machine learning & knowledge extraction

MDPI

Article

Analysing Semi-Supervised ConvNet Model Performance with Computation Processes

Elie Neghawi *,† **and Yan Liu** †

Gina Cody School of Engineering and Computer Science, Concordia University, Montreal, QC H3G 1M8, Canada; yan.liu@concordia.ca

* Correspondence: e_negh@live.concordia.ca

† These authors contributed equally to this work.

Abstract: The rapid development of semi-supervised machine learning (SSML) algorithms has shown enhanced versatility, but pinpointing the primary influencing factors remains a challenge. Historically, deep neural networks (DNNs) have been used to underpin these algorithms, resulting in improved classification precision. This study aims to delve into the performance determinants of SSML models by employing post-hoc explainable artificial intelligence (XAI) methods. By analyzing the components of well-established SSML algorithms and comparing them to newer counterparts, this work redefines semi-supervised computation processes for both data preprocessing and classification. Integrating different types of DNNs, we evaluated the effects of parameter adjustments during training across varied labeled and unlabeled data proportions. Our analysis of 45 experiments showed a notable 8% drop in training loss and a 6.75% enhancement in learning precision when using the Shake-Shake26 classifier with the RemixMatch SSML algorithm. Additionally, our findings suggest a strong positive relationship between the amount of labeled data and training duration, indicating that more labeled data leads to extended training periods, which further influences parameter adjustments in learning processes.

Keywords: deep learning; explainable artificial intelligence; convolutional neural networks (CNN); semi-supervised machine learning

Citation: Neghawi, E.; Liu, Y. Analysing Semi-Supervised ConvNet Model Performance with Computation Processes. *Mach. Learn. Knowl. Extr.* **2023**, *5*, 1848–1876. https://doi.org/10.3390/make5040089

Academic Editor: Andreas Holzinger

Received: 23 October 2023
Revised: 15 November 2023
Accepted: 22 November 2023
Published: 29 November 2023

1. Introduction

Recent advancements in semi-supervised machine learning (SSML) have led to the development of increasingly intricate models, architectures, and methods for parameter tuning, resulting in some of the most advanced algorithms to date [1]. However, as these algorithms become more sophisticated, so too does their complexity, which has led to a demand for greater transparency, particularly when the algorithms produce questionable results. When outcomes are inaccurate, it becomes even more essential to have clear explanations. Moreover, even with accurate results, the ability to trust in machine learning depends on being able to understand and justify how the decisions were made. This is crucial in high-stakes scenarios. As such, explainable artificial intelligence (XAI) methods are being developed to make the inner workings of these complex SSML algorithms more transparent, moving away from the notion of them as 'black boxes' to systems whose reasoning can be understood and trusted by users. There are two primary categories of XAI: post-hoc methods and model-based XAI. If an ML model's design ensures user transparency regarding its internal operations and decision-making, it is classified under model-based XAI. Due to the profound intricacy of SSML, this approach is unsuitable. Hence, post-hoc XAI, which evaluates the decision-making logic of these enigmatic AI models, is critical, especially for grasping SSML algorithms' overarching rationale.

Amidst XAI's complexities [2], labeled data acquisition remains a challenge [3]. The ML community often combines a vast volume of unlabeled data with minimal labeled counterparts in standard SSML algorithms. These algorithms typically include a network classifier and two primary costs: classification and consistency costs. While the former utilizes labeled data, the latter assesses discrepancies between network classifier augmentations using unlabeled data. Although familiar SSML algorithms incorporate designs such as the Π model [4] newer algorithms such as MixMatch and RemixMatch employ innovative patterns enhancing performance metrics.

DNNs as network classifiers have amplified learning accuracy [5]. Still, their intricacies often shroud performance justifications, leading to arbitrary adjustments in custom SSML algorithms. Recognizing this, our study systematically investigates SSML algorithm variations, focusing on learning accuracy and training effects related to DNN and SSML integration. Using prior frameworks [6], we unveil the fundamental factors influencing learning and training. Moreover, we employ XAI methodologies to devise a preprocessing-focused semi-supervised computation process (PF-SSCP), which essentially mirrors previous designs with slight reordering.

By integrating contemporary DNNs, including Shake-Shake26, DenseNet [7], and WideResNet [8], with each SSML algorithm, we establish a benchmark. Preliminary results highlight the performance superiority of Shake-Shake26 over DenseNet by 1.13% in terms of loss. However, this performance comes at the cost of extended training durations and increased resource consumption due to intricate interlayer connections. This raises our study's central question:

> Which are the most influential factors impacting computational costs and training accuracy when melding deep neural networks (DNN) with semi-supervised machine learning (SSML)?

Our research contributes in the following ways:

- We propose a framework method for connecting contemporary DNNs with SSML and illustrate a computation process with highlights of the most critical functions.
- We outline essential preprocessing steps when implementing these networks across varying labeled/unlabeled data ratios.
- We synergize the preprocessing-focused SSCP (PF-SSCP) with the classifier-focused SSCP (CF-SSCP), offering insights to maximize performance.
- We present SSML algorithms' anatomy, furnishing guidelines for future model design rooted in this anatomy's understanding.

2. Related Work

Several approaches exist for explaining CNNs, with one of the most notable being the unification and taxonomic view of graph neural networks (GNNs) [9]. This significant work delineates the commonalities and distinctions among existing methodologies, offering a foundation for future advancements [10]. The evaluation is streamlined by creating a suite of benchmark graph datasets specifically for GNN explainability. This encompasses metrics and datasets essential for appraising GNN explainability [11]. Nonetheless, complexities escalate when these concepts are applied to SSML algorithms due to the employment of multiple DNNs as network classifiers. Our approach delves into model-level explanations, directing attention to the taxonomy of SSMLs, and underscores precise changes and their driving forces.

An alternative study presented an interpretable compositional CNN approach [12,13], aiming to refine the conventional convolutional neural network (CNN) structure. This strategy is designed to facilitate the learning of filters that capture meaningful visual motifs within intermediate convolutional strata. While effective for illustrating a singular CNN, its applicability falters with SSML, where the network classifier integrates two or more CNNs.

Additionally, the XGNN method, which is a novel approach to graph generation, trains a graph generator to approximate predictions for target graphs rather than optimiz-

ing the input graph directly [14]. The graphs generated by this method are considered to be accurate explanatory targets that capture the unique patterns characteristic of graphs. The XGNN's graph generation process uses reinforcement learning, allowing it to incorporate any competent graph generation algorithm within its framework, which broadens the scope and applicability of its explanations. This approach enhances the overall understanding of how trained graph neural networks (GNNs) function. However, while XGNN demonstrates strong performance in explaining graph classification models, its application to SSML algorithms is less effective. To address this, our technique leverages the robust graph generation capabilities of XGNN but enhances them by adding a layer of decomposition based on instance-level explanations [15]. This serves to shed light on three critical aspects of the latest SSML algorithms, providing a more nuanced understanding.

Recent advancements in SSML extensively use certain techniques, including consistency regularization, self-training, and entropy minimization. The study reported in [16,17] indicates that SSML methods may experience a significant decline in performance when there is a discrepancy between the distribution of unlabeled data used during training and those encountered post-training, even if other variables such as initialization, preprocessing, regularization, and augmentation are well-managed. Conversely, our research focuses on developing a generalized approach to these techniques and elucidating the specific procedural steps that lead to better performance, independent of the specific characteristics of the unlabeled data.

Informed machine learning [16] integrates supplementary prior knowledge during model training. The referenced study provides a comprehensive synopsis of diverse strategies within this domain, meticulously outlining the informed machine learning design process and its numerous constituent elements. It also clearly distinguishes informed ML from conventional ML paradigms. Our research, however, zeroes in on SSML algorithms, starting with the initial design, then delving deeper into the taxonomy of each and revealing the influential factors in every segment.

3. Survey Analysis of SOTA Models with Modular Computation Components

This section conducts a technical survey of SOTA semi-supervised learning (SSL) models, focusing on dissecting and visualizing the computational components to examine their modularity. The purpose of this modular-based analysis is to abstract away from the intricate parameter-level specifics of each model and instead to concentrate on their functional attributes. The goal is to distill a computation process that encapsulates the shared computational components found within various models. Through this high-level analysis, we aim to pinpoint critical computational junctures that may be leveraged to enhance learning performance. The models under consideration, which include Temporal Ensembling, the Π model, Mean Teacher, MixMatch [18], and ReMixMatch [19], are explored for their design, taxonomy, and innovative contributions to the field of SSML. We intend to illustrate the interplay of these advanced techniques with the proposed semi-supervised computation process (SSCP), tracing the connections between modular computation components and the original models' equations.

3.1. Temporal Ensembling

The Temporal Ensembling structure [4] maintains a configuration that can be depicted in Figure 1, featuring a network classifier with a single DNN. The primary distinction between Temporal Ensembling and the generic form with multiple DNNs lies in the fact that, here, network augmentations are assessed only once per epoch, and network consistency relies on previous evaluations for the unsupervised loss component.

In the Temporal Ensembling model, the data x_i undergo the same preprocessing steps to achieve optimal performance. After this phase, the preprocessed data are fed into a neural classifier network, which determines the classification prediction z_i. In the subsequent iteration, z_i is aggregated, aiding in the computation of both consistency and classification costs in the current iteration. This mirrors the process followed in generic SSML; however, the computation of the consistency cost is exclusive to the unlabeled data. In Figure 1, the critical path for various steps, in the case of unlabeled data, is illustrated as a black path. For labeled data, both classification and consistency costs are calculated, and the forward propagation is depicted by the black and orange paths. Moreover, the total loss computation is articulated in Equation (12), mirroring that of the generic SSML.

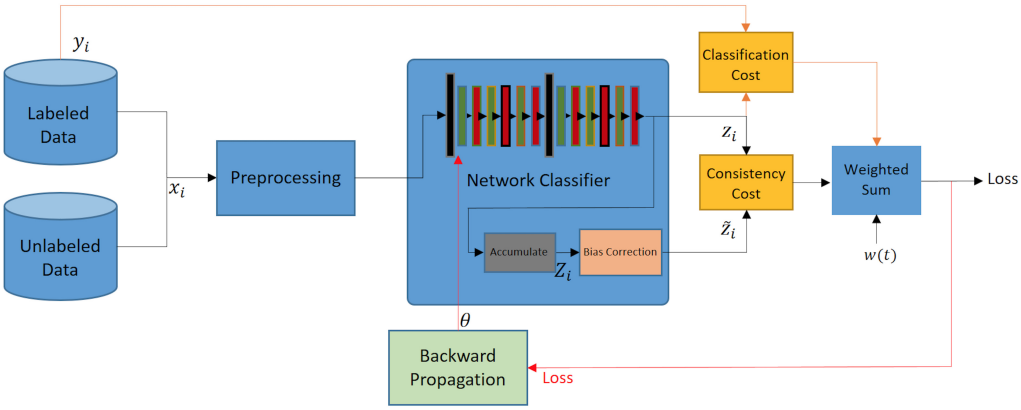

Figure 1. Representation of the Temporal Ensembling model structure.

The classification cost computation aligns precisely with that of generic SSML, as detailed in Equation (11), and is defined as the cross-entropy between the prediction z_i and the label y_i.

Regarding the consistency cost, despite being governed by the same Equation (10), there is significant variation in the target vector $\tilde{z}_i$. Here, $\tilde{z}_i$ is not derived from another neural network's assessment under varying dropout conditions. Instead, it is an accumulation of ensemble predictions preceded by a bias correction step. The neural network computes z_i, which is then amalgamated into the ensemble outputs Z_i following the update $Z_i \leftarrow \alpha Z_i + (1-\alpha)z_i$ [4], with α representing the momentum term that dictates the depth of historical training data influencing the ensemble. During the initial iteration ($i = 0$), z_{i-1} is set to zero, indicating that the initial values for $\tilde{z}$ and Z are also zero. Consequently, it is imperative to correct the startup bias in Z by dividing it by the factor $(1-\alpha^t)$ to obtain the training target $\tilde{z}_i$. In the context of backward propagation, this loss is applied solely to the current neural network.

Temporal Ensembling offers several advantages over other SSML algorithms, as outlined below:

- Since the network undergoes evaluation once for each input, the training process is expedited.
- With a training target characterized by reduced noise compared to the Π model, Temporal Ensembling can potentially realize superior accuracy performance.

However, Temporal Ensembling also presents certain limitations, detailed as follows:

- It necessitates the storage of auxiliary data and the introduction of a new hyperparameter, α, thereby demanding additional memory resources.
- Similar to the Π model, the Temporal Ensembling model lacks dynamic learning capabilities, necessitating system retraining for effective functionality.

3.2. Π *Model*

This model, a variant of SSML, employs a self-ensembling mechanism during training [4], ensuring network output consistency across two instances of the same input under different dropout conditions.

In Figure 2, both labeled and unlabeled data, x_i, are subjected to identical preprocessing, without discrepancy. Upon entry into the network classifier, the data are replicated across two analogous neural networks. These networks are differentiated by the following factors:

- Dropout regularization, infusing an element of randomness into these twin architectures.
- The types of augmentations employed, which expand the dataset and render identical labeled and unlabeled data as distinct inputs.

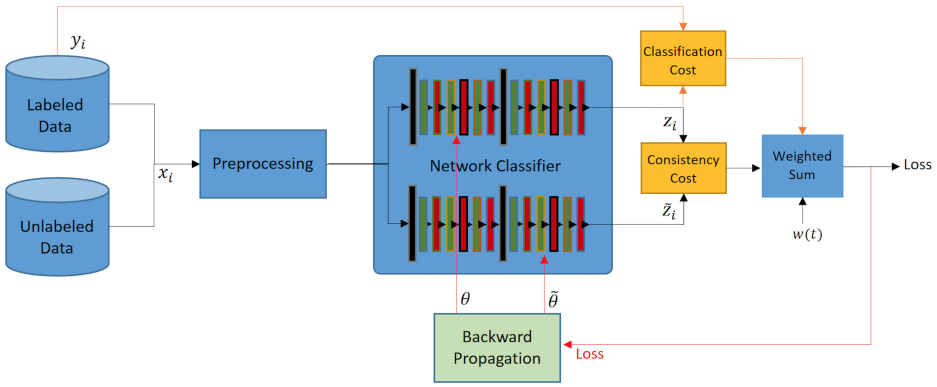

Figure 2. Π model structure representation.

These two neural networks yield the classifications z_i and $\tilde{z}_i$, which are instrumental in computing the total loss. When labeled data are input into the network classifier, both consistency and classification costs are derived. As shown in Figure 2, this scenario necessitates following the black and orange paths for forward propagation to ascertain the total loss.

Conversely, for unlabeled data, only the consistency cost is derived, meaning that the black path in Figure 2 is pursued during forward propagation. Ultimately, the losses across the entire dataset are aggregated via a weighted sum. A pivotal aspect here is the employment of the ramp-up function $w(t)$, which assigns weights to the consistency costs, compelling the semi-supervised system to prioritize classification costs in the initial stages. Throughout the loss computation, a backward propagation must be initiated to update the parameters aimed at loss reduction. In Figure 2, backpropagation is signified by the red path, indicating that both neural network parameters, θ and θ', are updated, thus revising the entire neural network flow.

Evaluating the performance of the Π model, we observe that it offers better accuracy than preceding SSML algorithms. However, this model has the following limitations:

- The network output is computed twice for the same input data, resulting in additional computational costs.
- The model is vulnerable to errors when fed incorrectly labeled data, negatively impacting its accuracy.
- This model cannot learn dynamically as it relies on batch-learning, in contrast with online learning neural networks.

3.3. *Mean Teacher*

To address the shortcomings of the two SSMLs given in the previous subsections, the Mean Teacher method calculates the averages of the model weights rather than predicting the outcome. This model is based on an average of consecutive student models; hence, it is referred to as the Mean Teacher method [4].

In this case, the raw data x_i are first preprocessed and then injected into the network classifier in the same manner as the previous SSML models. However, in this case, the

network classifier is not the same as previous models. Here, the network classifier consists of two identical DNNs with dropout conditions referred to as the student and teacher models. Primarily, the Mean Teacher model is exactly the same as the Π model during the computation of forward propagation for both unlabeled and labeled data. However, the Mean Teacher method diverges from the Π Model during the backward propagation phase and in the strategy it employs to update weights.

The backward propagation is simply computed for the the parameters θ of the student model. For the teacher model, the parameter update for $\tilde{\theta}$ happens using the parameters of the student model, θ, by calculating the exponential moving average (EMA) as shown in Figure 3. This can be observed in Equation (1) [4]:

$$\tilde{\theta}_t = \alpha\tilde{\theta}_{t-1} + (1-\alpha)\theta_t \tag{1}$$

where
$\tilde{\theta}_t$ specifies the parent weights;
$\tilde{\theta}_{t-1}$ signifies the parent weights of the previous run;
θ_t specifies the child weights; and
α specifies the smoothing parameter.

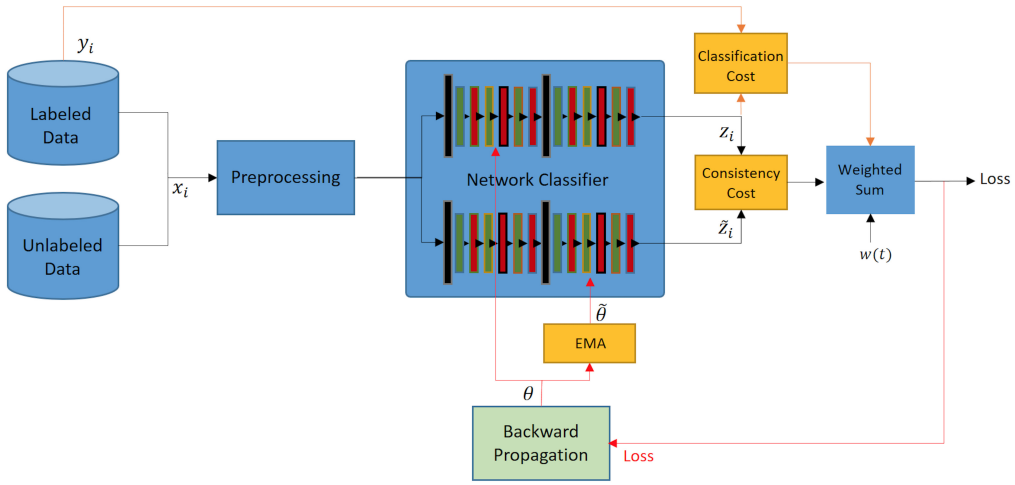

Figure 3. Mean Teacher model structure representation.

It has been observed that averaging the model weights over various training steps is more likely to provide a more accurate model compared to utilizing the final weights promptly. In the case of the Mean Teacher model, the information can be aggregated after each step rather than every epoch. However, previous research studies have overlooked a significant metric, which is the difference in the program's run-time. It has been observed that the Mean Teacher model can be trained in much less time than the Π model, and it achieves much improved accuracy compared to Temporal Ensembling. However, there are some drawbacks of the Mean Teacher model:

- Both the supervised networks perform forward propagation using different parameters. However, the student model is the only one with backward propagation to compute the gradients.
- The Mean Teacher model utilizes much greater memory resources compared to the other two models since the dropout conditions (η and η') must be kept and preserved during each epoch.

3.4. MixMatch

Compared to the previous methods discussed, MixMatch [18] has a different approach and follows the PF-SSCP method. In every batch, this model augments the raw labeled data x_b and keeps the labels y_b, generating the dataset $\hat{X}$. For the unlabeled data, the raw

input x_b is augmented K times, where K is a hyper-parameter. Then, those K augmented entries are passed through the network classifier to predict all their labels ($\tilde{z}_{b2}, \tilde{z}_{b2}$ to $\tilde{z}_{bk}$), and after that, their average, $\tilde{z}_{bavg}$, is taken as a prediction for all the K entries. $\tilde{z}_{bavg}$ is sharpened using temperature sharpening to reduce the entropy of the label distribution. This can be carried out as follows:

$$\text{Sharpen}(p, T)_i := \frac{p_i^{\frac{1}{T}}}{\sum_{j=1}^{L} p_j^{\frac{1}{T}}} \tag{2}$$

where
p is some input categorical distribution;
T is a hyperparameter that needs to be tuned; and
L is the number of labeled classes.

As such, the output of the sharpening will produce unlabeled data with the pseudo-labels $\bar{z}_b$, denoted as $\hat{U}$. Concatenating and shuffling both $\hat{X}$ and $\hat{U}$, we obtain W. MixUp is conducted on $\hat{X}$ and the first $|X|$ entries of W to obtain X', where $|X|$ is the size of the labeled data in the batch. Additionally, MixUp is applied on the unlabeled data in the batch $\hat{U}$ with the rest of the entries W to obtain U'. To understand how MixUp works, we need to understand how (x', p') is computed using two examples with their corresponding labels and probabilities, $(x_1, p_1), (x_2, p_2)$, as shown below:

$$\lambda \sim \text{Beta}(\alpha, \alpha) \tag{3}$$

$$\lambda' = \max(\lambda, 1 - \lambda) \tag{4}$$

$$x' = \lambda x_1 + (1 + \lambda')x_2 \tag{5}$$

$$p' = \lambda p_1 + (1 + \lambda')p_2 \tag{6}$$

where α is the hyperparameter to tune. Due to $\lambda \geq 0.5$ and the max function, the first term x_1 and its label p_1 have more importance than the second point x_2 and its label p_2. This makes the MixUp model's prediction for X' and U' correspond to labeled and unlabeled output guesses, respectively. As we know the exact output of labeled data, the classification cost is basically the cross-entropy. On the other hand, for the consistency cost, since we are guessing the labels of the unlabeled data, the L_2 loss [20] is used. These losses can be shown in the equations below:

$$L_X = \frac{1}{|X'|} \sum_{x,p \in X'} H(p, p_{model}(y|x;\theta)) \tag{7}$$

$$L_U = \frac{1}{L|U'|} \sum_{x,p \in U'} ||\bar{z}_b - p_{model}(y|u;\theta))||_2^2 \tag{8}$$

$$L = L_X + \lambda_U L_U \tag{9}$$

3.5. ReMixMatch

Following the same principle as MixMatch, ReMixMatch [19] adopts the PF-SSCP method but introduces several modifications. One significant alteration is the augmentation anchoring method. As depicted in Figure 4, the labeled input x_b undergoes a robust augmentation process using a variant of AutoAugment known as CTAugment, which concurrently learns an augmentation policy during model training. This strongly augmented labeled data $\hat{x}_b$, together with the label y_b, constitutes the dataset $\hat{X}$. For the unlabeled input x_b, the model employs multiple strong augmentations K, resulting in $\hat{x}_{b,1..k}$. Using the same unlabeled raw input x_b, a weakly augmented version is generated by applying only a

crop or a flip, yielding $\hat{X}_{bw}$. This $\hat{X}_{bw}$ is then processed through the network classifier to obtain the prediction $\tilde{z}_{bw}$.

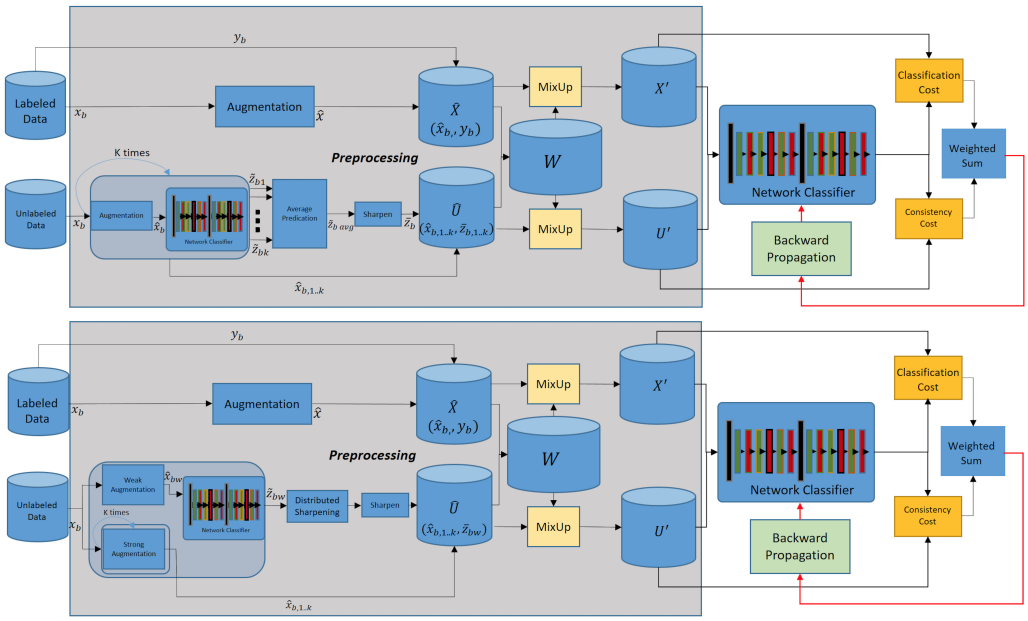

Figure 4. MixMatch (**Top**) and ReMixMatch (**Bottom**) model structure representation.

Another notable modification is the implementation of distribution alignment on $\tilde{z}_{bw}$. This adjustment involves maintaining a running average of the network classifier's predictions for the unlabeled data, denoted as $\tilde{p}(y)$, and estimating the marginal class distribution $p(y)$ for the labeled examples during training. The model then applies a ratio of $p(y)/\tilde{p}(y)$ to the network classifier's prediction of the label $z = p_{model}(y|u;\theta)$, followed by normalization using $\text{Normalize}(x)_i = x_i / \sum_j x_j$. Subsequently, temperature sharpening, as previously discussed in the context of MixMatch, is applied to produce the pseudo-labels $\bar{z}_{bw}$.

Utilizing both $\hat{x}_{b,1..k}$ and $\bar{z}_{bw}$, the model generates the unlabeled dataset $\hat{U}$. It is evident that the multiple strongly augmented versions of the unlabeled data are based on the weakly augmented variants. Beyond this stage, the procedure aligns precisely with that of MixMatch, adhering more generally to the PF-SSCP framework.

4. SSML Model Analysis Framework

As with many other ML algorithms, SSML algorithms maintain a similar structure, albeit with minor differences. Referring to Figure 5, we observe the network classifier-focused structure of previous SSML models, such as the Π model, Mean Teacher, and Temporal Ensembling. The preprocessing of labeled and unlabeled data is conducted, which are then injected into the network classifier without splitting, simultaneously. Throughout the preprocessing step, data quality is enhanced by suppressing undesirable distortions or enhancing certain image features essential for further processing. For instance, some of these operations could include filtering, geometric transformations, pixel brightness adjustments, or a combination thereof.

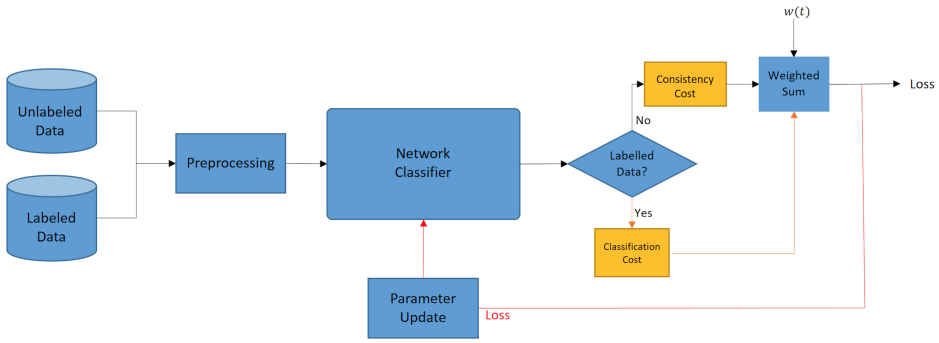

Figure 5. Network classifier-focused semi-supervised computation process (CF-SSCP).

Analyzing these SSML algorithms, the network classifier block comprises single or multiple instances of essentially the same DNN to execute the optimal data classification. Consequently, this unique network classifier design facilitates transformative parameter updates. This feature differentiates various SSML models. A network classifier can include the following types:

- One DNN: Here, a single DNN is employed in the classifier, conducting both backward and forward propagations.
- Two DNNs: This configuration utilizes two nearly identical DNNs, differing slightly in their dropout conditions and parameter initialization. Forward propagation occurs in both DNNs; however, backward propagation might not occur in one of these DNNs in some designs.
- Three or more DNNs: Here, three or more DNNs are employed, potentially with varying initialization conditions. Additionally, dropout conditions may or may not be integrated into these DNNs.

Depending on the total number of DNNs, the network classifier's output integrates one or more classifications for computing the loss. Based on the data type input, there are two distinct cases for loss computation:

- Unlabeled data: For unlabeled data, the consistency cost among two or more DNN instances is calculated, multiplied by the number of classes and the inverse of the batch size.
- Labeled data: In this scenario, the consistency cost among two or more DNN instances and the classification cost between one or more models are computed.

It is noteworthy that the classification cost is calculated only for labeled data. For unlabeled data, the classification cost is not computed and is set to zero.

Upon assessing the most recent state-of-the-art MixMatch and ReMixMatch, we noticed a deviation in the structure of the previous semi-supervised computation process. These alterations are depicted in Figure 6.

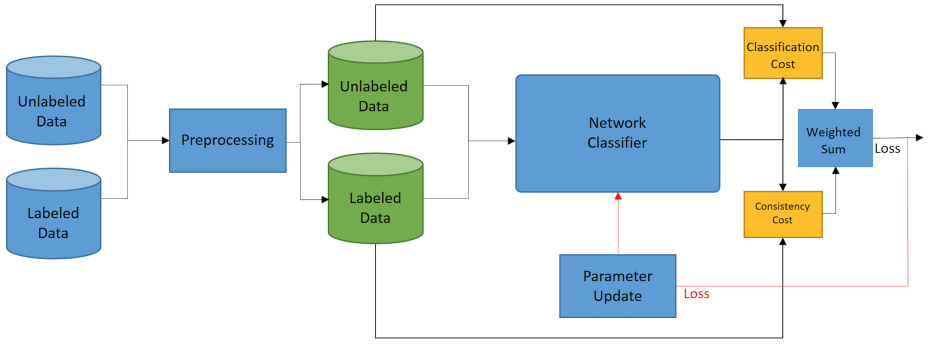

Figure 6. Preprocessing-focused semi-supervised computation process (PF-SSCP).

As illustrated, preprocessing is applied to both unlabeled and labeled datasets, subsequently outputting postprocessed unlabeled and labeled datasets. The preprocessing step generates pseudo-labels for the unlabeled datasets, effectively rendering both datasets labeled [21]. Regarding the network classifier, in most designs, it is simplified to one DNN; however, there is potential for further exploration. The network classifier's predictions are utilized for calculating the classification cost and the consistency cost. Examining each SSCP, the primary distinctions between PF-SSCP and CF-SSCP are as follows:

- The design's emphasis is placed on preprocessing steps rather than the network classifier.
- The classification cost is consistently calculated due to the pseudo-labels on the unlabeled datasets.

While these established differences might seem minor, when evaluating these distinctions and conducting experiments on these SSML algorithms, the following research question was prompted:

RQ1: How do PF-SSCP compare to CF-SSCP in parameter updates in network classifiers and preprocessing techniques, respectively, measured in training time, learning accuracy, and training loss?

Furthermore, the consistency cost (also known as the unsupervised loss) can be computed using the classification outputs of one or more DNN instances, denoted as z_i and $\tilde{z}_i$. Moreover, in instances where only one DNN is employed, $\tilde{z}_i$ represents the evaluations of the previous network, not a second network's evaluation. Generally, the consistency cost is formulated as follows [4,22]:

$$\begin{aligned} \text{Consistency Cost} &= w(t) * \text{ Squared Difference} \\ &= w(t) * \frac{1}{C|B|} \sum_{i \in B} \|z_i - \tilde{z}_i\|^2 \end{aligned} \tag{10}$$

where
z_i represents $f_\theta(g(x_{i \in B}))$ and θ denotes the function's training parameter.
$\tilde{z}_i$ denotes $f_{\theta'}(g(x_{i\ inB}))$, and θ' represents the function's training parameter.
$|B|$ is the batch size.
$w(t)$ signifies the time-dependent weighting function (discussed subsequently).
C represents the number of classes, with $y_i \in \{1 \dots C\}$ (e.g., cat, tree, dog, car, etc., for CIFAR-10).

When data are labeled, z_i is used in conjunction with the label y_i to compute the classification cost (or the supervised loss). The supervised loss, or classification cost, is essentially the log of the product of classification and the corresponding label. For mini-batches of size B, this is the inverse of the mini-batch size multiplied by the sum of the losses' negative values, as shown in Equation (11) [4]:

$$\text{Classification Loss } = -\frac{1}{|B|} \sum \log z_i[y_i] \tag{11}$$

Combining both losses provides the total loss used for updating the network's parameters. In light of these computations and the observed variations in SSCP designs, the following research question emerges:

$$\begin{aligned} \text{Loss } &= \text{Classification Cost} + \text{Consistency Cost} \\ &= \text{ Cross Entropy } + w(t) * \text{ Squared Difference} \\ &= -\frac{1}{|B|} \sum_{i \in (B \cap L)} y_i * \log z_i + w(t) * \frac{1}{C|B|} \sum_{i \in B} \|z_i - \tilde{z}_i\|^2 \end{aligned} \tag{12}$$

where

z_i specifies the $f_\theta(g(x_{i \in B}))$ and θ specifies the function training parameter.

$|B|$ represents the batch size.
L includes the labeled input indices.
y_i is the label for input x_i.
$w(t)$ specifies the time-dependent weighting function (discussed later).
C specifies the number of different classes $y_i \in \{1 \dots C\}$ (cat, tree, dogs, car, etc..)

As the loss has been decomposed, and it highly depends on the labeled/unlabeled data ratio, it raised another subquestion during our experiments:

RQ2: Given the consistency and classification costs' computation methods in PF-SSCP and CF-SSCP, how do these costs impact the effectiveness of the semi-supervised learning models in terms of accuracy and loss during the training and inference phases?

The research questions RQ1 and RQ2 delve deep into the comparative analysis of the PF-SSCP and CF-SSCP methods, focusing primarily on their structural differences and the implications these hold on the effectiveness of semi-supervised machine learning models.

5. Empirical Evaluation Method

Before diving into the experiments and assigning values to specific hyperparameters, it is important to discuss the methodology used to address the research questions. The adoption of various techniques for applying XAI methods [3,23] to analyze trained models has steered XAI towards post-hoc analysis. It is essential to remember that the complexity of a machine learning model is often inversely related to its interpretability [24]. Generally speaking, the more complicated and unconstrained a model is, the more difficult it is to interpret or explain clearly [25]. Thus, integrating CNNs within SSML algorithms results in non-monotonic and non-linear response outputs, contributing to the creation of models that are among the least interpretable. In this context, our methodology is model-specific, concentrating on global measures that will enable us to comprehend fully the inputs and their complex modeled correlations with the output predictions.

5.1. Datasets

In this section, we present the dataset utilized for training semi-supervised machine learning (SSML) classifiers. The CIFAR-10 dataset was selected, comprising 60,000 distinct 32×32 color images evenly distributed across ten different classes, with each class containing 6000 images. From this dataset, we allocated 10,000 images for the testing process and the remaining 50,000 images for the training phase.

In terms of the dataset, CIFAR-10 was selected for its simplicity and its common use in benchmarking the chosen state-of-the-art SSML classifiers. We reduced the labeled data in various ratios primarily to address RQ2 while also evaluating the comparison of PF-SSCP and CF-SSCP on aspects such as training time, training loss, and learning accuracy, thereby contributing to the resolution of RQ1. Given our objective of testing the SSML algorithms with varying quantities of labeled data, we intentionally withheld some of the labels during the training of these SSML algorithms, as detailed in Table 1.

Table 1. CIFAR-10 Training Datasets.

Labeled Data	Unlabeled Data	Labeled Data per Class	Unlabeled Data per Class
1000	49,000	100	4900
4000	46,000	400	4600
50,000	0	5000	0

As depicted in Table 1, both labeled and unlabeled data are proportionately distributed across the ten classes, indicating an impartial approach with no class preference. The substantial gap between the 4000 and the 50,000 labeled datasets serves as an ideal measure for assessing the efficacy of the unsupervised components relative to the supervised elements

within various SSML algorithms. In the case of the 1000 labeled datasets, the SSML algorithms undergo rigorous testing, primarily relying on the unsupervised components of the network, thereby approaching the realm of unsupervised machine learning algorithms [26].

5.2. Preprocessing

The preprocessing step primarily hinges on the specific SSML approach and the dataset in use. For the purposes of this paper, preprocessing is tailored to either the PF-SSCP or the CF-SSCP methodologies. In CF-SSCP, we execute zero component analysis (ZCA) [21], a step adopted by both the Mean Teacher and Π models in their processing of the CIFAR-10 datasets, which contributes to enhanced accuracy and diminished loss. Conversely, preprocessing in the context of PF-SSCP, as demonstrated in strategies such as MixMatch and ReMixMatch, adopts a more intricate approach previously elaborated upon in Sections 3.4 and 3.5. This method capitalizes on the network classifier's capability to generate pseudo-labels, negating dependence on randomized algorithms. Additionally, a crucial element in both preprocessing-focused and classifier-focused SSCP is data augmentation, a strategy that significantly betters results by enriching the training dataset. We executed various data augmentation techniques, including image flips, zooms, shifts, and cropping.

5.3. SSML and DNN Combination

The choice of SSML model dictates the nature of integration to be employed, a necessity given the distinct behaviors exhibited by different architectural choices [27]. This combination evaluates five critical aspects:

- Performance, gauged through accuracy and loss metrics.
- The duration of the training process.
- Constraints imposed by hardware.
- The initial selection of hyperparameters.
- The appropriateness of the loss function.

Both PF-SSCP and CF-SSCP are guided by these metrics, though the degree of influence each one holds varies. This variance is attributable to specific design features outlined in the preceding section for each SSML type and their respective behaviors. In the context of CF-SSCP, most DNNs utilize dropout regularization, a technique instrumental in preventing network overfitting. Conversely, PF-SSCP does not mandate overfitting prevention measures within the DNNs.

5.4. Ramp-Up and Ramp-Down Functions

In CF-SSCP, a ramp-up period was initiated with 40,000 training steps at the onset of the training process. During this phase, both the learning rate and the consistency cost parameters were progressively increased from zero to their peak values using a sigmoid-shaped function. Conversely, for PF-SSCP, a linear ramp-up was employed, escalating from an initial value of 100 to the maximum over the initial 16,000 training steps.

5.5. SSML Performance Measurements

This segment details the approach adopted to assess the classification efficacy of the SSML networks, with a specific focus on the variety of network classifiers involved in the evaluation. As indicated earlier, our experimental analysis was confined to the CIFAR-10 dataset. Per the discussion in Section 4, any SSML algorithm can be abstracted into two distinct types of SSML architectures, as illustrated in Figure 5, by employing diverse network classifiers. In this study, we restricted our consideration to contemporary CNN models serving as the network classifiers in SSML networks. For the training and evaluation phases of the proposed systems, we utilized a GeForce GTX 2080 Ti GPU. This specific GPU was chosen for its proficiency in handling an array of SSML algorithm and network classifier combinations [28].

It is pertinent to mention that in CF-SSCP, for a streamlined analysis, we amalgamated the preprocessing stage with the feature selection process, as both these procedures precondition the datasets prior to their introduction to the network classifiers. In contrast, PF-SSCP placed a greater emphasis on the preprocessing stages as well as the generation of pseudo-labels, owing to the inherent design of PF-SSCP, to more significantly affect the dataset rather than the network classifier.

The ensuing performance metrics were established as the foundation for the appraisal of the proposed models:

- Performance accuracy and training loss, evaluated against varying proportions of unlabeled to labeled datasets.
- The duration of training necessitated by each amalgamation of SSML with diverse network classifiers, inclusive of the time expended during the preprocessing and feature selection stages.
- Utilization of parameters and the complexity intrinsic to various SSML algorithms, with an exposition on the distinct parameters requisite in each scenario and their subsequent influence on complexity.

5.6. Experimental Design and Framework Specifications

In order to rigorously evaluate the performance of semi-supervised learning models, our experiments incorporate two principal computational frameworks: the CF-SSCP and the PF-SSCP frameworks, as outlined in Table 2. These frameworks provide the basis for comparing models by assessing the impact of classifier complexity and preprocessing sophistication. The table showcases the relevant experiments conducted, presenting a clear framework for analysis, while a comprehensive list of experimental hyperparameters and additional experimental details are reserved for the Appendix A section of this paper.

Table 2. Experiments for different semi-supervised models under CF-SSCP and PF-SSCP frameworks.

Framework	Model	Experiment and Architecture	Summarized Purpose
CF-SSCP	Temporal Ensembling (TE)	Exp 1: TE and Shake-Shake26	Ensemble predictions over time for stability and consistency
		Exp 2: TE and DenseNet-121	Utilizing ensemble learning with a focus on network depth
		Exp 3: TE and WRN-40-2	Applying ensembles to widen and deepen network architectures
		Exp 4: TE and WRN-28-10	Ensemble methods combined with a wider network model
	Π Model	Exp 5: Π model and Shake-Shake26	Ensuring consistent network predictions without skip connections
		Exp 6: Π model and DenseNet-121	Consistency of prediction with depth-oriented network architectures
		Exp 7: Π model and WRN-40-2	Deeper networks under consistency constraints
		Exp 8: Π model and WRN-28-10	Wider networks maintaining prediction consistency
	Mean Teacher (MT)	Exp 9: MT and Shake-Shake26	Teacher–student model consistency without skip connections
		Exp 10: MT and DenseNet-121	Depth and skip connections in a teacher–student setup
		Exp 11: MT and WRN-40-2	Deeper architecture in a mean teacher framework
		Exp 12: MT and WRN-28-10	Enhanced width in the teacher–student model's architecture
PF-SSCP	MixMatch (MM)	Exp 13: MM and Shake-Shake26	Augmentation and mixing strategies for semi-supervised learning
		Exp 14: MM and DenseNet-121	Deep architecture applied to advanced mix-and-match techniques
		Exp 15: MM and WRN-40-2	Widening and deepening networks with semi-supervised mix–matching
		Exp 16: MM and WRN-28-10	Wide network structures in advanced mix–match learning scenarios
	ReMixMatch (RM)	Exp 17: RM and Shake-Shake26	Refinement of mix–match techniques with a preprocessing focus
		Exp 18: RM and DenseNet-121	Application of preprocessing strategies in deep learning models
		Exp 19: RM and WRN-40-2	Preprocessing alignment in wider and deeper network structures
		Exp 20: RM and WRN-28-10	Extensive preprocessing in a widened network scenario

The selection of architectures for our experiments was deliberate to encompass a broad range of complexities and capacities pertinent to semi-supervised learning:

- **DenseNet-121 (Dense Convolutional Network—121 layers):** Known for its dense connectivity, DenseNet-121 optimizes parameter efficiency and facilitates feature propagation and reuse. Its design is particularly beneficial for learning with limited labeled data, which is a common challenge in semi-supervised learning scenarios [29].
- **Shake-Shake Regularization Model (Shake-Shake26):** The Shake-Shake regularization approach, exemplified by the Shake-Shake26 model, introduces stochasticity [30]

into the training process. This method has been shown to enhance generalization on image classification tasks, presenting a unique advantage in semi-supervised learning frameworks [31].

- **Wide Residual Networks (WRN-40-2):** The WRN-40-2 architecture augments the network's width, offering an optimal trade-off between depth and width. This expanded network capacity allows it to represent more complex functions and data relationships, benefiting from the additional unlabeled data in semi-supervised learning setups [8].
- **WRN-28-10:** The WRN-28-10 extends the width of traditional residual networks even further, targeting the rigorous demands of high-complexity classification tasks. The architecture is designed to capitalize on the unlabeled data that are more prevalent in semi-supervised learning contexts [8].

The adoption of these architectures in our experiments allows us to thoroughly assess the efficacy of semi-supervised learning strategies across varying levels of network complexity and depth.

6. Evaluation Results and Analysis

6.1. CF-SSCP with Various Network Classifiers

Given CF-SSCP's substantial reliance on the network classifier, it becomes pertinent to investigate its performance dynamics across different network classifiers. Accordingly, our tests for CF-SSCP were conducted using both single (Mean Teacher and Temporal Ensembling) and double (Π model) network classifiers. The ensuing subsections detail our findings.

6.1.1. Results of Temporal Ensembling

The Temporal Ensembling method, utilizing a singular DNN within the network classifier, stands as the most straightforward among the three SSML algorithms. As evidenced in Figures 7 and 8, it is apparent that, under Temporal Ensembling, the efficacy of both WRN-40-2 and DenseNet-121 is overshadowed by the more robust Shake-Shake26 model.

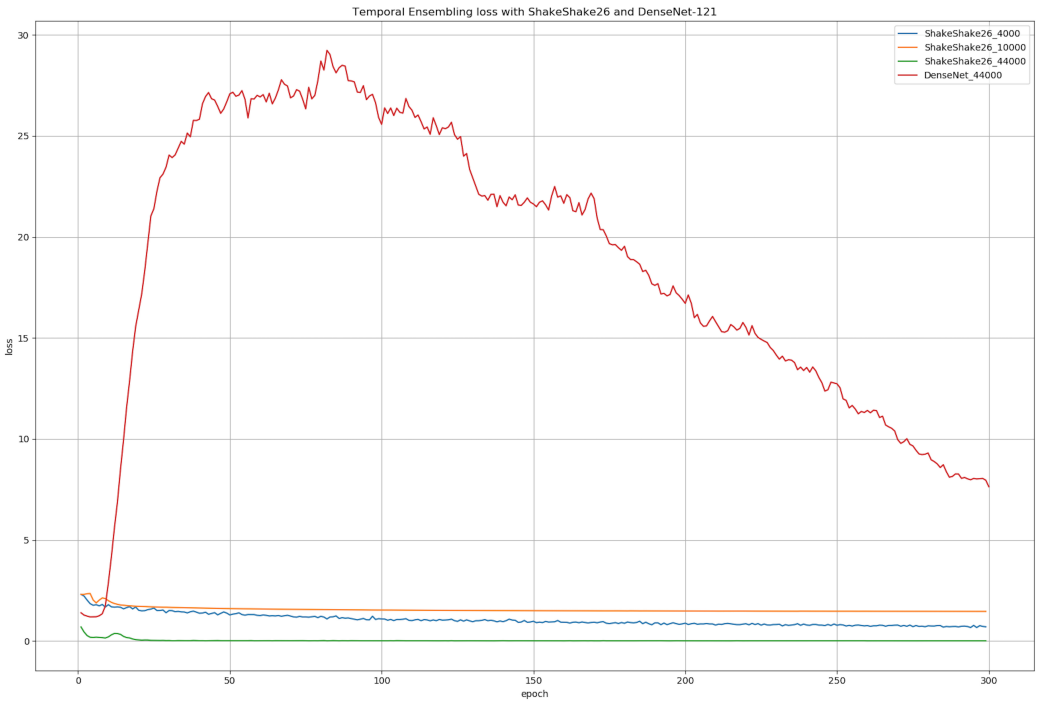

Figure 7. Loss comparison for Temporal Ensembling: DenseNet-121 vs. Shake-Shake26.

Figure 8. Loss comparison for Temporal Ensembling: WRN-40-2 vs. Shake-Shake26.

Inspection of the accuracy data in Figure 9 reveals that training accuracy marginally surpasses test accuracy, with the highest accuracy observed at 4000 labels.

Figure 9. Training and testing accuracy for Temporal Ensembling with Shake-Shake26.

However, as Figure 10 demonstrates, the training loss for Shake-Shake26 within the Temporal Ensembling framework fails to reach optimal performance even with extensive hyperparameter tuning. This outcome likely stems from the model's simplicity, concentrating solely on the training loss of a single DNN. Thus, exploration into the alternative models, namely the Π model and Mean Teacher model, is warranted.

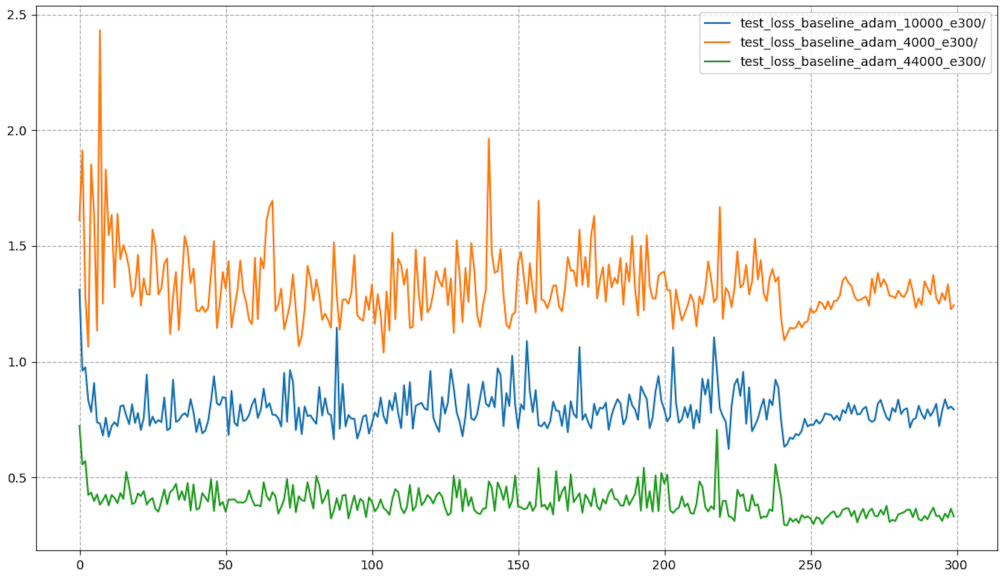

Figure 10. Training loss for Temporal Ensembling with Shake-Shake26.

Summary:

In all three experiments, it is evident that Temporal Ensembling experiences the highest loss compared to the other two models, the Π model and the Mean Teacher model. Analyzing the efficacy of various DNNs within the network classifier of Temporal Ensembling reveals that Shake-Shake26 incurs the least loss compared to the other two DNNs, namely, DenseNet and WideResNet. This suggests that a DNN with fewer skip connections can yield more accurate results and lower training loss. Additionally, employing two DNNs in a network classifier appears to facilitate better parameter updates during the computation of the consistency cost.

6.1.2. Results of Π Model and Mean Teacher

We conducted training and validation for the Π model and Mean Teacher model utilizing WideResNet as the primary network. This setup was designed to exhibit the performance metrics of each model by leveraging the CIFAR-10 dataset with varying percentages of labeled data.

- **WideResNet Core Network with Mean Teacher**: Figure 11 depicts the accuracy achieved by the Mean Teacher model using WideResNet as our core network. The graph indicates that the highest validation accuracy was achieved using the Stochastic Gradient Descent (SGD) [32] algorithm with 44,000 labels, the maximum in these experiments. Notably, accuracy diminishes as the label count decreases. This trend was anticipated since accuracy correlates with the total label count, assuming unchanged hyperparameters.
  However, the primary observation here concerns the discrepancy between the SGD and Adam optimizers. Clearly, the SGD optimizer [33] surpassed Adam in terms of implementation accuracy.
  Examining the loss in Figure 12, the Adam optimizer performs superiorly, with less test loss. Another notable aspect is both networks' initial struggle to smoothly reduce loss, resulting in a highly noisy transition of the loss function. This is attributed to the unsuitability of maintaining consistent hyperparameter values across different volumes of labeled data.
  Another critical point pertains to initialization. Different initialization parameters are crucial, evident from the 10,000-label dataset starting significantly lower than the 44,000-label set, causing the latter higher initial loss and hindering performance.

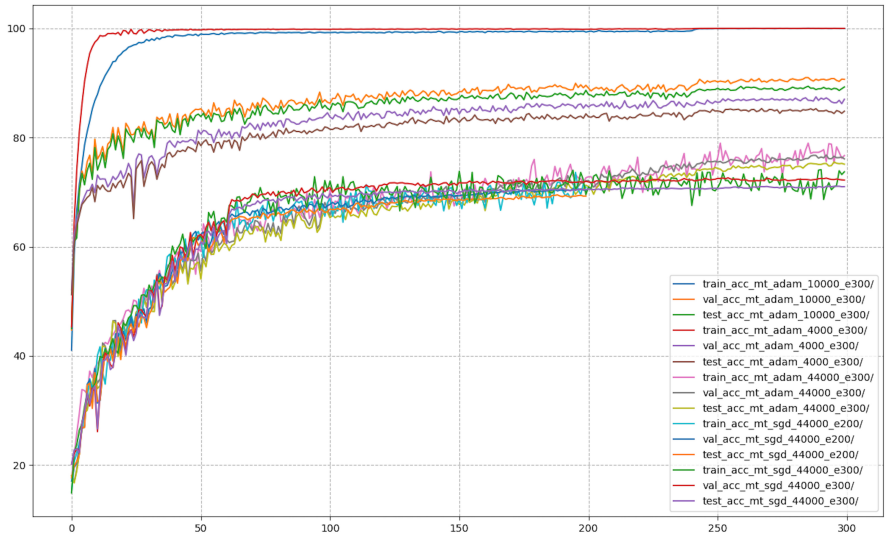

Figure 11. Mean Teacher accuracy with WideResNet across all label quantities.

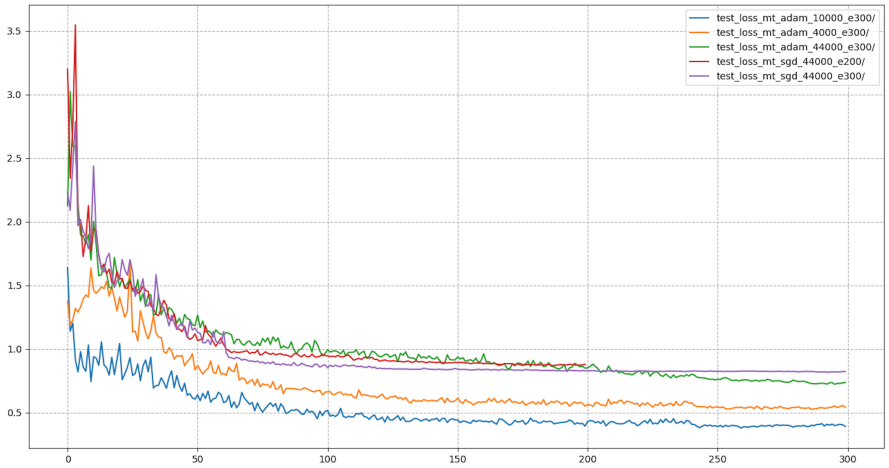

Figure 12. Mean Teacher loss with WideResNet across all label quantities.

- **WideResNet core network with Π model**: For the Adam optimizer, accuracy does not directly correlate with the number of labels, as distinctly seen in Figures 11 and 13.

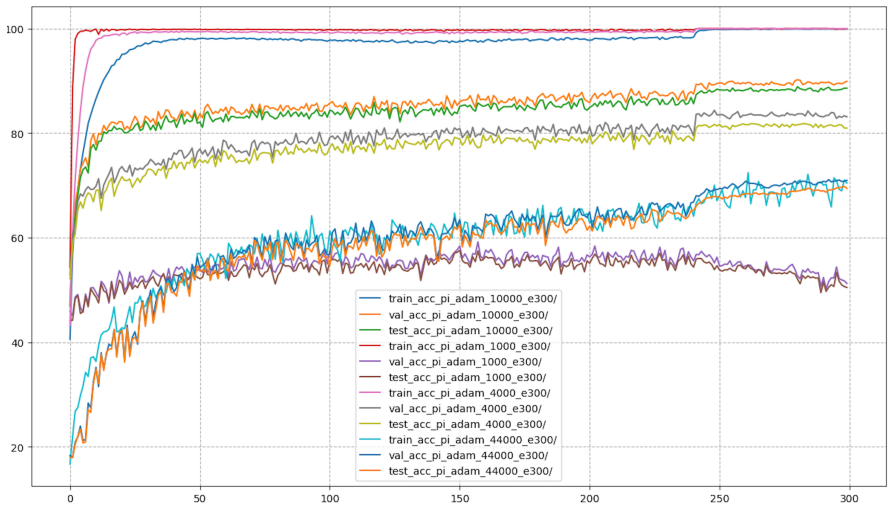

Figure 13. Π model accuracy with WideResNet across all label quantities.

Here, high training accuracy is noted with 1000, 4000, and 44,000 labels. Among these, the 1000-label dataset provides the highest accuracy. However, the 10,000-label dataset's training accuracy is 20% lower than the others, yet its testing accuracy, at 88.56%, surpasses the rest. Additionally, the 1000-label set displays an improvement in accuracy up to 58% before a decline, suggesting overfitting, as evidenced in Figure 14.

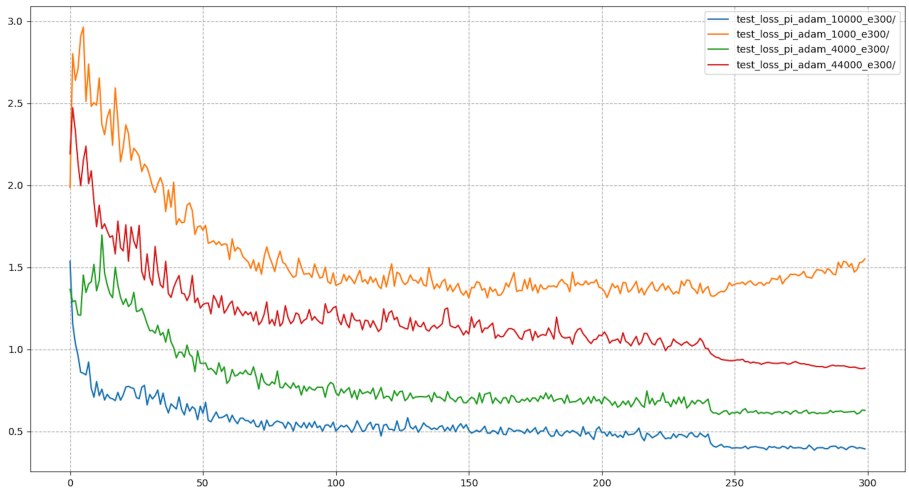

Figure 14. Π model loss with WideResNet across all label quantities.

Comparing this network with different label quantities, the loss is significantly lower at 10,000 labels, a result of hyperparameter tuning and network initialization. Even with 44,000 labels, the network struggles to reduce loss due to an unfavorable start. Moreover, the network's commencement varies across datasets, except for the 10,000-label set, where loss increases from 0 to 30 epochs. Significant fluctuations are particularly noticeable for the 1000-label data. These issues could be addressed through hyperparameter tuning using strategies such as Random Search or Grid Search.

- **Comparison of Mean Teacher with Π model using WideResNet as the core network**: Figure 15 compares the testing and validation accuracy of both networks, demonstrating the Mean Teacher model's superiority over the Π model with 4000-label datasets.

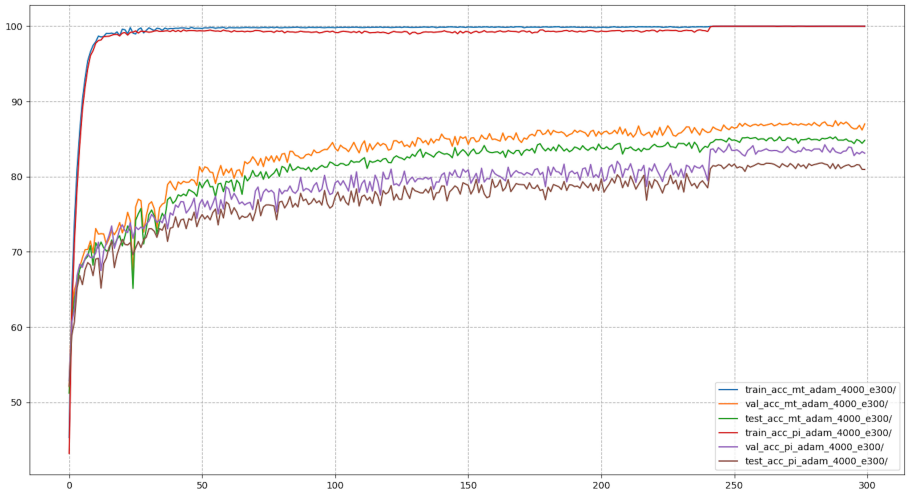

Figure 15. Mean Teacher vs. Π model in WideResNet accuracy at 4000 labels.

Furthermore, as shown in Figure 16, the Mean Teacher model excels in terms of loss, even with suboptimal hyperparameter tuning, highlighting the efficacy of the network design with the same core network and identical hyperparameters.

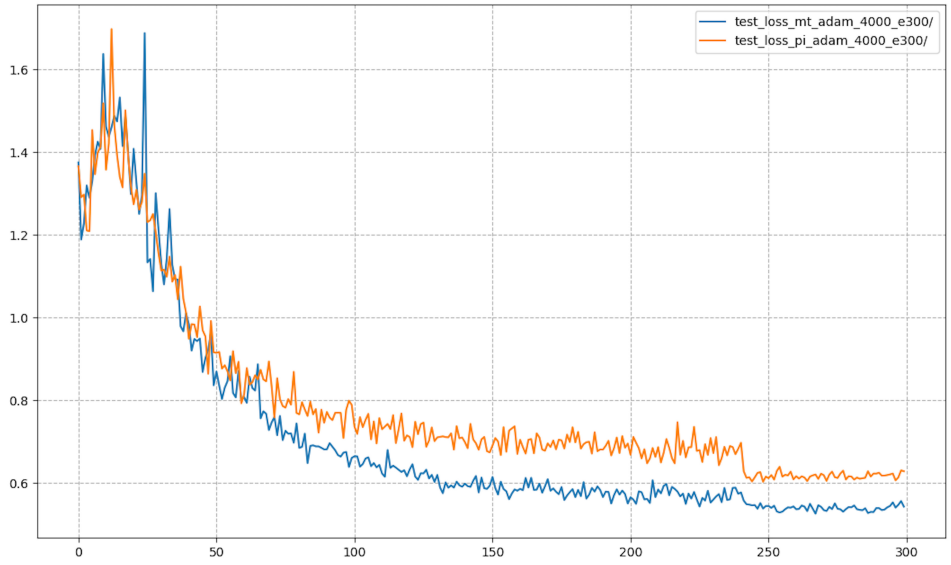

Figure 16. Mean Teacher vs. Π model in WideResNet loss at 4000 labels.

- **Mean Teacher with DenseNet and Shake-Shake26**: We deemed it essential to test the state-of-the-art SSML Mean Teacher model with a core network other than Shake-Shake26, hence the choice of DenseNet-121. Figure 17 shows that the training loss for both student networks is predictably lower, as the teacher network guides them using the EMA formula. These outcomes suggest that the teacher's parameters are optimized for the network's best overall performance. Moreover, the loss from the teacher represents the most optimal value attainable.

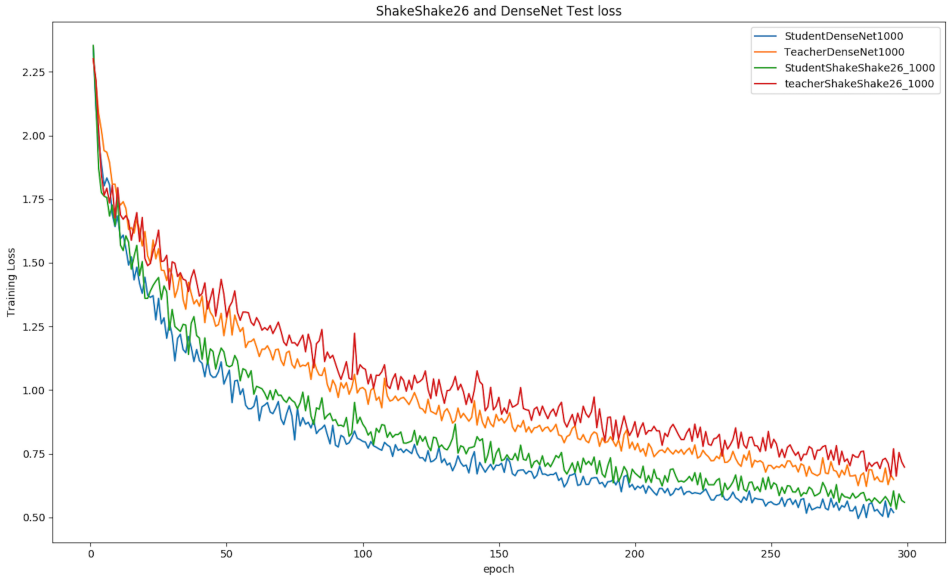

Figure 17. Student and teacher test loss in Mean Teacher for DenseNet and Shake-Shake26 at 1000 labels.

Additionally, the Shake-Shake26 network significantly outperforms DenseNet-121. As seen in Figure 18, this superiority is also reflected in accuracy performance. The primary reason is Shake-Shake26's broader and deeper network compared to DenseNet-121. However, it requires more computational time for training and hyperparameter tuning. System designers must consider this computational overhead, particularly when prioritizing performance over flexibility across datasets.

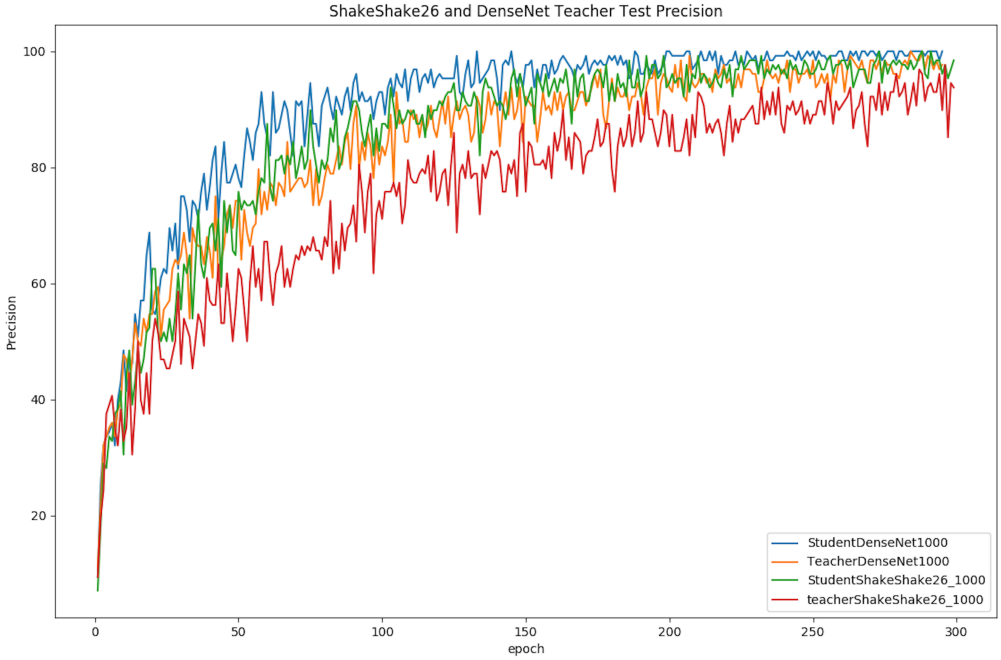

Figure 18. Student and teacher test accuracy in Mean Teacher for DenseNet and Shake-Shake26 at 1000 labels.

Figure 19 provides a few intriguing observations about the relationship between the teacher and the student models. First, with data with 4000 labels, the Shake-Shake26 network training process experiences a huge surge in the loss in the student nework. As a result, the teacher's network progress is also impacted. Secondly, the accuracy performance of a particular neural network is different for different datasets with various percentages of the labeled dataset. Therefore, it implies that the neural network design must be modified to incorporate dataset variations. Moreover, the hyperparameter tuning in the EMA can render better assumptions that would help in certain scenarios.

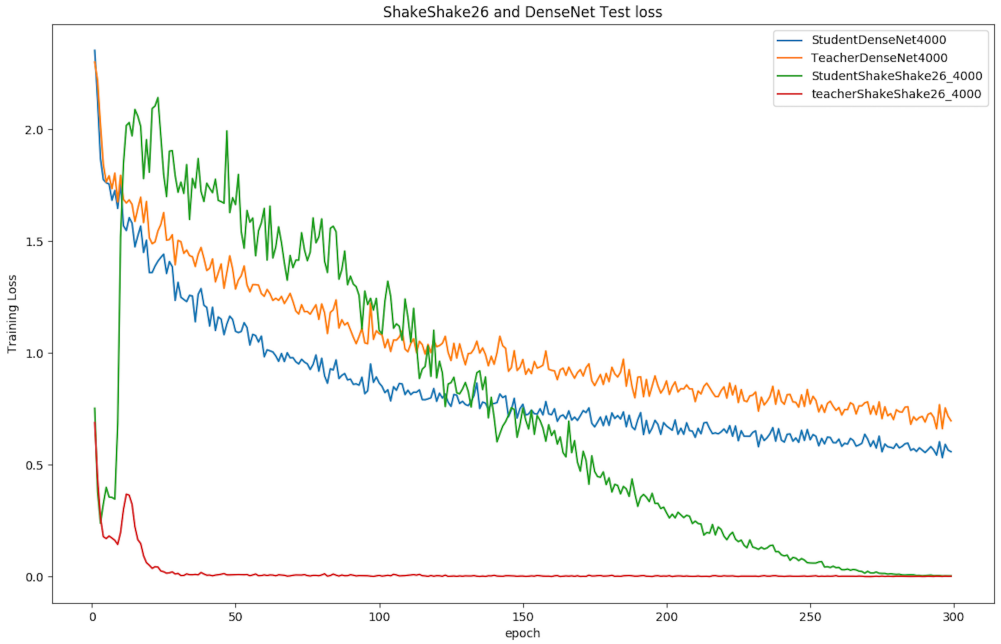

Figure 19. Shake-Shake26 vs. DenseNet-121 in Π model loss at 4000 labels.

Summary:

The incorporation of dual DNNs within a network classifier has shown to confer improved accuracy and reduced testing loss, a likely consequence of the combined classification prowess during forward propagation. This dual-network setup also injects additional variability into the calculation of the consistency cost, which in turn influences the total loss, with unlabeled data contributing to this effect. A comparative assessment of semi-supervised machine learning (SSML) models indicates that the Mean Teacher model surpasses the Π model in terms of accuracy and achieves a lower test loss.

When examining the performance across different SSML algorithms using Shake-Shake26, this model consistently presented a reduced loss when pitted against the DenseNet and WideResNet models, which may be attributable to its network width.

6.2. PF-SSCP with Different Network Classifiers

Understanding how PF-SSCP compares to CF-SSCP is crucial, but it is also vital to discern how the network classifier impactsPF-SSCP. We conducted tests using different network classifiers within both the MixMatch and ReMixMatch frameworks. The subsections below detail our analysis.

Results of MixMatch and ReMixMatch Analysis

- **MixMatch Compared to ReMixMatch with WideResNet as a Core Network**: Given PF-SSCP's emphasis on performance with the increased use of unlabeled data, we compared the outcomes from both frameworks using the same WideResNet-28-2, which has a depth of 28, a width of 2, and incorporates batch normalization [34]. As depicted in Figure 20, ReMixMatch slightly outperforms MixMatch with 4000 labels. This discrepancy widens with fewer labeled data.

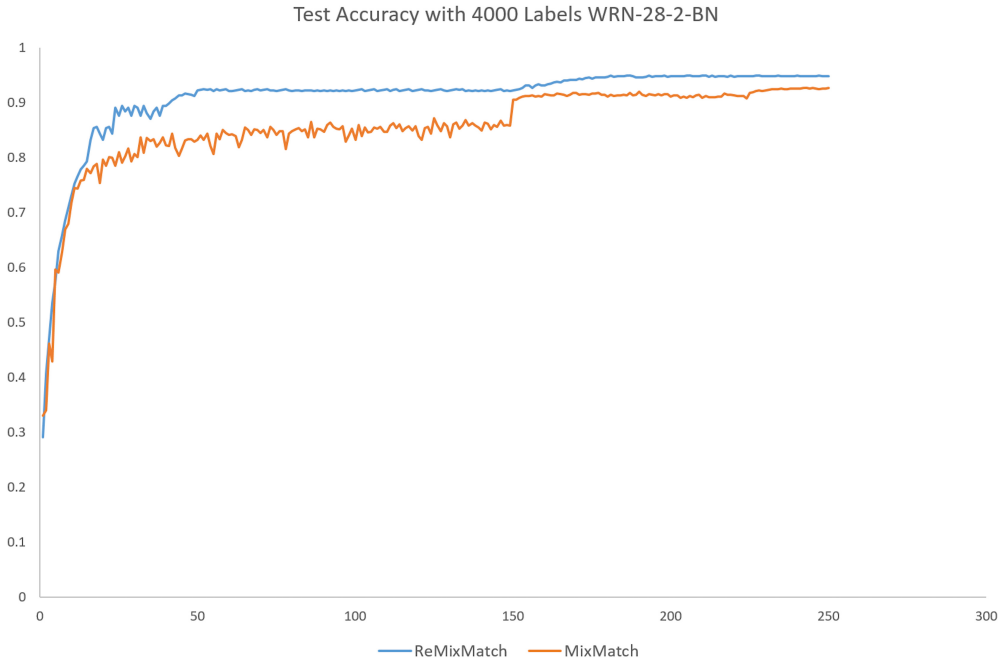

Figure 20. MixMatch vs. ReMixMatch at 4000 labels with WideResNet-28-2.

With 1000 labels, as shown in Figure 21, the performance gap becomes more pronounced. An intriguing observation is the convergence of the two semi-supervised machine learning models (SSMLs) within a limited number of epochs. ReMixMatch achieves higher initial accuracy, indicating its strength with fewer labeled data. However, overall performance differences are marginal, as MixMatch eventually catches up by the end of training.

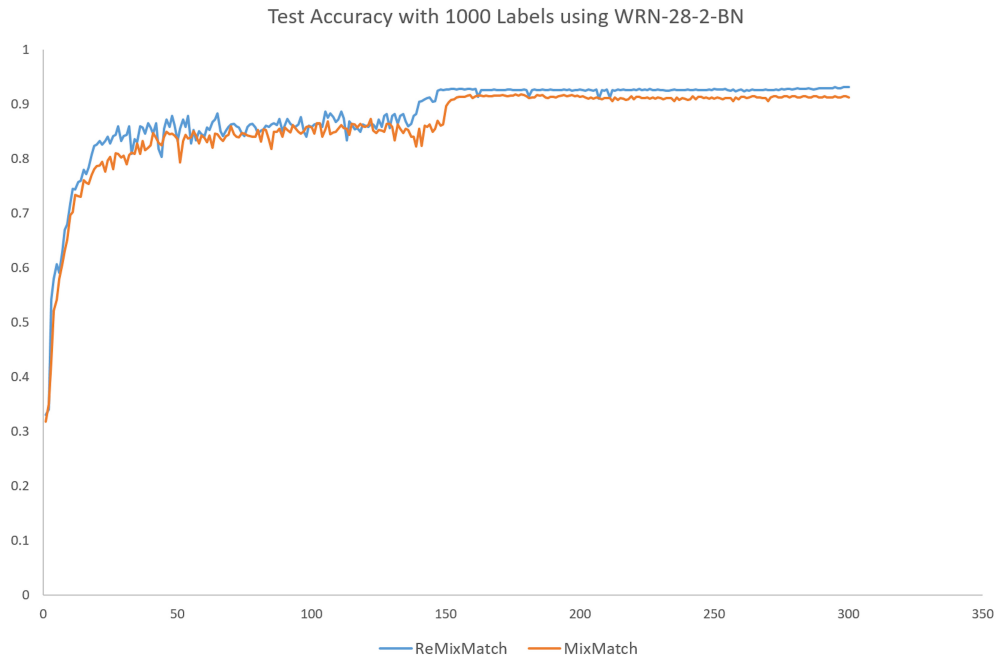

Figure 21. MixMatch vs. ReMixMatch at 1000 labels with WideResNet-28-2.

- **MixMatch and ReMixMatch with Shake-Shake26 as a core network**: Employing different core networks allows us to visualize the performance contributions of the SSMLs, independent of the underlying networks. By switching to Shake-Shake26, we sought to discern any performance variations. A close examination with 1000 labels, as seen in Figure 22, reveals an insignificant difference compared to WideResNet-28-2. Shake-Shake26 converges more rapidly, with marginally better accuracy, but this is not as pronounced when these Deep Neural Networks (DNNs) are used independently.

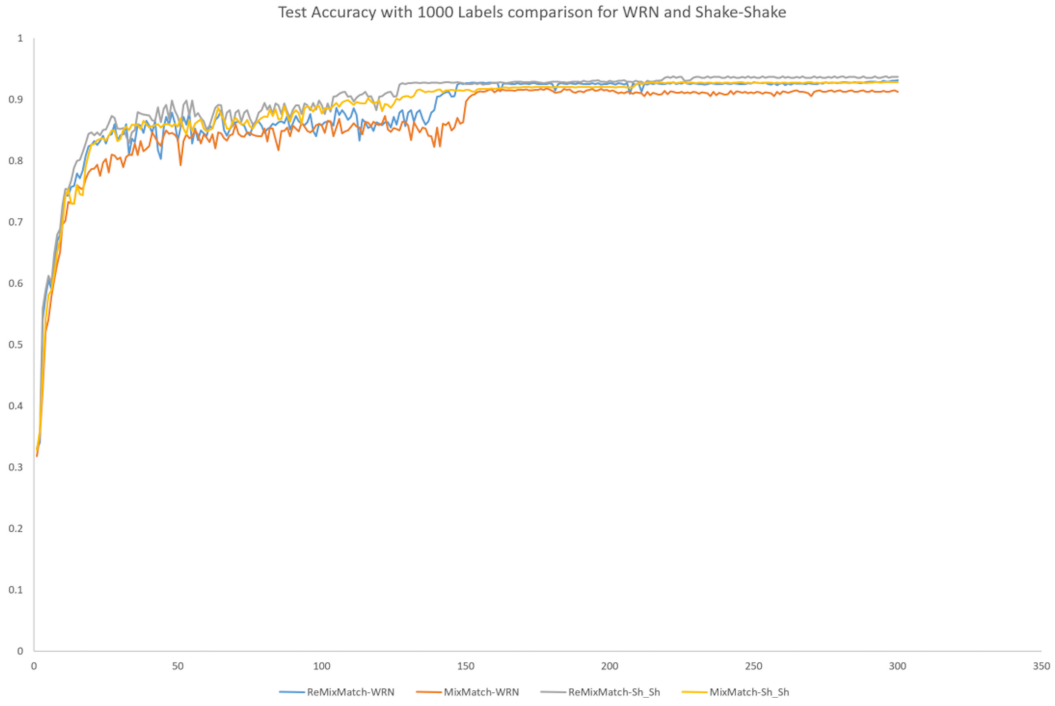

Figure 22. MixMatch and ReMixMatch at 1000 labels with WideResNet-28-2 and Shake-Shake26.

In Figure 23, with 4000 labels, ReMixMatch paired with Shake-Shake26 shows the highest accuracy. Notably, ReMixMatch with WideResNet and MixMatch with Shake-Shake26 are closely matched. The accuracy of ReMixMatch with WideResNet is slightly higher, but only by a narrow margin. Another key observation is the divergence in convergence patterns between MixMatch with Shake-Shake26 and MixMatch with WideResNet, as confirmed through five consecutive tests to ensure the finding's accuracy.

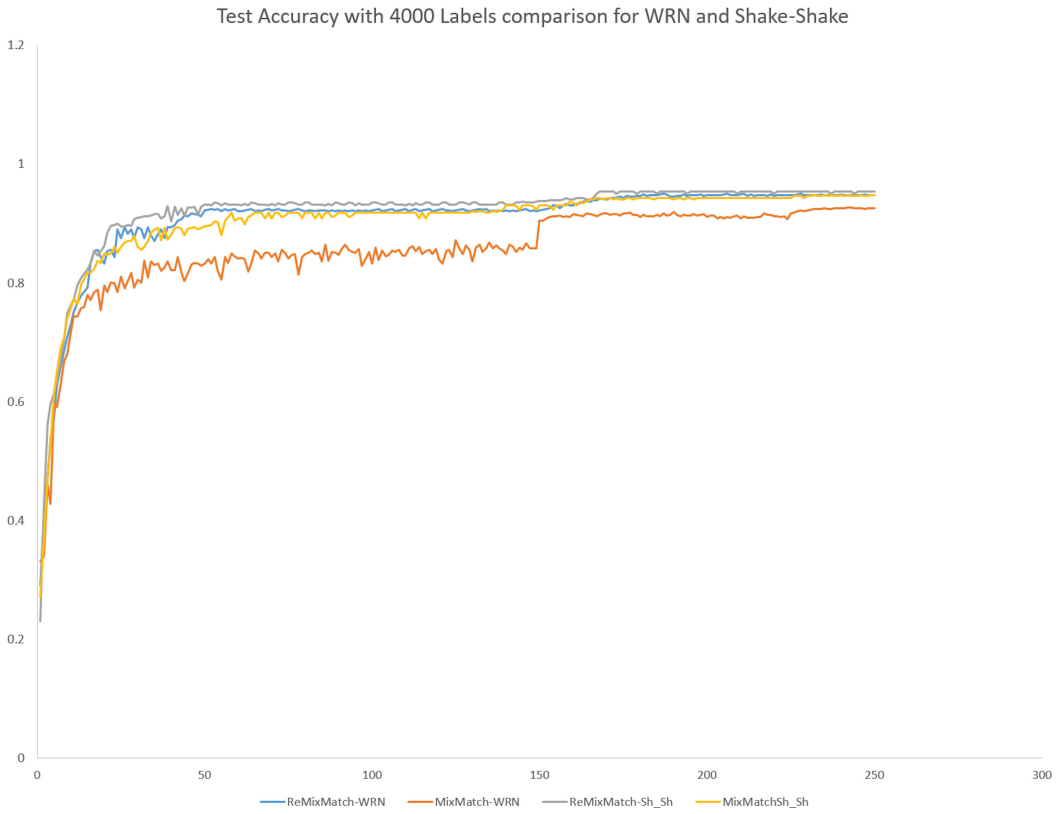

Figure 23. MixMatch and ReMixMatch at 4000 labels with WideResNet-28-2 and Shake-Shake26.

Summary:

A comparative review of the experimental data reveals that MixMatch and ReMixMatch algorithms yield higher accuracy improvements over the Mean Teacher, Π model, and Temporal Ensembling methods, underscoring the significance of preprocessing in model performance. When juxtaposing MixMatch with ReMixMatch, the latter exhibits a marginal lead in accuracy, indicating its slight edge within the preprocessing-enhanced learning approaches.

6.3. Training Time

Since the training time of a neural network in a production environment is critical, choosing the correct DNN for the SSML is a major decision in some cases where data change. Table 3 presents the training time results for every combination of all DNNs by executing all the SSMLs on the GPU—i.e., an Nvidia 2080 Ti GTX. The deeper and wider DNN is Shake-Shake26, and for this reason, the results demonstrate the highest training time in this case. On the other hand, the lowest training time was observed for DenseNet-121. This is due to the interconnection among the layers [35], making the total number of parameters lower as compared to the WideResNet model.

Table 3. Pearson correlation coefficients.

Network Classifier	SSML Algorithm	Labeled Data/ Unlabeled Data	Training Time (min)	PCC	Average PCC
Shake-Shake26	Temporal E.	49,000/1000 46,000/4000 0/50,000	1020 824 120	$PCC_{shake} = 0.7227$ p-value= 0.002003 $z'_{shake} = 0.929$	
	Mean Teacher	49,000/1000 46,000/4000 0/50,000	1161 1094 158		
	Π model	49,000/1000 46,000/4000 0/50,000	**1351** 1272 168		
	MixMatch	49,000/1000 46,000/4000 0/50,000	1211 1103 163		
	ReMixMatch	49,000/1000 46,000/4000 0/50,000	1239 1142 149		
DenseNet-121	Temporal E.	49,000/1000 46,000/4000 0/50,000	323 276 **110**	$\boldsymbol{PCC_{densenet} = 0.7585}$ **p-value= 0.001046** $\boldsymbol{z'_{densenet} = 0.993}$	$PCC_{average} = 0.701$ p-value= 0.003597 $z'_{avg} = 0.87$
	Mean Teacher	49,000/1000 46,000/4000 0/50,000	392 337 124		
	Π model	49,000/1000 46,000/4000 0/50,000	443 407 168		
	MixMatch	49,000/1000 46,000/4000 0/50,000	404 353 131		
	ReMixMatch	49,000/1000 46,000/4000 0/50,000	411 367 139		
WideResNet	Temporal E.	49,000/1000 46,000/4000 0/50,000	387 324 132	$PCC_{wide} = 0.5968$ p-value= 0.01904 $z'_{wide} = 0.688$	
	Mean Teacher	49,000/1000 46,000/4000 0/50,000	537 485 158		
	Π model	49,000/1000 46,000/4000 0/50,000	553 512 179		
	MixMatch	49,000/1000 46000/4000 0/50,000	551 512 149		
	ReMixMatch	49,000/1000 46,000/4000 0/50,000	567 517 151		

Furthermore, in case of the Π model, the training time is higher due to the model's ability to perform backward propagation for both the student and the teacher models. Expectedly, the lowest training time is still that of Temporal Ensembling, as it includes only one DNN in the classifier, which reduces the total number of parameters and the SSML's complexity.

Adding the PF-SSCPs, MixMatch and ReMixMatch require more training time as compared to the CF-SSCPs, excluding the Π model. This is the due to the complexity of the preprocessing step introduced and the pseudo-label calculation to generate the labeled and unlabeled training sets. Comparing MixMatch and ReMixMatch, it is clear that ReMixMatch requires more training time due to its more complex calculation of the pseudo-labels, as shown in Figure 4.

In order to see if the correlation would still withhold, Table 3 presents the Pearson correlation coefficients (PCC) [36] and the training time for all network classifiers independent of the SSCP with different SSML algorithms.

As shown previously, the training times for the Shake-Shake26 model as the DNN with a 49,000/1000 ratio of unlabeled/labeled data are shown as 1020, 1161, 1351, 1211, and 1239 min using the Temporal Ensembling model, Mean Teacher model, Π model, MixMatch, and ReMixMatch, respectively. On the other hand, the training times are 120, 158, 168, 163 and 149 min when there are no labels. In Table 3, we have similar results for

the other network classifiers. The Pearson correlation coefficients (PCC) for each network classifier are used to indicate a positive correlation between the training time and the labeled/unlabeled data ratio.

Looking at PCC_{shake}, we can observe a correlation of 0.7227 with a p-value of 0.002003, compared to utilizing only the CF-SSCP algorithm with a p-value of 0.0278 and PCC_{shake} of 0.7227. Moreover, we can observe that $0.73 > 0.7227 > 0.7$ and $0.002003 < 0.0278 < 0.05$, which indicates a higher correlation. We can find similar results for the DenseNet model with a $PCC_{DenseNet}$ value of 0.7585. However, for WideResNet, we see a weaker correlation, although it is noticeably positive with a $PCC_{WideResNet}$ of 0.5968. To evaluate the average correlation coefficient among all three network classifiers, we used Fisher's Z to transform each correlation coefficient. Subsequently, we calculated the mean z' value in each case. This means z'_{avg} value was then transformed to the correlation coefficient again. After performing these computations, the PCC_{avg} was found to be 0.701, which is greater than 0.692, as stated with CF-SSCP, which indicates a higher correlation.

7. Discussion

As observed in the experiments, SSML based on PF-SSCP demonstrates higher accuracy and lower loss compared to that based on CF-SSCP, directly informing our conclusion regarding RQ1. This improved performance can be attributed to the quantity of data inputted into the DNNs, which plays a pivotal role in an SSML's effectiveness. This is due to the utilization of parameter updates in each design. Additionally, for both PF-SSCP and CF-SSCP network classifiers, we observed that fewer skip connections enhanced accuracy and diminished loss. This diversification of parameter updates facilitated more effective forward propagation, influencing the consistency cost.

Another insight gleaned from these experiments pertains to the correlation between training time and the ratio of labeled to unlabeled data, as addressed in RQ2. The presence of more labeled data necessitated increased computational time to calculate the loss, primarily due to the classification cost. Alterations in the classification cost significantly affected the loss, prompting substantial parameter updates during backward propagation, which, in turn, impacted subsequent forward propagation.

8. Conclusions

In this paper, we delved into various facets of SSML through the lenses of post-hoc XAI and global scale analysis. We dissected SSML into two distinct SSCP components, highlighting the disparities in learning precision, training time, and design when integrated with cutting-edge DNNs as network classifiers. Through detailed graphs, we elucidated their operational mechanics, pinpointing the factors that contribute most significantly to their functionality.

Moreover, our research, grounded in 45 rigorous experiments, substantiated the impact of preprocessing techniques introduced within the PF-SSCP method. Our analyses revealed that the choice of a DNN as a network classifier, regardless of the SSML algorithm employed, profoundly influences the magnitude of parameter updates, both in terms of frequency and size, particularly when more skip connections are present. Additionally, our findings indicated that the preprocessing stages within SSML exert a more pronounced effect than the selection of the network classifier itself. This insight offers a strategic direction for ML developers and engineers, encouraging the adoption of innovative frameworks within SSMLs and DNNs to fully harness the potential of domain-specific data.

Building on prior discoveries, we further reinforced the observed correlation between training time and the ratio of labeled to unlabeled data, providing robust statistical evidence to support this relationship. In future endeavors, we aim to meticulously analyze and deconstruct groundbreaking SSML frameworks in conjunction with DNNs. Our goal is to authenticate the efficacy of the contributing factors identified in this study by exploring a broader spectrum of cases.

Author Contributions: Conceptualization, E.N. and Y.L.; methodology, E.N.; software, E.N.; validation, E.N. and Y.L.; formal analysis, E.N.; investigation, E.N.; resources, Y.L.; data curation, E.N.; writing—original draft preparation, E.N.; writing—review and editing, E.N. and Y.L.; visualization, E.N.; supervision, Y.L.; project administration, Y.L.; funding acquisition, Y.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The data presented in this study are available on request from the corresponding author.

Conflicts of Interest: E.N. and Y.L. declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

SSML	Semi-supervised machine learning
DNNs	Deep neural networks
XAI	Explainable artificial intelligence
CNN	Convolution neural networks
GNNs	Graph neural networks
SSCP	Semi-supervised machine learning process
PF-SSCP	Preprocessing-focused semi-supervised computation process
CF-SSCP	Network classifier-focused semi-supervised computation process
ResNet	Residual neural network
WRN	Wide residual network
DenseNet	Densely connected convolutional network
PCC	Pearson correlation coefficient

Appendix A

- **Experiment 1** *(Temporal Ensembling and Shake-Shake26)*: This trial involved processing $32 \times 32 \times 3$ images with ZCA and training on two Shake-Shake26 networks. A batch size of 128 was used over 300 epochs, with a dropout rate of 0.2, a learning rate of 0.2 (reduced at 50% and 75% of epochs), a momentum of 0.86, and a weight decay of 0.0002.
- **Experiment 2** *(Temporal Ensembling and DenseNet-121)*: Similar to Experiment 1 but utilizing two DenseNet-121 networks with a batch size of 64 and an initial learning rate of 0.1, which is reduced at the midpoint and three-quarter mark of epochs, alongside a momentum of 0.9 and a weight decay of 0.0001.
- **Experiment 3** *(Temporal Ensembling and WRN-40-2)*: Similar to Experiment 1 but using two WRN-40-2 networks, with a dropout of 0.1, learning rate of 0.1 (decreased at pre-set epochs), and a weight decay of 0.0005.
- **Experiment 4** *(Temporal Ensembling and WRN-28-10)*: Similar to Experiment 3 but with two WRN-28-10 networks and an increased dropout of 0.3.
- **Experiment 5** *(*Π *model and Shake-Shake26)*: Similar to Experiment 1 but under the Π model methodology with the same dropout rate and learning rate reduction schedule.
- **Experiment 6** *(*Π *model and DenseNet-121)*: Similar to Experiment 2 but following the Π model framework with identical dropout and learning rate scheduling.
- **Experiment 7** *(*Π *model and WRN-40-2)*: Similar to Experiment 3, applying the Π model approach with a dropout of 0.3.
- **Experiment 8** *(*Π *model and WRN-28-10)*: Similar to Experiment 7 but using WRN-28-10 networks.
- **Experiment 9** *(Mean Teacher and Shake-Shake26)*: Similar to Experiment 1 with the Mean Teacher model, applying an EMA of 0.999 and using the Adam optimizer with specified parameters.
- **Experiment 10** *(Mean Teacher and DenseNet-121)*: Similar to Experiment 2, adhering to the Mean Teacher methodology.

- **Experiment 11** *(Mean Teacher and WRN-40-2)*: Similar to Experiment 3 with the Mean Teacher framework and the same dropout rate.
- **Experiment 12** *(Mean Teacher and WRN-28-10)*: Similar to Experiment 7, utilizing the Mean Teacher strategy.
- **Experiment 13** *(MixMatch and Shake-Shake26)*: Similar to Experiment 1 but using the MixMatch approach, with specific augmentations and no dropout regularization.
- **Experiment 14** *(MixMatch and DenseNet-121)*: Similar to Experiment 2, employing the MixMatch method with a weight decay of 0.997.
- **Experiment 15** *(MixMatch and WRN-40-2)*: Similar to Experiment 3, following the MixMatch technique with an optimal weight decay.
- **Experiment 16** *(MixMatch and WRN-28-10)*: Similar to Experiment 7, incorporating the MixMatch framework.
- **Experiment 17** *(ReMixMatch and Shake-Shake26)*: Similar to Experiment 13 but applying the ReMixMatch methodology.
- **Experiment 18** *(ReMixMatch and DenseNet-121)*: Similar to Experiment 14, using the ReMixMatch strategy with a preferred weight decay.
- **Experiment 19** *(ReMixMatch and WRN-40-2)*: Similar to Experiment 15, employing the ReMixMatch technique with a specific weight decay.
- **Experiment 20** *(ReMixMatch and WRN-28-10)*: Similar to Experiment 16, following the ReMixMatch approach.
- **Experiment 21** *(Temporal Ensembling and Shake-Shake26 with reduced dropout)*: Similar to Experiment 1 but with a reduced dropout of 0.023.
- **Experiment 22** *(Temporal Ensembling and DenseNet-121 with reduced dropout)*: Similar to Experiment 2, with a lowered dropout rate of 0.045.
- **Experiment 23** *(Temporal Ensembling and WRN-40-2 with reduced dropout)*: Similar to Experiment 3, with a dropout rate adjusted to 0.087.
- **Experiment 24** *(Temporal Ensembling and WRN-28-10 with reduced dropout)*: Similar to Experiment 4, with a decreased dropout of 0.083.
- **Experiment 25** *(*Π *model and Shake-Shake26 with reduced dropout)*: Similar to Experiment 5, maintaining a dropout rate of 0.2.
- **Experiment 26** *(*Π *model and DenseNet-121 with reduced dropout)*: Similar to Experiment 6, with a lowered dropout of 0.022.
- **Experiment 27** *(*Π *model and WRN-40-2 with reduced dropout)*: Similar to Experiment 7, with a reduced dropout of 0.072.
- **Experiment 28** *(*Π *model and WRN-28-10 with reduced dropout)*: Similar to Experiment 8, with a dropout rate of 0.068.
- **Experiment 29** *(Mean Teacher and Shake-Shake26 with reduced dropout)*: Similar to Experiment 9, with a dropout of 0.03.
- **Experiment 30** *(Mean Teacher and DenseNet-121 with reduced dropout)*: Similar to Experiment 10, with a reduced dropout rate of 0.02.
- **Experiment 31** *(Mean Teacher and WRN-40-2 with reduced dropout)*: Similar to Experiment 11, with a decreased dropout of 0.082.
- **Experiment 32** *(Mean Teacher and WRN-28-10 with reduced dropout)*: Similar to Experiment 12, with a lowered dropout rate of 0.075.
- **Experiment 33** *(MixMatch and Shake-Shake26 with four augmentations)*: Similar to Experiment 13 but with an increased number of augmentations set to 4.
- **Experiment 34** *(MixMatch and DenseNet-121 with reduced weight decay)*: Similar to Experiment 14, with a modified weight decay of 0.997.
- **Experiment 35** *(MixMatch and WRN-40-2 with optimal weight)*: Similar to Experiment 15, with a tuned weight decay of 0.999.
- **Experiment 36** *(MixMatch and WRN-28-10 with optimal weight)*: Similar to Experiment 16, applying a fine-tuned weight.
- **Experiment 37** *(MixMatch and WRN-16-10 with optimal weight)*: Similar to Experiment 14, but using WRN-16-10 networks.

- **Experiment 38** *(ReMixMatch and Shake-Shake26 with dropout)*: Same as Experiment 17, but with a dropout of 0.953.
- **Experiment 39** *(ReMixMatch and DenseNet-121 with dropout)*: Similar to Experiment 18, with a dropout rate of 0.238.
- **Experiment 40** *(ReMixMatch and WRN-40-2 reduced weight decay)*: Similar to Experiment 19, with a lowered weight decay of 0.877.
- **Experiment 41** *(ReMixMatch and WRN-16-10 with optimal weight)*: Similar to Experiment 14, using WRN-16-10 networks.
- **Experiment 42** *(ReMixMatch and WRN-16-10 with dropout)*: Similar to Experiment 41, with a dropout rate of 0.38.
- **Experiment 43** *(ReMixMatch and WRN-16-10 with weight decay)*: Identical to Experiment 42, with a weight decay of 0.997.
- **Experiment 44** *(ReMixMatch and WRN-28-10 with dropout)*: Similar to Experiment 20, with a dropout rate of 0.233.
- **Experiment 45** *(ReMixMatch and WRN-28-10 with very low dropout)*: Similar to Experiment 20, with a very low dropout of 0.0233.

References

1. Chapelle, O.; Schölkopf, B.; Zien, A. (Eds.) *Semi-Supervised Learning*; The MIT Press: Cambridge, MA, USA, 2006.
2. Rudin, C. Stop Explaining Black Box Machine Learning Models for High Stakes Decisions and Use Interpretable Models Instead. 2019. Available online: http://xxx.lanl.gov/abs/1811.10154 (accessed on 3 April 2023).
3. Adadi, A.; Berrada, M. Peeking Inside the Black-Box: A Survey on Explainable Artificial Intelligence (XAI). *IEEE Access* **2018**, *6*, 52138–52160. [CrossRef]
4. Laine, S.; Aila, T. Temporal Ensembling for Semi-Supervised Learning. *CoRR* **2016**, abs/1610.02242. Available online: http://xxx.lanl.gov/abs/1610.02242 (accessed on 14 February 2022).
5. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. *CoRR* **2015**, abs/1512.03385. Available online: http://xxx.lanl.gov/abs/1512.03385 (accessed on 17 February 2022).
6. Neghawi, E.; Liu, Y. Evaluation of Parameter Update Effects in Deep Semi-Supervised Learning Algorithms. In Proceedings of the 2020 IEEE 44th Annual Computers, Software, and Applications Conference (COMPSAC), Madrid, Spain, 13–17 July 2020; pp. 351–360.
7. Lin, Y.; Han, S.; Mao, H.; Wang, Y.; Dally, W.J. Deep Gradient Compression: Reducing the Communication Bandwidth for Distributed Training. *CoRR* **2017**, abs/1712.01887. Available online: http://xxx.lanl.gov/abs/1712.01887 (accessed on 3 March 2023).
8. Zagoruyko, S.; Komodakis, N. Wide Residual Networks. 2017. Available online: http://xxx.lanl.gov/abs/1605.07146 (accessed on 19 March 2023).
9. Baldassarre, F.; Azizpour, H. Explainability Techniques for Graph Convolutional Networks. *CoRR* **2019**, abs/1905.13686. Available online: http://xxx.lanl.gov/abs/1905.13686 (accessed on 22 February 2023).
10. Yuan, H.; Yu, H.; Gui, S.; Ji, S. Explainability in Graph Neural Networks: A Taxonomic Survey. *CoRR* **2020**, abs/2012.15445. Available online: http://xxx.lanl.gov/abs/2012.15445 (accessed on 3 March 2023).
11. Xie, N.; Ras, G.; van Gerven, M.; Doran, D. Explainable Deep Learning: A Field Guide for the Uninitiated. *CoRR* **2020**, abs/2004.14545. Available online: http://xxx.lanl.gov/abs/2004.14545 (accessed on 7 March 2023).
12. Shen, W.; Wei, Z.; Huang, S.; Zhang, B.; Fan, J.; Zhao, P.; Zhang, Q. Interpretable Compositional Convolutional Neural Networks. *CoRR* **2021**, abs/2107.04474. Available online: http://xxx.lanl.gov/abs/2107.04474 (accessed on 8 February 2023).
13. Doshi-Velez, F.; Kim, B. Towards a rigorous science of interpretable machine learning. *arXiv* **2017**, arXiv:1702.08608.
14. Yuan, H.; Tang, J.; Hu, X.; Ji, S. XGNN: Towards Model-Level Explanations of Graph Neural Networks. *CoRR* **2020**, abs/2006.02587. Available online: http://xxx.lanl.gov/abs/2006.02587 (accessed on 20 December 2022).
15. Pope, P.E.; Kolouri, S.; Rostami, M.; Martin, C.E.; Hoffmann, H. Explainability Methods for Graph Convolutional Neural Networks. In Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 16–20 June 2019; pp. 10764–10773. [CrossRef]
16. Kim, G. Recent Deep Semi-supervised Learning Approaches and Related Works. *CoRR* **2021**, abs/2106.11528. Available online: http://xxx.lanl.gov/abs/2106.11528 (accessed on 17 February 2022).
17. Tarvainen, A.; Valpola, H. Weight-averaged consistency targets improve semi-supervised deep learning results. *CoRR* **2017**, abs/1703.01780. Available online: http://xxx.lanl.gov/abs/1703.01780 (accessed on 22 December 2022).
18. Berthelot, D.; Carlini, N.; Goodfellow, I.J.; Papernot, N.; Oliver, A.; Raffel, C. MixMatch: A Holistic Approach to Semi-Supervised Learning. *CoRR* **2019**, abs/1905.02249. Available online: http://xxx.lanl.gov/abs/1905.02249 (accessed on 22 December 2022).
19. 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* **2020**, arXiv:1911.09785.
20. Goodfellow, I.; Bengio, Y.; Courville, A. *Deep Learning*; MIT Press: Cambridge, MA, USA, 2016.

21. Pal, K.K.; Sudeep, K.S. Preprocessing for image classification by convolutional neural networks. In Proceedings of the 2016 IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT), Bangalore, India, 20–21 May 2016.
22. Murphy, K.P. *Machine Learning: A Probabilistic Perspective*; MIT Press: Cambridge, MA, USA, 2013.
23. Arrieta, A.B.; Díaz-Rodríguez, N.; Ser, J.D.; Bennetot, A.; Tabik, S.; Barbado, A.; García, S.; Gil-López, S.; Molina, D.; Benjamins, R.; et al. Explainable Artificial Intelligence (XAI): Concepts, Taxonomies, Opportunities and Challenges toward Responsible AI. 2019. Available online: http://xxx.lanl.gov/abs/1910.10045 (accessed on 17 April 2023).
24. Chen, J.; Song, L.; Wainwright, M.J.; Jordan, M.I. Learning to Explain: An Information-Theoretic Perspective on Model Interpretation. *CoRR* **2018**, abs/1802.07814. Available online: http://xxx.lanl.gov/abs/1802.07814 (accessed on 22 February 2023).
25. Lipton, Z.C. The mythos of model interpretability. *Queue* **2018**, *16*, 30. [CrossRef]
26. He, K.; Fan, H.; Wu, Y.; Xie, S.; Girshick, R. Momentum Contrast for Unsupervised Visual Representation Learning. 2020. Available online: http://xxx.lanl.gov/abs/1911.05722 (accessed on 22 February 2023).
27. Szegedy, C.; Vanhoucke, V.; Ioffe, S.; Shlens, J.; Wojna, Z. Rethinking the Inception Architecture for Computer Vision. *CoRR* **2015**, abs/1512.00567. Available online: http://xxx.lanl.gov/abs/1512.00567 (accessed on 5 April 2023).
28. Hadjis, S.; Zhang, C.; Mitliagkas, I.; Ré, C. Omnivore: An Optimizer for Multi-device Deep Learning on CPUs and GPUs. *CoRR* **2016**, abs/1606.04487. Available online: http://xxx.lanl.gov/abs/1606.04487 (accessed on 18 February 2022).
29. Huang, G.; Liu, Z.; Weinberger, K.Q. Densely Connected Convolutional Networks. *CoRR* **2016**, abs/1608.06993. Available online: http://xxx.lanl.gov/abs/1608.06993 (accessed on 17 February 2022).
30. Mania, H.; Pan, X.; Papailiopoulos, D.; Recht, B.; Ramchandran, K.; Jordan, M.I. Perturbed Iterate Analysis for Asynchronous Stochastic Optimization. 2016. Available online: http://xxx.lanl.gov/abs/1507.06970 (accessed on 23 August 2023).
31. Gastaldi, X. Shake-Shake regularization. *CoRR* **2017**, abs/1705.07485. Available online: http://xxx.lanl.gov/abs/1705.07485 (accessed on 23 August 2023).
32. Tieleman, T.; Hinton, G. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *Coursera Neural Netw. Mach. Learn.* **2012**, *4*, 26–31.
33. Goyal, P.; Dollár, P.; Girshick, R.B.; Noordhuis, P.; Wesolowski, L.; Kyrola, A.; Tulloch, A.; Jia, Y.; He, K. Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour. *CoRR* **2017**, abs/1706.02677. Available online: http://xxx.lanl.gov/abs/1706.02677 (accessed on 7 March 2023).
34. Ioffe, S.; Szegedy, C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *CoRR* **2015**, abs/1502.03167. Available online: http://xxx.lanl.gov/abs/1502.03167 (accessed on 7 March 2023).
35. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778. [CrossRef]
36. Vapnik, V.N. *The Nature of Statistical Learning Theory*; Springer: New York, NY, USA, 1995.