TransBoost: Improving the Best ImageNet Performance using Deep Transduction

Anonymous Author(s)
Affiliation
Address
email

Abstract

This paper deals with *deep transductive learning*, and proposes *TransBoost* as a procedure for fine-tuning any deep neural model to improve its performance on any (unlabeled) test set provided at training time. TransBoost is inspired by a large margin principle and is efficient and simple to use. The ImageNet classification performance is consistently and significantly improved with TransBoost on many architectures such as ResNets, MobileNetV3-L, EfficientNetB0, ViT-S, and ConvNext-T. Additionally we show that TransBoost is effective on a wide variety of image classification datasets.

1 Introduction

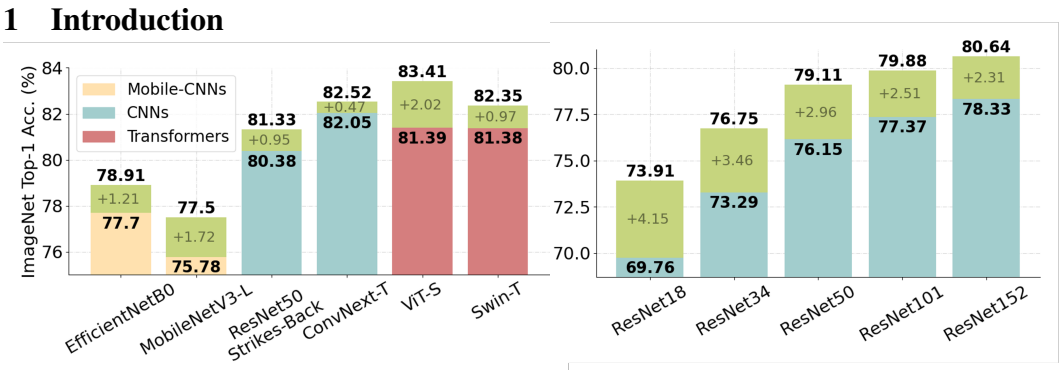

Figure 1: ImageNet top-1 accuracy gains of TransBoost (green) using representatives of CNNs (blue), Mobile-CNNs (yellow) and Transformers (red), in comparison to the standard inductive (fully supervised) performance.

Alternative learning frameworks that make use of unlabeled data have gained considerable attention in recent years. These include *semi-supervised learning* and *transductive learning*, both of which were extensively studied in classical and statistical machine learning [1, 2, 3] before the advent of deep learning. In this article we focus on deep *transductive* classification. In this setting, both a labeled training sample and an (unlabeled) test sample are provided at training time. The goal is to predict *only* the labels of the given test instances as accurately as possible. In contrast, in standard *inductive* classification, the goal is to train a general model capable of predicting the labels of unseen test instances.

In semi-supervised learning (SSL) we are also given a set of unlabeled examples, which is used for selecting a model. Despite a commonly held misconception, however, the goal of SSL remains inductive rather than transductive, and the trained model must predict labels for new instances (i.e., the unlabeled instances are used for training only).

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

Transductive models deliver higher accuracy gains compared with traditional inductive models, as we demonstrate in this paper in the context of image classification. Much of the power of transductive learning is achieved through analyzing the given test set as a group (see Section 5.4). Transductive prediction is thus especially useful whenever we can accumulate a test set of instances and then train a specialized model to predict their labels. While there are many possible relevant use cases, one that stands out is medical diagnosis. Here, daily or weekly medical records can be accumulated and sent to the transductive predictor as a set.

Perhaps the best-known transductive classification method in classical machine learning is the transductive support vector machine (TSVM) [4, 5]. While a support vector machine (SVM) [6] seeks to achieve a general decision function, a TSVM attempts to reduce misclassifications of just the target test instances. To the best of our knowledge, in deep learning, transductive settings have only been addressed in the case of few-shot learning (t-FSL).

In this paper we address deep transductive learning and introduce *TransBoost*, a novel procedure that takes a pretrained neural classification model, along with a (labeled) training set, and an (unlabeled) test set. The procedure fine-tunes the model to improve its performance on this particular test set. The TransBoost optimization procedure is inspired by TSVM and its objective attempts to approximate a *large margin principle*. TransBoost is implemented through a simple transductive loss component, which is combined with the standard cross-entropy loss.

We examine the effectiveness of TransBoost by using a variety of pretrained neural networks, datasets, and baseline models. Our extensive study examines a spectrum of ResNet architectures and a variety of modern architectures such as ConvNext [7], MobileNetv3 [8] and ViT [9]. Figure 1 highlights some of TransBoost's results on the ImageNet dataset, which demonstrate impressive and consistent accuracy gains, leading to state-of-the-art performance compared to standard (inductive) classification as well as versus SSL and t-FSL methods (see full results in Section 5.2).

To summarize, the contributions of this paper are: (1) A novel deep transductive loss function, inspired by a large margin principle, which optimizes performance on a specific test set. (2) A simple and efficient transductive procedure for fine-tuning a pretrained neural model in order to boost its performance over an inductive setting. (3) A comprehensive empirical study of the proposed fine-tuning procedure showing great benefits across a large number of architectures, datasets and baseline techniques.

2 Problem Formulation

As transductive learning has not been extensively researched in the deep learning community, we begin by defining the transductive learning problem within the context of deep learning classification. We follow Vapnik's standard formulation from statistical learning theory [4]. Let $P(X, Y)$ be a probability distribution over $\mathcal{X} \times \mathcal{Y}$, where $\mathcal{X}$ represents an input space (e.g., raw image data), and $\mathcal{Y}$ represents a label set corresponding to C classes. The learner is provided with a set of labeled instances, $S_l \triangleq \{(x_1, y_1), \ldots, (x_L, y_L)\}$, where $x_i \in \mathcal{X}$, $y_i \in \mathcal{Y}$, and a finite set of unlabeled instances, $X_u \triangleq \{x_{L+1}, \ldots, x_{L+U}\}$, where $x_i \in \mathcal{X}$. The objective is to label the unlabeled instances based on this data. Given a deep neural network $f_\theta : \mathcal{X} \to \mathcal{Y}$, where θ denotes its parameters, and a loss function $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$, we follow [1] (Setting 2) and define the (true) *risk* over the unlabeled set,

$$E_{f_\theta}(X_U) \triangleq \frac{1}{U} \sum_{i=1}^{U} \ell(f_\theta(x_{L+i}), y_{L+i}). \tag{1}$$

Thus, we are given a labeled training set $S_l \triangleq (X_l, Y_l)$ selected i.i.d. according to $P(X, Y)$, as well as a pretrained classification model f_θ, and we presume it was trained with the labeled set S_l. An independent test set $S_u \triangleq (X_u, Y_u)$, of U samples is randomly and independently selected in the same manner, and we are required to revise the given model based on both S_l and X_u, so as to minimize Equation (1) without knowing the labels Y_u.

3 Related Work

Transductive learning (or *transduction*) was first formulated by Vapnik [4, 10] who also introduced the first transductive algorithm – TSVM [4, 5], which learns a large margin hyperplane classifier from labeled training data while simultaneously forcing it to take into account the (unlabeled) test data. Whereas transduction received considerable attention in the context of classical machine learning [1, 2, 3], it has only been briefly addressed in the context of deep neural networks.

Recall that in (deep) transductive classification we are provided with both a labeled training sample and an unlabeled test sample. A related but different setting is *semi-supervised learning* (SSL), where in addition to the labeled training set, we are also given a (typically large) unlabeled training sample. The objective is *inductive* – namely, to guess the label of any *unseen* test sample (that was drawn from the same distribution). A simple observation is that any SSL algorithm can be applied in transduction by using the given (unlabeled) test sample as the unlabeled SSL training sample. Thus, algorithms such as [11, 12, 13] can be used to solve transductive learning. This fact may explain a common confusion between SSL and transduction. Nevertheless, an inductive SSL algorithm does not need to use (and cannot rely on) the fact that it will only be asked about the given test points. In contrast, a (meaningful) transductive algorithm must rely on this knowledge, as we contend in this paper. In fact, in Section 5.2 we demonstrate that our proposed transductive procedure substantially outperforms the *SimCLR* [11] and *SimCLRv2* [12] algorithms, where SimCLRv2 currently achieves state-of-the-art SSL performance according to [14].

Recently, there has been a flurry of research concerning *transductive few-shot learning* (t-FSL) [15, 16, 17]. In t-FSL we are given a classifier for a certain classification task, such as cats versus dogs, as well as a training set (called the *support* set) for unseen classes (referred to as *ways*), such as elephants, tigers and lizards, containing only few labeled samples per unseen class. A test set (called the *query* set), containing instances from the unseen classes, is given and the goal is to learn the additional classes so as to correctly classify these test instances. Clearly, this setting is transductive; however, the goal here is to learn from a few labeled examples (e.g., one-shot, five-shot). Formally, a t-FSL algorithm can be applied to transductive classification (by treating the training set as a support set). For the most part, however, methods for t-FSL cannot be effectively applied to large (non few-shot) training sets (e.g., iLPC [17]). Nevertheless, in order to conduct a complete study in transductive classification, we chose two t-FSL representative algorithms as baselines for our experiments (see Section 5). Specifically, we consider the entropy minimization (Ent-min) method of [15], and the TIM-GD method of [16]. With Ent-min, the Shannon entropy is reduced by fine-tuning, whereas TIM-GD optimizes the mutual information of test instances using only the classifier weights with fine-tuning. Ent-min is a common t-FSL baseline while TIM-GD is a top performing 5-shot t-FSL algorithm according to the few-shot leaderboard [18].

4 TransBoost: Fine-Tuning via Transductive Learning

We now present *TransBoost*, a procedure that takes a pretrained neural model along with its (labeled) training set as well as an (unlabeled) test set. The procedure then fine-tunes the model to improve its performance for this specific test set. The TransBoost optimization objective, $\mathcal{L}$, is driven by Equation (3) and approximated by Algorithm 1, which includes a novel transductive loss component in addition to the standard cross-entropy loss. Our transductive component, $\mathcal{L}_{\text{TransBoost}}$, is introduced in Equation (3) as a regularization term and is defined in Equation (2). $\mathcal{L}_{\text{TransBoost}}$ is inspired by TSVMs [4, 5] and follows a *large margin principle*.

Let us elaborate on our loss function, $\mathcal{L}_{\text{TransBoost}}$ (2), which is applied on the set of test samples, X_u. This loss function includes an unsupervised non-negative symmetric pairwise similarity function, $\mathcal{S} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, which measures the similarity of two input instances. In our implementation, we simply used an L_2-based score function applied on prediction (softmax) vectors generated by f_θ; see Equation (4). $\mathcal{S}$ is only applied on pairs of test instances that are likely to belong to different classes. The likelihood is determined by using an implementation of the Kronecker delta function,

Algorithm 1: TransBoost Procedure

Input : Labeled train sample $S_l \triangleq (X_l, Y_l)$, unlabeled test sample X_u, a model f_θ that was pretrained on S_l, a supervised pointwise loss function ℓ, implementations of TransBoost's functions: $\mathcal{S}$, δ and κ. Hyperparameters: Number of epochs E, labeled batch size L', unlabeled batch size U', regularization parameter λ.

for $epoch \leftarrow 1$ **to** E **do**
  `// Sample labeled and unlabeled batches in a cyclical manner`
  for $S'_l \triangleq (X'_l, Y'_l) \subset S_l$, $X'_u \subset X_u$ **to** $\max\{\lceil \frac{L}{L'} \rceil, \lceil \frac{U}{U'} \rceil\}$ **do**
    `// Prepare random test sample pairs`
    ${X'_u}^{\pi} \leftarrow$ generate a random permutation of X'_u
    `// Approximate the TransBoost loss`
    $L_{\text{TransBoost}} \leftarrow 0$
    for $i \leftarrow 0$ **to** U' **do**
      $L_{\text{TransBoost}} \leftarrow L_{\text{TransBoost}} + \kappa(x_i)\kappa({x_i}^{\pi})\delta(x_i, {x_i}^{\pi})\mathcal{S}(x_i, {x_i}^{\pi})$
    end for
    $L_{\text{TransBoost}} \leftarrow L_{\text{TransBoost}} / \sum_{i=1}^{U'} \delta(x_i, {x_i}^{\pi})$ if $\sum_{i=1}^{U'} \delta(x_i, {x_i}^{\pi}) \neq 0$ else 0
    `// Apply an optimization step`
    $\theta \leftarrow$ optimize $\left[\frac{1}{L'} \sum_{i=1}^{L'} \ell(f_\theta(x_i), y_i) + \lambda \cdot L_{\text{TransBoost}}\right]$
  end for
end for

$\delta : \mathcal{X} \times \mathcal{X} \rightarrow \{0, 1\}$, which obtains 1 iff the instances are predicted to belong to different classes; see Equation (5) for our implementation. Additionally, for each pair we intensify the loss using the model's confidence in its predictions. We define $\kappa : \mathcal{X} \rightarrow \mathbb{R}^+$ to be a confidence function, which for each instance, x, gives its class prediction confidence; see Equation (6) for our implementation. Finally, our transductive loss function, $\mathcal{L}_{\text{TransBoost}}$, is

$$\mathcal{L}_{\text{TransBoost}}(X_u | f_\theta, \mathcal{S}, \delta, \kappa) \triangleq \frac{1}{U_\delta} \sum_{1 \leq i < j \leq U} \kappa(x_i)\kappa(x_j)\delta(x_i, x_j)\mathcal{S}(x_i, x_j), \tag{2}$$

where $U_\delta \triangleq \sum_{1 \leq i < j \leq U} \delta(x_i, x_j)$. The final loss function of our fine-tuning procedure includes the transductive loss term (2) as a regularization term as follows,

$$\mathcal{L}(X_l, Y_l, X_u | f_\theta, \mathcal{S}, \delta, \kappa) \triangleq \underbrace{\frac{1}{L} \sum_{i=1}^{L} \ell(f_\theta(x_i), y_i)}_{\text{labeled/inductive (standard) loss}} + \lambda \cdot \underbrace{\mathcal{L}_{\text{TransBoost}}(X_u | f_\theta, \mathcal{S}, \delta, \kappa)}_{\text{unlabeled/transductive loss}}, \tag{3}$$

where ℓ is a standard (inductive) pointwise loss, e.g., the cross-entropy loss and $\lambda \in \mathbb{R}$ is a regularization hyperparameter.

Using the optimization objective in Equation (3), we incentivize test sample pairs that are likely to be different in their classes to also be different in their empirical class probabilities while preserving the prior knowledge of f_θ. Accordingly, given two test samples, (x_i, x_j), which are predicted to be different in their classes by δ, we obtain a higher cost (by $\mathcal{S}$) if x_i is close (similar) to x_j in its prediction vector than if x_i is far (dissimilar) from x_j, depending on the model's confidence in its predictions utilizing κ.

The computation of the $\mathcal{L}_{\text{TransBoost}}$ loss component (2) is quadratic in U – namely, its time complexity is $O(\binom{U}{2})$. For large test sets (e.g., ImageNet), this is prohibitively large. Therefore, we propose to approximate $\mathcal{L}_{\text{TransBoost}}$ by using random subsets of pairs from the test set. This random sampling is applied for each minibatch. The pseudo-code of the optimization procedure is presented in Algorithm 1. As can be seen, the size of the set of sampled test pairs used to approximate (2) is taken to be the batch size (U'), which allows for very fast computation of $\mathcal{L}_{\text{TransBoost}}$ but could be sub-optimal (this size was not optimized).

4.1 TransBoost Implementation

While $\mathcal{S}$, δ and κ in Equation (2) can be implemented in many ways, we now describe our proposed implementation that was used to obtain all the results described in Section 5. As can be seen below, we attempted to instantiate Equation (2) in the simplest possible manner. The consideration of more sophisticated methods is left for future work (and an alternative choice is discussed in Section 5.5).

The unsupervised symmetric pairwise similarity function $\mathcal{S}_f$ was straightforwardly implemented using the L_2 norm as follows,

$$\mathcal{S}_f(x_i, x_j) \triangleq \sqrt{2} - \left\|\hat{p}(x_i|f_\theta) - \hat{p}(x_j|f_\theta)\right\|_2, \tag{4}$$

where $\hat{p}(x|f_\theta)$ denotes the empirical class probability of f_θ applied on x. It is easy to show that $\mathcal{S}_f$ is non-negative[1] (see Appendix A for a proof).

Additionally, we implemented δ_f (5) and κ_f (6) based on the pretrained weights of the model before it was fine-tuned with our procedure, which we refer to as θ_0. Thereafter, the unsupervised symmetric pairwise selection function δ_f is implemented using pseudo-labels computed by the given pretrained model f_{θ_0},

$$\delta_f(x_i, x_j) \triangleq \begin{cases} 1 & f_{\theta_0}(x_i) \neq f_{\theta_0}(x_j) \\ 0 & \text{otherwise,} \end{cases} \tag{5}$$

and the confidence function κ_f is implemented using the standard softmax response, i.e.,

$$\kappa_f(x) \triangleq \max\{\hat{p}_j(x|f_{\theta_0})\}_{j=1}^{C}, \tag{6}$$

where $\hat{p}(x|f_{\theta_0})$ is the empirical class probabilities of f_{θ_0} applied to x, and C is the number of classes.

Finally, the optimization objective (3) is implemented using the standard cross-entropy loss, $\text{CE}(X_l, Y_l|f_\theta) \triangleq \frac{1}{L}\sum_{i=1}^{L} -\log \hat{p}_{y_i}(x_i|f_\theta)$, as well the proposed $\mathcal{S}_f$, δ_f and κ_f,

$$\mathcal{L}(X_l, Y_l, X_u|f_\theta, \mathcal{S}_f, \delta_f, \kappa_f) = \text{CE}(X_l, Y_l|f_\theta) + \lambda \cdot \mathcal{L}_{\text{TransBoost}}(X_u|f_\theta, \mathcal{S}_f, \delta_f, \kappa_f), \tag{7}$$

where $\lambda \in \mathbb{R}$ is a regularization hyperparameter.

4.2 A Large Margin Analogy of TransBoost

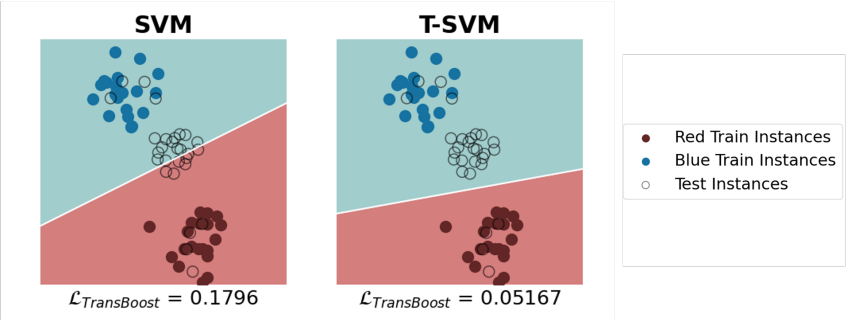

Figure 2: A toy quantitative example indicates that our $\mathcal{L}_{\text{TransBoost}}$ encourages behavior similar to TSVM.

TransBoost is inspired by TSVM, in which a large margin principle is applied to the given (unlabeled) test sample to leverage the knowledge of this specific test set. In Figure 2 we present a toy quantitative (graphical) example indicating that our transductive loss component, $\mathcal{L}_{\text{TransBoost}}$, encourages behavior similar to TSVM. Consider Figure 2 (left) showing the decision boundary obtained by SVM for the given training set (and ignoring the test points). Using SVM to implement δ (to determine if two test points are in the same class), we get that $\mathcal{L}_{\text{TransBoost}} = 0.1796$ (shown at the bottom). Clearly, this loss value is significantly larger than the corresponding loss (0.05167) obtained when using TSVM (right), where a large margin principle of test instances is applied.

[1] $\mathcal{S}_f(x_i, x_j) = 0$ iff $\hat{p}(x_i|f_\theta)$ and $\hat{p}(x_j|f_\theta)$ are one-hot vectors where the ones indicate different classes.

5 Empirical Study

In this section we present a comprehensive empirical study of TransBoost. We begin by describing our experimental design and then present our studies in detail.

5.1 Experimental Design and Details

This section describes the experimental details: datasets used, architectures used, and our TransBoost's procedure details and hyperparameters.

Datasets and Preprocessing. Most of our study of TransBoost is done using the well-known ImageNet-1k ILSVRC-2012 dataset [19], which contains 1,281,167 training instances and 50,000 test instances in 1,000 categories. Preprocessing for training is standard and includes resizing and random cropping to 224×224×3, followed by a random horizontal flip. For testing, the images are resized and center cropped. In some of our experiments involving advanced architectures from the Timm repository [20], we applied the test augmentations (at test time only) using its code from the Timm repository. Additionally, we investigated the effectiveness of TransBoost on the Food-101 [21], CIFAR-10 and CIFAR-100 [22], SUN-397 [23], Stanford Cars [24], FGVC Aircraft [25], the Describable Textures Dataset (DTD) [26] and Oxford 102 Flowers [27]. For each of these datasets, we applied the same preprocessing as in ImageNet, resulting in an image size of 224×224×3, with the exception of CIFAR-10 and CIFAR-100, where we used random cropping with padding (cropping to 32×32×3), followed by a random horizontal flip, a single RandAugment [28] operation and MixUp [29].

Architectures. We examined various inductive (standard) pretrained architectures as baselines for TransBoost. Specifically, we used a variety of ResNet [30] architectures: ResNet18, ResNet34, ResNet50, ResNet101 and ResNet152, which were pretrained using the standard PyTorch training recipe [31]. We also considered ResNet50-StrikesBack [32], which is the ResNet50 architecture trained with Timm's [20] advanced training recipe. Additionally, we show results with more advanced architectures spanning various families: EfficientNetB0 [33] and MobileNetV3-L [8], ConvNext-T [7], and the Transformer models ViT-S [9] and Swin-T [34], which were pretrained with advanced procedures of Timm's library [20] in accordance to their original papers.

Details on the TransBoost Fine-Tuning Procedure. Throughout all TransBoost's experiments, we followed Algorithm 1, and our functions were implemented as described in Section 4.1. We used the LAMB optimizer [35] with a total batch size of 1024 (including 512 labeled training instances and 512 unlabeled test instances) for 60 epochs, unless otherwise specified. Following [35], our learning rate is scaled using a square-root scaling law, specifically, $8.25 \times 10^{-6} \cdot \sqrt{\text{batch size}}$. The learning rate starts at 2.64×10^{-4} $(= 8.25 \times 10^{-6} \cdot \sqrt{1024})$ with no warmup and then decayed with a cosine decay schedule. A weight decay of 10^{-4} is used. The regularization hyperparameter of our loss in all our experiments was fixed to $\lambda = 1$; see Equation (7). Our hyperparameters were fine-tuned based on a validation set that was sampled from the training set over ImageNet.

5.2 ImageNet Experiments: Transductive vs. Inductive

This study examines the **transductive** performance of TransBoost applied to various pretrained models (all described in Section 5.1), and the transductive performance of the other baselines (SSL and t-FSL) versus the **inductive** performance of all baselines. Table 1 presents all the results. We note that some of these results were highlighted in Figure 1.

Table 1 is divided into several horizontal sections. In the first section we present TransBoost applied to a number of ResNet architectures. In the second section, we discuss modern architectures, including two types of Vision Transformers, and also ConvNext-T as representative of the ConvNext family, which presently achieves state-of-the-art results over ImageNet [14]. Next we present the SimCLR and SimCLRv2 SSL methods (see Appendix B for more details on how we trained them in transduction). We note that SimCLRv2 presently dominates SSL performance over ImageNet according to [14]. The last table section considers two popular t-FSL methods (which are described

Method	Architecture	Params (M)	Inductive / Transductive Top1 Acc. (%)	Improv. (%)
ResNet [30] architectures that were pretrained using standard simple procedures:				
TransBoost	ResNet18	11.69	69.76 / **73.91**	+4.15
	ResNet34	21.80	73.29 / **76.75**	+3.46
	ResNet50	25.56	76.15 / **79.11**	+2.96
	ResNet101	44.55	77.37 / **79.88**	+2.51
	ResNet152	60.19	78.33 / **80.64**	+2.31
Advanced vision architectures that were pretrained using advanced procedures:				
TransBoost	EfficientNetB0 [33]	5.29	77.70 / **78.91**	+1.21
	MobileNetV3-L [8]	5.48	75.78 / **77.50**	+1.72
	ResNet50-StrikesBack [32]	25.56	80.38 / **81.33**	+0.95
	ConvNext-T [7]	28.59	82.05 / **82.52**	+0.47
	ViT-S [9]	22.05	81.39 / **83.41**	+2.02
	Swin-T [34]	28.29	81.38 / **82.35**	+0.97
Baseline representatives of semi-supervised learning algorithms in transduction:				
SimCLR [11]	ResNet50	25.56	**76.15** / 75.95	-0.20
	ResNet152	60.19	**78.33** / 78.00	-0.33
SimCLRv2 [12]	ResNet50	25.56	**76.15** / 74.68	-1.47
	ResNet152	60.19	**78.33** / 76.64	-1.69
Baseline representatives of transductive few-shot learning algorithms:				
Ent-min [15]	ResNet50	25.56	76.15 / **77.55**	+1.40
TIM-GD [16]	ResNet50	25.56	**76.15** / 71.51	-4.64

Table 1: **Inductive vs. Transductive.** A comprehensive analysis of the transductive performance on ImageNet compared to the standard inductive (fully supervised) performance. TransBoost's performance is highlighted in green, SSL's performance is highlighted in blue, and t-FSL's performance is highlighted in yellow.

in Section 3). The column structure of Table 1 is straightforward but the most important column that deserves some explanation is the *Inductive / Transductive* column (4th column) in which we compare the transductive performance of each experiment to the relevant inductive fully supervised performance. For example, TransBoost is capable of increasing the ImageNet top-1 accuracy of ViT-S by two percent.

Table 1 and Figure 1 demonstrate that fine-tuning with TransBoost consistently and significantly improves the inductive top-1 accuracy performance in all cases. Even the tiny version of the top performing ConvNext architecture family (presently the state-of-the-art according to [14]) is improved by TransBoost. Surprisingly, ViT-S performed better than ConvNext-T when taking advantage of the given test samples using TransBoost, indicating that members of the ViT family could be the best transductive performers on ImageNet. A second interesting fact is that ViT-S and Swin-T perform similarly in the inductive setting while the ViT-S top-1 accuracy gain is improved by over x2 relative to Swin-T. Could this advantage be related to the architecture? Interestingly, TransBoost also improves the performance of ResNet50-StrikesBack, which is the standard ResNet50 architecture trained using a sophisticated training procedure [32] that boosts its top-1 accuracy by +4.2%. Surprisingly, TransBoost adds almost +1% on top of this phenomenal improvement. Another striking related result is that TransBoost improves the standard ResNet50 so that it achieves performance close to ResNet50-StrikesBack despite the fact that our procedure is much simpler than the sophisticated training procedure of [32].

Consider the first section of Table 1 (the ResNets section). While TransBoost always improves the inductive baselines, it is evident that its relative advantage monotonically decreases with the model size/performance. For example, the best transductive improvement is achieved for ResNet18 yielding a top-1 accuracy gain of +4.15% and resulting in a top-1 accuracy of 73.91%[2]. With respect to the other related baselines, TransBoost consistently outperforms both the SSL and t-FSL baseline algorithms. Comparing the results of SimCLR to the results of SimCLRv2, we see that SimCLR

[2]This ResNet18 transductive performance even outperforms the current state-of-the-art inductive performance [14] for ResNet18 on ImageNet (73.19%) [36], which is achieved with a fancy training recipe.

outperforms SimCLRv2 in the transductive setting (they mainly differ by the additional stage of knowledge distillation [37] that is added to SimCLRv2), which suggests that the stage of knowledge distillation can hinder the transductive performance. We hypothesize that distilling knowledge from test instances leads to overfitting of incorrect class predictions that lead to model confusion. Finally, when evaluating the t-FSL representatives, we observe that Ent-min improves its inductive baseline while TIM-GD (which is an upgraded version of Ent-min) is inferior to its inductive baseline.

5.3 TransBoost's Performance on Additional Vision Datasets

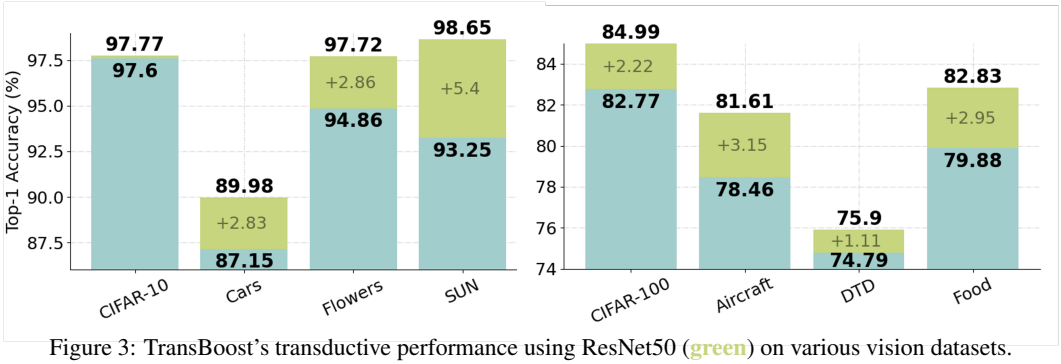

Figure 3: TransBoost's transductive performance using ResNet50 (green) on various vision datasets.

We now examine the performance of TransBoost across eight other datasets (all described in Section 5.1). Throughout this study, we start with a ResNet50 architecture pretrained on ImageNet (using PyTorch's standard training recipe). Then we apply inductive fine-tuning for each specific dataset and, finally, we apply our TransBoost procedure.

The results are visually depicted in the graph of Figure 3 and are also shown in Table 3 in the Appendix. Clearly, TransBoost outperforms its inductive baseline in all cases. The improvement is large and significant in all datasets with the exception of CIFAR-10 where the improvement is minor. The most prominent result is obtained for the SUN-397 dataset, where we observe a striking +5.4% top-1 accuracy gain from 93.25% to 98.65%. To the best of our knowledge, this is presently the best reported result for SUN-397. For a more in-depth discussion, we refer the reader to Appendix C.

5.4 Transductive Performance Improves with Test Set Size

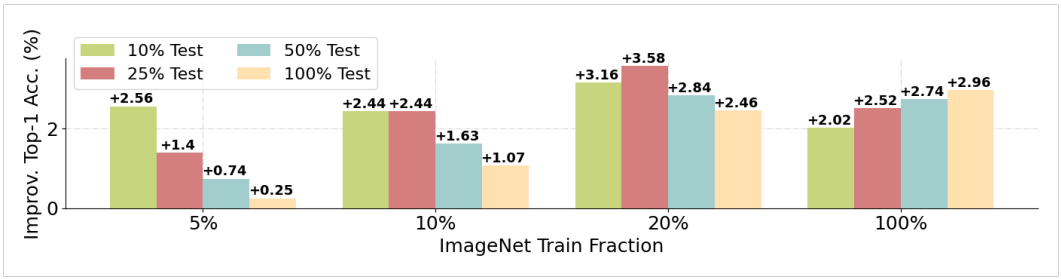

Figure 4: TransBoost's transductive performance using ResNet50 on ImageNet training and test subsets.

In this study, we consider various transductive settings, where each setting is characterized by a certain training set size and a certain test set size. We are interested in examining the final ImageNet transductive behavior of ResNet50 as a function of these sizes. The sizes of our training sets are: 5%, 10%, 20%, 100%; and for testing sets are: 10%, 25%, 50%, 100%, all taken as a fraction of the original train/validation set sizes, respectively.

Consider Figure 4 showing the top-1 accuracy gains for all combinations (4×4 experiments in total). Importantly, we observe that whenever TransBoost uses the entire training set, its performance increases as the test set size grows. This strongly indicates that TransBoost leverages the test set as a whole, a desirable property of a transductive learning algorithm. On the other hand, and quite surprisingly, when TransBoost only used 5% of the training set, increasing the test set size worsened

its performance. We hypothesize that this type of behavior is a result of poor prior knowledge (very small training set) that leads the models to wrongly analyze the relationships (similarities in our case) between pairs. This overall bad effect increases as the test set size expands.

In addition to the observations on the effect of a sufficiently large training set, we also observe the importance of a sufficiently large test when considering the performance of TransBoost that used 100% of the test set (marked in yellow in Figure 4). It is evident that with this largest test set, performance monotonically increases with the training set size. For a detailed description/discussion of this section's experiments, see Appendix D.

5.5 Bringing the Like Together or Separating the Difference?

TransBoost is driven by Equation (2) whose informal desideratum is: revise the model so as to take apart the representations of two instances that are likely (κ) to belong to different classes (δ), which currently appear to be similar ($\mathcal{S}$). Rather than, or in addition to separating instances that are likely to be different, we could consider grouping together instances that are likely to belong to the same class. In this section we summarize our experiments to apply TransBoost for this alternative objective and for a combination of both objectives. In our study we used the ResNet50 architecture, which was pretrained on ImageNet (using PyTorch's standard training recipe). The transductive performance of TransBoost was analyzed using three variants of the transductive loss component: (1) the proposed separation component, $\mathcal{L}_{\text{TransBoost}}(X_u|\mathcal{S}_f, \delta_f, \kappa_f)$, as in Equation (7); (2) a "reciprocal" instantiation of this objective – namely, $\mathcal{L}_{\text{TransBoost}}(X_u|\sqrt{2} - \mathcal{S}_f, 1 - \delta_f, \kappa_f)$, which brings together similar instances; and (3) a variant that considers both these objectives together. The results are presented in Table 2.

Transductive Loss	**Inductive / Transductive** Top1 Acc. (%)	Improv. (%)
$\mathcal{L}_{\text{TransBoost}}(X_u\|\mathcal{S}_f, \delta_f, \kappa_f)$	76.15 / **79.11**	+2.96
$\mathcal{L}_{\text{TransBoost}}(X_u\|\sqrt{2} - \mathcal{S}_f, 1 - \delta_f, \kappa_f)$	**76.15** / 75.14	-1.01
$\mathcal{L}_{\text{TransBoost}}(X_u\|\mathcal{S}_f, \delta_f, \kappa_f) + \mathcal{L}_{\text{TransBoost}}(X_u\|\sqrt{2} - \mathcal{S}_f, 1 - \delta_f, \kappa_f)$	76.15 / **79.00**	+2.85

Table 2: Performance of transductive loss variants using ResNet50. Our proposed loss is highlighted in **green**.

Clearly, the best performance is achieved using our proposed implementation (row highlighted in green). Interestingly, the reciprocal variation significantly degrades performance and the combination of both objectives slightly reduces the top-1 accuracy.

6 Concluding Remarks

We presented TransBoost, a novel and powerful transductive fine-tuning procedure that can be efficiently applied on any pretrained model. TransBoost is a deep transductive classification algorithm that is inspired by a large margin principle such as the TSVM algorithm. Strikingly, TransBoost consistently and significantly improves the inductive ImageNet top-1 performance of many architectures including the most advanced ones. Moreover, TransBoost is effective on a broad range of image classification datasets.

As described in Algorithm 1, TransBoost is a general optimization procedure that can be implemented in a range of ways depending on the similarity function ($\mathcal{S}$), the selection function (δ), and the confidence function (κ). We instantiated these functions simply and straightforwardly. A question for future research may be to ask whether more sophisticated choices will lead to better transduction performance. The ViT-S architecture achieved the best transductive classification performance in our experiments and accrued the largest gain relative to induction among advanced architectures. It would be interesting to explore the question of how this Transformer architecture facilitates this gain. In general, we can ask what is the best architecture for transductive classification. Finally, it would be very interesting to see whether such large performance gains are also sustained in NLP applications.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A] .
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] All used models are provided at online repositories. The code for our detection method will be provided upon acceptance.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Section 5.1.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [N/A]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

References

[1] Philip Derbeko, Ran El-Yaniv, and Ron Meir. Explicit learning curves for transduction and application to clustering and compression algorithms. *Journal of Artificial Intelligence Research*, 22:117–142, 2004.

[2] Ran El-Yaniv and Dmitry Pechyony. Transductive rademacher complexity and its applications. *Journal of Artificial Intelligence Research*, 35:193–234, 2009.

[3] Ran El-Yaniv, Dmitry Pechyony, and Vladimir Vapnik. Large margin vs. large volume in transductive learning. *Machine Learning*, 72(3):173–188, 2008.

[4] Vladimir Vapnik. *The nature of statistical learning theory*. Springer science & business media, 1999.

[5] Thorsten Joachims et al. Transductive inference for text classification using support vector machines. In *Icml*, volume 99, pages 200–209, 1999.

[6] Corinna Cortes and Vladimir Vapnik. Support vector machine. *Machine learning*, 20(3):273–297, 1995.

[7] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. *arXiv preprint arXiv:2201.03545*, 2022.

[8] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, et al. Searching for mobilenetv3. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1314–1324, 2019.

[9] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021.

[10] Vladimir Vapnik. *Estimation of dependences based on empirical data*. Springer Science & Business Media, 2006.

[11] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.

[12] Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey E Hinton. Big self-supervised models are strong semi-supervised learners. *Advances in neural information processing systems*, 33:22243–22255, 2020.

[13] Mahmoud Assran, Mathilde Caron, Ishan Misra, Piotr Bojanowski, Armand Joulin, Nicolas Ballas, and Michael Rabbat. Semi-supervised learning of visual features by non-parametrically predicting view assignments with support samples. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8443–8452, 2021.

[14] Papers with code: The latest in machine learning. `https://paperswithcode.com`.

[15] Guneet Singh Dhillon, Pratik Chaudhari, Avinash Ravichandran, and Stefano Soatto. A baseline for few-shot image classification. In *International Conference on Learning Representations*, 2019.

[16] Malik Boudiaf, Imtiaz Ziko, Jérôme Rony, José Dolz, Pablo Piantanida, and Ismail Ben Ayed. Transductive information maximization for few-shot learning. *Advances in Neural Information Processing Systems*, 33:2445–2457, 2020.

[17] Michalis Lazarou, Tania Stathaki, and Yannis Avrithis. Iterative label cleaning for transductive and semi-supervised few-shot learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8751–8760, 2021.

[18] Few-shot classification leaderboard. `https://fewshot.org`.

[19] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

[20] Ross Wightman. Pytorch image models. `https://github.com/rwightman/pytorch-image-models`, 2022.

[21] Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101–mining discriminative components with random forests. In *European conference on computer vision*, pages 446–461. Springer, 2014.

[22] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[23] Jianxiong Xiao, James Hays, Krista A Ehinger, Aude Oliva, and Antonio Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *2010 IEEE computer society conference on computer vision and pattern recognition*, pages 3485–3492. IEEE, 2010.

[24] Jonathan Krause, Jia Deng, Michael Stark, and Li Fei-Fei. Collecting a large-scale dataset of fine-grained cars. 2013.

[25] Subhransu Maji, Esa Rahtu, Juho Kannala, Matthew Blaschko, and Andrea Vedaldi. Fine-grained visual classification of aircraft. *arXiv preprint arXiv:1306.5151*, 2013.

[26] Mircea Cimpoi, Subhransu Maji, Iasonas Kokkinos, Sammy Mohamed, and Andrea Vedaldi. Describing textures in the wild. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3606–3613, 2014.

[27] Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *2008 Sixth Indian Conference on Computer Vision, Graphics & Image Processing*, pages 722–729. IEEE, 2008.

[28] Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. Randaugment: Practical automated data augmentation with a reduced search space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 702–703, 2020.

[29] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations*, 2018.

[30] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[31] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[32] Ross Wightman, Hugo Touvron, and Hervé Jégou. Resnet strikes back: An improved training procedure in timm. *arXiv preprint arXiv:2110.00476*, 2021.

[33] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pages 6105–6114. PMLR, 2019.

[34] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10012–10022, 2021.

[35] Yang You, Jing Li, Sashank Reddi, Jonathan Hseu, Sanjiv Kumar, Srinadh Bhojanapalli, Xiaodan Song, James Demmel, Kurt Keutzer, and Cho-Jui Hsieh. Large batch optimization for deep learning: Training bert in 76 minutes. In *International Conference on Learning Representations*, 2020.

[36] Zhiqiang Shen and Marios Savvides. Meal v2: Boosting vanilla resnet-50 to 80%+ top-1 accuracy on imagenet without tricks. *arXiv preprint arXiv:2009.08453*, 2020.

[37] Geoffrey Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. In *NIPS Deep Learning and Representation Learning Workshop*, 2015.

A The Similarity Function is Bounded

In the following, we claim and prove that $\mathcal{S}_f$ is bounded.

Claim 1. *Let $f : \mathcal{X} \to \{1, \dots C\}$ be a classification model and $x_1, x_2 \in \mathcal{X}$ input instances, where C is the number of classes. Accordingly, our implementation of TransBoost's symmetric pairwise similarity function, $\mathcal{S}_f$ (4), satisfies the following:*

$$0 \leq \mathcal{S}_f(x_1, x_2) \leq \sqrt{2}.$$

Proof.

Recalling our implementation of $\mathcal{S}_f$ (4),

$$\mathcal{S}_f(x_1, x_2) \triangleq \sqrt{2} - \|\hat{p}(x_1|f) - \hat{p}(x_2|f)\|_2 \,,$$

where $\hat{p}(x|f)$ is the empirical class probabilities of f applied on x. Since $|| \cdot ||$ is non-negative, we obtain $\mathcal{S}_f(x_1, x_2) \leq \sqrt{2}$. Now we need to show that $\mathcal{S}_f(x_1, x_2) \geq 0$, which holds iff, $\|\hat{p}(x_1|f) - \hat{p}(x_2|f)\|_2 \leq \sqrt{2}$, iff

$$\|\hat{p}(x_1|f) - \hat{p}(x_2|f)\|_2^2 \leq 2.$$

Considering the left-hand side of the inequality,

$$\|\hat{p}(x_1|f) - \hat{p}(x_2|f)\|_2^2 = \|\hat{p}(x_1|f)\|_2^2 + \|\hat{p}(x_2|f)\|_2^2 - 2\langle \hat{p}(x_1|f), \hat{p}(x_2|f)\rangle.$$

Following a sub case of Hölder inequality, $\forall v \in \mathbb{R}^n$; $||\boldsymbol{v}||_2 \leq ||\boldsymbol{v}||_1$. Therefore, using the last equation we obtain:

$$\begin{aligned}
\|\hat{p}(x_1|f) - \hat{p}(x_2|f)\|_2^2 &= \|\hat{p}(x_1|f)\|_2^2 + \|\hat{p}(x_2|f)\|_2^2 - 2\langle \hat{p}(x_1|f), \hat{p}(x_2|f)\rangle \\
&\leq \|\hat{p}(x_1|f)\|_1^2 + \|\hat{p}(x_2|f)\|_1^2 - 2\langle \hat{p}(x_1|f), \hat{p}(x_2|f)\rangle \\
&= 2 - 2\langle \hat{p}(x_1|f), \hat{p}(x_2|f)\rangle \qquad (8) \\
&\leq 2. \qquad (9)
\end{aligned}$$

Where (8, 9) hold since $\hat{p}$ is a probability vector. Hence for any $x \in \mathcal{X}$, $\|\hat{p}(x|f)\|_1 = \sum_{i=1}^{C} |\hat{p}_i(x|f)| = 1$ and $\forall 1 \leq i \leq C$; $0 \leq \hat{p}_i(x|f) \leq 1$. ■

B Transduction Training Details of SimCLR and SimCLRv2

The SimCLR [11] algorithm consists of self-supervised pretraining and supervised fine-tuning. SimCLRv2 [12] incorporates approximately the same steps, but adds a final knowledge distillation [37] stage that has been found powerful in inductive SSL (see [12] for further details).

SimCLR (and SimCLRv2), however, are designed for SSL rather than transduction; therfore, we will now elaborate how they were trained for transduction. As a first point, we note that both SimCLR and SimCLRv2 were applied to the ResNet50 and ResNet152 architectures using pretrained weights provided by the authors. Next, we fine-tuned the model, using both the labeled training set as well as the unlabeled test set, for another 100 epochs. We followed the pretraining scheme as suggested by the authors. In the second step of supervised fine-tuning, we followed the SimCLRv2 instructions given by the authors. For the final stage (self-distillation), SimCLRv2 optimizes a weighted loss function based on labeled training instances applied to standard CE and unlabeled training instances applied to the CE distillation loss. In our case, the unlabeled set is the transductive set.

C Supplement for the Additional Vision Datasets Experiments

Here we discuss the details in the experiments on TransBoost using the additional vision datasets that were introduced in Section 5.3, as well as an additional observation offered in Appendix C.1. Table 3 compares the TransBoost performance to the standard inductive (fully supervised) performance and highlights some relevant properties of the datasets we used.

Dataset	Train (+Validation)	Test	Classes	Resolution	Inductive / Transductive Top1 Acc. (%)	Improv. (%)
CIFAR-10 [22]	50,000	10,000	10	32	97.60 / **97.77**	+0.17
CIFAR-100 [22]	50,000	10,000	100	32	82.77 / **84.99**	+2.22
Stanford Cars [24]	8,144	8,014	196	224	87.15 / **89.98**	+2.83
Flowers-102 [27]	7,169	1,020	102	224	94.86 / **97.72**	+2.86
SUN-397 [23]	87,003	21,750	397	224	93.25 / **98.65**	+5.40
FGVC Aircraft [25]	6,800	3,400	102	224	78.46 / **81.61**	+3.15
DTD [26]	3,760	1,880	47	224	74.79 / **75.90**	+1.11
Food-101 [21]	75,750	25,250	101	224	79.88 / **82.83**	+2.95

Table 3: A study comparing TransBoost transductive performance to standard inductive (fully supervised) performance on various vision datasets (green) using ResNet50. TransBoost's best result is highlighted in yellow.

C.1 Transductive Performance Gain Improves as the Number of Classes Grows

Table 3 displays an interesting pattern regarding the number of classes. For experiments conducted using both non-CIFAR and CIFAR groups, we observe that TransBoost performs better on datasets with many classes than on datasets with a few classes. Specifically, among non-CIFAR experiments, it can be seen that the highest top-1 accuracy gain is achieved on SUN-397 while the experiment on DTD attained the smallest improvement. SUN-397 and DTD have the largest and the smallest number of classes, respectively, among non-CIFAR datasets. This effect is also observed when the CIFAR datasets are used. On the basis of these observations, we hypothesize that the more classes the dataset has, the more likely similar representations, which belong to different classes, will be found and, therefore, there are more target test instances to improve.

D Supplement for the ImageNet Subsets Experiments

This section provides a supplementary in-depth discussion of our experiments on ImageNet training and test subsets, which are varying in their sizes (introduced in Section 5.4). In this study, we used the ResNet50 architecture that was pretrained on ImageNet combinations of training and test subsets (using PyTorch's standard training recipe). For experiments that do not utilize the entire training set, we applied the TransBoost procedure with a batch size of 128. Although TransBoost was optimized for a specific test set, here we evaluate its performance in various inductive settings, where unseen test instances are examined at test time. Additionally, we present a comprehensive analysis of TransBoost's performance in various transductive settings as well as a standard inductive (fully supervised) performance analysis as a baseline. The complete results are presented in Table 4.

Table 4 presents 16 experiments of TransBoost's procedure performed on all combinations of the ImageNet training set fractions: 5%, 10%, 20%, 100%; and the ImageNet test set fractions: 10%, 25%, 50%, 100%. The table's sections represent training set fractions, and the rows in each section represent test set fractions. There are two main column groups highlighted in green and blue. Green columns present comprehensive results in transductive settings, whereas blue columns present comprehensive results in inductive settings. Across the two settings (transductive / inductive), TransBoost's performance is compared to the standard inductive (fully supervised) performance. For instance, using 20% of the training set and 25% of the test set, we obtained the best top-1 accuracy gain of +3.58% in the transductive setting while the performance in the inductive setting degraded by -1.21%. We note that the performance in the inductive setting (highlighted in blue) is evaluated using the test instances that were not used at training time.

As can be seen, TransBoost consistently outperforms the baselines in transductive settings while the baselines consistently outperform TransBoost in inductive settings. This behavior is indeed what one may expect from real transductive models, which will only be asked for the given test set. Interestingly, TransBoost's best top-1 accuracy gain – an impressive improvement of +3.58% – was

ImageNet Fraction / Train	Instances Per Class Test	Inductive Baseline / Transductive TransBoost Top-1 (%)	Improv. (%)	Inductive Baseline / Inductive TransBoost Top-1. (%)	Improv (%)
5% / 64	10% / 5	35.68 / **38.24**	+2.56	**36.08** / 35.02	-1.06
	25% / 13	36.03 / **37.43**	+1.40	**36.04** / 35.22	-0.82
	50% / 25	36.30 / **37.04**	+0.74	**35.78** / 35.23	-0.55
	100% / 50	36.04 / **36.29**	+0.25	-	-
10% / 128	10% / 5	50.70 / **53.14**	+2.44	**50.27** / 48.34	-1.93
	25% / 13	50.39 / **52.83**	+2.44	**50.29** / 48.94	-1.35
	50% / 25	50.63 / **52.26**	+1.63	**50.00** / 49.10	-0.90
	100% / 50	50.32 / **51.39**	+1.07	-	-
20% / 256	10% / 5	62.20 / **75.38**	+3.16	**62.20** / 60.72	-1.48
	25% / 13	61.78 / **65.36**	+3.58	**62.35** / 61.14	-1.21
	50% / 25	62.22 / **65.06**	+2.84	**62.18** / 61.40	-0.78
	100% / 50	62.20 / **64.66**	+2.46	-	-
100% / ∼1300	10% / 5	75.38 / **77.40**	+2.02	**76.24** / 75.18	-1.06
	25% / 13	75.53 / **78.05**	+2.52	**76.36** / 75.45	-0.91
	50% / 25	75.91 / **78.65**	+2.74	**76.38** / 75.39	-0.99
	100% / 50	76.15 / **79.11**	+2.96	-	-

Table 4: A comprehensive analysis of TransBoost's performance on ImageNet subsets, both in transduction and induction, compared to the standard inductive (fully supervised) performance using ResNet50. The performance of TransBoost in transduction is highlighted in green and the performance of TransBoost in induction is highlighted in blue.

achieved by the experiment that used 20% of the training set and 25% of the test set. Our best top-1 accuracy (79.11%) was attained in the experiment that used the entire training and test sets. For further details, we refer the reader to Appendix D.1 and Appendix D.2, below.

D.1 TransBoost's Inductive Performance Increases with Test Set Size

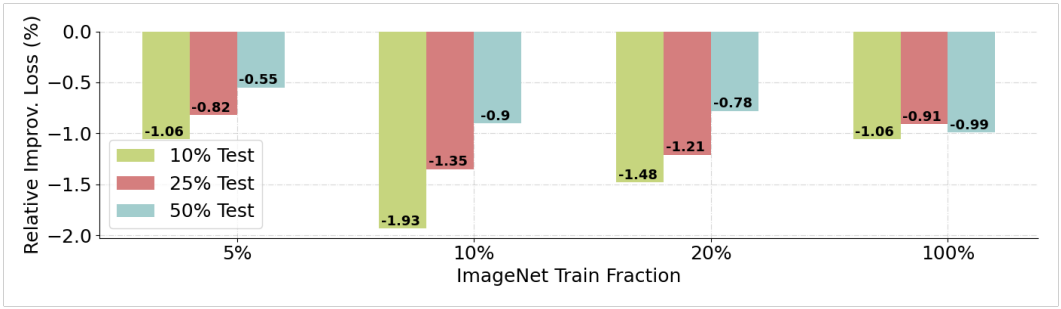

Figure 5: TransBoost's inductive performance using ResNet50 on ImageNet training and test subsets.

In Figure 5, we present an interesting trend of TransBoost's top-1 accuracy gains in various inductive settings (derived from Table 4). Note that the fractions described in the figure legend refer to the test instances that participated in the training procedure, while the inductive performance was examined using the rest of the instances. We have already discussed the fact that all inductive baselines outperform TransBoost in all inductive settings. This can be expected from a transductive model that specializes in a particular test set. Figure 5, however, reveals an interesting pattern when the entire training set is not used: the more test instances are provided at training time, the better TransBoost's performance in the inductive setting. This observation implies that our proposed optimization procedure learns better to generalize unseen instances when the ratio of test instances (out of the training and test instances combined) is significant.

D.2 TransBoost's Loss is Consistently Improved

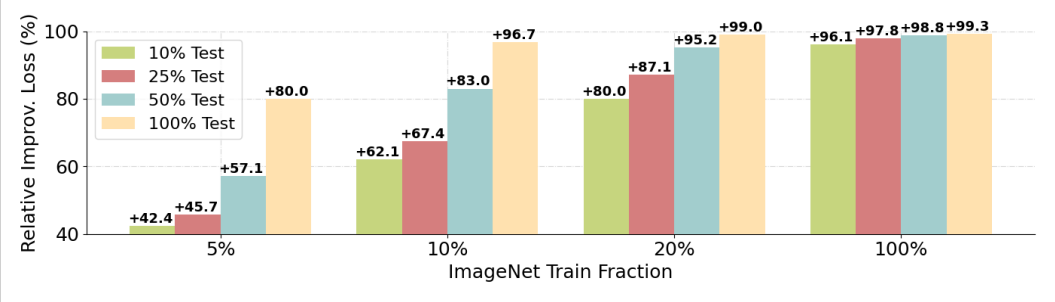

Figure 6: Relative improvements in our proposed loss component, $\mathcal{L}_{\text{TransBoost}}$, using ResNet50 on ImageNet training and test subsets.

Here we provide an additional analysis of the relative improvement in our proposed loss component, $\mathcal{L}_{\text{TransBoost}}$ (see Section 4.1), across the experiments that we described above. Specifically, we approximated TransBoost's loss following Algorithm 1 (the approximation stage only) on the inductive baselines as well as on TransBoost's output models in the transductive setting. Then, we calculated their relative improvements in (%) and presented the results in Figure 6. For example, our best relative improvement in $\mathcal{L}_{\text{TransBoost}}$ was given by the experiment that used the entire training and test sets with a relative improvement of +99.3%.

As shown in Figure 6, the larger the training and/or test sets TransBoost uses, the larger is the improvement of the loss $\mathcal{L}_{\text{TransBoost}}$. This suggests that the greater the prior knowledge of the model is, the easier it is to optimize $\mathcal{L}_{\text{TransBoost}}$. Moreover, the more test instances we use in the TransBoost optimization procedure, the more $\mathcal{L}_{\text{TransBoost}}$ does improve. We contend that this is how a transductive procedure should behave, since the test instances are used as a group rather than treating test instances independently.