Machine Learning
Final Project - By the Paper: "A More Tolerant Teacher Educates Better Students"

Maxim Bragilovski: 205695612 , Omer Reichstein: 307860296

July 21, 2021

1 Introduction

The goal of the final exercise is to evaluate the performance of Deep Learning ensemble methods that were published in the professional literature that weren't covered during the course. In the following section, we will introduce three algorithms. Random Forest that we used as our baseline, algorithm from [1], and improved algorithm.

1.1 Random-Forest

Random forest is a supervised learning algorithm. The "forest" it builds, is an ensemble of decision trees, usually trained with the "bagging" method. The general idea of the bagging method is that a combination of learning models increases the overall result. Random-forest builds multiple decision trees and merges them to get a more accurate and stable prediction. The algorithm works as follows: for each tree in the forest, we select a bootstrap sample from S where S (i) denotes the ith bootstrap. We then learn a decision tree using a modified decision-tree learning algorithm. The algorithm is modified as follows: at each node of the tree, instead of examining all possible feature-splits, we randomly select some subset of the features f ⊆ F. where F is the set of features. The node then splits on the best feature in f rather than F. In practice, f is much, much smaller than F. Deciding on which feature to split is oftentimes the most computationally expensive aspect of decision tree learning. By narrowing the set of features, we drastically speed up the learning of the tree[1].

1.2 A More Tolerant Teacher Educates Better Students

The algorithm that we choose to implement is from the article "A More Tolerant Teacher Educates Better Students" [1]. The challenge that the article is focused on is training a deep neural network in generations (Hu et al. 2016). The purpose of training in generations is to optimize the target network (student) with the help of another network (teacher) with the same architecture.

To address this idea, training in generations consists of two steps:

[1]$https://builtin.com/data-science/random-forest-algorithm$

$$\mathcal{L}^{\mathrm{S}}\left(\mathcal{B};\boldsymbol{\theta}^{\mathrm{S}}\right)=-\frac{1}{|\mathcal{B}|}\sum_{(\mathbf{x}_n,\mathbf{y}_n)\in\mathcal{B}}\left\{\lambda\cdot\mathbf{y}_n^{\top}\ln\mathbf{f}\left(\mathbf{x}_n;\boldsymbol{\theta}^{\mathrm{S}}\right)+\right.$$
$$\left.(1-\lambda)\cdot\mathrm{KL}\left[\mathbf{f}\left(\mathbf{x}_n;\boldsymbol{\theta}^{\mathrm{T}}\right)\|\mathbf{f}\left(\mathbf{x}_n;\boldsymbol{\theta}^{\mathrm{S}}\right)\right]\right\}.$$

Figure 2: Teacher student loss

$$\mathcal{L}^{\mathrm{T}}\left(\mathcal{B};\boldsymbol{\theta}^{\mathrm{T}}\right)=\frac{1}{|\mathcal{B}|}\sum_{(\mathbf{x}_n,\mathbf{y}_n)\in\mathcal{B}}\left\{-\eta\cdot\mathbf{y}_n^{\top}\ln\mathbf{f}\left(\mathbf{x}_n;\boldsymbol{\theta}^{\mathrm{T}}\right)+\right.$$
$$\left.(1-\eta)\cdot\left[f_{a_1}-\frac{1}{K-1}\sum_{k=2}^{K}f_{a_k}\right]\right\}.$$

Figure 3: Top score difference (TSD

- **Training the patriarch (the first teacher):** The patriarch trains with a simple cross entropy loss function shown in Fig. 1

$$\mathcal{L}(\mathcal{B};\boldsymbol{\theta})=-\frac{1}{|\mathcal{B}|}\sum_{(\mathbf{x}_n,\mathbf{y}_n)\in\mathcal{B}}\mathbf{y}_n^{\top}\ln\mathbf{f}(\mathbf{x}_n;\boldsymbol{\theta}).$$

Figure 1: Cross entropy loss

- **Teacher-student optimization:** This stage can be repeated several times (each stage is called generations). We train a student model called (MS) with help of the student from the previous stage that is now called a teacher (MT). This process is done by the loss function (Furlanello et al. 2018) that appears in Fig. 2. The implantation exits here [2] .

The main contribution of the article is about the first stage, training the patriarch. Instead of training the patriarch with simple cross-entropy loss, they used a different loss that is appropriate for training in generation tasks. They added to the cross-entropy loss an additional part. They pick up a few classes which have been assigned with the highest confidence scores and assume that these classes are more likely to be semantically similar to the target class. They set a fixed integer K which stands for the number of semantically reasonable classes for each image, including the primary class. [1] . Then, the additional part of the loss computes the gap between the confidence scores of the primary class and other $K-1$ classes with the highest scores as shown in Fig. 3. To adapt their method to our data sets we did one change. The change is about change K correlated to the number of classes in the data. If the number of classes is less than 5 we set K to 2, if it is between 6 to 10 we set it to 3, otherwise we set K to 4.

The main advantage of this paper is about optimizing deep networks in generations. The paper proposes a new method of optimizing that leads to better performances in terms of accuracy and error rate with the same architecture. Although, this method increases the training time that is the main disadvantage. The authors did an experiment on two detests are CIFAR10 and ILSVRC2012. It will be interesting to check on larger data if the training time increment is

[2] $https : //keras.io/examples/vision/knowledge_distillation/$

worth the accuracy improvement. In addition, it will be interesting to examine the performance of this method on data where there are fewer classes and each class is different from the other.

1.3 Improvement

The improved algorithm that we suggest is more suitable for small data than standard deep neural networks that can't handle them compared to classic ML algorithms such as Random Forest. To handle this problem we suggest the following algorithm.

Algorithm 1 Improved Algorithm

```
1: $train \leftarrow transform(train)$
2: $transfer\_layer \leftarrow RestNet50$
3: $new\_model \leftarrow build\_model(transfer\_layer)$
4: $teacher \leftarrow new\_model.fit(train)$
5: **for** $i\ in\ range(generations)$ **do**
6:     $student \leftarrow simple\_nn\_model()$
7:     $generation \leftarrow Distillation.fit(student = student, teacher = teacher)$
8:     $teacher \leftarrow generation$
9: **end for**
```

Line 1 is transforming tabular data into images using Tab2Img library[3]; Line 2 loads a pre-trained model, after several experiments we found that RestNet50 leading to bast performances in terms of accuracy and training time; Line 3 is building a deep learning model using a transfer learning that trained on image-net; Line 4 fitting the first teacher (patriarch) with images that represent a tabular data using loss function from Fig. 3. Lines 5 -6 are responsible for teaching in generations. Each generation we teaching a student with the teacher (that is the student from the previous stage) using the loss function from Fig. 2.

We assume that this method will lead to better results due to using pre-trained weights. With small datasets, it is well known that deep learning models can't converge with a few amount of samples. With good initializing weights, we might have fewer samples.

2 Experiments

2.1 Settings

We conducted classification tasks for each of our three subjected algorithms. For the baseline algorithm, we used Random forest, then we test the Tolerant Teacher With five generations algorithm as proposed in [1] and finally we tested the same algorithm with our improvement as described in section 2. We tested each algorithm performance on 20 different data sets that shown in Table 1.

2.2 Hyper-parameter Optimization

For evaluation of the algorithms, we used 10-fold Cross-Validation - to separate between the Train and Test data. The results of each fold for every dataset are presented in [4]. The

[3]https://pypi.org/project/tab2img/

[4]$https://github.com/Maximbrg/Deep-learning---Batter-teacher-batter-performance$

Table 1: Data settings

Dataset Name	Sampels	Features	Classes	Balanced
Abalon	4177	8	3	Yes
Annealing	898	31	5	No
Arrhythmia	452	262	13	No
Audiology-std	196	159	18	No
Autos	205	25	6	No
Balance-Scale	625	4	3	No
Baseball	1340	16	3	No
Car	1728	6	4	No
Cardiotocography-3classes	2126	21	3	No
Cardiotocography-10classes	2126	21	10	No
Conn_Bench-Vowel-Deterding	990	11	11	Yes
Contrac	1473	9	3	Yes
Dermatology	366	34	6	No
Ecoli	336	7	8	No
Energy-y1	768	8	3	Yes
Energy-y2	768	8	3	Yes
Flags	194	28	8	No
Glass	214	9	6	No
Heart-VA	200	12	5	No
Iris	150	4	3	Yes

subjected algorithms have non-learnable hyperparameters, and for every fold, we chose the most promising hyper-parameters using "Random Search" of sk-learn package [5]. For the random forest, we tested: n estimators (number of trees in the forest), max features, max depth, min samples split, min samples leaf, and bootstrapping due to that that they are the most influent features. For the Tolerant Teacher With 5 generations algorithm, we tested: l2 penalty, size of the hidden layers, and the dropout rate for the same reason above. For our improved version, we tested the number of epochs, batch size, and the optimal generation due to the fact that we focused on the challenge of minimizing the training time, in other words, we want to achieve better performances in terms of training time. The performance metrics we measure are Accuracy, TPR, FPR, Precision, AUC – Area Under the ROC Curve, Area under the Precision-Recall, Training time, Inference time for 1000 instances (although the training time was measured under different hardware in different algorithms).

2.3 Results

The results of the experiment appears in Table 2.3 and in Table 3. These are the results of the three algorithms. The base is Random-Forest, then comes the algorithm of the article that we choose to implement, and last is our improved algorithm. Each row represents the average results of the 10-cross validation. The full results of each fold appear in our Git repository [6]. By looking on the tablas 2.3 3 we can notice that our improved algorithm outperforms all other algorithms most of the time except the training time.

[5] $https : //scikit-learn.org/stable/modules/generated/sklearn.model_selection.RandomizedSearchCV.html$

[6] https://github.com/Maximbrg/Deep-learning—Batter-teacher-batter-performance/tree/main

Table 2: Results 1

Dataset Name	Algorithm Name	Accuracy	TPR	TNR	Precision	AUC	PR-Curve	Training Time	Inference Time
abalon	Base	0.664	0.651	0.172	0.654	0.739	0.575	135.852	0.15
abalon	A More Tolerant Teacher With 5 generations	0.697	0.682	0.158	0.687	0.762	0.577	127.983	0.166
abalon	improvement	0.726	0.708	0.145	0.716	0.688	**0.578**	119.614	0.161
annealing	Base	0.759	0.74	0.129	0.749	0.712	0.58	111.776	0.177
annealing	A More Tolerant Teacher With 5 generations	0.79	0.765	0.113	0.78	0.639	0.583	103.439	0.181
annealing	improvement	0.816	0.787	0.101	0.806	0.657	**0.585**	95.241	0.189
arrhythmia	Base	0.837	0.796	0.086	0.837	0.668	0.589	86.883	0.181
arrhythmia	A More Tolerant Teacher With 5 generations	0.868	0.811	0.071	0.868	0.684	0.592	78.5	0.184
arrhythmia	improvement	0.897	0.84	0.057	0.897	0.609	**0.596**	70.099	0.181
audiology-std	Base	0.928	0.87	0.042	0.928	0.535	0.601	61.319	0.167
audiology-std	A More Tolerant Teacher With 5 generations	0.957	0.888	0.029	0.957	0.55	0.605	53.089	0.172
audiology-std	improvement	0.94	0.835	0.03	0.94	0.454	**0.609**	55.854	0.155
autos	Base	0.912	0.801	0.032	0.912	0.454	0.612	58.685	0.148
autos	A More Tolerant Teacher With 5 generations	0.897	0.763	0.035	0.897	0.357	0.615	61.308	0.131
autos	improvement	0.867	0.708	0.037	0.867	0.357	**0.617**	64.649	0.131
balance-scale	Base	0.848	0.682	0.037	0.848	0.265	0.618	67.432	0.119
balance-scale	A More Tolerant Teacher With 5 generations	0.828	0.669	0.04	0.828	0.18	0.619	70.558	0.127
balance-scale	improvement	0.802	0.589	0.043	0.802	0.091	**0.62**	73.553	0.117
baseball	Base	0.783	0.548	0.047	0.783	0.091	0.619	76.32	0.109
baseball	A More Tolerant Teacher With 5 generations	0.763	0.502	0.051	0.763	0.091	0.619	79.139	0.105
baseball	improvement	0.746	0.482	0.052	0.746	0	**0.617**	82.567	0.1
car	Base	0.735	0.44	0.052	0.735	0	0.616	79.49	0.112
car	A More Tolerant Teacher With 5 generations	0.758	0.475	0.048	0.758	0	0.615	76.494	0.127
car	improvement	0.751	0.416	0.045	0.751	0	**0.614**	72.675	0.127
cardiotocography-3clases	Base	0.776	0.382	0.042	0.776	0	0.614	68.414	0.122
cardiotocography-3clases	A More Tolerant Teacher With 5 generations	0.803	0.42	0.037	0.803	0	0.613	64.846	0.123
cardiotocography-3clases	improvement	0.792	0.36	0.037	0.792	0	**0.612**	62.536	0.125
cardiotocography-10clases	Base	0.8	0.36	0.034	0.8	0	0.612	58.651	0.127
cardiotocography-10clases	A More Tolerant Teacher With 5 generations	0.777	0.306	0.031	0.777	0	0.612	55.093	0.125
cardiotocography-10clases	improvement	0.778	0.256	0.028	0.778	0	**0.611**	51.775	0.133

Table 3: Results 2

Dataset Name	Algorithm Name	Accuracy	TPR	TNR	Precision	AUC	PR-Curve	Training Time	Inference Time
conn-bench-vowel-deterding	Base	0.79	0.191	0.026	0.79	0	0.611	47.373	0.133
conn-bench-vowel-deterding	A More Tolerant Teacher With 5 generations	0.791	0.263	0.029	0.791	0	0.61	46.849	0.125
conn-bench-vowel-deterding	improvement	0.787	0.265	0.03	0.787	0	**0.61**	46.548	0.123
contrac	Base	0.779	0.341	0.036	0.779	0	0.61	46.625	0.117
contrac	A More Tolerant Teacher With 5 generations	0.736	0.395	0.049	0.736	0	0.609	46.807	0.116
contrac	improvement	0.717	0.365	0.054	0.717	0	**0.609**	46.731	0.112
dermatology	Base	0.717	0.413	0.058	0.717	0	0.609	45.461	0.105
dermatology	A More Tolerant Teacher With 5 generations	0.713	0.479	0.061	0.713	0	0.609	45.406	0.097
dermatology	improvement	0.746	0.545	0.06	0.746	0.082	**0.608**	45.607	0.109
ecoli	Base	0.747	0.625	0.063	0.747	0.082	0.608	45.413	0.105
ecoli	A More Tolerant Teacher With 5 generations	0.747	0.719	0.064	0.747	0.082	0.609	45.399	0.097
ecoli	improvement	0.762	0.709	0.066	0.762	0.159	**0.609**	45.564	0.098
energy-y1	Base	0.77	0.682	0.067	0.77	0.239	0.609	45.497	0.091
energy-y1	A More Tolerant Teacher With 5 generations	0.792	0.672	0.065	0.792	0.319	0.61	45.451	0.098
energy-y1	improvement	0.835	0.682	0.057	0.835	0.399	**0.611**	45.357	0.094
energy-y2	Base	0.843	0.678	0.059	0.843	0.478	0.612	45.463	0.098
energy-y2	A More Tolerant Teacher With 5 generations	0.873	0.694	0.053	0.873	0.558	0.614	45.435	0.102
energy-y2	improvement	0.886	0.692	0.054	0.886	0.636	**0.616**	45.719	0.114
flags	Base	0.879	0.685	0.064	0.879	0.627	0.618	45.577	0.102
flags	A More Tolerant Teacher With 5 generations	0.889	0.669	0.065	0.889	0.706	0.62	45.773	0.116
flags	improvement	0.883	0.638	0.072	0.883	0.783	**0.622**	45.82	0.111
glass	Base	0.89	0.64	0.081	0.89	0.78	0.624	48.8	0.106
glass	A More Tolerant Teacher With 5 generations	0.893	0.658	0.082	0.893	0.788	0.627	51.907	0.108
glass	improvement	0.898	0.649	0.092	0.898	0.779	**0.63**	55.408	0.127
heart-va	Base	0.903	0.657	0.09	0.903	0.783	0.633	58.487	0.127
heart-va	A More Tolerant Teacher With 5 generations	0.905	0.643	0.099	0.905	0.772	0.637	61.43	0.125
heart-va	improvement	0.905	0.658	0.109	0.905	0.774	**0.64**	64.558	0.128
iris	Base	0.907	0.647	0.123	0.907	0.762	0.644	67.599	0.12
iris	A More Tolerant Teacher With 5 generations	0.925	0.633	0.134	0.925	0.749	0.648	71.034	0.136
iris	improvement	0.926	0.625	0.146	0.926	0.739	**0.652**	73.988	0.123

Table 4: Nemenyi-Friedman post-hoc test

	Base	**Teacher-student**	**Improved**
Base	1	0.51	0.001
Teacher-student	0.51	1	0.03
Improved	0.001	0.03	1

2.4 Statistical significance testing

To test if there are statistically significant differences between we used the "Friedman" test on the metric AUC-PR with a significant level of $\alpha = 0.05$. Our null hypothesis (H_0) is that all the algorithms are having equal AUC-PR scores on the AUC-PR metric. The results of the test are the statistical value of 30.63 and a p-value of 0.044 which is lower than 0.05 and in that case, we reject the null hypothesis and conclude that there is at least one algorithm that outperforms the others. using nemenyi-Friedman post-hoc test with the following parameters : L=3 , N=20, a = 0.05 $CD = q_\alpha \cdot \sqrt{\frac{L\cdot(L+1)}{6N}}$ we get the statistic result of 0.74 shown in Table 4.

2.5 Discussion

As shown in Table 4, the tolerant teacher-student algorithm as described in the paper [1] achieves statistically significant better results than random forest. Our improved algorithm works even better than the other two algorithms. We notice that in the smaller datasets e.g autos and flags random forest achieves better results but not in a significant matter. We can explain these results with a simple conjuncture that for small data sets, deep learning algorithms have the struggle to find optimal learnable parameters contrary to algorithms such as a Random Forest. Our improved version uses Resnet as a transfer layer, and it causes our model to be more generative and less vulnerable to unseen data. We have shown that on small datasets the tolerant teacher having difficulty converging. From the accuracy metric, we infer that all the three algorithms achieve mostly similar results and we associate this to our mostly unbalanced datasets.

3 Conclusion

In this assignment we were asked to implement a deep learning algorithm and compare it with a well known baseline algorithm. We chose to implement the algorithm from the paper [1]. As for the baseline algorithm we used the famous random forest algorithm. We suggested an improvement to to the algorithm and used the tabular data as image and we combine the teacher - student algorithm with the use of transfer learning and got promising results. For conclusion we successfully proved that the Teacher - Student algorithm is statistically significant stronger by getting pr-curve scores than random forest algorithm and our own improvement is even better than it predecessor.

References

[1] C. Yang, L. Xie, S. Qiao, and A. L. Yuille. Training deep neural networks in generations: A more tolerant teacher educates better students. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 5628–5635, 2019.