CL) approach deals with the training such that first easy example are given to the network for training and the difficulty of the examples is gradually increased until the end of the training which is done with the most difficult example from the training set. The purpose of this method is to train the network like humans by designing a curriculum which changes from easy to hard with time.
Another approach is **Self-Paced Learning** (SPL) in which some examples are removed from the training set or their impact in the risk function is reduced.

CL and SPL are common in a way that they both either subselect some data from training set or they re-arrange examples for the training but they do not manipulate or transform the data itself.
Friendly Training (FT) on the other hand is a recent approach introduced which transforms or simplifies the data for training. The process include training with simplified data but data simplification is gradually removed with training in a way that a the end of the training the data provided is in its original format. It is expected that FT can be beneficial for noisy examples or datasets annotated with noisy labels. The data transformation process in FT is independent for each example i.e. FT perturbates each data example from scratch.

2. Motivation (from FT to NFT)

The Friendly Training data simplification process may have some regularities that are common among several training examples. For this reason similar simplifications for nearby data in training may work better rather than independent data simplifications. This way of data simplification is not discovered by FT. To tackle this, an auxiliary multi-layer network is introduced which is responsible for transforming the input data. This auxiliary network learns how to transform the data to improve the learning process. The weights of auxiliary network defines the state of the network which is constantly updated along the training process. This network is binded with the actual network i.e. the classifier network (in case of supervised classification setting). The auxiliary network extends the classifier network as it is added as extra initial layers which transforms the given input before passing through the classifier network. The effect of auxiliary network is reduced gradually during the training in a way that at the end of the training, it fades away. This process of training is named as **Neural Friendly Training** (NFT). The contributions of NFT are stated below:

1. A training strategy to simplify training data by using an auxiliary network that fades out at the end.
2. Extended use of FT to non-artificial data.
3. Comparison of FT and NFT using convolutional and fully connected neural architectures.

3. Neural Friendly Training

A generic classification problem is considered to explain Neural Friendly Training in detail.

Given a training set $\mathcal{X}$ with n number of pair examples and labels given as: $\mathcal{X} = \{(x_k, y_k), k = 1, ..., n\}$ where $x_k \in \mathbb{R}^d$. The neural network (classifier) is denoted by $f(x, w)$ where x is the input and w is a vector with weights

and biases of the network.
We denote $\mathcal{B}$ as mini-batch and $\mathcal{L}$ as the empirical risk which measures a mismatch between the ground truth and the predicted labels.
The empirical risk is given as:

$$\mathcal{L}(\mathcal{B}, w) = \frac{1}{|\mathcal{B}|} \sum_{i=1}^{|\mathcal{B}|} \ell(f(x_i, w), y_i) \tag{1}$$

where $\mathcal{B} \subset \mathcal{X}$ of size $|\mathcal{B}| \geq 1$. $(x_i, y_i) \in \mathcal{B}$ and ℓ is the loss function. ℓ is averaged over a mini-batch. Mini-Batches are non-overlapping and randomly sampled at each training epoch. This is also known is Classic Training (CT).

Friendly Training – In Classic Training, the training data is given to the network independently on the state of the network. The data in each batch $\mathcal{B}$ in this case could be heterogeneous i.e. the examples could have outliers or their distribution could be multi-modal. Curriculum Learning tries to solve this by arranging the input data in increasing order of difficulty or complexity. Usually humans determine the complexity which could be different than what the complexity is for a machine. With Self-Paced Learning approach, the value of the loss function ℓ is used to assess the complexity. The examples with largest loss values are either removed or their contribution is reduced.

Friendly Training, on the other hand transforms the training data according to the state of the network. The main purpose of doing this it to remove parts of the information which are too complex for the network to learn with current weights while keeping the information which can be learned easily. This process is done in two phases in which the first phase the training data is transformed to make sure it is easily manageable for the the network while in the second phase, the network is updated using Classic Training approach but only using the simplified data rather than the original data. The whole process is designed in a way that the alteration of data is reduced gradually with training until the end of the training where the data is used in its original form. While testing, the classifier network deployed does not rely on transformed data any more.
FT perturbs the training data by estimating the variation δ_i.

$$\tilde{x_i} = x_i + \delta_i \tag{2}$$

For each sample x_i, δ_i is estimated from scratch. The main goal remains the same i.e. to minimize $\mathcal{L}$ in Equation 1 by replacing x_i with the transformed $\tilde{x_i}$. Accurate δ_i is determined by an optimization procedure and the strength of perturbation is progressively reduced with time.

Neural Friendly Training – In the simplification of data in FT, there might be some regularities in the data. To tackle this, a more structured transformation function is introduced which is shared by all the examples in the data. A training sample x_i is transformed into $\tilde{x_i}$ by using a transformation function given in Equation 3.

$$\tilde{x_i} = s(x_i, \theta) \tag{3}$$

where $(x_i, \tilde{x_i}) \in \mathbb{R}^d \times \mathbb{R}^d$ and θ is a set of learnable parameters shared by all examples. In the settings of this paper, s is a neural network called *auxiliary network* parameterized by θ. δ_i has the same definition in NFT setting i.e. $\delta_i = \tilde{x_i} - x_i$. The *main network* is the network which implements the function f also known as the classifier network. A proposed sketch of NFT model is shown in Figure 1

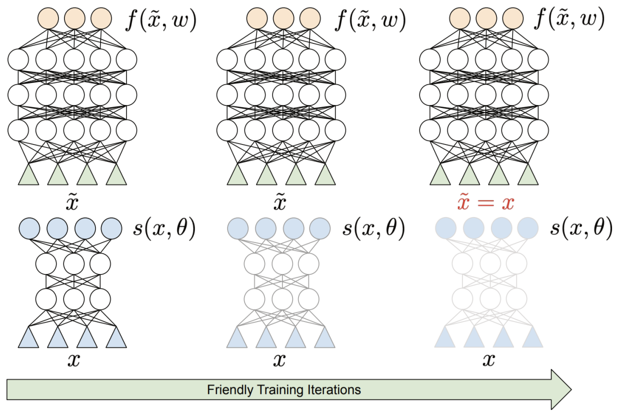

Figure 1. Neural Friendly Training - proposed sketch of NFT model: main deep network (top) and auxiliary network (bottom).

The risk function $\mathcal{L}$ from Equation 1 is redefined to meet the current settings below:

$$\mathcal{L}(\mathcal{B}, w, \theta) = \frac{1}{|\mathcal{B}|} \sum_{i=1}^{|\mathcal{B}|} \Bigg(\ell(f(\underbrace{s(x_i, \theta)}_{\tilde{x_i}}, w), y_i) + \eta || \underbrace{s(x_i, \theta) - x_i}_{\delta_i} ||^2 \Bigg) \tag{4}$$

Where $\eta > 0$ is the weight of the squared Euclidean norm of the perturbation δ_i. $\gamma \geq 1$ indicated NFT iteration index where each iteration has two phases. In first phase the auxiliary network is updated by minimizing Equation 4 with respect to θ while in the second phase, the main network is updated by minimizing the same equation with respect to w. γ_{max} is the maximum NFT iterations and it is ensured that after $\gamma_{max\_simp} < \gamma_{max}$ the data is not perturbed anymore. At each iteration γ in training the value of η is computed using Equation 5.

$$\eta = \eta_{max}\left(1 - \left[1 - \frac{\gamma - 1}{\gamma_{max\_simp} - 1}\right]_+^2\right) \quad (5)$$

In the equation above, $\eta \in [0, \eta_{max}]$. At γ_{max_simp} iterations, the penalty on $||\delta_i||^2$ reaches its maximum weighting which enforces the function $s(., \theta)$ to get close to the identity function. For this reason after γ_{max_simp} iterations the auxiliary network does not modify the data, hence it will be dropped and the data takes its original form and then becomes the input for the main network. The implementation of NFT is described in detail in later sections.

4. Experiments

The experiments focus on evaluating the behavior of Neural friendly Training (NFT) versus Friendly Training (FT). The authors experimented with both approaches on 3 main activities considering the Easy-Example First Training (EEFT) and Classic Training (CT) as a baseline.

1. Advanced digit and shape recognition.
2. Sentiment analysis.
3. Image classification.

4.1. Models and Architectures

During the experiments, the authors used the 4 classifiers that are in (Marullo et al.2021):

- **FC-A** and **FC-B** are two feed forward Fully-Connected multi-layer neural network architectures. **FC-A** is a simple one-hidden layer (10 hidden neurons) and **FC-B** have 5 hidden layers (2500-2000-1500-1000-500 neurons) with batch normalization and ReLU activations.
- **CNN-A** is composed of 2 convolutional layers (32-64 filters) with max pooling, dropout and 2 fully connected layers (9216-128 neurons).
- **CNN-B** consists of 4 convolutional layers (32-48-64-64 filters) and 2 fully connected layers(5184-128 neurons).
- Both **CNNs** have **cross-entropy** as loss function and **Adam** as an optimizer.

4.2. Training configurations

The authors adapted the choice of the auxiliary network according to the type of input it is supposed to simplify. In the cases of experiments involving images (advanced digit and shape recognition, and Image classification), the auxiliary network is inspired by the U_{Net}, which progressively down-samples the image by encoding contextual information into convolutional feature maps that will be passed on to the decoder part of the U_{Net}, then up-samples and transforms the data until its size matches the input size by exploiting hop connections. And for sentiment analysis, they chose a fully connected network with 256 hidden neurons.
The **initialization conditions** of the networks were done in a random way but in order to have the same initialization for the FT, NFT and CT and the results averaged over 3 executions.
As a **metric**, they chose the error rate which is the most used metric in classification problems.
The **Hyper-parameters** set as follows:

- For CT, Adam's learning rate was fixed by a preliminary experiment.
- For FT, the grid search was used to fix the hyper-parameters (η_{max}, γ_{max_simp}, α β, n_f), such that :
  - γ index the number of epoch (iteration number).
  - η is the weight of the euclidean norm in Equation 4.
  - (α, β) are the parameters of the Adam optimizer. α is the learning rate, and β represent the exponential decays.
  - γ_{max_simp} the maximum iterator index after which the data is not simplified anymore

4.3. Advanced digit and shape recognition

4.3.1. Dataset

For the purposes of this test, they formed a group of 5 other datasets:

- mnist-rot: MNIST digits, rotated by a random angle θ, such that $\theta \in [0, 2\pi]$
- mnist-back-image: MNIST digits, with the background replaced with patches extracted from random images of public domain.
- mnist-rot-back-image: MNIST digits, with both the rotation and background changes combined.
- rectangles-image: white rectangles, wide or tall, with the inner and outer regions filled with patches taken from random images.
- convex: convex or non-convex white regions on a black background.

Hyperparameters values:

Parameters	Values
γ_{max}	200 epochs
η_{max}	$\{500, 1000, 2000\}$
γ_{max_simp}	$\{0.25, 0.5, 0.85\} \times \gamma_{max}$
β	$\{10^{-5}, 10^{-4}, 5.10^{-4}\}$

4.3.2. Results

By running the different models on the different datasets, the authors obtained the following results:

		mn-back	mn-rot-back	mn-rot	rectangles	convex
FC-A	CT	$28.34_{\pm 0.09}$	$64.06_{\pm 0.31}$	$43.16_{\pm 0.51}$	$24.31_{\pm 0.21}$	$33.91_{\pm 0.44}$
	EEF	$\mathbf{28.18}_{\pm 0.47}$	$64.27_{\pm 0.19}$	$43.91_{\pm 0.73}$	$24.48_{\pm 0.11}$	$\mathbf{33.17}_{\pm 0.93}$
	FT	$28.66_{\pm 0.06}$	$64.14_{\pm 0.36}$	$43.24_{\pm 0.43}$	$24.64_{\pm 0.37}$	$34.38_{\pm 0.22}$
	NFT	$\mathbf{28.15}_{\pm 0.04}$	$64.55_{\pm 0.14}$	$\mathbf{42.96}_{\pm 0.58}$	$24.57_{\pm 0.19}$	$34.25_{\pm 1.03}$
FC-B	CT	$21.06_{\pm 0.39}$	$51.71_{\pm 0.79}$	$10.13_{\pm 0.27}$	$25.10_{\pm 0.20}$	$27.24_{\pm 0.05}$
	EEF	$21.38_{\pm 0.18}$	$52.95_{\pm 0.63}$	$\mathbf{10.04}_{\pm 0.17}$	$\mathbf{24.84}_{\pm 0.32}$	$28.21_{\pm 0.96}$
	FT	$21.74_{\pm 0.26}$	$\mathbf{51.02}_{\pm 0.07}$	$11.19_{\pm 0.37}$	$\mathbf{24.14}_{\pm 0.53}$	$27.49_{\pm 0.07}$
	NFT	$\mathbf{20.91}_{\pm 0.52}$	$\mathbf{50.20}_{\pm 0.16}$	$\mathbf{10.09}_{\pm 0.32}$	$\mathbf{25.09}_{\pm 0.09}$	$\mathbf{26.81}_{\pm 0.15}$
CNN-A	CT	$7.25_{\pm 0.16}$	$29.05_{\pm 0.45}$	$7.48_{\pm 0.14}$	$9.86_{\pm 0.32}$	$8.24_{\pm 0.09}$
	EEF	$\mathbf{7.02}_{\pm 0.08}$	$29.12_{\pm 0.34}$	$7.61_{\pm 0.22}$	$12.82_{\pm 0.70}$	$8.72_{\pm 0.74}$
	FT	$\mathbf{6.80}_{\pm 0.19}$	$\mathbf{28.74}_{\pm 0.29}$	$\mathbf{7.36}_{\pm 0.06}$	$\mathbf{9.72}_{\pm 0.20}$	$8.59_{\pm 1.44}$
	NFT	$\mathbf{6.59}_{\pm 0.09}$	$\mathbf{28.67}_{\pm 0.35}$	$\mathbf{7.17}_{\pm 0.17}$	$10.99_{\pm 1.89}$	$\mathbf{8.03}_{\pm 0.23}$
CNN-B	CT	$5.15_{\pm 0.15}$	$23.05_{\pm 0.21}$	$6.58_{\pm 0.06}$	$8.10_{\pm 1.90}$	$3.01_{\pm 0.41}$
	EEF	$\mathbf{4.82}_{\pm 0.19}$	$\mathbf{22.89}_{\pm 0.49}$	$7.02_{\pm 0.28}$	$8.35_{\pm 1.01}$	$3.75_{\pm 0.58}$
	FT	$\mathbf{5.03}_{\pm 0.11}$	$\mathbf{22.81}_{\pm 0.36}$	$6.95_{\pm 0.12}$	$\mathbf{7.32}_{\pm 1.31}$	$\mathbf{2.87}_{\pm 0.42}$
	NFT	$\mathbf{4.96}_{\pm 0.34}$	$\mathbf{22.22}_{\pm 0.62}$	$\mathbf{6.48}_{\pm 0.25}$	$\mathbf{6.27}_{\pm 0.62}$	$\mathbf{2.78}_{\pm 0.34}$

Figure 2. Comparison of different classifiers (FC-A, FC-B,CNN-A, CNN-B) and learning algorithms (CT, EEF, FT from (Marullo et al. 2021) and our NFT)

These results confirm that the models of the Friendly training family perform better in general than the baseline models (CT and EEFT). The proposed NFT almost always improves the results of FT, supporting the idea of using an auxiliary network to capture regularities in the simplification process (Except for Rectangles).

4.4. Sentiment analysis

4.4.1. Data and training conditions

The authors also tried to see how NFT behaves in NLP, and they applied it to the task of Sentiment Analysis which is based on 2 datasets.

- The IMDB dataset which is a large movie review dataset that has a collection of 50k highly-polar reviews from the IMDB database. Each review is represented by its TF-IDF representation with a vocabulary of the 20K most frequent words.
- The Wines dataset which collects 130k wine reviews scored in [80,100], that the authors divided into 2 classes [80,90) and [90,100]. The textual representation of each review is obtained thanks to a pretrained transformer-based architecture DistilRoBERTa.

Hyperparameters values:

Parameters	Values
γ_{max}	30 epochs
η_{max}	$\{10, 100, 500, 1000, 2000\}$
γ_{max_simp}	$\{0.05, 0.1, 0.25, 0.5, 0.85\} \times \gamma_{max}$
β	$\{10^{-5}, 10^{-4}, 5.10^{-4}\}$

4.4.2. Results of the Authors

	imdb		wines
FC-B CT	$13.27_{\pm 0.19}$	FC-B CT	$17.38_{\pm 0.15}$
FC-B FT	$13.66_{\pm 0.69}$	FC-B FT	$\mathbf{17.07}_{\pm 0.11}$
FC-B NFT	$\mathbf{11.93}_{\pm 0.09}$	FC-B NFT	$\mathbf{17.15}_{\pm 0.12}$

Figure 3. Comparison of classifiers with different architectures and learning algorithms (CT, FT), Mean test error is reported with standard deviation

CT's performance is consistently improved by NFT, achieving lower error rates in both data sets. These results confirm the versatility of NFT. The sentence classification task seems to be slightly more difficult in WINES due to the fact that wine critics are less dispersed.

4.5. Image classification

4.5.1. Data and training conditions

The authors of the paper described how they performed their experiments on the following datasets:

- CIFAR-10 dataset is a popular Image Classification dataset that has 60k 32x32 colored images with 10 classes. The CIFAR-10 dataset was later divided into training and validation sets such that 10k examples are used for the validation.
- CIFAR-10N10 is CIFAR-10 dataset with some noise introduced by randomly swapping 10% of the target labels.

Hyperparameters values:

Parameters	Values
γ_{max}	250 epochs
η_{max}	$\{500, 1000, 2000\}$
γ_{max_simp}	$\{0.25, 0.5, 0.7\} \times \gamma_{max}$
β	$\{10^{-5}, 10^{-4}, 5.10^{-4}\}$

They performed a more specific experimental activity by replacing CNN-B with a ResNet18 (He et al. 2016), inheriting all the carefully selected optimization parameters and tricks that give state-of-the-art results in CIFAR-10.

4.5.2. Results of the Authors

The Table 3 represents the test errors obtained from the paper and the test errors obtained from our reproduced experiments.
In the table 3 and specifically the experiments of the authors (Original), FT does not improve CT results while NFT slightly improves it for the CNN-B Architecture, whereas FT and NFT reduce the test error for the ResNet Architecture.

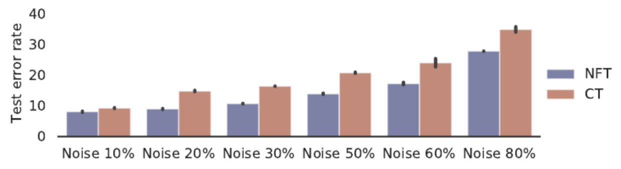

Figure 4. ResNet18 on CIFAR-10 dataset for different amounts of noisy labels. Error bars include standard deviation

The impact of NFT is becoming increasingly evident in highly noisy environments, confirming that data simplification helps the main network to better discard noisy information.

5. Reproduced Experiments

We have reproduced some of the experiments in this paper. The experimental settings are kept the same as described in the paper. The authors of this paper have used heavy computation power to run all the experiments several time. Unlike them, we are not equipped with GPUs and due to that reason we were not able to run all the experiments multiple times. We have used a combination of Personal Computers (without GPUs) and Google Colab to get some results.

We have chosen the three experiments with the following datasets:

- Wine dataset - Text dataset
- IMDB dataset - Text dataset
- CIFAR-10 dataset - Image dataset

The experimental settings, parameters and hyper parameters are kept as described in the section above. In Table 1 we compare our results for the Wine dataset with the results presented in the paper. We observe that, even without high computational power, the results are closer to the original.

Table 1. Wine dataset results (test error rate)

Training Method	Original	Reproduced
FC-B CT	17.38 ± 0.15	17.38
FC-B FT	$\mathbf{17.07} \pm \mathbf{0.11}$	17.25
FC-B NFT	$\mathbf{17.15} \pm \mathbf{0.12}$	**17.17**

In Table 2, we present the results for IMDB dataset. We failed to reproduce one experiment because we got a CUDA error and we are not able to fix it. We observe that for CT the result is close the the interval given by the paper but for NFT we have a fairly pronounced error in our experiment compared to their experiment.

Table 2. IMDB dataset results (test error rate)

Training Method	Original	Reproduced
FC-B CT	13.27 ± 0.19	13.41
FC-B FT	13.66 ± 0.69	Failed to reproduce
FC-B NFT	$\mathbf{11.93} \pm \mathbf{0.09}$	**12.63**

Some of the experiments for CIFAR-10 dataset were not run for number of epochs defined for the experiments because of the limit of time to use Google Colab but we still share the available results for comparison with the original results in Table 3. We observe that although the error rates from our results are not that close to the original results but they still show similar pattern and can be reproduced closer to the original with sufficient resources. For CNN-B models we have better results than authors for CT and FT and the result of NFT is close to what they get.

Table 3. CIFAR-10 dataset results (test error rate)

Training Method	Original	Reproduced
CNN-B CT	29.75 ± 0.37	28.91
CNN-B FT	30.19 ± 0.53	28.60
CNN-B NFT	$\mathbf{29.00} \pm 0.36$	29.45
ResNet CT	9.30 ± 0.16	14.64
ResNet FT	$\mathbf{8.92} \pm 0.23$	**10.03**
ResNet NFT	$\mathbf{8.10} \pm 0.19$	14.12

We show the error rate and the loss for this experiment with Classical Training (CT), Friendly Training (FT) and Neural Friendly Training (NFT) using CNN-B in Figure 5 and Figure 6. Figure 5 and Figure 6 show the same for ResNet. For the test loss, we can see that all learning methods have approximately the same error loss with some difference at the begging depending on the model. For the test error we can see that the FT and NFT have a better scores than CT but unfortunately we do not have the test error because of Codalab.

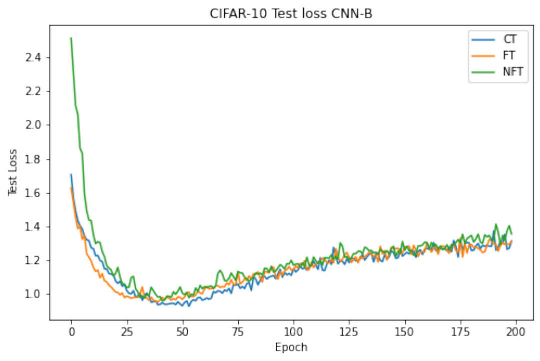

Figure 5. CIFAR-10 Test Loss with CNN-B

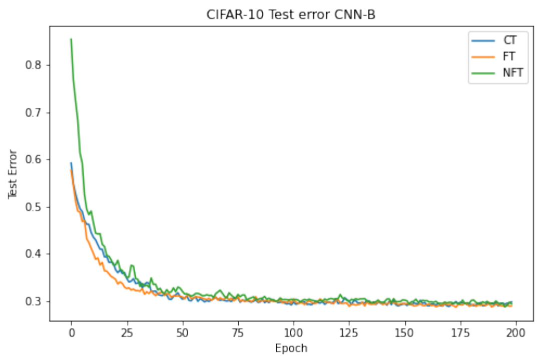

Figure 6. CIFAR-10 Test Error with CNN-B

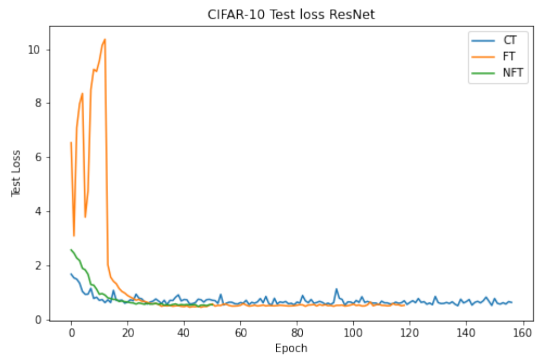

Figure 7. CIFAR-10 Test Loss with ResNet

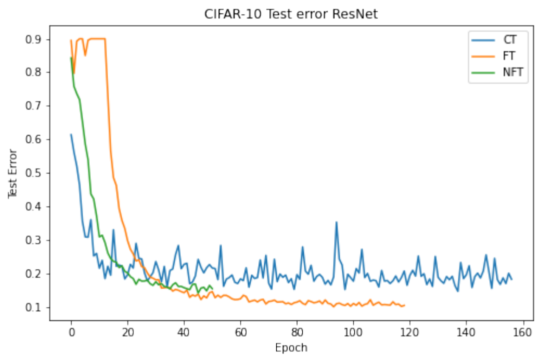

Figure 8. CIFAR-10 Test Error with ResNet

The authors noted that state-of-the-art CNNs achieve much lower test errors on the CIFAR-10 dataset, and they focused on the ResNet18 which is fine-tuned specifically to achieve state-of-the-art results on the CIFAR-10 and they tried to improve it using FT and NFT. Another CNN that was considered during the experiments was the CNN-B architecture.
The Figure 8 represents the changes of the reproduced test error with the ResNet18 when using CT, FT and NFT.
From the Figure 8, we can see that FT is better than NFT. The test error during CT is not stable and changes between distant values even after 100 Epoch.
The Figure 7 represents the changes of the reproduced test loss with ResNet18 when suing CT, FT and NFT.
From the Figure 7, we see that the test loss during CT, FT and NFT converge almost to the same place. At the beginning the loss shows significant instability for FT. The loss in FT seems to converge to a slightly better values compared to CT and NFT.
The Figures 5 and 6 represents respectively the changes of the test loss and the test error obtained for the CNN-B architecture when trained on CIFAR-10 using CT, FT and NFT.
The test loss in Figure 5 and the test error in Figure 6 has almost the same behavior for CT, FT and NFT. Even though the test loss is increasing after 50 Epochs, the test error remains low.

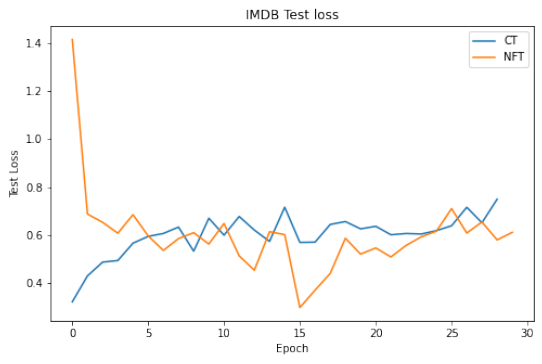

Figure 9. IMDB Test Loss

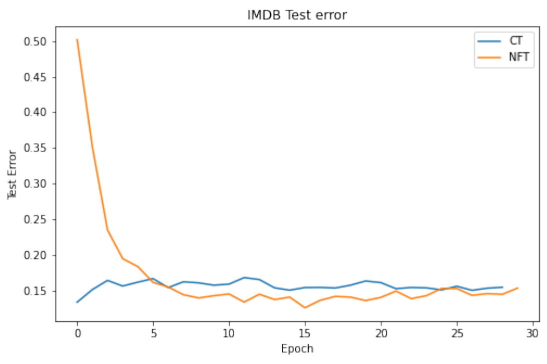

Figure 10. IMDB Test Error

The FC-B architecture was trained for 30 Epochs using the IMDB and Wines datasets.
The Figure 9 represents the variations of the test loss during 30 Epochs, and we can see that NFT achieves lower test loss values at the Epoch 15.
The Figure 10 represents the variations of the test error during 30 Epochs, and we can see that the NFT converges to lower errors than CT.
Both the test loss and test error start high for the NFT and then drops to lower values, but in CT they don't have a behavior where we can say that the loss improved or not, because they start with a certain values and almost stays in the neighborhood of that value until the end of the training.

6. Conclusion and Discussion

This paper has proposed a very interesting way of modification of the training data before giving it to the Neural Network. The use of an auxiliary network for all the examples in the dataset and the fading-out of the role of auxiliary network till the end of the training is a great technique. The paper is very well written with easy language. The experimental settings and all the figures are well explained. The results produced in the paper are also very clear and can be reproduced as the code is published by the authors. We can note that there is a big difference for the CIFAR-10 using ResNet but we can not conclude due to the fact we did not finish the experiment with full epochs and we run it just one time.

For researchers (specially students) with low resources and low computing power, the reproducibility of the results is difficult however even with low resources, the results produced are close to the original results presented in the paper.

The paper gives good insight into the modification of data before training for the purpose of simplifying the data for the neural network. We would like to discover and work more on this topic in the future to establish a good benchmark for Neural Friendly Training.