Self-Supervision Can Benefit Fully-Supervised Classification Tasks

By

ROEE CATES

Under the supervision of

PROF. DAPHNA WEINSHALL

THE HEBREW
UNIVERSITY
OF JERUSALEM

Faculty of Computer Science and Engineering
THE HEBREW UNIVERSITY OF JERUSALEM

A dissertation submitted to the Hebrew University of Jerusalem as a partial fulfillment of the requirements of the degree of MASTER OF SCIENCE in the Faculty of Computer Science and Engineering.

DECEMBER 2020

Abstract

Overfit is a fundamental problem in machine learning in general, and in deep learning in particular. In order to reduce overfit and improve generalization in the classification of images, some employ invariance to a group of transformations that images of certain objects may inherently possess, such as rotations and reflections. However, since not all objects exhibit necessarily the same invariance, it seems desirable to allow the network to learn the useful level of invariance from the data. To this end, motivated by self-supervision, we introduce an architecture enhancement for existing neural network models based on input transformations, termed 'TransNet', together with a training algorithm suitable for it. Our model can be employed during training time only and then pruned for prediction, resulting in an equivalent architecture to the base model. Thus pruned, we show that our model improves performance on various data-sets while exhibiting improved generalization, which is achieved in turn by enforcing soft invariance on the convolutional kernels of the last layer in the base model. Theoretical analysis is provided to support the proposed method.

Dedication and acknowledgements

I wish to thank my academic adviser, Professor Daphna Weinshall, who guided me in this process, this thesis wouldn't been written without you. I also wish to thank my beloved family who supported me throughout my whole academic journey.

Table of Contents

List of Tables

Table **Page**

List of Figures

Figure **Page**

CHAPTER 1

INTRODUCTION

Deep neural network models currently define the state of the art in many computer vision tasks, as well as speech recognition and other areas. These expressive models are able to model complicated input-output relations. At the same time, models of such large capacity are often prone to overfit, *i.e.* performing significantly better on the training set as compared to the test set. This phenomenon is also called the *generalization gap*.

We propose a method to narrow this generalization gap. Our model, which is called *TransNet*, is defined by a set of input transformations. It augments an existing Convolutional Neural Network (CNN) architecture by allocating a specific head - a fully-connected layer which receives as input the penultimate layer of the base CNN - for each input transformation (see Fig. 1.1). The transformations associated with the model's heads are not restricted apriori.

The idea behind the proposed architecture is that each head can specialize in a different yet related classification task. We note that any CNN model can be viewed as a special case of the *TransNet* model, consisting of a single head associated with the identity transformation. The overall task is typically harder when training *TransNet*, as compared to the base CNN architecture. Yet by training multiple heads, which share the convolutional backbone, we hope to reduce the model's overfit by providing a form of regularization.

Chapter 3 presents the approach. In Section 3.1 we define the basic model and the training algorithm designed to train it (see Alg. 1). We then discuss the type of

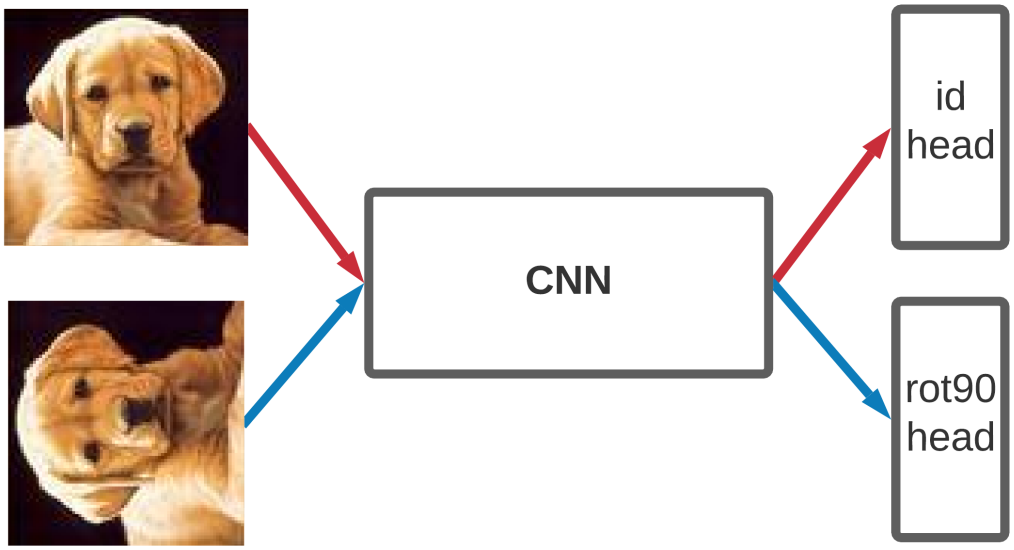

Figure 1.1: An illustration of the *TransNet* architecture, which consists of 2 heads associated with 2 transformations, the identity and rotation by 90°. Each head classifies images transformed with its associated transformation, while both share the same convolutional layers.

transformations that can be useful when learning to classify images. We also discuss the model's variations: (i) pruned version that employs multiple heads during training and then keeps only the head associated with the identity transformation for prediction; (ii) the full version where all heads are used in both training and prediction.

Theoretical investigation of this model is provided in Section 3.2, using the dihedral group of transformations (D_4) that includes rotations by 90^o and reflections. We first prove that under certain mild assumptions, instead of applying each dihedral transformation to the input, one can compile it into the CNN model's weights by applying the inverse transformation to the convolutional kernels. In order to obtain intuition about the inductive bias of the model's training algorithm in complex realistic frameworks, we analyze the model's inductive bias using a simplified framework.

In Chapter 4 we describe our empirical results. We first introduce a novel invariance score (IS), designed to measure the model's kernel invariance under a given group of transformations. IS effectively measures the inductive bias imposed on the model's weights by the training algorithm. To achieve a fair comparison, we compare a regular CNN model traditionally trained, to the same model trained like a *TransNet* model as follows: heads are added to the base model, it is trained as a *TransNet* model, and then the extra heads are pruned. We then show that training as *TransNet* improves test accuracy as compared to the base model. This improvement was achieved while

keeping the optimized hyper-parameters of the base CNN model, suggesting that further improvement by fine tuning may be possible. We demonstrate the increased invariance of the model's kernels when trained with *TransNet*.

1.1 Our Contribution

- Introduce *TransNet* - a self-supervised model for supervised learning that imposes partial invariance to a group of transformations.
- Introduce an invariance score (IS) for CNN convolutional kernels.
- Theoretical investigation of the inductive bias implied by the *TransNet* training algorithm.
- Demonstrate empirically how both the full and pruned versions of *TransNet* improve accuracy.

CHAPTER 2

RELATED WORK & BACKGROUND

In this chapter we survey the various methods of the main topics related to our work: overfit, self-supervised learning and equivariant CNNs. Next, we elaborate on specifically related methods as well as providing the relevant background.

2.1 Related Work

2.1.1 Overfit

A fundamental and long-standing issue in machine learning, overfit occurs when a learning algorithm minimizes the train loss, but generalizes poorly to the unseen test set. Many methods were developed to mitigate this problem, including *early stopping* - when training is halted as soon as the loss over a validation set starts to increase, and *regularization* - when a penalty term is added to the optimization loss. Other related ideas, which achieve similar goals, include dropout [27], batch normalization [14], transfer learning [25, 29], and data augmentation [3, 33].

2.1.2 Self-Supervised Learning

A family of learning algorithms that train a model using self generated labels (*e.g.* the orientation of an image), in order to exploit unlabeled data as well as extract more information from labeled data. Self training algorithms are used for representation learning, by training a deep network to solve pretext tasks where labels can be produced

directly from the data. Such tasks include colorization [16, 32], placing image patches in the right place [7, 22], inpainting [23] and orientation prediction [10]. Typically, self-supervision is used in unsupervised learning [8], to impose some structure on the data, or in semi-supervised learning [12, 31]. Our work is motivated by *RotNet*, an orientation prediction method suggested by [10]. It differs from [12, 31], as we allocate a specific classification head for each self-supervised label rather than predicting the self-supervised label with a separate head.

2.1.3 Equivariant CNNs

Many computer vision algorithms are designed to exhibit some form of invariance to a transformation of the input, including geometric transformations [20], transformations of time [28], or changes in pose and illumination [24]. Equivariance is a more relaxed property, exploited for example by CNN models when translation is concerned. Work on CNN models that enforces strict equivariance includes [1, 2, 6, 9, 21, 26]. Like these methods, our method seeks to achieve invariance by employing weight sharing of the convolution layers between multiple heads. But unlike these methods, the invariance constraint is soft. Soft equivariance is also seen in works like [5], which employs a convolutional layer that simultaneously feeds rotated and flipped versions of the original image to a CNN model, or [30] that appends rotation and reflection versions of each convolutional kernel.

2.2 Background

2.2.1 Self-supervised learning

This field of research deals with training a model to solve a pretext task using self generated labels, *i.e.* labels that can be derived directly from the input of the model. Self-supervised learning is mainly used in an unsupervised framework for representation learning. However, lately it was incorporated in semi-supervised and supervised learning frameworks as discussed below.

2.2.1.1 Unsupervised learning

As mentioned earlier, there are many pretext tasks suggested in the context of self-supervised learning. We will elaborate on the *RotNet* model, suggested by Gidaris et al.

[10]. This method, randomly rotate the input image in multiples of 90° ([0°, 90°, 180°, 270°]). Then the model is trained to predict the rotation applied on the input image - a 4-way classification problem (see Fig. 2.1).

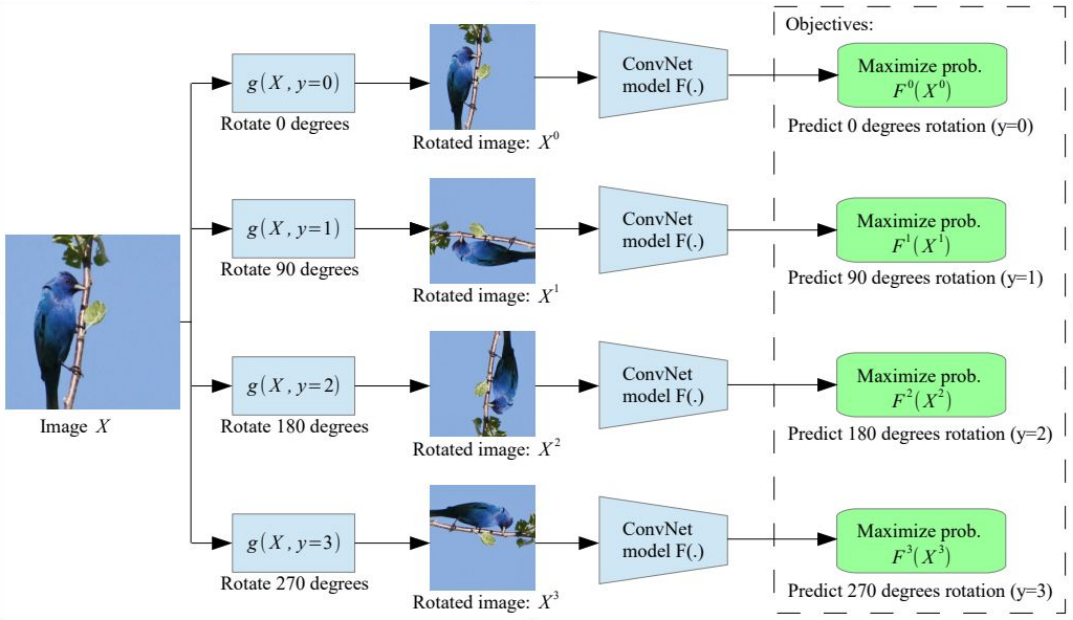

Figure 2.1: Illustration of the self-supervised *RotNet* model which learns to predict the rotation applied on the input image (Image source [10]).

2.2.1.2 Semi-supervised learning

Motivated by the quickly advancing field of self-supervised learning, Zhai et al. [31] suggested to incorporate it to semi-supervised learning, yielding the S^4L model. Their model is a 2 headed CNN model - a classification head as well as a rotation prediction head as suggested by Gidaris et al. [10], each head is a fully-connected layer which receives as input the penultimate layer of the base CNN (see Fig. 2.2). During training - the classification head receives the labeled images while the rotation prediction head is receives all the images (including unlabeled) as it's labels are self generated.

2.2.1.3 Supervised learning

Self-supervision was incorporated to a supervised framework by Hendrycks et al. [12], they suggested a similar model to the S^4L model (see Fig. 2.2) with a different loss

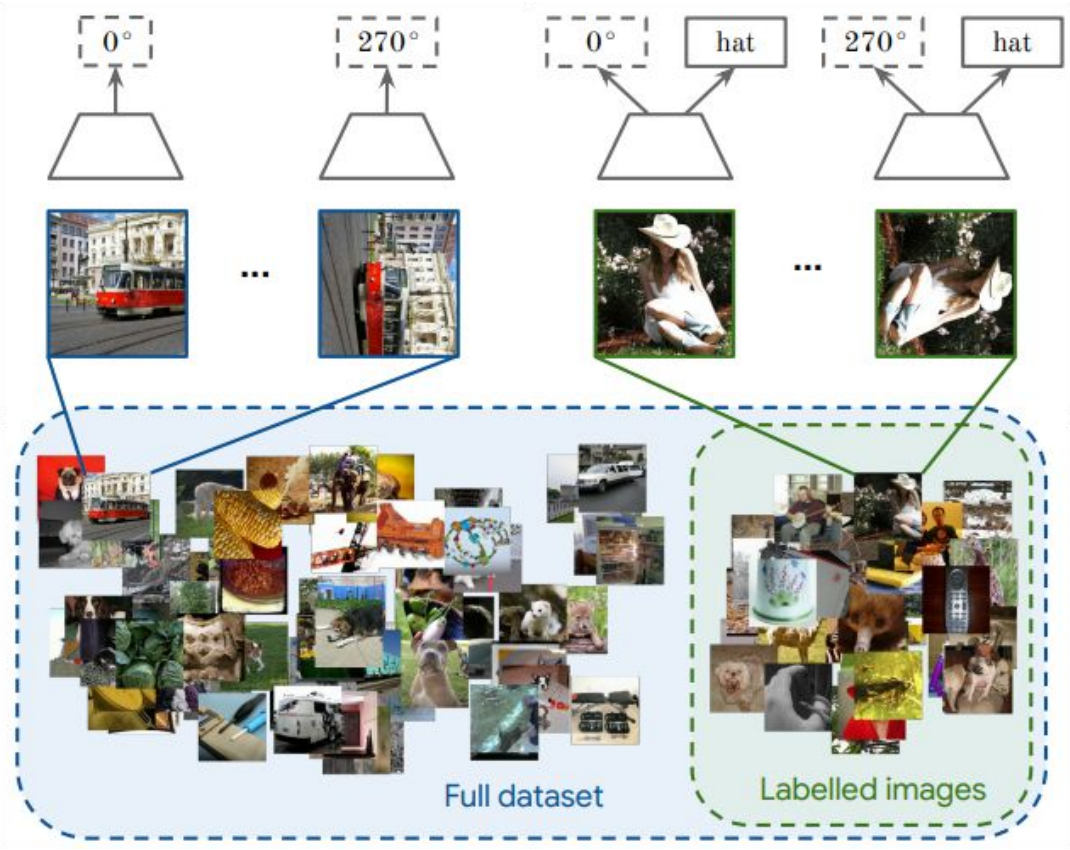

Figure 2.2: Illustration of the S^4L model, which incorporates self-supervised learning (manifested by the rotation prediction head) in a semi-supervised learning framework (Image source [31]).

function defined by:

$$\mathscr{L}(x, y; \boldsymbol{\theta}) = \mathscr{L}_{CE}(y, p(y|PGD(x); \boldsymbol{\theta}) + \lambda \mathscr{L}(PGD(x); \boldsymbol{\theta})$$

where PGD stands for projected gradient descent, a white box attack algorithm which adds a small perturbation to the input aiming to increase the model loss and thus make it predict a wrong class. They show this model improves robustness (to adversarial examples, label and input corruption) as well as out-of-distribution detection.

2.2.2 Equivariant CNNs

Throughout this thesis we use the the terms "invariance" and "equivariance", which can be defined as follows.

Let G denote a group acting on a set X, and let $f : X \to X$ denote a function.

- $x \in X$ is **invariant** under G if $\quad g \circ x = x \quad \forall g \in G$

- f is **invariant** under G if $\quad f(g \circ x) = f(x) \quad \forall x \in A \ \forall g \in G$
- f is **equivariant** under G if $\quad f(g \circ x) = g \circ f(x) \quad \forall x \in A \ \forall g \in G$

We'll demonstrate these concepts and their relation in the context of CNNs *w.r.t.* the following framework: Let X be the set of all $3D$ tensors while f is a convolutional layer represented by a convolutional kernel w_f.

Example 1. *A convolutional layer is equivariant under translation. In that case G is the group of translations. Indeed, the equivariant condition is met - a translated image convolved with a kernel is the same as the translation of that convolved image (up to the added padding in the edges of the feature map).*

Example 2. *These two concepts (invariance and equivariance) share a special case relation, an invariant convolution kernel under a sub-group of the dihedral group D_4 represents an equivariant convolutional layer under that sub-group.* E.g. *let G be the group of the identity and the horizontal reflection transformations, an invariant convolutional kernel under G (*i.e. *w_f is invariant under G) represents an equivariant convolutional layer under G (*i.e. *f is equivariant under G).*

2.2.2.1 Hard vs. soft equivariance

We distinguish between soft and hard equivariance models (where soft equivariance is referred to an approximately equivariance model).

Hard equivariance. Group equivariant convolutional neural networks (G-CNNs), is a representative model exhibiting (hard) equivariance to a specified group (a subgroup of the dihedral group D_4, *e.g.* rotations) was proposed by Cohen and Welling [2]. The G-CNN model is based on G-convolution - a new layer which is equivariant to a specified group, this layer extends the regular convolution layer which is translation equivariant. The G-CNN model processes structured feature maps (this structure is defined by an additional dimension corresponded to the specified group). The G-convolution layer convolves transformed versions of its structured kernels with the structured input, and computes the dot-product of each such convolution, thus resulting again in a structured output (see Fig. 2.3).

Soft equivariance. Several works were proposed to encourage models to exhibit soft equivariance, here we'll mention 2 of them which are akin to our work.

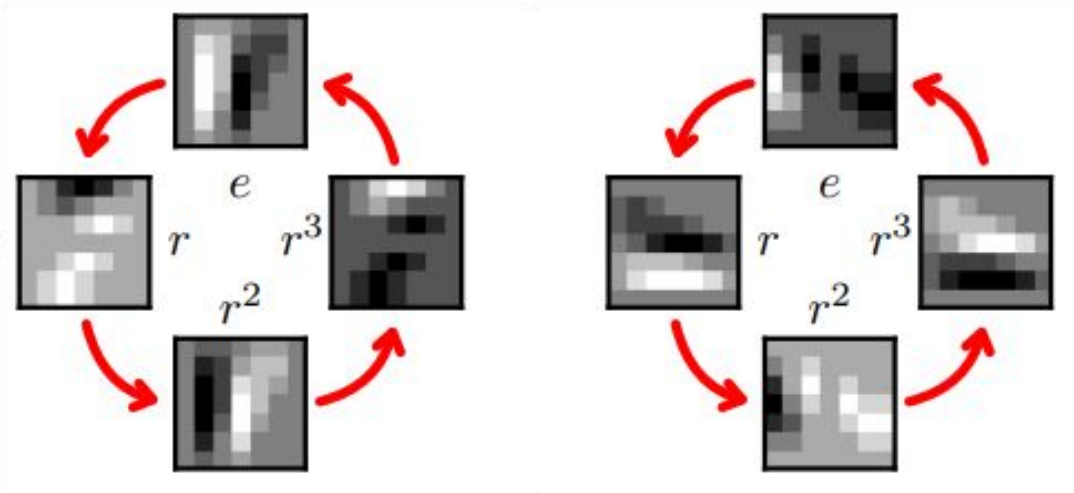

Figure 2.3: 2 structured output feature maps of the G-convolution layer. The right input was rotated by 90°, the right output is equal to the left output up to rotation of the structure (Image source [2]).

Dieleman et al. [5] proposes a CNN architecture which simultaneously feeds cropped rotated versions of the original image to a CNN model and concatenates the results to a single head (see Fig. 2.4). This method encourages parameter sharing *w.r.t.* rotation and translation which improves CNN's performance on the galaxy morphology prediction task.

Wu et al. [30] proposes 2 convolution layers: "rotate-pooling convolution" (RPC) and "flip-rotate-pooling convolution" (FRPC). The RPC layer convolves the input with rotated versions of each convolutional kernel, resulting in 8 feature maps, the multiple feature maps are then max-pooled (channel wise) resulting in a single output feature map. The FRPC layer is similar to the RPC layer except that the input is convolved with 16 versions (the previous 8 and their flip) of the convolutional kernel. The CNN model consists of these layers shows increased robustness under rotations as well as slightly improved accuracy.

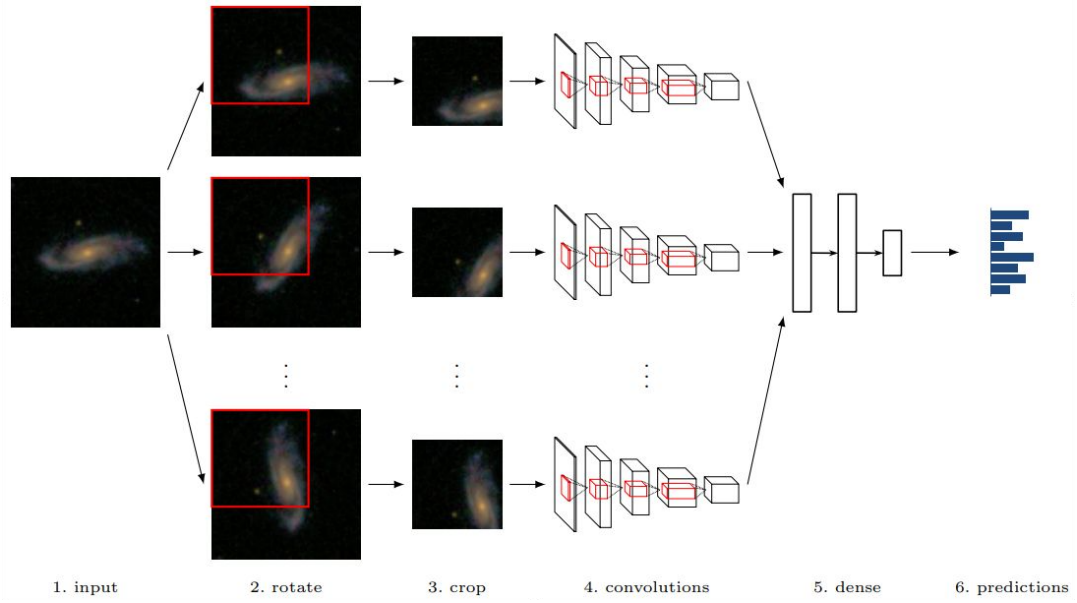

Figure 2.4: Illustration of the CNN architecture designed to exploit transitional and rotational symmetry (Image source [5]).

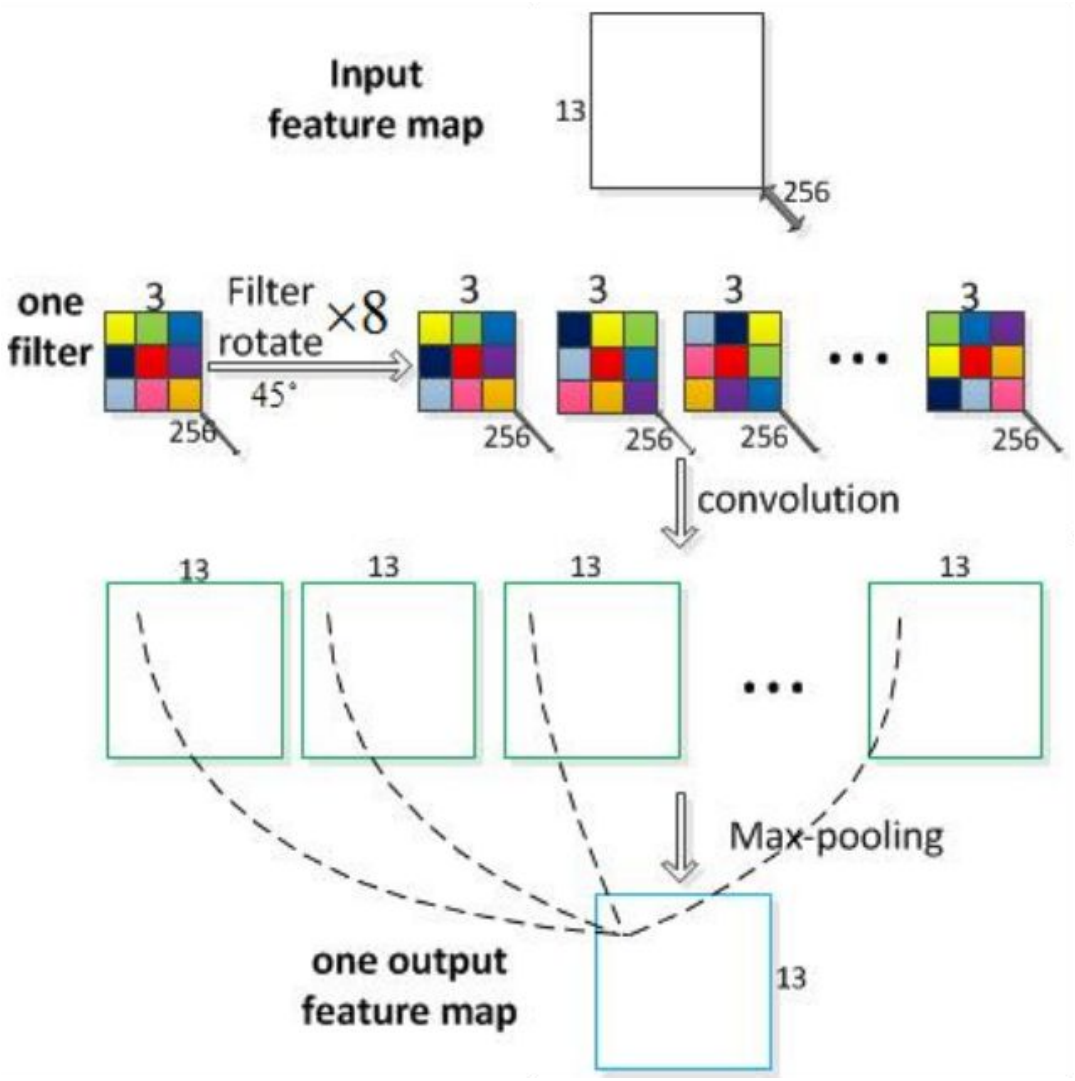

Figure 2.5: Illustration of the rotate-pooling convolution layer, this layer utilizes weight sharing to produce multiple feature maps which are then pooled, resulting in a single output feature map per kernel (Image source [30]).

CHAPTER 3

OUR METHOD

In this chapter, we present our novel CNN architecture as well as a suitable training algorithm for it. We then analyze theoretically our model considering a simplified framework.

3.1 TransNet

3.1.1 Notations and definitions

Let $\mathbb{X} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ denote the training data, where $\boldsymbol{x}_i \in \mathbb{R}^d$ denotes the i-th data point and $y_i \in [K]$ its corresponding label. Let $\mathscr{D}$ denote the data distribution from which the samples are drawn. Let $\mathscr{H}$ denote the set of hypotheses, where $h_{\boldsymbol{\theta}} \in \mathscr{H}$ is defined by its parameters $\boldsymbol{\theta}$ (often we use $h = h_{\boldsymbol{\theta}}$ to simplify notations). Let $\ell(h, \boldsymbol{x}, y)$ denote the loss of hypothesis h when given sample $(\boldsymbol{x}, y)$. The overall loss is:

$$\mathscr{L}(h, \mathbb{X}) = \mathbb{E}_{(\boldsymbol{x},y)\sim\mathscr{D}}[\ell(h, \boldsymbol{x}, y)] \tag{3.1}$$

Our objective is to find the optimal hypothesis:

$$h^* := \arg\min_{h\in\mathscr{H}} \mathscr{L}(h, \mathbb{X}) \tag{3.2}$$

For simplicity, whenever the underlying distribution of a random variable isn't explicitly defined we use the uniform distribution, *e.g.* $\mathbb{E}_{a\in\mathbb{A}}[a] = 1/|\mathbb{A}| \sum_{i=1}^{|\mathbb{A}|} a$.

3.1.2 Model architecture

The *TransNet* architecture is defined by a set of input transformations $\mathbb{T} = \{t_j\}_{j=1}^{m}$, where each transformation $t \in \mathbb{T}$ operates on the inputs ($t : \mathbb{R}^d \to \mathbb{R}^d$) and is associated with a corresponding model's head. Thus each transformation operates on datapoint $\boldsymbol{x}$ as $t(\boldsymbol{x})$, and the transformed data-set in defined as:

$$t(\mathbb{X}) := \{(t(\boldsymbol{x}_i), y_i)\}_{i=1}^{n} \tag{3.3}$$

Given an existing NN model h, henceforth called the *base model*, we can split it to two components: all the layers except for the last one denoted f, and the last layer g assumed to be a fully-connected layer. Thus $h = g \circ f$. Next, we enhance model h by replacing g with $|\mathbb{T}| = m$ heads, where each head is an independent fully connected layer g_t associated with a specific transformation $t \in \mathbb{T}$. Formally, each head is defined by $h_t = g_t \circ f$, and it operates on the corresponding transformed input as $h_t(t(\boldsymbol{x}))$.

The full model, with its m heads, is denoted by $h_{\mathbb{T}} := \{h_t\}_{t \in \mathbb{T}}$, and operates on the input as follows:

$$h_{\mathbb{T}}(\boldsymbol{x}) := \mathbb{E}_{t \in \mathbb{T}}[h_t(t(\boldsymbol{x}))]$$

The corresponding loss of the full model is defined as:

$$L_{\mathbb{T}}(h_{\mathbb{T}}, \mathbb{X}) := \mathbb{E}_{t \in \mathbb{T}}[\mathscr{L}(h_t, t(\mathbb{X}))] \tag{3.4}$$

Note that the resulting model (see Fig. 1.1) essentially represents m models, which share via f all the weights up to the last fully-connected layer. Each of these models can be used separately, as we do later on.

3.1.3 Training algorithm

Our method uses SGD with a few modifications to minimize the transformation loss (3.4), as detailed in Alg. 1. Relying on the fact that each batch is sampled i.i.d. from $\mathscr{D}$, we can prove (see Lemma 1) the desirable property that the sampled loss $L_{\mathbb{B}}$ is an unbiased estimator for the transformation loss $L_{\mathbb{T}}(h_{\mathbb{T}}, \mathbb{X})$. This justifies the use of Alg. 1

to optimize the transformation loss.

Algorithm 1: Training the *TransNet* model

input : *TransNet* model $h_{\mathbb{T}}$, batch size b, maximum iterations num MAX_ITER

output : trained *TransNet* model

for $i = 1 \dots MAX\_ITER$ **do**
  sample a batch $\mathbb{B} = \{(\boldsymbol{x}_k, y_k)\}_{k=1}^{b} \overset{iid}{\sim} \mathscr{D}^b$
  forward:
  for $t \in \mathbb{T}$ **do**
    $\mathscr{L}(h_t, \mathbb{B}) = \frac{1}{b} \sum_{k=1}^{b} \ell(h_t, t(\boldsymbol{x}_k), y_k)$
  end
  $L_{\mathbb{T}}(h_{\mathbb{T}}, \mathbb{B}) = \frac{1}{m} \sum_{t \in \mathbb{T}} \mathscr{L}(h_t, \mathbb{B})$
  backward (SGD):
  update the model's weights by differentiating the sampled loss $L_{\mathbb{T}}(h_{\mathbb{T}}, \mathbb{B})$
end

Lemma 1. *Given batch $\mathbb{B}$, the sampled transformation loss $L_{\mathbb{T}}(h_{\mathbb{T}}, \mathbb{B})$ is an unbiased estimator for the transformation loss $L_{\mathbb{T}}(h_{\mathbb{T}}, \mathbb{X})$.*

Proof.

$$
\begin{aligned}
&\mathbb{E}_{\mathbb{B} \sim \mathscr{D}^b}[L_{\mathbb{T}}(h_{\mathbb{T}}, \mathbb{B})] \\
&= \mathbb{E}_{\mathbb{B} \sim \mathscr{D}^b}[\mathbb{E}_{t \in \mathbb{T}}[\mathscr{L}(h_t, t(\mathbb{B}))]] \\
&= \mathbb{E}_{t \in \mathbb{T}}[\mathbb{E}_{\mathbb{B} \sim \mathscr{D}^b}[\mathscr{L}(h_t, t(\mathbb{B}))]] \quad (\mathbb{B} \overset{iid}{\sim} \mathscr{D}^b) \\
&= \mathbb{E}_{t \in \mathbb{T}}[\mathscr{L}(h_t, t(\mathbb{X}))] \\
&= L_{\mathbb{T}}(h_{\mathbb{T}}, \mathbb{X})
\end{aligned} \tag{3.5}
$$

■

3.1.4 Transformations

Which transformations should we use? Given a specific data-set, we distinguish between transformations that occur naturally in the data-set versus such transformations that do not. For example, horizontal flip can naturally occur in the CIFAR-10 data-set, but not in the MNIST data-set. *TransNet* can only benefit from transformations that do not occur naturally in the target data-set, in order for each head to learn a well defined

and non-overlapping classification task. Transformations that occur naturally in the data-set are often used for data augmentation, as by definition they do not change the data domain.

Dihedral group D_4. As mentioned earlier, the *TransNet* model is defined by a set of input transformations $\mathbb{T}$. We constrain $\mathbb{T}$ to be a subset of the dihedral group D_4, which includes reflections and rotations by multiplications of 90°. We denote a horizontal reflection by m and a counter-clockwise 90° rotation by r. Using these two elements we can express all the D_4 group elements as $\{r^i, m \circ r^i \mid i \in 0,1,2,3\}$. These transformations were chosen because, as mentioned in [10], their application is relatively efficient and does not leave artifacts in the image (unlike scaling or change of aspect ratio).

Note that these transformations can be applied to any 3D tensor while operating on the height and width dimensions, including an input image as well as the model's kernels. When applying a transformation t to the model's weights $\boldsymbol{\theta}$, denoted $t(\boldsymbol{\theta})$, the notation implies that t operates on the model's kernels separately, not affecting other layers such as the fully-connected ones (see Fig. 3.1).

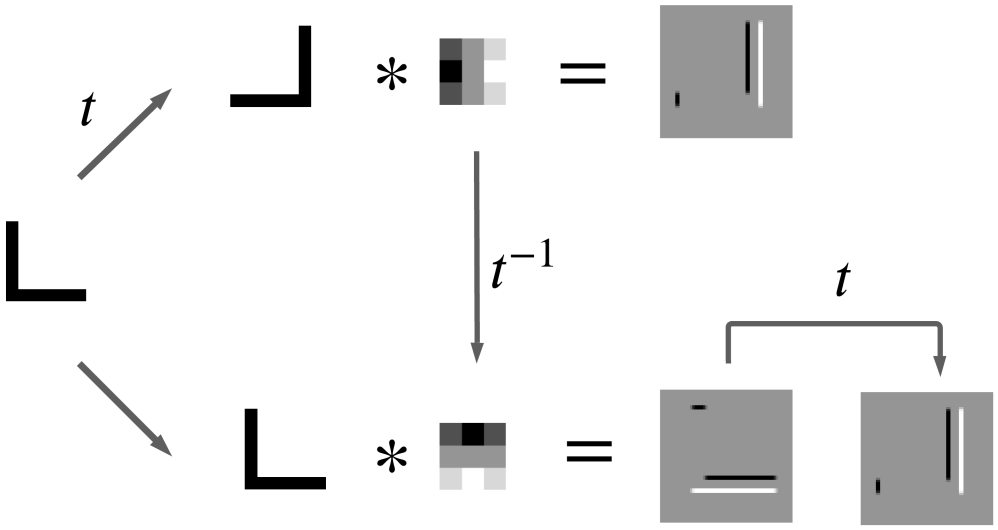

Figure 3.1: The transformed input convolved with a kernel (upper path) equals to the transformation applied on the output of the input convolved with the inversely transformed kernel (lower path).

3.1.5 Model variations

Once trained, the full *TransNet* model can be viewed as an ensemble of m shared classifiers. Its time complexity is linear with the number of heads, almost equivalent to

an ensemble of the base CNN model, since the time needed to apply each one of the D_4 transformations to the input is negligible as compared to the time needed for the model to process the input. Differently, the space complexity is almost equivalent to the space complexity of only one base CNN model[1].

We note that one can prune each one of the model's heads, thus leaving a smaller ensemble of up to m classifiers. A useful reduction prunes all the model's heads except one, typically the one corresponding to the identity transformation, which yields a regular CNN that is equivalent in terms of time and space complexity to the base architecture used to build the *TransNet* model. Having done so, we can evaluate the effect of the *TransNet* architecture's and its training algorithm's inductive bias solely on the training procedure, by comparing the pruned *TransNet* to the base CNN model (see Section 4.1).

3.2 Theoretical Analysis

In this section we analyze theoretically the *TransNet* model. We consider the following basic CNN architecture:

$$h_{\boldsymbol{\theta}} = g \circ l_{inv} \circ \prod_{i=1}^{k} c_i \tag{3.6}$$

where g denotes a fully-connected layer, l_{inv} denotes an invariant layer under the D_4 transformations group (*e.g.* a global average pooling layer - GAP), and $\{c_i\}_{i\in[k]}$ denote convolutional layers[2]. The *TransNet* model extends the basic model by appending additional heads:

$$h_{\mathbb{T},\boldsymbol{\theta}} = \{g_t \circ l_{inv} \circ \prod_{i=1}^{k} c_i\}_{t\in\mathbb{T}} \tag{3.7}$$

We denote the parameters of a fully-connected or a convolutional layer by subscripts of w (weight) and b (bias), *e.g.* $g(\boldsymbol{x}) = g_w \cdot \boldsymbol{x} + g_b$.

3.2.1 Transformation compilation

Transformations in the dihedral D_4 group satisfy another important property, expressed by the following proposition:

[1]Each additional head (another fully-connected layer) adds 102K (~0.45%) and 513K (~0.90%) extra parameters to the basic ResNet18 model when training CIFAR-100 and ImageNet-200 respectively.

[2]While each convolutional layer may be followed by ReLU and Batch Normalization [14] layers, this doesn't change the analysis so we obviate the extra notation.

Proposition 1. *Let $h_{\boldsymbol{\theta}}$ denote a CNN model where the last convolutional layer is followed by an invariant layer under the D_4 group. Then any transformation $t \in D_4$ applied to the input image can be compiled into the model's weights $\boldsymbol{\theta}$ as follows:*

$$\forall t \in D_4 \quad \forall \boldsymbol{x} \in \mathbb{X}: \quad h_{\boldsymbol{\theta}}(t(\boldsymbol{x})) = h_{t^{-1}(\boldsymbol{\theta})}(\boldsymbol{x}) \tag{3.8}$$

Proof. By induction on k we can show that:

$$\prod_{i=1}^{k} c_i \circ t(\boldsymbol{x}) = t \circ \prod_{i=1}^{k} t^{-1}(c_i)(\boldsymbol{x}) \tag{3.9}$$

(see Fig. 3.1). Plugging (3.9) into (3.6), we get:

$$\begin{aligned} h_{\boldsymbol{\theta}}(t(\boldsymbol{x})) &= g \circ l_{inv} \circ \prod_{i=1}^{k} c_i \circ t(\boldsymbol{x}) \\ &= g \circ l_{inv} \circ t \circ \prod_{i=1}^{k} t^{-1}(c_i)(\boldsymbol{x}) \\ &= g \circ l_{inv} \circ \prod_{i=1}^{k} t^{-1}(c_i)(\boldsymbol{x}) \quad (l_{inv} \circ t = l_{inv}) \\ &= h_{t^{-1}(\boldsymbol{\theta})}(\boldsymbol{x}) \end{aligned}$$

■

Implication. The ResNet model [11] used in our experiments satisfies the pre-condition in the proposition stated above, since it contains a GAP layer [19] after the last convolutional layer, and GAP is invariant under D_4.

3.2.2 Single vs. multiple headed model

In order to acquire intuition regarding the inductive bias implied by training algorithm Alg. 1, we consider two cases, a single and a double headed model, trained with the same training algorithm. A single headed model is a special case of the full multi-headed model, where all the heads share weights $h_t(t(\boldsymbol{x})) = h(t(\boldsymbol{x}))\ \forall t$, and the loss in line 5 of Alg. 1 becomes $\mathscr{L}(h, \mathbb{B}) = \frac{1}{b}\sum_{k=1}^{b} \ell(h, t(\boldsymbol{x}_k), y_k)$.

As it's hard to analyze non-convex deep neural networks, we focus on a simplified framework and consider a convex optimization problem where the loss function is convex *w.r.t.* the model's parameters $\boldsymbol{\theta}$. We also assume that the model's transformations in $\mathbb{T}$ form a group[3].

[3] $\mathbb{T}$ being a group is a technical constraint needed for the analysis, not required by the algorithm.

Single Headed model Analysis. In this simplified case, we can prove the following strict proposition:

Proposition 2. *Let $h_{\boldsymbol{\theta}}$ denote a CNN model satisfying the pre-condition of Prop. 1, and $\mathbb{T} \subset D_4$ a transformations group. Then the optimal transformation loss $L_{\mathbb{T}}$ (see Eq. 3.4) is obtained by invariant model's weights under the transformations $\mathbb{T}$. Formally:*

$$\exists \boldsymbol{\theta}_0 : (\forall t \in \mathbb{T} : \boldsymbol{\theta}_0 = t(\boldsymbol{\theta}_0)) \wedge (\boldsymbol{\theta}_0 \in \arg\min_{\boldsymbol{\theta}} L_{\mathbb{T}}(\boldsymbol{\theta}, \mathbb{X}))$$

Proof. To simplify the notations, henceforth we let $\boldsymbol{\theta}$ denote the model $h_{\boldsymbol{\theta}}$.

$$\begin{aligned}
&L_{\mathbb{T}}(\boldsymbol{\theta}, \mathbb{X}) \\
&\quad = \mathbb{E}_{t\in\mathbb{T}}[\mathscr{L}(\boldsymbol{\theta}, t(\mathbb{X}))] \\
&\quad = \mathbb{E}_{t\in\mathbb{T}}[\mathbb{E}_{(\boldsymbol{x},y)\sim\mathscr{D}}[\ell(\boldsymbol{\theta}, t(\boldsymbol{x}), y)]] \\
&\quad = \mathbb{E}_{t\in\mathbb{T}}[\mathbb{E}_{(\boldsymbol{x},y)\sim\mathscr{D}}[\ell(t^{-1}(\boldsymbol{\theta}), \boldsymbol{x}, y)]] \quad \text{(by Prop. 1)} \\
&\quad = \mathbb{E}_{(\boldsymbol{x},y)\sim\mathscr{D}}[\mathbb{E}_{t\in\mathbb{T}}[\ell(t^{-1}(\boldsymbol{\theta}), \boldsymbol{x}, y)]] \\
&\quad \geq \mathbb{E}_{(\boldsymbol{x},y)\sim\mathscr{D}}[\ell(\mathbb{E}_{t\in\mathbb{T}}[t^{-1}(\boldsymbol{\theta})], \boldsymbol{x}, y)] \quad \text{(Jensen's inequality)} \\
&\quad = \mathbb{E}_{(\boldsymbol{x},y)\sim\mathscr{D}}[\ell(\bar{\boldsymbol{\theta}}, \boldsymbol{x}, y)] \quad (\bar{\boldsymbol{\theta}} := \mathbb{E}_{t\in\mathbb{T}}[t(\boldsymbol{\theta})], \quad \mathbb{T} = \mathbb{T}^{-1}) \\
&\quad = \mathscr{L}(\bar{\boldsymbol{\theta}}, \mathbb{X}) \\
&\quad = \mathbb{E}_{t\in\mathbb{T}}[\mathscr{L}(t^{-1}(\bar{\boldsymbol{\theta}}), \mathbb{X})] \quad (\bar{\boldsymbol{\theta}} \text{ is invariant under } \mathbb{T}) \\
&\quad = \mathbb{E}_{t\in\mathbb{T}}[\mathscr{L}(\bar{\boldsymbol{\theta}}, t(\mathbb{X}))] \quad \text{(by Prop. 1)} \\
&\quad = L_{\mathbb{T}}(\bar{\boldsymbol{\theta}}, \mathbb{X})
\end{aligned}$$

Above we use the fact that $\bar{\boldsymbol{\theta}}$ is invariant under $\mathbb{T}$ since $\mathbb{T}$ is a group and thus $t_0\mathbb{T} = \mathbb{T}$, hence:

$$t_0(\bar{\boldsymbol{\theta}}) = t_0(\mathbb{E}_{t\in\mathbb{T}}[t(\boldsymbol{\theta})]) = \mathbb{E}_{t\in\mathbb{T}}[t_0 \circ t(\boldsymbol{\theta})] = \mathbb{E}_{t\in\mathbb{T}}[t(\boldsymbol{\theta})] = \bar{\boldsymbol{\theta}}$$

■

Double headed model. In light of Prop. 2 we now present a counter example, which shows that Prop. 2 isn't true for the general *TransNet* model.

Example 3. *Let $\mathbb{T} = \{t_1 = r^0, t_2 = m \circ r^2\} \subset D_4$ denote the transformations group consisting of the identity and the vertical reflection transformations. Let $h_{\mathbb{T},\boldsymbol{\theta}} = \{h_i = g_i \circ GAP \circ c\}_{i=1}^2$ denote a double headed* TransNet *model, which comprises a single convolutional layer (1 channel in and 2 channels out), followed by a GAP layer and then 2 fully-connected*

layers $\{g_i\}_{i=1}^2$*, one for each head. Each* g_i *outputs a vector of size 2. The data-set* $\mathbb{X} = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2)\}$ *consists of 2 examples:*

$$\boldsymbol{x}_1 = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, y_1 = 1, \quad \boldsymbol{x}_2 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}, y_2 = 2$$

Note that $\boldsymbol{x}_2 = t_2(\boldsymbol{x}_1)$ [4].

Now, assume the model's convolutional layer c *is composed of 2 invariant kernels under* $\mathbb{T}$*, and denote it by* c_{inv}*. Let* $i \in 1,2$*, then:*

$$\begin{aligned} h_i(\boldsymbol{x}_2) = h_i(t_2(\boldsymbol{x}_1)) &= g_i \circ GAP \circ c_{inv} \circ t_2(\boldsymbol{x}_1) \\ &= g_i \circ GAP \circ c_{inv}(\boldsymbol{x}_1) = h_i(\boldsymbol{x}_1) \end{aligned} \tag{3.10}$$

In this case both heads predict the same output for both inputs with different labels, thus:

$$\mathscr{L}(h_i, t_i(\mathbb{X})) > 0 \implies L_{\mathbb{T}}(h_{\mathbb{T},\boldsymbol{\theta}}, \mathbb{X}) > 0.$$

In contrast, by setting $c_w = (\boldsymbol{x}_1, \boldsymbol{x}_2), c_b = (0,0)$*, which isn't invariant under* $\mathbb{T}$*, as well as:*

$$g_{1,w} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, g_{1,b} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad g_{2,w} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, g_{2,b} = \begin{bmatrix} 0 \\ 0 \end{bmatrix},$$

we obtain:

$$\mathscr{L}(h_i, t_i(\mathbb{X})) = 0 \implies L_{\mathbb{T}}(h_{\mathbb{T},\boldsymbol{\theta}}, \mathbb{X}) = 0.$$

We may conclude that the optimal model's kernels aren't invariant under $\mathbb{T}$*, as opposed to the claim of Prop. 2.*

Discussion. The intuition we derive from the analysis above is that the training algorithm (Alg. 1) implies an invariant inductive bias on the model's kernels as proved in the single headed model, while not strictly enforcing invariance as shown by the counter example of the double headed model.

[4] This example may seem rather artificial, but in fact this isn't such a rare case. *E.g.* , the airplane and the ship classes, both found in the CIFAR-10 data-set, that share similar vertically flipped blue background.

CHAPTER 4

EXPERIMENTAL EVALUATION

In this chapter, we empirically evaluate our *TransNet* model performance as well as its generalization capacity and its invariance level.

4.1 Experimental Results

Data-Sets. For evaluation we used the 5 image classification data-sets detailed in Table 4.1. These diverse data-sets allow us to evaluate our method across different image resolutions and number of predicted classes.

Name	**Classes**	**Train/Test Samples**	**dim**
CIFAR-10 [15]	10	50K/10K	32
CIFAR-100 [15]	100	50K/10K	32
ImageNette [13]	10	10K/4K	224
ImageWoof [13]	10	10K/4K	224
ImageNet-200	200	260K/10K	224

Table 4.1: The data-sets used in our experiments. The dimension of each example, a color image, is **dim**×**dim**×3 pixels. ImageNette represents 10 easy to classify classes from ImageNet [4], while ImageWoof represents 10 hard to classify classes of dog breeds from ImageNet. ImageNet-200 represents 200 classes from ImageNet (same classes as in [17]) of full size images.

Implementation Details. We employed the ResNet [11] architecture, specifically the ResNet18 architecture for all the data-sets except for the ImageNet-200 which was

MODEL	CIFAR-10	CIFAR-100	ImageNette	ImageWoof	ImageNet-200
base-CNN	95.57 ± 0.08	76.56 ± 0.16	92.97 ± 0.16	87.27 ± 0.15	84.39 ± 0.07
PT2-CNN	**95.99** ± 0.07	**79.33** ± 0.15	93.84 ± 0.14	**88.09** ± 0.30	**85.17** ± 0.10
PT3-CNN	95.87 ± 0.04	79.08 ± 0.06	**94.15** ± 0.16	87.79 ± 0.11	84.97 ± 0.95
PT4-CNN	95.73 ± 0.05	77.98 ± 0.17	93.94 ± 0.06	85.81 ± 0.79	84.02 ± 0.71

Table 4.2: Accuracy of models with the same space and time complexity, comparing the Base CNN with pruned *TransNet* models "PTm-CNN", where $m = 2, 3, 4$ denotes the number of heads in training. Mean and standard error for 3 repetitions are shown.

MODEL	CIFAR-10	CIFAR-100	ImageNette	ImageWoof	ImageNet-200
base-CNN	95.57 ± 0.08	76.56 ± 0.16	92.97 ± 0.16	87.27 ± 0.15	84.39 ± 0.07
T2-CNN	96.22 ± 0.10	80.35 ± 0.06	94.02 ± 0.13	88.36 ± 0.33	85.47 ± 0.14
T3-CNN	**96.33** ± 0.06	**80.92** ± 0.08	94.39 ± 0.07	**88.79** ± 0.25	**85.68** ± 0.20
T4-CNN	96.17 ± 0.01	79.94 ± 0.16	**94.67** ± 0.06	87.05 ± 0.75	85.54 ± 0.11

Table 4.3: Accuracy of models with similar space complexity and different time complexity, comparing the Base CNN with full *TransNet* models. With m denoting the number of heads, chosen to be 2,3 or 4, the prediction time complexity of the respective *TransNet* model "Tm-CNN" is m times larger than the base CNN. Mean and standard error for 3 repetitions are shown.

evaluated using the ResNet50 architecture. It's important to notice that we haven't changed the hyper-parameters used by the regular CNN architecture which *TransNet* is based on. This may strengthen the results as one may fine tune these hyper-parameters to suit best the *TransNet* model.

We used a weight decay of 0.0001 and momentum of 0.9. The model was trained with a batch size of 64 for all the data-sets except for ImageNet-200 where we increased the batch size to 128. We trained the model for 300 epochs, starting with a learning rate of 0.1, divided by 10 at the 150 and 225 epochs, except for the ImageNet-200 model which was trained for 120 epochs, starting with a learning rate of 0.1, divided by 10 at the 40 and 80 epochs. We normalized the images as usual by subtracting the image's mean and dividing by the image's standard deviation (color-wise).

We employed a mild data augmentation scheme - horizontal flip with probability of 0.5. For the CIFAR data-sets we padded each dimension by 4 pixels and cropped randomly (uniform) a 32×32 patch from the enlarged image [18] while for the ImageNet family data-sets we cropped randomly (uniform) a 224×224 patch from the original image.

In test time, we took the original image for the CIFAR data-sets and a center crop for the ImageNet family data-sets. The prediction of each model is the mean of the model's output on the original image and a horizontally flipped version of it. Note that

a horizontal flip occurs naturally in every data-set we use for evaluation and therefore isn't associated with any of the *TransNet* model's heads that we evaluate.

Notations.

- "base CNN" - a regular convolutional neural network, identical to the *TransNet* model with only the head corresponding to the identity transformation.
- "PTm-CNN" - a pruned *TransNet* model trained with m heads, where a single head is left and used for prediction[1]. It has the same space and time complexity as the base CNN.
- "Tm-CNN" - a full *TransNet* model trained with m heads, where all are used for prediction. It has roughly the same space complexity[1] and m times the time complexity as compared to the base CNN.

4.1.1 Models accuracy, comparative results

We now compare the accuracy of the "base-CNN", "PTm-CNN" and "Tm-CNN" models, where $m = 2, 3, 4$ denotes the number of heads of the *TransNet* model, and their ensembles, across all the data-sets listed in Table 4.1.

Models with the same space and time complexity. First, we evaluate the pruned *TransNet* model by comparing the "PTm-CNN" models with the "base-CNN" model, see Table 4.2. Essentially, we evaluate the effect of using the *TransNet* model only for training, as the final "PTm-CNN" models are identical to the "base-CNN" model regardless of m. We can clearly see the inductive bias implied by the training procedure. We also see that *TransNet* training improves the accuracy of the final "base-CNN" classifier across all the evaluated data-sets.

Models with similar space complexity, different time complexity. Next, we evaluate the full *TransNet* model by comparing the "Tm-CNN" models with the "base-CNN" model, see Table 4.3. Despite the fact that the full *TransNet* model processes the (transformed) input m times more as compared to the "base-CNN" model, its architecture is not significantly larger than the base-CNN's. The full *TransNet* adds to the "base-CNN" a negligible number of parameters, in the form of its multiple heads[1]. Clearly the full *TransNet* model improves the accuracy as compared to the "base-CNN" model, and also

[1] In our experiments we chose the head associated with the identity (r^0) transformation when evaluating a pruned *TransNet*. Note, however, that we could have chosen the best head in terms of accuracy, as it follows from Prop. 1 that its transformation can be compiled into the model's weights.

as compared to the pruned *TransNet* model. Thus, if the additional runtime complexity during test is not an issue, it is beneficial to employ the full *TransNet* model during test time. In fact, one can process the input image once, and then choose whether to continue processing it with the other heads to improve the prediction, all this while keeping roughly the same space complexity.

Ensembles: models with similar time complexity, different space complexity. Here we evaluate ensembles of pruned *TransNet* models, and compare them to a single full *TransNet* model that can be seen as a space-efficient ensemble: full *TransNet* generates m predictions with only $1/m$ parameters, where m is the number of *TransNet* heads. Results are shown in Fig. 4.1. Clearly an ensemble of pruned *TransNet* models is superior to an ensemble of base CNN models, suggesting that the accuracy gain achieved by the pruned *TransNet* model doesn't overlap with the accuracy gain achieved by using an ensemble of classifiers. Furthermore, we observe that the full *TransNet* model exhibits competitive accuracy results, with 2 and 3 heads, as compared to an ensemble of 2 or 3 base CNN models respectively. This is achieved while utilizing $1/2$ and $1/3$ as many parameters respectively.

Accuracy vs. generalization. In Fig. 4.1 we can see that 2 heads improve the model's performance across all data-sets, 3 heads improve it on most of the data-sets, and 4 heads actually undermine performance on all the data-sets except ImageNette. We hypothesize that too many heads impose too strict an inductive bias on the model's kernels. Thus, although generalization is improved, test accuracy is reduced due to insufficient variance. Further analysis is presented in the next section.

4.1.2 Generalization

We've seen in Section 4.1.1 that the *TransNet* model, whether full or pruned, achieves better test accuracy as compared to the base CNN model. This occurs despite the fact that the transformation loss $L_{\mathbb{T}}(h_{\mathbb{T}}, \mathbb{X})$ minimized by the *TransNet* model is more demanding than the loss $\mathscr{L}(h, \mathbb{X})$ minimized by the base CNN, and appears harder to optimize. This conjecture is justified by the following Lemma:

Lemma 2. *Let $h_{\mathbb{T}}$ denote a* TransNet *model that obtains transformation loss of $a := L_{\mathbb{T}}(h_{\mathbb{T}}, \mathbb{X})$. Then there exists a reduction from $h_{\mathbb{T}}$ to the base CNN model h that obtains a loss of at most a,* i.e. *$\mathscr{L}(h, \mathbb{X}) \leq a$.*

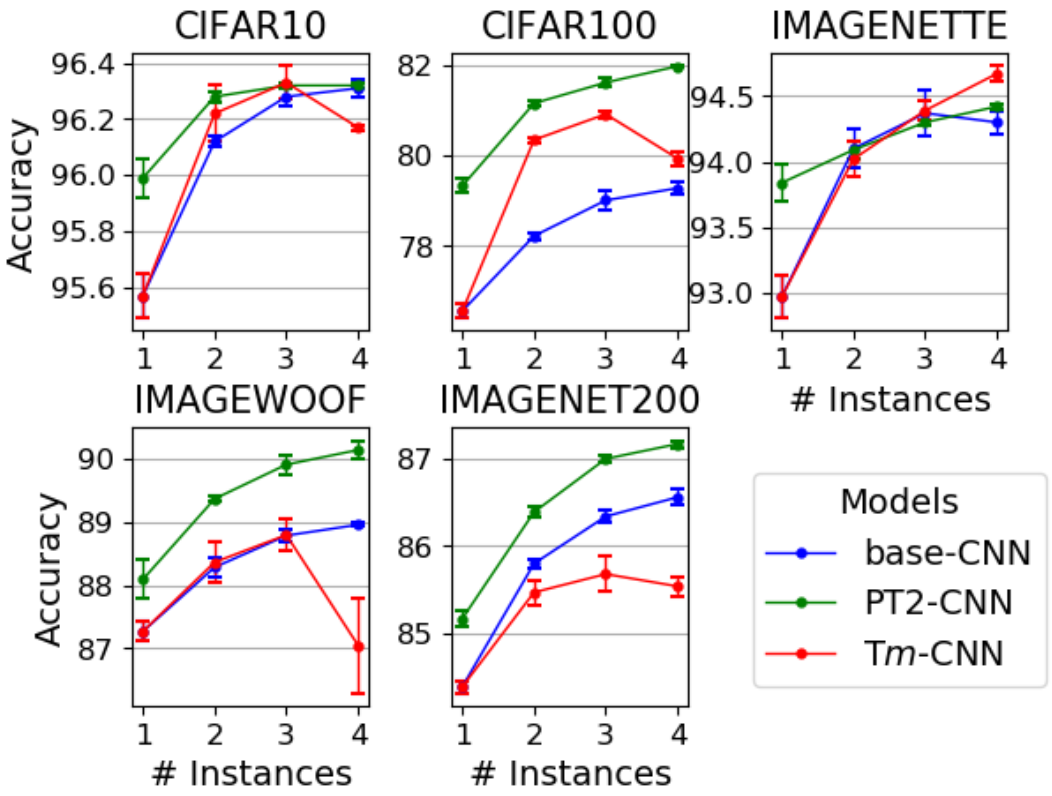

Figure 4.1: Model accuracy as a function of the number of instances (X-axis) processed during prediction. Each instance requires a full run from input to output. An ensemble of m instances includes: m independent base CNN classifiers for "CNN"; m pruned *TransNet* trained with 2 heads for "PT2-CNN"; and a single *TransNet* model with m heads, where m is the ensemble size, for "Tm-CNN".

Proof. $a = L_{\mathbb{T}}(h_{\mathbb{T}}, \mathbb{X}) = \mathbb{E}_{t\in\mathbb{T}}[\mathscr{L}(h_{\boldsymbol{\theta}_t}, t(\mathbb{X}))]$, so there must be a transformation $t \in \mathbb{T}$ s.t. $\mathscr{L}(h_{\boldsymbol{\theta}_t}, t(\mathbb{X})) \leq a$. Now, one can compile the transformation t into $h_{\boldsymbol{\theta}_t}$ (see Prop. 1) and get a base CNN: $\tilde{h} = h_{t^{-1}(\boldsymbol{\theta}_t)}$ which obtains $\mathscr{L}(\tilde{h}, \mathbb{X}) = \mathscr{L}(h_{t^{-1}(\boldsymbol{\theta}_t)}, t(\mathbb{X})) = \mathscr{L}(h_{\boldsymbol{\theta}_t}, t(\mathbb{X})) \leq a$. ■

Why is it, then, that the *TransNet* model achieves overall better accuracy than the base CNN? The answer lies in its ability to achieve a better generalization.

In order to measure the generalization capability of a model *w.r.t.* a data-set, we use the ratio between the test-set and train-set loss, where a lower ratio indicates better generalization. As illustrated in Fig. 4.2, clearly the pruned *TransNet* models exhibit better generalization when compared to the base CNN model. Furthermore, the generalization improvement increases with the number of *TransNet* model heads, which are only used for training and then pruned. The observed narrowing of the generalization gap occurs because, although the *TransNet* model slightly increases the training loss, it

more significantly decreases the test loss as compared to the base CNN.

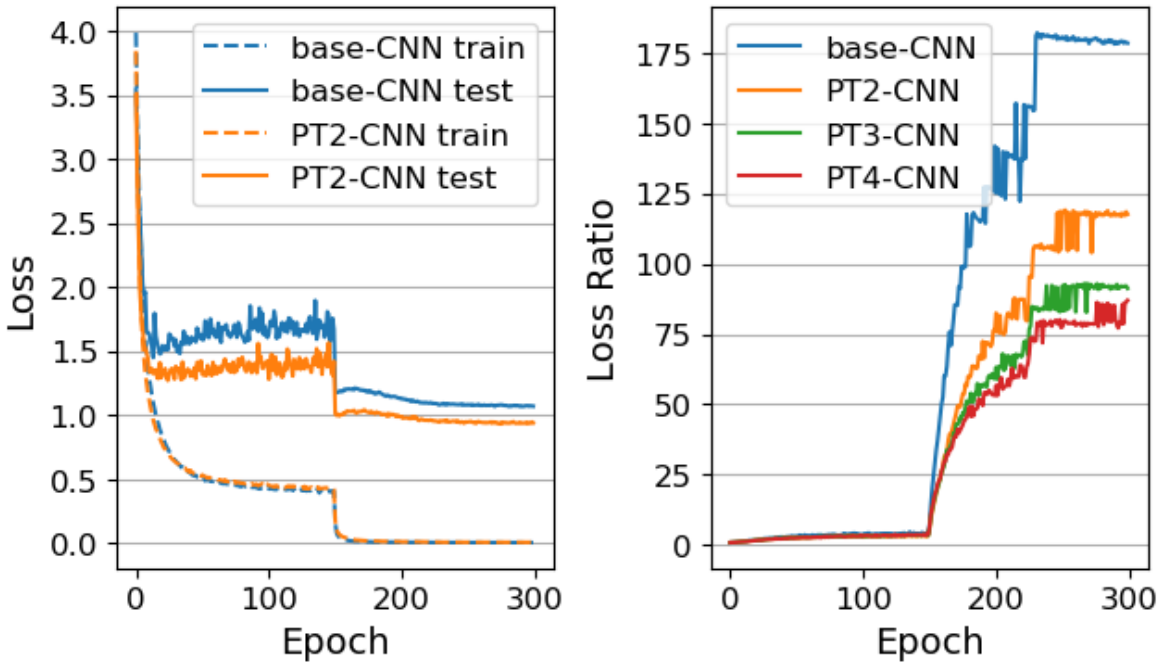

Figure 4.2: CIFAR-100 results. Left panel: learning curve of the Base CNN model ("base-CNN") and a pruned *TransNet* model ("PT2-CNN"). Right panel: generalization score, test-train loss ratio, measured for the base-CNN model and various pruned *TransNet* models with a different number of heads.

We note that better generalization does not necessarily imply a better model. The "PT4-CNN" model generalizes better than any other model (see right panel of Fig. 4.2), but its test accuracy is lower as seen in Table 4.2.

4.1.3 Kernel invariance

What characterizes the beneficial inductive bias implied by the *TransNet* model and its training algorithm Alg. 1?. To answer this question, we investigate the emerging invariance of kernels in the convolutional layers of the learned network, *w.r.t.* the *TransNet* transformations set $\mathbb{T}$.

We start by introducing the "Invariance Score" (IS), which measures how invariant a 3D tensor is *w.r.t.* a transformations group. Specifically, given a convolutional kernel denoted by $\boldsymbol{w}$ (3D tensor) and a set of transformations group $\mathbb{T}$, the IS score is defined as follows:

$$IS(\boldsymbol{w}, \mathbb{T}) := \min_{\boldsymbol{u} \in INV_{\mathbb{T}}} \|\boldsymbol{w} - \boldsymbol{u}\| \tag{4.1}$$

where $INV_{\mathbb{T}}$ is the set of invariant kernels (same shape as $\boldsymbol{w}$) under $\mathbb{T}$, *i.e.* $(\boldsymbol{u} \in INV_{\mathbb{T}}) \iff (\forall t \in \mathbb{T} : \boldsymbol{u} = t(\boldsymbol{u}))$.

Lemma 3. $\arg\min_{\boldsymbol{u} \in INV_{\mathbb{T}}} \|\boldsymbol{w} - \boldsymbol{u}\| = \mathbb{E}_{t \in \mathbb{T}}[t(\boldsymbol{w})]$

Proof. Let $\boldsymbol{u}$ be an invariant tensor under $\mathbb{T}$. Define $f(\boldsymbol{u}) := \|\boldsymbol{w} - \boldsymbol{u}\|^2$. Note that $\arg\min_{u \in INV_{\mathbb{T}}} \|\boldsymbol{w} - \boldsymbol{u}\| = \arg\min_{u \in INV_{\mathbb{T}}} f(u)$.

$$
\begin{aligned}
f(\boldsymbol{u}) &= \|\boldsymbol{w} - \boldsymbol{u}\|^2 \\
&= \mathbb{E}_{t \in \mathbb{T}}[\|\boldsymbol{w} - t(\boldsymbol{u})\|^2] \quad (\boldsymbol{u} \text{ is invariant under } \mathbb{T}) \\
&= \mathbb{E}_{t \in \mathbb{T}}[\left\|t^{-1}(\boldsymbol{w}) - \boldsymbol{u}\right\|^2] \\
&= \mathbb{E}_{t \in \mathbb{T}}[\|t(\boldsymbol{w}) - \boldsymbol{u}\|^2] \quad (\mathbb{T} = \mathbb{T}^{-1}) \\
&= \mathbb{E}_{t \in \mathbb{T}}[\sum_{i=1}^{size(\boldsymbol{w})} (t(\boldsymbol{w})_i - \boldsymbol{u}_i)^2]
\end{aligned}
$$

where index i runs over all the tensors' elements. Finally, we differentiate f to obtain its minimum:

$$
\begin{aligned}
&\frac{\partial f}{\partial \boldsymbol{u}_i} = \mathbb{E}_{t \in \mathbb{T}}[-2(t(\boldsymbol{w})_i - \boldsymbol{u}_i)] = 0 \\
&\implies \boldsymbol{u}_i = \mathbb{E}_{t \in \mathbb{T}}[[t(\boldsymbol{w})_i] \implies \boldsymbol{u} = \mathbb{E}_{t \in \mathbb{T}}[t(\boldsymbol{w})]
\end{aligned}
$$

■

Lemma 3 gives a closed-form expression for the IS gauge:

$$IS(\boldsymbol{w}, \mathbb{T}) = \|\boldsymbol{w} - \mathbb{E}_{t \in \mathbb{T}}[t(\boldsymbol{w})]\| \tag{4.2}$$

Equipped with this gauge, we can inspect the invariance level of the model's kernels *w.r.t.* a transformations group. Note that this measure allows us to compare the full *TransNet* model with the base CNN model, as both share the same convolution layers. Since the transformations of the *TransNet* model don't necessarily form a group, we use the minimal group containing these transformations - the group of all rotations $\{r^i\}_{i=1}^4$.

In Fig. 4.3 we can see that the full *TransNet* model "T2-CNN" and the base CNN model demonstrate similar invariance level in all the convolutional layers but the last one. In Fig. 4.4, where the distribution of the IS score over the last layer of 4 different models is fully shown, we can more clearly see that the last convolutional layer of full *TransNet* models exhibits much higher invariance level as compared to the base CNN. This phenomenon is robust to the metric used in the IS definition: similar results can be

obtained when using other metrics such as "Pearson Correlation" and "Cosine Similarity". The increased invariance in the last convolutional layer is monotonically increasing with the number of heads in the *TransNet* model, which is consistent with the generalization capability of these models (see Fig 4.2).

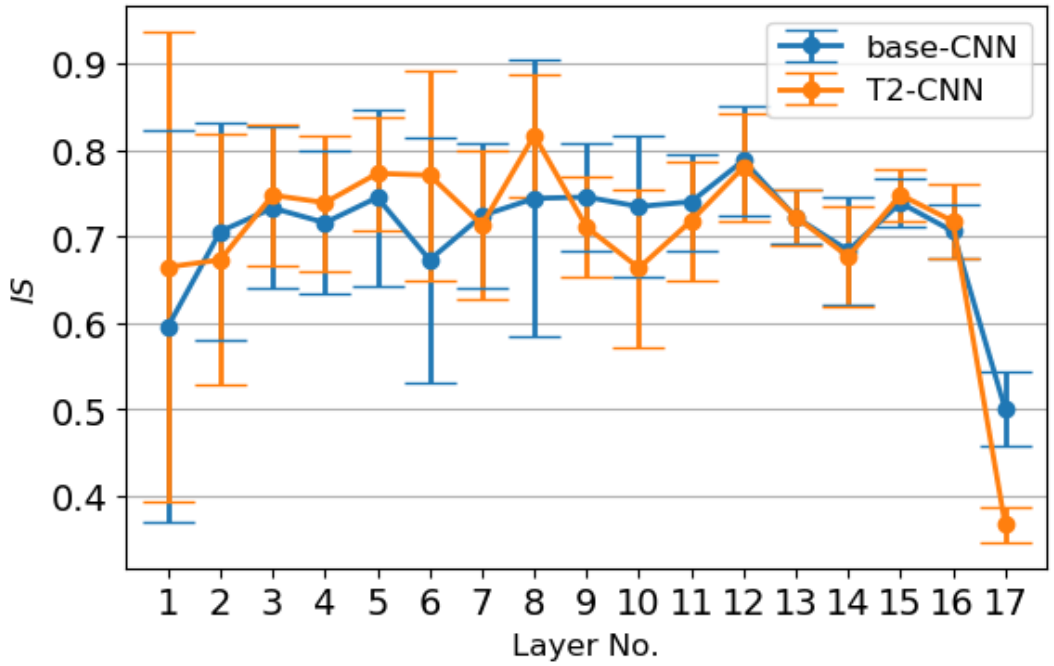

Figure 4.3: CIFAR-100 results, plotting the distribution of the IS scores (mean and std) for the kernels in each layer of the different models. Invariance is measured *w.r.t.* the group of 90° rotations.

The generalization improvement achieved by the *TransNet* model, as reported in Section 4.1.2, may be explained by this increased level of invariance, as highly invariant kernels have fewer degrees of freedom, and should therefore be less prone to overfit.

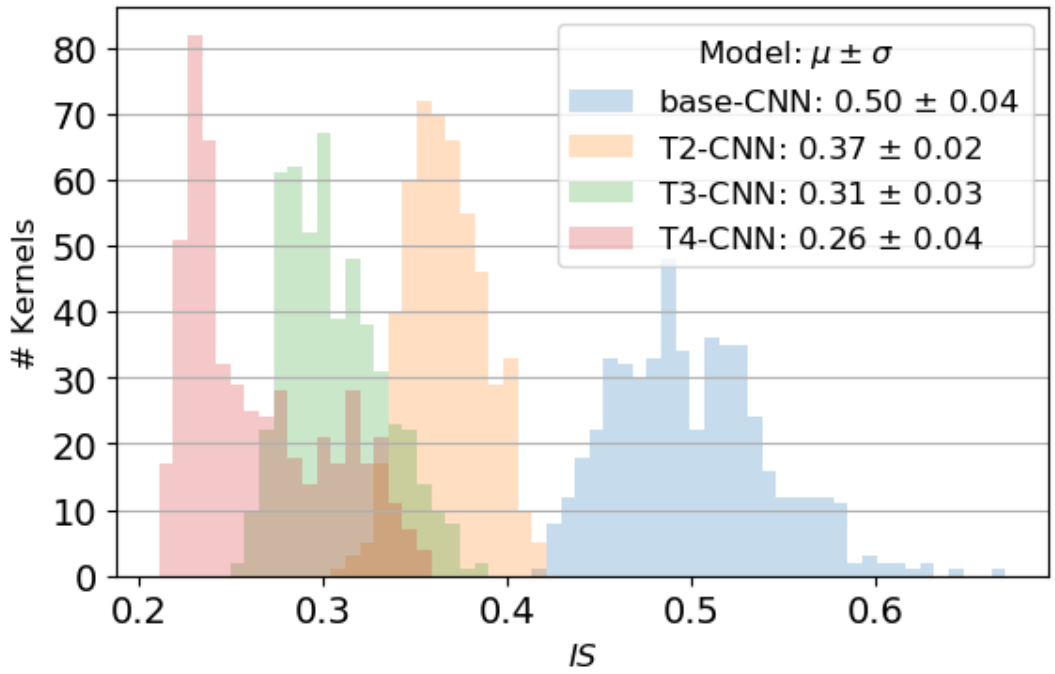

Figure 4.4: CIFAR-100 results, plotting the full distribution of the IS scores for the kernels in the last (17-th) layer of the different models. Invariance is measured *w.r.t.* the group of 90° rotations.

CHAPTER 5

SUMMARY AND DISCUSSION

We introduced a model inspired by self-supervision, which includes a base CNN model attached to multiple heads, each corresponding to a different transformation from a fixed set of transformations. The self-supervised aspect of the model is crucial, as the chosen transformations must not occur naturally in the data. When the model is pruned back to match the base CNN, it achieves better test accuracy and improved generalization, which is attributed to the increased invariance of the model's kernels in the last layer. We observed that excess invariance, while improving generalization, eventually curtails the test accuracy.

We evaluated our model on various image data-sets, observing that each data-set achieves its own optimal kernel's invariance level, *i.e.* there's no optimal number of heads for all data-sets. Finally, we introduced an invariance score gauge (IS), which measures the level of invariance achieved by the model's kernels. IS may be leveraged to determine the optimal invariance level, as well as potentially function as an independent regularization term.

5.1 Future Work

We suggest 2 main research direction following this paper: The first is evaluating and analyzing the *TransNet* model based on different self-supervision transformations rather than the D_4 group. The second is trying to regularize a regular CNN model by adding to it a regularization term in the form of a scale factor multiplied by the IS score of the

model's kernels. This regularization technique would have 2 main considerations: which group of transformations should we choose to define the IS score? and On which kernels should we apply this regularization. An initial estimate to the aforementioned question could be derived by analyzing the IS measures exhibited by the *TransNet* model.

BIBLIOGRAPHY

[1] Christopher Clark and Amos Storkey.
Training deep convolutional neural networks to play go.
In *International conference on machine learning*, pages 1766–1774, 2015.

[2] Taco Cohen and Max Welling.
Group equivariant convolutional networks.
In *International conference on machine learning*, pages 2990–2999, 2016.

[3] Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le.
Autoaugment: Learning augmentation strategies from data.
In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 113–123, 2019.

[4] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei.
Imagenet: A large-scale hierarchical image database.
In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[5] Sander Dieleman, Kyle W Willett, and Joni Dambre.
Rotation-invariant convolutional neural networks for galaxy morphology prediction.
Monthly notices of the royal astronomical society, 450(2):1441–1459, 2015.

[6] Sander Dieleman, Jeffrey De Fauw, and Koray Kavukcuoglu.
Exploiting cyclic symmetry in convolutional neural networks.
arXiv preprint arXiv:1602.02660, 2016.

[7] Carl Doersch, Abhinav Gupta, and Alexei A Efros.
Unsupervised visual representation learning by context prediction.
In *Proceedings of the IEEE international conference on computer vision*, pages 1422–1430, 2015.

[8] Alexey Dosovitskiy, Philipp Fischer, Jost Tobias Springenberg, Martin Riedmiller, and Thomas Brox.
Discriminative unsupervised feature learning with exemplar convolutional neural networks.
IEEE transactions on pattern analysis and machine intelligence, 38(9):1734–1747, 2015.

[9] Robert Gens and Pedro M Domingos.
Deep symmetry networks.
In *Advances in neural information processing systems*, pages 2537–2545, 2014.

[10] Spyros Gidaris, Praveer Singh, and Nikos Komodakis.
Unsupervised representation learning by predicting image rotations.
arXiv preprint arXiv:1803.07728, 2018.

[11] 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.

[12] Dan Hendrycks, Mantas Mazeika, Saurav Kadavath, and Dawn Song.
Using self-supervised learning can improve model robustness and uncertainty.
In *Advances in Neural Information Processing Systems*, pages 15663–15674, 2019.

[13] Jeremy Howard.
Imagewang.
URL `https://github.com/fastai/imagenette/`.

[14] Sergey Ioffe and Christian Szegedy.
Batch normalization: Accelerating deep network training by reducing internal covariate shift.
arXiv preprint arXiv:1502.03167, 2015.

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

[16] Gustav Larsson, Michael Maire, and Gregory Shakhnarovich.
Learning representations for automatic colorization.
In *European conference on computer vision*, pages 577–593. Springer, 2016.

[17] Ya Le and Xuan Yang.
Tiny imagenet visual recognition challenge.
CS 231N, 7, 2015.

[18] Chen-Yu Lee, Saining Xie, Patrick Gallagher, Zhengyou Zhang, and Zhuowen Tu.
Deeply-supervised nets.
In *Artificial intelligence and statistics*, pages 562–570, 2015.

[19] Min Lin, Qiang Chen, and Shuicheng Yan.
Network in network.
arXiv preprint arXiv:1312.4400, 2013.

[20] Joseph L Mundy, Andrew Zisserman, et al.
Geometric invariance in computer vision, volume 92.
MIT press Cambridge, MA, 1992.

[21] Jiquan Ngiam, Zhenghao Chen, Daniel Chia, Pang W Koh, Quoc V Le, and Andrew Y Ng.
Tiled convolutional neural networks.
In *Advances in neural information processing systems*, pages 1279–1287, 2010.

[22] Mehdi Noroozi and Paolo Favaro.
Unsupervised learning of visual representations by solving jigsaw puzzles (2016).
arXiv preprint arXiv:1603.09246.

[23] Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A. Efros.
Context encoders: Feature learning by inpainting, 2016.

[24] Pascal Paysan, Reinhard Knothe, Brian Amberg, Sami Romdhani, and Thomas Vetter.
A 3d face model for pose and illumination invariant face recognition.
In *2009 Sixth IEEE International Conference on Advanced Video and Signal Based Surveillance*, pages 296–301. Ieee, 2009.

[25] Ling Shao, Fan Zhu, and Xuelong Li.
Transfer learning for visual categorization: A survey.
IEEE transactions on neural networks and learning systems, 26(5):1019–1034, 2014.

[26] Laurent Sifre and Stéphane Mallat.
Rotation, scaling and deformation invariant scattering for texture discrimination.
In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1233–1240, 2013.

[27] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov.
Dropout: a simple way to prevent neural networks from overfitting.
The journal of machine learning research, 15(1):1929–1958, 2014.

[28] Pavan Turaga and Rama Chellappa.
Locally time-invariant models of human activities using trajectories on the grassmannian.
In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2435–2441. IEEE, 2009.

[29] Karl Weiss, Taghi M Khoshgoftaar, and DingDing Wang.
A survey of transfer learning.
Journal of Big data, 3(1):9, 2016.

[30] Fa Wu, Peijun Hu, and Dexing Kong.
Flip-rotate-pooling convolution and split dropout on convolution neural networks for image classification.
arXiv preprint arXiv:1507.08754, 2015.

[31] Xiaohua Zhai, Avital Oliver, Alexander Kolesnikov, and Lucas Beyer.
S4l: Self-supervised semi-supervised learning.
In *Proceedings of the IEEE international conference on computer vision*, pages 1476–1485, 2019.

[32] Richard Zhang, Phillip Isola, and Alexei A Efros.
Colorful image colorization.
In *European conference on computer vision*, pages 649–666. Springer, 2016.

[33] Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang.
Random erasing data augmentation.
In *AAAI*, pages 13001–13008, 2020.