Jagiellonian University

Faculty of Mathematics and Computer Science

Michał Filek

Album nr: 1157814

Text classification using the semi-supervised methods

Master Thesis
Field of study: Computer Science

Supervisor:
Dr Marek Śmieja
Institute of Computer Science and Computational Mathematics

Cracow 2020

Author's statement

Being aware of legal responsibility, I declare that this diploma thesis was written by myself and does not contain content obtained in a manner inconsistent with applicable regulations.

I also certify that the presented thesis has not been subject to procedures related to obtaining a professional title at a university..

....

Cracow, date Author's signature

Thesis supervisor's statement

I confirm that this work has been prepared under my supervision and is eligible to be presented in the procedure for granting the professional title.

....

Cracow, date Thesis supervisor's signature

Abstract

The work is devoted to the issue of **s**emi-**s**upervised **l**earning for the tasks of text classification. One of the aims is to test the semi-supervised learning algorithm called FixMatch in the NLP domain. This method is considered to be state-of-the-art for problems of image classification with very few training labeled examples. In order to test FixMatch for text classification, it is necessary to create realistic augmentations, which will be proposed in the following work. In order to evaluate the FixMatch properly, two other popular SSL methods (VAT, Pseudo-Label) and supervised model, will be trained as comparative baselines. Another goal of the work is to check how the result on the test set are affected by training process and selection of hyperparameters using different ratio of the training and validation set sizes. The last goal will be to examine the quality of the chosen hyperparameters on small validation sets, also by checking the results of the SLL methods trianing them on more labelled and unlabeled data.

Results of conducted experiments for the FixMatch algorithm indicate that it was not possible to transfer state-of-the-art scores from the computer vision domain into the NLP domain. After observing the results of successful experiments with different SSL methods, a conclusion appears that the number of examples in the validation set should be slightly less or equal to the number of examples in the training set. For tested SSL methods, the error on the test set is decreasing with training on an additional number of labeled data, while in the case of unlabeled data, there is no such relation.

Contents

1 Introduction

Nowadays, the field of machine learning has been undergoing an unprecedented development and finds numerous applications in solving problems of science and business. Particularly noteworthy is the **D**eep **L**earning, which is an sub-discipline of machine learning. DL algorithms have basically become the basis for computer vision applications. This success is related to the complexity of patterns that these algorithms can match, the scaling of effectiveness in solving problems together with the increase in the amount of training data and the possibility of omit the extraction of features in the data preparation process. The most popular DL algorithms are deep artificial neural networks, which are obtaining the best results when trained in a supervised manner. Supervised learning requires a large amount of labeled data, which often involves high costs and requires, for example, the work of data taggers and/or the employment of an expert in the field of particular domain. Very often in commercial machine learning systems there is a considerable amount of unlabelled data, which is not used in any way. In order to improve performances of the systems, a number of semi-supervised learning methods have been developed, which make it possible to use unlabeled data. In recent years consistency training methods have become very popular. They relies on using different types of transformations on the unlabeled data set to increase the effectiveness of the trained models. One of the methods using consistency training is the FixMatch algorithm, which with a very small amount of labeled data and data augmentations allows to train an effective model. This algorithm was used in computer vision problems and obtained state-of-the-art results. In this work, there will be a trial of testing this algorithm in the NLP domain. Another important issue in the case of using SSL methods, is to get knowledge about the minimum amount of labeled data necessary for proper training of the algorithm and the estimation of the hyperparameters required by algorithm. It is essential that the verification of above question is carried out in a realistic way, i.e. with a small amount of validation data.

In the first part of this work, basic machine learning concepts for supervised and semi-supervised learning will be explained. Apart from the FixMatch method, two other popular algorithms will be discussed: Pseudo-Label and VAT. In the second part, experiments with the SSL methods mentioned above, will be conducted, with a goal of checking the effectiveness of the FixMatch algorithm in the NLP domain and examing ratio between training and validation datasets.

2 Machine Learning basic concepts

The field of machine learning is concerned with the automatic discovery of regularities in data through the use of computer algorithms. Machine learning systems can tackle problems involving knowledge of the real world and make predictions based on new data. With better algorithms and more data, these systems can make better decisions and solve more difficult problems. There is a question - but what exactly mean that algorithm is learning?

"A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E." [1]

For machine learning algorithms experience E in above sentence mean - the data which are a collection of features that have been quantitatively measured from some object or event. Those features are typically represented as a vector $\boldsymbol{x} \in \mathbb{R}^n$, and $i \in \{1, \ldots, n\}$ where each value x_i is another feature.

2.1 Machine Learning Tasks

Generally, there have been two essentially distinct categories of tasks in machine learning:

2.1.1 Unsupervised Learning

Unsupervised learning try to solve a group of machine learning tasks that are looking for previously undetected patterns in a data set with no pre-existing labels and with a minimum of human guidance. Let $\mathbb{X} := \{x_0, x_1, \ldots, x_n\}$ be a collection of n learning examples, where $\mathrm{x}_i \in p_{\mathrm{data}}$ for all i $\in [n] := \{0, 1, \ldots, n\}$. Commonly it is presumed that the examples are sampled

independently and **i**dentically **d**istributed (i.i.d.) from a common data generating distribution p_{data}. There might be many goals based on setting mention above. One will be clustering which mean discovering groups of similar examples within the data [2]. Another might be to determine the probability density function if x is continuous [3] or a probability mass function if x is discrete [4] on the space that the examples were drawn from. Unsupervised learning can also be used to project the data from a high-dimensional space down to two or three dimensions for the purpose of visualization [5] [6]. [7]

2.1.2 Supervised Learning

The goal of supervised learning is to learn a mapping from $\mathbb{X}$ to $\mathbb{Y}$, taking into account the training set of pairs (x_i, y_i). $y_i \in \mathbb{Y}$ are called labels or targets of the x_i. One more time, as with unsupervised learning, the standard requirement for pairs (x_i, y_i) is that they are sampled i.i.d. from some data generating distribution p_{data} which here ranges over joint distribution $p(X, Y)$. If the requested output consists of one or more continuous variables, then such a task is called regression and learning algorithm is asked to derive the function $f : \mathbb{R}^n \to \mathbb{R}$. The case where the goal of learning algorithm is to produce a function that assigns each input to one of a finite number of discrete categories $f : \mathbb{R}^n \to \{0, 1, \ldots, n\}$, is called classification. The optimal scenario will allow the algorithm to correctly determine the vector labels for a set of previously unseen examples called test data. This requires a learning algorithm to have an ability to generalize knowledge from training to testing examples. [8] [9] [10]

Generative Models The generative models are a statistical models that can capture the common distribution $P(X, Y)P(X)$, use it to compute the conditional probability $P(Y|X = x)$ and for instance, perform classification tasks based on calculated distributions. Furthermore these models may either explicitly or implicitly model the input and output distribution and tell how likely is a given pair (x_i, y_i). Usually, because of sampling ability, it is possible for them to generate synthetic data points from the input space.

Discriminative Models Unlike generative models, discriminative models do not attempt to estimate how x_i were generated. Instead of this they

simply tells how likely the label might be applied to the example, so they evaluate $p(y|x)$ directly.

In the next sections the emphasis will be on supervised learning, discriminative models and classification tasks due to topic of this thesis.

2.2 Loss Function

As it was previously stated the goal of classification algorithm is to produce model $M(\mathbb{Y} \mid \mathbb{X};\ \theta)$ based on training data $\mathbb{X}_{train}$ and its labels $\mathbb{Y}_{train}$ that will predict $\mathbb{Y}_{test}$ for new, previously unseen test examples $\mathbb{X}_{test}$. Therefore the model has to solve decision problem which depends on choosing an label $y_{(n)}$ that for classification tasks belong to finite set of $\{0, 1, \ldots, n\}$ categories. To begin the process of learning, there must be made a choice of loss function, $\mathrm{J}(\theta,\ (\mathrm{y}_{(i)},\ x_{(i)})))$, which measures how compatible model's actions are with true targets values. Therefore the task is to estimate an unknown model's parameters θ on the basis of observations $\mathbb{X}_{train}$. To solve this problem it will be usefull to use likelihood function $p_m(\mathbb{Y} \mid \mathbb{X};\ \theta)$ as the conditional probability of $\mathbb{X}$ given $\mathbb{Y}$ and with the model weights which might be consider as hypothesis θ.[7]

2.2.1 MAP

One common way for creating loss function using likelihood (θ will be perceived as a random variable) is to apply Bayes theorem to compute $p(\mathbb{X} \mid \theta)$. After some mathematical transformations [7], equation of **M**aximum **A** **P**osterori emerge :

$$\hat{\theta}^{MAP} = \underset{\theta}{\operatorname{argmax}}\ \mathrm{p}(\mathbb{Y} \mid\ \mathbb{X}; \theta)p(\theta) \tag{1}$$

2.2.2 MLE

In the machine learning literature, a common used loss functions are those based on **M**aximum **L**ikelihood **E**stimation. MLE is considered as the negative log of the MAP function which is created by observation that likelihood term depends essentially on number of samples in $\mathbb{X}$. While there is more and more data, the prior stays constant and the MAP estimate converges towards MLE with assumption that $p(\theta)$ is uniform. Because the

negative logarithm is a monotonically decreasing function, maximizing the likelihood is equivalent to minimizing the error. [7]

$$\hat{\theta}^{MLE} = \arg\min_{\theta} \; -\log \mathrm{p}(\mathbb{Y} \mid \mathbb{X};\; \theta) \tag{2}$$

2.3 Model performence metrices

What exactly mean that model $f(\boldsymbol{x}; \boldsymbol{\theta}_1)$ will be better than $f(\boldsymbol{x}; \boldsymbol{\theta}_2)$?

The most intuitive approach will be to measure what was a sum of true positive (i.e $\boldsymbol{y}_i == 1 \wedge \boldsymbol{y}_i == f(\boldsymbol{x}_i; \boldsymbol{\theta})$) and true negative (i.e $\boldsymbol{y}_i == 0 \wedge \boldsymbol{y}_i == f(\boldsymbol{x}_i; \boldsymbol{\theta})$) classified examples to all predictions including its wrong classified counterparts i.e false negative and false positive examples.

Such defined metric is called accuracy (3) and is a common choice when dealing with balanced datasets (same amount of examples per every distinct label). Another metric which is popular in evaluatiog model performence is error rate (4).

$$accuracy = \frac{True\ Positive + True\ Negative}{True\ Positive + True\ Negative + False\ Positive + False\ Negative} \tag{3}$$

$$error\ rate = 1 - accuracy \tag{4}$$

$$precision = \frac{True\ Positive}{True\ Positive + False\ Positive} \tag{5}$$

$$recall = \frac{True\ Positive}{True\ Positive + False\ Negative} \tag{6}$$

$$F1 = \frac{2 \cdot precision \cdot recall}{precision + recall} \tag{7}$$

If datasets is inbalanced it is reasonable to use other common metrics such as precision (5), recall (6) or harmonic mean of those two - called F1 (7).

2.4 Overfitting, Underfitting, capacity

General drawback of choosing maximum likelihood estimator for loss function is an existence of overfitting which relies on overconfidence of predicted distribution. The simple procedure of training machine learning algorithm is to sample the training set, then use it to choose the model's parameters via reducing training set error and then, sample the test set. Under this procedure, the expected test error should be greater than or equal to the expected value of training error.

To make an evaluation of how well algorithm will perform on new unseen data its crucial to:

1. Decrease training error.
2. Decrease the gap between training and test error.

If executing one of these two steps fail we can observed underfitting and overfitting respectively. Underfitting refers to a model that can neither model the training data nor generalize to new data so its train error is too high. Overfitting occurs when the gap between the training error and test error is too large. We can alter model's behaviour by changing it's complexity also know as capacity. Model's capacity is its capability to fit a wide variety of functions. Models with low capacity can have difficulties to fit the training set. Models with high capacity can fit every minor variation in the input, since this is more likely to be noise than true signal and that can lead to high test error. [10] [7] [9]

2.5 Regularization

Regularization allows models with high capacity to be trained on relatively small datasets without significant overfitting, essentially by preventing model from becoming too complex.

2.5.1 L2

One technique that is often used to control the overfitting behaviour involves adding a penalty term to the error function in order to prevent the coefficients from reaching large values. This extra term is known as L2 -norm or Tikhonov

regularization and leads to a modification of error function

$$J(\theta) = MSE_{train} + \lambda\theta^{\top}\theta \tag{8}$$

In equation (8) adjustment of hyperparameter λ will affect the quantative importance of the regularization term compared with the MSE term and will control the effective complexity of the model and hence determines the degree of overfitting. In the neural networks literature, this is called weight decay, since it encourages small weights, and hence simpler models. [11] [10]

2.5.2 Early stopping

When training deep networks with sufficient representational capacity to overfit the task, there is popular observation that training error decreases constantly over time, but validation error begins to rise again. Simple method to prevent this is called early stopping, which means stopping the training procedure when the error on the validation set starts to increase. This procedure leads to obtaining a model with better validation set error by returning to the parameter setting at the point in time with the lowest validation error. Every time the error on the validation set decrease, a copy of the model parameters is saved. When the training algorithm terminates, the saved parameters are loaded, rather than the latest parameters. The training terminates when no parameters have improved over the best recorded validation error for some arbitrarily chosen number of iterations. This strategy is known as early stopping. It is probably the most commonly used form of regularization in deep learning. Its popularity is due to both its effectiveness and its simplicity. [10]

2.5.3 Dropout

Dropout can be thought of as a method of making bagging practical for ensembles of very many large neural networks. Bagging involves training multiple models and evaluating multiple models on each test example. This seems impractical when each model is a large neural network, since training and evaluating such networks is costly in terms of runtime and memory. Dropout provides an inexpensive approximation to training and evaluating a bagged ensemble of exponentially many neural networks. Specifically, dropout trains the ensemble consisting of all subnetworks that can be formed

by removing none output units from an underlying base network. In most modern neural networks, based on a series of affine transformations and nonlinearities, we can effectively remove a unit from a network by multiplying its output value by zero. [12] [10]

2.5.4 Data Augmentation

There is common observation that for a given model capacity, the overfitting behaviour become less significant as the size of the data set increases. Another way to say this is that the larger the data set, the more complex must the model to fit into the data. Therefore many modern implicit regularization techniques have an impressive results with models that have orders of magnitude more parameters than training examples and control overfitting with the help of data augmentation techniques [13] [14], [15].

3 Semi-supervised Learning

Machine learning systems often achieve their strong performance through supervised learning that requires a labeled dataset. Benefit conferred by the use of a larger datasets can therefore come at a significant cost since labeling data often requires a lot of human labor. This cost can be particularly extreme when labeling must be done by an expert (for example, a doctor in medical applications).

Semi-supervised learning (SSL) is halfway between supervised and unsupervised learning. In addition to unlabeled data, the algorithm is provided with some supervision information – but not necessarily for all examples. The traning dataset of for SSL methods $\mathbb{X} := \{x_0, x_1, \ldots, x_n\}$ can be divided into two parts: the samples $\mathbb{X}_l := \{x_0, x_1, \ldots, x_l\}$, for which labels $\mathbb{Y} := \{y_0, y_1, \ldots, y_l\}$ are given, and the samples $\mathbb{X}_{unl} := \{x_{l+1}, x_{l+2}, \ldots, x_{l+n}\}$ without superivsed information i.e without targets. Since unlabeled data can often be obtained with minimal human labor, any performance boost gained by SSL algorithms often comes with low cost.

Semi-supervised learning methods shows that in comparison with a supervised algorithms they can achieve a more accurate prediction by taking into account the unlabeled points [16]. However, there is an important prerequisite: the distribution of examples, which the unlabeled data will

help explain must be relevant for the classification problem. In a more mathematical formulation, it could be said that the knowledge on $p(x)$ that is gained through the unlabeled data has to carry information that is useful in the inference of $p(y|x)$. If this is not the case, semi-supervised learning methods will not yield an improvement over supervised learning. It might even happen that using the unlabeled data degrades the prediction accuracy by misguiding the inference [8]. Semi-supervised learning, like other machine learning algorithms, needs assumptions on which the knowledge generalization mechanism will be based. [8]

3.1 Semi-supervised learning assumptions

3.1.1 Smoothness assumption

"If two points x_1, x_2 in a high-density region are close, then so should be the corresponding outputs y_1, y_2."

This assumption also implies that if two points are linked by a path of high density (e.g., if they belong to the same cluster), then their outputs are likely to be close. If, on the other hand, they are separated by a low-density region, then their outputs needn't be close. [8]

3.1.2 Cluster assumption

"If points are in the same cluster, they are likely to be of the same class"

Assume that the points of each class tended to form a cluster. Then the unlabeled data might be helpful in finding the boundary of each cluster more accurately: one could run a clustering algorithm and use the labeled points to assign a class to each cluster. That is in fact one of the earliest forms of semi-supervised learning [8]. The cluster assumption can easily be seen as a special case of the above semi-supervised smoothness assumption. Considering that clusters are frequently defined as being sets of points that can be connected by short curves which traverse only high-density regions. [8]

3.1.3 Manifold assumption

"The (high-dimensional) data lie (roughly) on a low-dimensional manifold."

It may be unclear how, at first glance, such an assumption might be helpful in designing a semi-supervised learning algorithm. A well-known problem of many statistical methods and machine learning algorithms is the so-called curse of dimensionality. It is problem related to the fact that volume grows exponentially with the number of dimensions, and an exponentially growing number of examples is required for statistical tasks such as the reliable estimation of densities. This is a problem that directly affects generative approaches that are based on density estimates in input space. A related problem of high dimensions, which may be more severe for discriminative methods, is that pairwise distances tend to become more similar, and thus less expressive. If the data happen to lie on a low-dimensional manifold, however, then the learning algorithm can essentially operate in a space of corresponding dimension, thus avoiding the curse of dimensionality. Making projection from high-dimensional to low-dimensional manifold as an approximation of the high-density regions, then the semi-supervised smoothness assumption might be applied on the manifold. [8]

3.2 SSL Approaches

Using semi-supervised assumptions for creating new algorithms results in many different approaches for training models on a large sizes of data without requiring a large amount of labels. In this work, particular emphasis will be placed on methods related to deep neural networks.

3.2.1 Entropy Minimization

Entropy Minimization approach is based on previously discussed assumption that in many semi-supervised learning methods the classifier's decision boundary should not pass through high-density regions of the marginal data distribution (smoothing assumption). Unlabeled data should be used to ensure that classes are well-separated. This can be achieved by encouraging the model's output distribution to have low entropy (i.e., to make "high-confidence" predictions) on unlabeled data. This is done explicitly in [17] with a loss term which minimizes the entropy of model $p_m(\mathbb{Y} \mid \mathbb{X};\ \theta)$ which task is to classified distinct labels from set $\mathbb{K} := \{y_0, \dots y_K\}$ using training dataset $\mathbb{X}$ of size N.

$$J^{EntMin}(\mathbb{Y}, \mathbb{X}) = MLE_{train} + \lambda \sum_{n=1}^{N} \sum_{k=1}^{K} p_m(y_k \mid x_i) \log p_m(y_k \mid x_i) \quad (9)$$

θ model parameter was skipped to simplified above equation .

Pseudo-Label [18] does entropy minimization implicitly by constructing "hard labels" (i.e one-hot encoded) on unlabeled data and using them as training targets in a standard cross-entropy loss. Pseudo-labeling leverages the idea that the model itself obtain artificial labels for unlabeled data. There is an underlaying assumption that $argmax$ function applied to a probability distribution $q_b = p_m(y_i \mid x_i)$ produces a valid "one-hot" probability distribution. The use of a hard label makes pseudo-labeling closely related to entropy minimization , where the model's predictions are encouraged to be low-entropy (i.e. high-confidence) on unlabeled data.

$$\mathrm{J}^{PS}(\mathbb{X}_{unl}, \mathbb{Y}, \mathbb{X}_l) = MLE_{train}(\mathbb{Y}, \mathbb{X}_l) + \lambda \sum_{n=1}^{N} H(argmax(q_b);\ q_b)) \quad (10)$$

Usually linear scheduler is used to generate λ coefficient which will be a scaling factor of unlabeled loss function term. Such scheduler is parametrized by hyperparameters:

- λ_{Max} - maxium value of λ
- T_1 - the number of step in which scheduler start to work
- T_2 - the number of steps until achieving λ_{Max}

3.2.2 Label Propagation

Various graph-based algorithms for semi-supervised rely on the idea of building a graph whose nodes are data points (labeled and unlabeled ones) and edges represent similarities between points. Known labels are used to propagate information through the graph in order to label all nodes. Given the graph $\mathcal{G}$, a simple idea for semi-supervised learning label propagation is to propagate labels on the graph. Starting with nodes $1, 2, \ldots, l$ labeled with their known label (tagged as e.g 1) and nodes $l_{+1}, l_{+2}, \ldots, n$ labeled with 0. Each node starts to propagate its label to its neighbors, and the process is repeated until convergence. In the experimental part of this thesis none of label propagation methods will be used, so that approach won't be further discussed. [8]

3.2.3 Consistency Training

Consistency training framework was first proposed in [19] and [20]. In a nutshell, this SSL method simply regularize model predictions to be invariant

to noise applied to either input examples [21] [22] or hidden states [23]. This procedures make sense intuitively because a good model should be robust to any small change in an input example or hidden states. Under this framework, methods differ mostly in how and where the noise injection is applied. Typical noise injection methods are additive Gaussian noise, dropout noise or adversarial noise.

$$\mathrm{J}(\mathrm{Y},\, \mathrm{X}_l, \mathbb{X}_{unl}) = MLE_{train} + \lambda \sum_{n=1}^{N} p_m(y \mid \alpha(x_i)) \log p_m(y \mid \alpha(x_i)) \quad (11)$$

Model is trained both via a standard supervised classification loss on train samples $\mathbb{X}_l$ with labels $\mathbb{Y}$ and with special unsupervised loss term with unlabeled data $\mathbb{X}_{unl}$ of size N. Note that α and p_m are stochastic transformations, so the two terms in (16) are not identical. Extensions to this idea include using an adversarial perturbations in place of α. [24]

Virtual adversarial training [21] Let D[p, p'] be a non-negative function that measures the divergence between two distributions p and p', e.g. Kullback–Leibler divergence or cross-entropy. Model's parametrized output conditional distribution will be denoted as $p_m(y \mid x;\, \theta)$.

$$LDS(x, \theta) = D[p_m(y \mid x, \hat{\theta}), p_m(y \mid x + r_{vadv}, \theta)] \quad (12)$$

Virtual adversarial training is consistency regularization method based partially on virtual adversarial loss LDS: a measure of local smoothness of the conditional label distribution given input (12). It use the virtual adversarial direction, i.e a vector r_{vadv}, that can most greatly deviate the current inferred output distribution from the status quo.

$$r_{vadv} = \nabla D[p(y \mid x, \hat{\theta}), p(y \mid x + r, \theta)], \quad ||x|| \leq \epsilon \quad (13)$$

To compute r_{vadv} without label information it is necessary to first generate "virtual" labels from $p(y \mid x;\, \hat{\theta})$ (i.e model output distribution at a specific iteration step during training) and use them in place of labels that are unknown to the user. Then it is possible, for neural networks, using random generated vector r (with norm smaller or equal to ϵ), to efficiently compute r_{vadv} (13) using backpropagation algorithm [10].

Thus the full objective function is given by:

$$\mathrm{J}^{VAT}(\mathbb{Y}, \mathbb{X}_l, \mathbb{X}_{unl}) = MLE_{train}(\mathbb{X}_l, \mathbb{Y}) + \lambda LDS(\mathbb{X}_{unl}, \theta) \quad (14)$$

One notable advantage of VAT is that there are just two scalar-valued hyperparameters:

1. the norm constrainted by $\epsilon > 0$ for the adversarial direction
2. the regularization coefficient $\lambda > 0$ that controls the relative balance between the supervised loss term and the LDS.

Consistency Training with data augmentation In recent works [16], [25], [22] $\alpha(x)$ perturbations from equation (16) are considered as data augmentations, which are well know methods of regularization in supervised learning. Data augmentation tend to create novel and realistic-looking training data by applying a transformation to an example, without changing its label. Formally, let $\hat{x_i} = \alpha(x_i)$ be the augmentation transformation from which it is possible to draw augmented examples $\hat{x_i}$ based on an original example x_i with label y_i. For an augmentation transformation to be valid, it is required that any example $\hat{x_i}$ drawn from the distribution shares the same ground-truth label as y_i . Given a valid augmentation transformation, the negative log-likelihood can be simply minimize. Thus, the design of the augmentation transformation has become critical. As it was previously mentioned, consistency training is an important component of many recent state-of-the-art SSL algorithms e.g Fixmatch.

FixMatch Fix-Match produces artificial labels using both consistency regularization and pseudo-labeling. The artificial label is produced based on a weakly-augmented unlabeled image (e.g., using only flip-and-shift data augmentation) which is used as a target when the model is fed with a strongly-augmented version of the same image. Augmentations used in FixMatch for strong augmentation are inspired by [16] and [22] including methods such as CutOut [15] and RandAugment [14].Those methods all produce heavily distorted versions of a given original image. Cutout is a regularization technique of randomly masking out square regions of input during training.RandAugment relies on randomly selecting transformations from a given collection (e.g., color inversion, translation, contrast adjustment, etc.) for each sample in a minibatch. In FixMatch an artificial label is retain if the model assigns a high probability to one of the possible classes.

Let $\mathbb{B} = (x_b; y_b) : b \in (1, 2 \ldots; B_s)$ be a batch of L labeled examples, where x_l are the training examples and y_l are one-hot labels. Let $\mathbb{U} = x_u$:

$u \in (1; \ldots; U_s)$ be a batch of U_s unlabeled examples. Let $p_m(y \mid x;\ \theta)$ be the predicted class distribution produced by the model for input x with weights θ. The cross-entropy between two probability distributions p and p' is denotes as $H(p;\ p')$. Two types of augmentations are performed: strong and weak, denoted by $A(\cdot)$ and $\alpha(\cdot)$ respectively.

$$\mathrm{J}_{unl}(\mathbb{U}) = \sum_{n=1}^{U_s} 1(max(q_b) \geq \tau) H(\hat{q_b}; p_m(y_i \mid A(u_i))) \tag{15}$$

For unlabeled data, FixMatch computes an artificial label for each example and then used in a standard cross entropy loss. To obtain an artificial label, a weakly augmented version of a given unlabeled image: $q_b = p(y \mid \alpha(x);\ \theta)$ is computed and the model predicts it's class distribution.Then $\hat{q_b} = argmax(q_b)$ is used in the cross-entropy loss against the model's output for a strongly-augmented version of x_b. A scalar hyperparameter τ is denoting the threshold above which a pseudo-label is retained and loss is computed. In sum, the loss minimized by FixMatch is:

$$\mathrm{J}^{FM}(\mathbb{B}, \mathbb{U}) = MLE_{train}(\mathbb{B}) + \lambda J_{unl}(\mathbb{U}) \tag{16}$$

where λ is a fixed scalar hyperparameter denoting the relative weight of the unlabeled loss.

It is worth to note that Fixmatch algorithm is similar to the pseudo-label one. The crucial difference is that the artificial label is computed based on a weakly-augmented image and the loss is enforced against the model's output for a strongly augmented image. This introduces a form of consistency regularization which, is crucial to FixMatch's success. As training progresses, the model's predictions become more confident and it is more frequently the case that $argmax(q_b) > \tau$. [24]

4 Experiments

The SSL methods used in experiments in the NLP domain are : Pseudo-Label, Vat and FixMatch. The results from each method will be compared to the supervised model, trained, validated on the same amount of training and validation data. The implementation has been written in the Pytorch and in the Pytorch Lightning framework. Code fragments were used from [26], [27] All experiments were conducted in one code base [28].

4.1 Testing implementation on MNIST

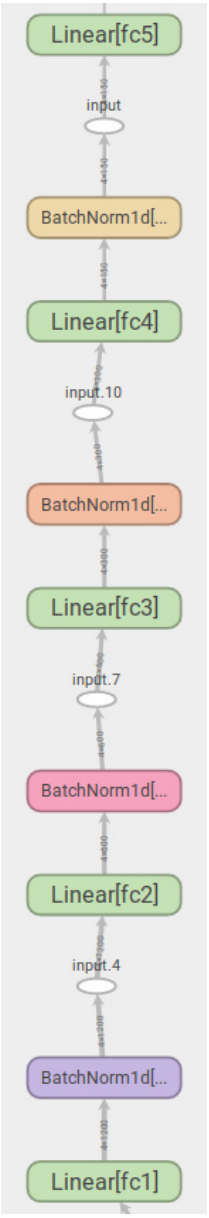

Figure 1: Feed Forward Network Architecture with BatchNorm layers, visualized in TensorBoard.

In order to test the correct implementation of used SSL algorithms, a series of experiments will be performed on the MNIST data set. [29] Originally, the MNIST consists of a training set containing 60000 examples and 10000 examples from the test set. In both cases, the number of examples for each class is equal.
As an artificial neural network architecture will be used a simple Feed Forward Network from figure 1.
For the semi-supervised problem, the original dataset will be divided into labeled and unlabeled data. On the labelled set, another division into training and validation data will be made.
The images will be standardized with regard to the average and the variance of the whole set, i.e. successively:

- mean: 0.1307
- standard deviation: 0.3081

Cross entropy will be used as a cost function. The FixMatch method uses the RandAugment [14] with the appropriate amount of transformations marked as N_{weak} and their strength M_{weak} and similarly for strong augmentations. Additionally, CutOut [15] was used for strong augmentation.
The notation that will be used in the result tables:

- V - validation dataset size
- X_l - training labeled dataset size
- X_{unl} - training unlabeled dataset size
- T - test dataset size
- B - labeled batch size
- U - unlabeled batch size

The results, together with the selected hyperparameters, are placed in the table 1. It may be observed that FixMatch, VAT and pseudolabeeling were the best models respectively, which is consistent with the results of these algorithms in other [30], [24] works and may be an argument for a successful implementation of these methods.

	Supervised	**Pseudo-Label**	**VAT**	**FixMatch**
Hyperparameters:				
X_l	96	96	96	96
T	10000	10000	10000	10000
V	960	960	960	960
B	32	32	64	32
U	-	32	256	224
Max steps	10000	10000	10000	6000
Initial lr	1e-05	1e-05	2e-05	3e-2
Optimizer	Adam	Adam	Adam	SGD
Momentum	-	-	-	0.9
Weight decay	-	-	-	0.0005
Lr scheduler	-	-	StepLR	LinearWarmupLR
Decay γ	-	-	0.5	-
Initial decay step	-	-	2000	-
Decay step size	-	-	2000	-
Warmup steps	-	-	-	1000
T_1	-	1	-	-
T_2	-	2000	-	-
Max λ	-	1	-	-
λ	-	-	1	-
ε	-	-	3	-
K	-	-	1	-
ξ	-	-	1e-06	-
τ	-	-	-	0.999
N weak	-	-	-	1
M weak	-	-	-	2
N strong	-	-	-	3
M strong	-	-	-	5
Val. Error rate	26.6	24.6	23.6	22.5
Test Error rate	24.1	21.4	20.4	18.1

Table 1: Best validation error rates and test error rates for MNIST obtained by Supervised, Pseudo-Label, VAT and FixMatch models with chosen and tuned hyperparameters.

4.2 Experiments in NLP domain

4.2.1 Components of compared SLL algorithms

Validated machine learning systems will consist of the following elements:

- Processed datasets:
  - IMDB [31]
  - MR [32]
- Representation of data:
  - the FastText algorithm was used to represent texts in the form of **300** dimensional vectors. [33]
- Model Architecture:
  - the chosen architecture of the artificial neural network will be the convolution model proposed in [34]. The choice of this architecture is associated with a relatively short processing time of training data, which allows to conduct many experiments.
  - hyperparameters associated with the architecture of this model, except for the dropout value, have been chosen arbitrarily and will not change during the process of estimating other algorithm hyperparameters. Their values are:
    - number of filters - 32
    - filter sizes - 1, 2, 3, 5
- SSL algorithms:
  - the same implementations of SSL methods that were tested during the experiments on the MNIST data set will be used:
    - Pseudo-Label
    - VAT
    - FixMatch

One of the goals of the work is also to see how the FixMatch method works in the NLP domain. The following text augmentation methods were used to achieve this:

1. Weak type augmentations:
   - substitute character by keyboard distance

- insert character randomly
- substitute character randomly
- delete char randomly
- swap character randomly
- insert word by tf-idf similarity
- split word into two tokens randomly
- swap word randomly
- substitute word by it's antonym
- substitute word by spelling mistake from special words dictionary

2. Strong type augmentation:
   - back translation i.e translation of the text from English into French and back into English, using the package Tensor2Tensor [35]

4.2.2 Experiments methodology

The experiments were carried out according to some recommendations from [30]:

1. the same type and models's architecture and one code base for all evaluated SSL algorithms
2. realistically small validation and training datasets used to select hyperparameters
3. validation of SSL algorithms with different amount of labeled and unlabeled data
4. comparison of SSL algorithms with models trained only on labeled data with carefully tuned hyperparameters

In the process of hyperparameters tuning, **200** samples from labeled sets and **400** samples from unlabelled sets were used. In order to investigate the effect of the ratio of the validation set size to the training set size on the classifier's effectiveness, 3 validation sets of the sizes of **40**, **60**, and **100** were arbitrarily selected. In order to check how a random small sample of the real data set affects each of the SSL algorithms, training and validation of the model took place on the **3** folds with different training and validation data.

With the tuned hyperparameters, each of the SSL algorithms will be

trained on all of the available data and will receive a score on a test set for each of the folds. The results obtained in this procedure will indicate the quality of selected hyperparameters in the validation process and the impact of a random sample from the training set on the algorithms' final results.

In order to investigate even more deeply the obtained hyperparameters and the scalability of the SLL algorithms, further experiments will be performed:

1. The algorithms will be trained and tested on the additional amount of labeled and unlabeled data simultaneously:
   - $D_l = 400, D_{ul} = 800$
   - $D_l = 600, D_{ul} = 1200$
2. The algorithms will be trained and tested on additional unlabeled data:
   - $D_l = 200, D_{ul} = 800$
   - $D_l = 200, D_{ul} = 1200$

4.2.3 Experiments results on IMDB dataset

Hyperparameter estimation Table no. 2 shows hyperparameters and the best results are visible on the different validation sets for 3 folds (with different training and validation data) for Pseudo-Label and Supervised algorithms.

- Pseudo-Label:
  - the lowest average and variance of the error rate was obtained by a model validated on 40 samples.
  - the largest average and variance of error rate was obtained with a model validated on 100 samples.
- Supervised:
  - the lowest average and error rate variances were obtained by a model validated on 60 samples.
  - the largest average and error rate variance was obtained with a model validated on 40 samples.

Table no. 3 shows hyperparameters and the best results on the different validation sets for 3 folds (with different training and validation data) for FixMatch and VAT algorithms.

- FixMatch:
  - the smallest average and the largest variance of the error rate was obtained by a model validated on 40 samples.
  - the smallest variance of the error rate was obtained with a model validated on 60 samples.
  - the largest average error rate was obtained with a model validated on 100 samples.
- VAT:
  - the same results as for FixMatch

Results on test datasets In table no. 4 it can be observed:

- for Pseudo-Label models
  - that among the models validated on 40 samples the smallest average error was obtained by the model with the ratio of the number of labeled data to the number of unlabeled data equal to 600/1200. The same model on the validation set obtained the smallest average error and the smallest variance. Adding new labelled and unlabeled data reduces the average error rate.
  - that among the models validated on 100 samples, the smallest average error had a model with the ratio of labelled to unlabeled data equal to 200/400. The same model on the validation set obtained the largest average error rate and the largest variance error rate. The model has a relatively low error, but the results do not scale with the new data, it seems that the new data spoil the effectiveness of the model.
- for Supervised models
  - that among the models validated on 40 samples the smallest average error was obtained in the model with the ratio of the number of labelled to unlabelled data equal to 400/800. The same model on the validation set obtained the largest average and error rate variance. Adding new labelled and unlabeled data does not reduce the average error rate.

- that among the models validated on 60 samples the smallest average error was obtained by the model with the ratio of the number of labeled data to the number of unlabeled data equal to 400/800. The same model on the validation set obtained the smallest average and the smallest variance for error rate. Adding new labeled and unlabeled data does not reduce the average error rate.

The following conclusions can be drawn from the above observations:

- in the case of the Pseudo-Label algorithm, it turns out that in order to choose the model that best classifies on the test set, it would be completely non-intuitive to choose the model with the greatest variance and the greatest error on the validation set. The results obtained by the algorithm are lower than those for supervised baseline except for one model. It can be assumed that too little validation data was used and/or the algorithm was trained with too few steps to estimate the correct hyperparameters for efficient classification with less error rate score than supervised baseline.
- in the case of supvervised baseline, the choice of the model taking into account the average error or its variance on the validation set does not affect the results on the test set to be significantly better.

In table no. 5 it can be observed:

- for FixMatch models:
  - that among the models validated on 40 samples the smallest average error was obtained by the model with the ratio of the number of labeled data to the number of unlabeled data equal to 200/1200. The same model on the validation set obtained the smallest average error and the largest variance. This model basically makes random decisions with an error close to 50. Addition of new labeled and unlabeled data does not reduce the average error rate.
  - that among the models validated on 60 samples the smallest average error was obtained by the model with the ratio of the

number of labeled data to the number of unlabeled data equal to 400/800. The same model, on the validation set, obtained the smallest variance of error rate. This model basically makes random decisions when the error is close to 50. Adding new labeled and unlabeled data does not reduce the average error rate.

  - that among the models validated on 100 samples the smallest average error had a model with the ratio of the number of labeled data to the number of unlabeled data equal to 400/800/ The same model on the validation set obtained the largest average error rate. As with the rest of the models for the FixMatch algorithm, the average error rate does not change after adding labeled or unlabelled data.

- for VAT models
  - that among the models validated on 40 samples the smallest average error was obtained by the model with the ratio of the number of labeled data to the number of unlabeled data equal to 600/1200. The same model on the validation set has the smallest average error rate. The model will yield slightly better results for more labeled and unlabeled data.
  - that among the models validated on 60 samples the smallest average error was obtained by the model with the ratio of the number of labeled data to the number of unlabeled data equal to 400/800. The same model on the validation set obtained the smallest variance. The model gets much better effects with the increase of tagged and unlabeled data, and slightly better with the increase of unlabeled data.
  - that among the models validated on 100 samples the smallest average error was obtained by the model with the ratio of the number of labeled data to the number of unlabeled data equal to 400/800. The same model on the validation set has obtained the largest average error. Adding new labeled and unlabeled data does not reduce the average error rate.

The following conclusions can be drawn from the above observations:

- in the case of the FixMatch algorithm, the best generalizing predictor

for test set is the smallest error rate variance on the validation set. It can be assumed that too little validation data was used for FixMatch and/or the algorithm was trained with too few steps to estimate the proper hyperparameters to make the correct classification with less error rate than supervised baseline.

- in case of the VAT algorithm, the smallest error variance on the validation set translates into results on the test set. The trade off between the amount of validation and training data which equal to 60 validation samples proved to be optimal, but the effectiveness of the model is still much lower than for the supervised model. It can be assumed that an increase in the number of training steps would result in a better result on the test set.

4.2.4 Experiments results on MR dataset

Hyperparameter estimation The table no. 6 shows hyperparameters and the best results on the different validation sets for 3 folds (with different training and validation data) for Pseudo-Label and Supervised algorithms.

- Pseudo-Label:
  - the smallest average error rate and the largest variance of the error rate was obtained by a model validated on 40 samples.
  - the largest average error rate and the smallest variance was obtained with a model validated on 100 samples.
- Supervised:
  - the smallest average error rate and the largest variance of the error rate was obtained by a model validated on 60 samples.
  - the largest average error rate was obtained with a model validated on 100 samples.
  - the smallest variance of the error rate was obtained with a model validated on 40 samples.

Table no. 7 shows hyperparameters and the best results on the different validation sets for 3 folds (with different training and validation data) for FixMatch and VAT algorithms.

- FixMatch:

  - the smallest average of the error rate was obtained with a model validated on 100 samples.
  - the smallest variance of the error rate obtained a model validated on 40 samples.
  - the largest average and variance of the error rate obtained a model validated on 60 samples.
- VAT:
  - the largest average error rate was obtained with a model validated on 100 samples.
  - the smallest variance of the error rate was obtained with a model validated on 60 samples.
  - the smallest average and the largest variance of the error rate was obtained with a model validated on 40 samples.

Results on test datasets In table no. 8 it can be observed

- for Pseudo-Label models
  - that among the models validated on 40 samples the smallest average error was obtained by the model with the ratio of the number of labeled data to the number of unlabeled data equal to 600/1200. The same model on the validation set obtained the smallest average and the largest variance for the error rate. The average error rate decreases with the addition of labeled and unlabeled data.
  - that among the models validated on 100 samples the smallest average error was obtained by the model with the ratio of the number of labeled data to the number of unlabeled data equal to 400/800. The same model on the validation set obtained the largest average error rate and the smallest variance error rate. The average error decreases with the addition of labeled and unlabeled data.
- for Supervised models
  - that among the models validated on 40 samples the smallest

average error rate was obtained by the model with the ratio of the number of labeled data to the number of unlabeled data equal to 600/1200. The same model on the validation set obtained the largest average and variance of the error rate. The average error rate decreases with the addition of new labeled and unlabeled data.

- that among the models validated on 60 samples the smallest average error was obtained by the model with the ratio of the number of labeled data to the number of unlabeled data equal to 400/800. The same model on the validation set obtained the smallest average and the largest variance of the error rate. The average error rate does not decrease with the addition of labelled and unlabeled data.
- that among the models validated on 100 samples the smallest average of the error rate was obtained by the model with the ratio of the number of labeled data to the number of unlabeled data equal to 600/1200. The same model on the validation set has obtained the largest average error. The average error decreases with the addition of labeled and unlabeled data.

The following conclusions can be drawn from the above observations:

- in the case of the Pseudo-Label algorithm, it turns out that not the average error result on the validation set but it's variance is a good indicator of successful model generalization on the test set. The results obtained by the model are lower than those for supervised baseline. It can be assumed that too little validation data was used and/or the algorithm was trained with too few steps to estimate the correct hyperparameters for correct classification with error rate smaller than supervised baseline.
- in the case of supervised baseline, similarly as in the Pseudo-Label algorithm, not an average but the variance of the error rates is a good indicator of the quality of the models' results on the test set.

In table no. 9 it can be observed:

- for FixMatch models:
  - that among the models validated on 40 samples the smallest average error was obtained by the model with the ratio of the number of labeled data to the number of unlabeled data equal to 200/1200. The same model on the validation set has obtained the smallest average error. The average error does not decrease with the addition of labeled and unlabeled data.
  - that among the models validated on 60 samples the smallest average error was obtained by the model with the ratio of the number of labeled data to the number of unlabeled data equal to 600/1200. The same model on the validation set obtained the largest average error and the largest variance of error. The average error does not decrease with the addition of labeled and unlabeled data.
  - that among the models validated on 100 samples the smallest average error was obtained by the model with the ratio of the number of labeled data to the number of unlabeled data equal to 200/400. The same model on the validation set has obtained the smallest average error. The average error does not decrease with the addition of labeled and unlabeled data.
- for VAT models:
  - that among the models validated on 40 samples the smallest average error was obtained by the model with the ratio of the number of labeled data to the number of unlabeled data equal to 600/1200. The same model on the validation set obtained the smallest average and the largest error rate variance. The average error rate decreases with the addition of labeled data.
  - that among the models validated on 60 samples the smallest average error was obtained by the model with the ratio of the number of labeled data to the number of unlabeled data equal to 400/800. The same model on the validation set obtained the smallest variance. The average error decreases with the addition of labeled data. The model obtains smaller average error than supervised baseline.
  - that among the models validated on 100 samples the smallest

average error was obtained by the model with the ratio of the number of labeled data to the number of unlabeled data equal to 400/800. The same model on the validation set has obtained the largest average error. The average error decreases with the addition of labeled data.

The following conclusions can be drawn from the above observations:

- for the FixMatch algorithm, the average error on the validation set translates into results on the test set. It can be assumed that too little validation data was used for FixMatch and/or the algorithm was trained with too few steps to estimate the right hyperparameters to make the correct classification with less error rate than supervised baseline.
- in case of the VAT algorithm, the smallest error variance on the validation set translates into results on the test set. One of the trained models achieved a smaller error than supervised baseline. It can be assumed that too few steps were used to teach the model and/or too few validation examples were used to correctly select the hyperparameters that will give the model better classification results as the unlabeled number of samples increase.

5 Conclusions and Future Works

It was not possible to train a FixMatch model that would have been more effective than the Pseudo-Labeling and VAT method, and even supervised baseline on the used amount of training and validation data. It can be hypothesized that the number of validation data and/or the duration of the algorithm training was too low. Depending on the used SSL algorithm, the variance or average error on the folds are the parameters that should considered when choosing the best hyperparameters. In most cases better results on the test set were given by models trained and validated on the validation set of size 60 and 100, so it can be hypothesized that it is worth to use slightly less or equal number of ratio between the validation data samples to training data samples, in case of small amount of labelled data. Most of the VAT and PS models have scaled up as the amount of labelled

data increases. Only the VAT method from the tested SSL methods used on the MR set proved to be better than the supervised baseline for any amount of the proposed validation data. On this basis it can be hypothesized that it is a method that works well with a very small amount of training and validation data. In order to improve the result of the FixMatch algorithm and other SSL methods, it would be necessary to increase the number of steps and perform experiments on larger amounts of labelled and unlabeled data. Such experiments will be future work that should be done in order to analyze more thoroughly the discussed SSL methods.

References

[1] Thomas M. Mitchell. *Machine Learning*. McGraw-Hill, Inc., USA, 1 edition, 1997.

[2] Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep Clustering for Unsupervised Learning of Visual Features. *arXiv e-prints*, page arXiv:1807.05520, July 2018.

[3] Ertunc Erdil, Krishna Chaitanya, and Ender Konukoglu. Unsupervised out-of-distribution detection using kernel density estimation. *arXiv e-prints*, page arXiv:2006.10712, June 2020.

[4] Dragi Anevski, Richard Gill, and Stefan Zohren. Estimating a probability mass function with unknown labels. *arXiv.org (math.ST) 1312.1200 Preprint*, 12 2013.

[5] Laurens van der Maaten and Geoffrey Hinton. Viualizing data using t-sne. *Journal of Machine Learning Research*, 9:2579–2605, 11 2008.

[6] Dong Hyun Jeong, Caroline Jeong, William Ziemkiewicz, Remco Ribarsky, and Chang. Understanding principal component analysis using a visual analytics tool. 01 2010.

[7] Kevin P. Murphy. *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012.

[8] Olivier Chapelle, Bernhard Schölkopf, and Alexander Zien. *Semi-*

Supervised Learning (Adaptive Computation and Machine Learning). The MIT Press, 2006.

[9] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg, 2006.

[10] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[11] Guodong Zhang, Chaoqi Wang, Bowen Xu, and Roger Grosse. Three Mechanisms of Weight Decay Regularization. *arXiv e-prints*, page arXiv:1810.12281, October 2018.

[12] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 06 2014.

[13] Alex Hernández-García and Peter König. Data augmentation instead of explicit regularization. *arXiv e-prints*, page arXiv:1806.03852, June 2018.

[14] Ekin D. Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V. Le. RandAugment: Practical automated data augmentation with a reduced search space. *arXiv e-prints*, page arXiv:1909.13719, September 2019.

[15] Terrance DeVries and Graham W. Taylor. Improved Regularization of Convolutional Neural Networks with Cutout. *arXiv e-prints*, page arXiv:1708.04552, August 2017.

[16] Qizhe Xie, Zihang Dai, Eduard Hovy, Minh-Thang Luong, and Quoc V. Le. Unsupervised Data Augmentation for Consistency Training. *arXiv e-prints*, page arXiv:1904.12848, April 2019.

[17] Yves Grandvalet and Y. Bengio. Semi-supervised learning by entropy minimization. *Adv. Neural Inform. Process. Syst.*, 17, 01 2004.

[18] Dong-Hyun Lee. Pseudo-label : The simple and efficient semi-supervised learning method for deep neural networks. *ICML 2013 Workshop : Challenges in Representation Learning (WREPL)*, 07 2013.

[19] Mehdi Sajjadi, Mehran Javanmardi, and Tolga Tasdizen. Regularization With Stochastic Transformations and Perturbations for Deep Semi-Supervised Learning. *arXiv e-prints*, page arXiv:1606.04586, June 2016.

[20] Samuli Laine and Timo Aila. Temporal Ensembling for Semi-Supervised Learning. *arXiv e-prints*, page arXiv:1610.02242, October 2016.

[21] Mehdi Sajjadi, Mehran Javanmardi, and Tolga Tasdizen. Regularization With Stochastic Transformations and Perturbations for Deep Semi-Supervised Learning. *arXiv e-prints*, page arXiv:1606.04586, June 2016.

[22] David Berthelot, Nicholas Carlini, Ekin D. Cubuk, Alex Kurakin, Kihyuk Sohn, Han Zhang, and Colin Raffel. ReMixMatch: Semi-Supervised Learning with Distribution Alignment and Augmentation Anchoring. *arXiv e-prints*, page arXiv:1911.09785, November 2019.

[23] Antti Tarvainen and Harri Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. *arXiv e-prints*, page arXiv:1703.01780, March 2017.

[24] Kihyuk Sohn, David Berthelot, Chun-Liang Li, Zizhao Zhang, Nicholas Carlini, Ekin D. Cubuk, Alex Kurakin, Han Zhang, and Colin Raffel. FixMatch: Simplifying Semi-Supervised Learning with Consistency and Confidence. *arXiv e-prints*, page arXiv:2001.07685, January 2020.

[25] David Berthelot, Nicholas Carlini, Ian Goodfellow, Nicolas Papernot, Avital Oliver, and Colin Raffel. MixMatch: A Holistic Approach to Semi-Supervised Learning. *arXiv e-prints*, page arXiv:1905.02249, May 2019.

[26] Jungdae Kim. Unofficial PyTorch implementation of "FixMatch: Simplifying Semi-Supervised Learning with Consistency and Confidence" , March 2020. Available at `https://github.com/kekmodel/FixMatch-pytorch`.

[27] Ingyo Chung. Virtual Adversarial Training (VAT) for semi-supervised MNIST written in PyTorch, February 2019. Available at `https://github.com/jik0730/VAT-pytorch`.

[28] Michał Filek. Text classification using the semi-supervised learn-

ing, September 2020. Available at `https://github.com/M1F1/Text-classification-using-the-semi-supervised-learning-methods`.

[29] Yann LeCun and Corinna Cortes. MNIST handwritten digit database. 2010.

[30] Avital Oliver, Augustus Odena, Colin Raffel, Ekin D. Cubuk, and Ian J. Goodfellow. Realistic Evaluation of Deep Semi-Supervised Learning Algorithms. *arXiv e-prints*, page arXiv:1804.09170, April 2018.

[31] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics.

[32] Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. Thumbs up? sentiment classification using machine learning techniques. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2002.

[33] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching Word Vectors with Subword Information. *arXiv e-prints*, page arXiv:1607.04606, July 2016.

[34] Yoon Kim. Convolutional Neural Networks for Sentence Classification. *arXiv e-prints*, page arXiv:1408.5882, August 2014.

[35] Ashish Vaswani, Samy Bengio, Eugene Brevdo, Francois Chollet, Aidan N. Gomez, Stephan Gouws, Llion Jones, Łukasz Kaiser, Nal Kalchbrenner, Niki Parmar, Ryan Sepassi, Noam Shazeer, and Jakob Uszkoreit. Tensor2tensor for neural machine translation. *CoRR*, abs/1803.07416, 2018.

	Pseudo-Label			Supervised		
Validation sizes	40	60	100	40	60	100
Hyperparameters:						
B	32	32	32	32	32	32
U	32	32	32	-	-	-
Max steps	1000	1000	1000	200	200	200
Initial lr	1e-2	1e-3	1e-3	1e-2	1e-2	1e-2
Dropout	0.1	0.2	0.1	0.2	0.2	0.1
Optimizer	Adam	Adam	Adam	Adam	Adam	Adam
T_1	1	1	1	-	-	-
T_2	1000	1000	1000	-	-	-
$Max\ \lambda$	1	1	1	-	-	-
Lr scheduler	StepLR	StepLR	StepLR	StepLR	StepLR	StepLR
Decay gamma	0.5	0.5	0.5	0.5	0.5	0.5
Initial decay step	400	200	250	50	50	50
Decay step size	40	1	1	1	1	1
Error rate	37.7 ± 0.4	42.0 ± 6.2	42.5 ± 6.4	41.4 ± 12.2	37.2 ± 1.4	39.3 ± 1.5

Table 2: Best validation error rates for IMDB obtained by Supervised and Pseudo-Label models on 3 training folds with tuned hyperparameters on validation datasets with 40, 60 and 100 sample sizes.

	FixMatch			VAT		
Validation sizes	40	60	100	40	60	100
Hyperparameters:						
B	32	32	32	64	64	64
U	224	224	224	256	256	256
Max steps	300	300	300	300	300	300
Optimizer	SGD	SGD	SGD	Adam	Adam	Adam
Dropout	0.1	0.1	0.2	0.1	0.2	0.25
Momentum	0.9	0.9	0.9	-	-	-
Weight decay	0.0005	0.0005	0.0005	-	-	-
Initial lr	3e-2	3e-2	3e-2	2e-5	2e-5	2e-5
Lr scheduler	-	CosineLR	CosineLR	StepLR	StepLR	StepLR
Decay γ	-	-	-	0.5	0.5	0.5
Initial decay step	-	-	-	20	20	20
Decay step size	-	-	-	100	75	80
Warmup steps	-	50	50	-	-	-
λ	0.1	0.1	0.1	1	1	1
ε	-	-	-	3	3	3
K	-	-	-	1	1	1
ξ	-	-	-	1e-6	1e-6	1e-6
τ	0.999	0.999	0.999	-	-	-
N weak	1	1	2	-	-	-
Error rate	34.2 ± 10.4	35.5 ± 3.6	45.5 ± 10.3	45.6 ± 11.0	46.3 ± 6.9	47.6 ± 7.4

Table 3: Best validation error rates for IMDB obtained by FixMatch and VAT models on 3 training folds with tuned hyperparameters on validation datasets with 40, 60 and 100 sample sizes.

	Pseudo-Label			Supervised		
Validation sizes	40	60	100	40	60	100
$\boldsymbol{X_l/X_{unl}}$						
200/400	34.6 ± 4.0	33.6 ± 3.7	28.8 ± 0.6	32.5 ± 2.2	32.5 ± 0.1	32.3 ± 3.8
400/800	36.7 ± 4.3	31.8 ± 1.4	32.2 ± 2.8	31.0 ± 1.2	30.4 ± 0.5	29.3 ± 2.5
600/1200	31.2 ± 0.9	32.0 ± 2.0	33.4 ± 3.9	31.2 ± 0.9	32.0 ± 2.0	33.4 ± 3.9
200/800	33.9 ± 1.1	32.9 ± 3.0	33.3 ± 4.3	-	-	-
200/1200	33.9 ± 1.1	32.8 ± 3.0	33.3 ± 4.3	-	-	-

Table 4: Test error rates for IMDB obtained by Pseudo-Label and Supervised models on 3 training folds(with different labeled and unlabeled sample sizes). Hyperparameters for those models were tuned using 200 labeled and 400 unlabeled training samples on validation datasets with 40, 60 and 100 sample sizes.

	FixMatch			VAT		
Validation sizes	40	60	100	40	60	100
$\boldsymbol{X_l/X_{unl}}$						
200/400	50.9 ± 2.2	46.8 ± 4.3	47.1 ± 3.3	41.0 ± 1.5	42.6 ± 1.8	42.3 ± 1.7
400/800	50.9 ± 2.6	46.1 ± 3.8	46.7 ± 2.7	41.0 ± 1.9	37.0 ± 2.0	40.2 ± 3.6
600/1200	49.8 ± 1.5	46.3 ± 5.5	47.6 ± 2.9	40.7 ± 0.5	37.5 ± 0.1	41.5 ± 1.1
200/800	50.8 ± 1.9	50.4 ± 1.9	47.4 ± 2.8	41.0 ± 1.3	41.3 ± 2.0	42.3 ± 1.7
200/1200	48.5 ± 6.4	50.3 ± 1.6	47.1 ± 3.3	41.0 ± 1.4	41.3 ± 2.0	43.2 ± 1.7

Table 5: Test error rates for IMDB obtained by FixMatch and VAT models on 3 training folds(with different labeled and unlabeled sample sizes). Hyperparameters for those models were tuned using 200 labeled and 400 unlabeled training samples on validation datasets with 40, 60 and 100 sample sizes.

	Pseudo-Label			Supervised		
Validation sizes	40	60	100	40	60	100
Hyperparameters:						
B	32	32	32	32	32	32
U	32	32	32	-	-	-
Max steps	200	1000	200	200	200	200
Initial lr	1e-2	1e-1	1e-1	1e-3	1e-2	1e-2
Dropout	0.1	0.1	0.2	0.1	0.1	0.3
Optimizer	Adam	Adam	Adam	Adam	Adam	Adam
T_1	1	1	1	-	-	-
T_2	200	200	200	-	-	-
Max λ	1	1	1	-	-	-
Lr scheduler	StepLR	-	StepLR	-	-	-
Decay γ	0.5	-	0.5	-	-	-
Initial decay step	400	-	60	-	-	-
Decay step size	40	-	20	-	-	-
Error rate	25.0 ± 6.2	34.1 ± 6.2	37.8 ± 2.5	42.3 ± 1.2	42.1 ± 4.8	45.2 ± 3.1

Table 6: Best validation error rates for MR obtained by Supervised and Pseudo-Label models on 3 training folds with tuned hyperparameters on validation datasets with 40, 60 and 100 sample sizes.

	FixMatch			VAT		
Validation sizes	40	60	100	40	60	100
Hyperparameters:						
B	32	32	32	64	64	64
U	224	224	224	256	256	256
Max steps	300	300	300	300	600	600
Dropout	0.1	0.1	0.1	0.1	0.2	0.2
Optimizer	SGD	SGD	SGD	Adam	Adam	Adam
Momentum	0.9	0.9	0.9	-	-	-
Weight decay	5e-4	5e-4	1e-3	-	-	-
Initial lr	3e-2	3e-2	3e-2	2e-5	2e-5	2e-5
Lr scheduler	-	CosineLR	CosineLR	-	StepLR	StepLR
Decay γ	-	-	-	-	0.5	0.5
Initial decay step	-	-	-	-	50	20
Decay step size	-	-	-	-	200	180
Warmup steps	-	50	50	-	-	-
λ	0.1	0.1	0.1	1	1	1
ϵ	-	-	-	3	3	3
K	-	-	-	1	1	1
ξ	-	-	-	1e-6	1e-6	1e-6
τ	0.999	0.999	0.999	-	-	-
N weak	1	2	2	-	-	-
Error rate	46.3 ± 3.8	47.2 ± 6.0	44.9 ± 4.4	41.4 ± 19.1	45.6 ± 10.0	46.4 ± 10.3

Table 7: Best validation error rates for MR obtained by FixMatch and VAT models on 3 training folds with tuned hyperparameters on validation datasets with 40, 60 and 100 sample sizes.

	Pseudo-Label			Supervised		
Validation sizes	40	60	100	40	60	100
$\boldsymbol{X_l/X_{unl}}$						
200/400	44.5 ± 3.4	48.0 ± 2.6	44.0 ± 3.6	44.2 ± 4.6	44.2 ± 5.9	43.2 ± 4.6
400/800	42.6 ± 1.3	41.3 ± 3.3	41.2 ± 2.7	43.9 ± 4.3	41.8 ± 3.8	40.5 ± 1.3
600/1200	41.6 ± 2.3	41.9 ± 1.2	38.4 ± 0.4	39.4 ± 2.4	43.5 ± 7.2	38.1 ± 2.7
200/800	44.5 ± 3.4	47.2 ± 3.8	44.0 ± 3.6	-	-	-
200/1200	44.2 ± 3.9	47.2 ± 3.8	44.0 ± 3.6	-	-	-

Table 8: Test error rates for MR obtained by Pseudo-Label and Supervised models on 3 training folds(with different labeled and unlabeled sample sizes). Hyperparameters for those models were tuned using 200 labeled and 400 unlabeled training samples on validation datasets with 40, 60 and 100 sample sizes.

	FixMatch			VAT		
Validation sizes	40	60	100	40	60	100
$\boldsymbol{X_l/X_{unl}}$						
200/400	47.7 ± 2.4	48.7 ± 2.8	46.3 ± 4.8	42.9 ± 4.4	43.4 ± 3.8	43.7 ± 33.3
400/800	47.9 ± 8.2	49.6 ± 1.2	50.4 ± 1.2	39.7 ± 3.6	45.4 ± 1.7	42.1 ± 4.2
600/1200	48.4 ± 6.6	48.8 ± 0.7	50.0 ± 2.2	38.4 ± 5.3	31.2 ± 1.3	33.3 ± 4.3
200/800	50.6 ± 8.6	51.9 ± 9.1	47.6 ± 3.0	45.1 ± 1.9	45.1 ± 1.7	44.0 ± 2.8
200/1200	47.1 ± 8.4	51.4 ± 8.7	48.5 ± 2.8	46.1 ± 1.3	45.6 ± 2.0	43.9 ± 2.7

Table 9: Test error rates for MR obtained by FixMatch and VAT models on 3 training folds(with different labeled and unlabeled sample sizes). Hyperparameters for those models were tuned using 200 labeled and 400 unlabeled training samples on validation datasets with 40, 60 and 100 sample sizes.