ABSTRACT

CHEN, ZEXI. Towards Semi-supervised Video Action Recognition. (Under the direction of Ranga Raju Vatsavai.)

Video action recognition is the task of assigning one of several class labels to a short video clip containing an action. In recent years, deep learning models have been introduced into this recognition task and proven to obtain better recognition accuracy compared to traditional models with hand-crafted features. A primary reason for the success of the deep learning models on video recognition task is the usage of extensive well-annotated videos. However, extracting video clips consisting of sole action precisely from long untrimmed videos and annotating them accurately at a large scale are very expensive and require expert knowledge. Semi-supervised learning (SSL) provides an alternative solution that reduces the heavy burden on collecting labeled data by allowing the model to leverage cheaply available unlabeled data. In this dissertation, we explore supervised and semi-supervised deep learning approaches for image classification and video recognition tasks. This work lays the foundation for future deep semi-supervised video action recognition.

In this dissertation, we made the following novel contributions: (1) developed a new variant of Long Short-Term Memory, namely Relational LSTM model that utilizes a non-local operation in place of the fully connected operation in a vanilla LSTM to address the challenge of relational reasoning across space and time between objects in videos; (2) developed a novel local clustering method to mitigate the effect of confirmation bias issue in consistency-based SSL methods; (3) designed a new composite consistency regularization method to address the prediction inconsistency issue in Generative Adversarial Networks (GANs) based SSL methods.

Towards Semi-supervised Video Action Recognition

by
Zexi Chen

A dissertation submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the Degree of
Doctor of Philosophy

Computer Science

Raleigh, North Carolina

2020

APPROVED BY:

Tianfu Wu	Min Chi
Dennis Bahler	Ranga Raju Vatsavai Chair of Advisory Committee

ACKNOWLEDGEMENTS

First of all, I would like to thank my advisor, Dr. Ranga Raju Vatsavai, for his guidance, inspiration, and support throughout my Ph.D. studies at NC State University. I became very interested in machine learning research after taking Dr. Raju's Automated Learning and Data Analysis (CSC 522) course during my first semester of Ph.D. study, and approached him for further research opportunities after the completion of the course. I felt very grateful that he agreed to work with me immediately in our first meeting and provided endless support during my thesis research. He always understood my struggles and concerns about my research work and helped me overcome various hurdles by sharing his own experiences and always provided invaluable suggestions. Although I have changed my research topics several times, he always supported me firmly and provided helpful advice to get me started on the new topic. I am very grateful for his understanding, compassion, and advice that allowed me to successfully complete my Ph.D. dissertation.

I also would like to thank Dr. Tianfu Wu for his guidance and collaboration on my dissertation research work. His expertise in computer vision research and his suggestions on video action recognition research, helped me refine several aspects of my dissertation. I enjoyed my time discussing research ideas with him and feel amazed by his unique insights and great vision.

I would also like to express my sincere gratitude to my committee members – Dennis Bahler and Min Chi – for serving on my committee and providing constructive suggestions on my dissertation research. Both Dr. Bahler and Dr. Chi's advice and feedback have greatly helped me in refining this thesis.

Special thanks to my close collaborators and co-authors who made this dissertation possible. I would like to sincerely thank Bharathkumar (Tiny) Ramachandra for his tremendous support for both my daily life and research. Whenever I got stuck and felt depressed, he could always save me with his optimism and timely suggestions. I also appreciate Benjamin Dutton for his great support. I really enjoyed our time together at his house with Tiny, when we have many invaluable discussions on our research work. And I would extend my thanks to the rest of our STAC lab members - Krishna Karthik Gadiraju, Seokyong Hong, Qiang Zhang, John Jernigan, Ashwin Shashidharan and Umesh Gupta. Additionally, I would like to thank my friends including Ziwei Wu, Shuai Yang, Yuepeng Qi, Zhewei Hu, Liang Dong, Lingnan Gao for their encouragement and support during my Ph.D.

Finally, I would like to thank my parents for their love, encouragement, and support throughout my academic life, which culminated in the successful completion of my Ph.D. They always understood my needs and provided unconditional support that kept me going. I treat them as my best audience and supporters. My success in this Ph.D. study would not have been possible without them.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER

1

RESEARCH OUTLINE

Video action recognition (also known as video action classification) is a very popular topic in the computer vision field, partly because of the key role it plays in many video analysis tasks such as video anomaly detection [Ram19], human-machine interaction, video retrieval [Rat13], video search and video captioning [Yu18]. An action is a sequence of object movements, and may involve the movements of more than one object concurrently. For instance, the "Jump Rope" action consists of a sequence of human and rope movements at the same time. Although video action recognition, at first sight, might seem to be a natural extension to image classification where a temporal dimension has been added, its inherent complexity and higher dimension make the recognition task much more complicated.

In recent years, we have witnessed a series of remarkable success in fully-supervised video action recognition task using deep learning models [Sim14a; Car17; Wu18]. For deep learning models to work well for video action recognition task, one of the biggest challenges is to obtain large-scale well-annotated action datasets. However, action annotation is very expensive and requires expert knowledge due to the large variations in viewpoint, scale, object motion speed and appearance within a video clip [Bar12]. This becomes even more problematic when one considers that the video clip containing an action is usually part of an untrimmed long video and first needs to be extracted before annotation. Insufficient labeled data problem in supervised classification is addressed in the literature by combining limited labeled samples with a large number of freely available unlabeled samples, giving raise to a variety of semi-supervised learning (SSL) algorithms [Ras15; Lai16; Tar17; Miy18; Sal16; Ber19a]. These advances in SSL and deep learning laid a foundation for developing new techniques in supervised and semi-supervised image classification and video action recognition tasks in this dissertation.

1.1 Thesis Statement

"Quality of semi-supervised video action recognition can be significantly improved by jointly modeling long-term trajectory features and object interaction features in videos, and by giving careful consideration to the confirmation bias and consistency issues that arise in various deep semi-supervised learning frameworks."

1.2 Thesis Contributions

In summary, the key contributions of this research are organized as following:

Chapter 2- Relational Long Short-Term Memory for Video Action Recognition: As a first step, we work on video action recognition in a fully-supervised setting. We note that there are four key features crucial for recognizing actions in videos: appearance features, short-term motion features, long-term trajectory features, and object interaction features. Existing video action recognition methods have not modeled them jointly. Hence, it motivated us to develop a two-branch neural architecture to model these features in a unified framework. The two-branch neural architecture has a spatial-temporal pooling based local branch for capturing local spatial appearance and short-term features, and a Relational LSTM based non-local branch for capturing long-term trajectory features and object interaction features. The proposed Relational LSTM module in non-local branch is a new variant of Long Short Term Memory where we utilize a non-local operation [Wan18] to substitute the fully connected operation in the vanilla LSTM. Experimental results on two benchmark action recognition datasets show that our model achieves new state-of-the-art performance among LSTM-based methods. Moreover, our method achieved a minimum gain of 3.2% over state-of-the-art methods on more complex large-scale Charades dataset, showing the effectiveness in complex video understanding.

Chapter 3- Local Clustering with Mean Teacher for Semi-supervised Learning: In this chapter, we focus on the semi-supervised image classification problem. Among various approaches to SSL with deep neural networks, consistency-based methods have set state-of-the-art performance on multiple benchmark datasets. The primary goal of consistency-based methods is to encourage consistent probability predictions for the same data under either (1) different noise conditions or (2) different network parameterizations. However, a well-known problem with these recent consistency-based methods is the confirmation bias [Tar17]. We developed a local clustering method to mitigate the confirmation bias issue, where we cluster data points locally by minimizing the pairwise distance between neighboring points in feature space, aiming at correcting the predictions of unlabeled data points with the assistance of neighboring data points. Accordingly, the model would discover class-specific information from unlabeled data and as well enables better generalization on test data.

Chapter 4- Composite Consistency Regularization with Generative Adversarial Networks for Semi-supervised Learning: Recently developed Generative Adversarial Networks (GANs) have

shown promising results on image generation tasks, where they can generate high-quality realistic images even indistinguishable by humans. However, existing semi-supervised extensions of GANs are under-performing compared to the state-of-the-art non-GAN based semi-supervised approaches. Following the work in the previous chapter, we identified that the performance of GAN-based semi-supervised approaches are limited by the lack of consistency in class probability predictions under various local perturbations. We note that this problem has been addressed by consistency regularization in the consistency-based methods. In this chapter, we present a new composite consistency regularization by combining two consistency-based techniques – Mean Teacher and Interpolation Consistency Training, and incorporate it into the vanilla semi-GAN to address the prediction inconsistency problem. Our evaluation on benchmark datasets showed that our method achieves the new state-of-the-art performance among GAN-based semi-supervised approaches.

Chapter 5- Conclusions: Finally, we summarize our thesis contributions and discuss potential future research directions.

CHAPTER

2

RELATIONAL LONG SHORT-TERM MEMORY FOR VIDEO ACTION RECOGNITION

Spatial and temporal relationships, both short-range and long-range, between objects in videos are key cues for recognizing actions. It is a challenging problem to model them jointly.

In this chapter, we first present a new variant of Long Short-Term Memory, namely Relational LSTM, to address the challenge of relation reasoning across space and time between objects. In our Relational LSTM module, we utilize a non-local operation similar in spirit to the recently proposed non-local network [Wan18] to substitute the fully connected operation in the vanilla LSTM. By doing this, our Relational LSTM is capable of capturing long and short-range spatio-temporal relations between objects in videos in a principled way. Then, we propose a two-branch neural architecture consisting of the Relational LSTM module as the non-local branch and a spatio-temporal pooling based local branch. The local branch is utilized for capturing local spatial appearance and/or short-term motion features. The two branches are concatenated to learn video-level features from snippet-level ones which are then used for classification.

Experimental results on UCF-101 and HMDB-51 datasets show that our model achieves state-of-the-art results among LSTM-based methods, while obtaining comparable performance with other state-of-the-art methods (which use not directly comparable schema). Further, on the more complex large-scale Charades dataset we obtain a large 3.2% gain over state-of-the-art methods, verifying the effectiveness of our method in complex understanding.

2.1 Introduction

Action recognition (or video classification) as we use it is the task of assigning one of many class labels to a short video clip containing an action. Long untrimmed videos further carry a burden of multi-label assignment. Action recognition in videos plays a crucial role in many applications, e.g. visual surveillance [Lin08], sport video analysis [Soo14], human-machine interaction [Van13], video object tracking [Wan16a], etc. It has piqued the interest of the computer vision and deep learning communities, owing to the fact that the performances of state-of-the-art approaches are still well below human-level performance. Action recognition is a more complicated task when compared to still image classification because the temporal domain introduces variations in motion and viewpoints which have to be accounted for. Additionally, the use of a moving camera rather than a static camera introduces variations that could make optical flow based features less reliable. Besides, the interactions of multiple objects in actions make the classification task even more challenging.

Intuitively, action recognition requires models capable of learning key features such as:

Appearance features: Some actions are defined by certain special objects. For the "Blowing Candles" class in the UCF-101 dataset [Soo12], the presence of a candle in any one frame is sufficient to correctly classify the action.

Short-term motion features: Some actions are characterized by particular short-term motion with large variations of appearance. For the "Boxing Speed Bag" class in the UCF-101 dataset, a short optical flow snippet of the video clip is sufficient to correctly classify the action.

Long-term trajectory features: Similarly, some actions depend on long-term object trajectories which are defined by appearance and motion jointly. For the "Golf Swing" class in the UCF-101 dataset, a longer optical flow snippet of the video clip composed of the long-term motion of the arm and golf club is required to correctly classify the action.

Object interaction features: Multi-object based actions are frequently observed and entails modeling of object interactions. For the "Frisbee Catch" class in the UCF-101 dataset, the interaction of the Frisbee moving between 2 persons is required to correctly classify the action. Note that this category of features can be further divided into object interactions *across space* and *across time*, but exploring specific scenarios where they are applicable in an isolated fashion becomes harder.

Clearly, key features stated above do not present themselves in a mutually exclusive form in popular benchmark datasets; appearance features would be useful regardless of whether the action possessed interactions between objects. Nevertheless, one popular approach that has brought recent success to modeling these features is that of signal decomposition. Inspired by the discovery that the Human Visual System has separate processing pathways for different types of signals such as fast and slow motion [Kai96], researchers have employed multi-stream architectures to process each of these features separately. This paper is also motivated by the idea of signal decomposition.

Most significantly, the two-stream architecture of Simoyan and Zisserman [Sim14a] pioneered research in this area for action recognition from videos. They design two parallel 2D ConvNet streams to learn appearance features from RGB images and motion features from optical flow

fields. An alternative to the optical flow stream also presented itself in the form of 3D convolutions [Ji13], although they have been used together too since then [Car17]. Since 2D convolution on an optical flow stream and 3D convolution were deemed to not be able to capture sequential information required to model long-term trajectory features well, researchers turned to explore ways to address this next. The use of LSTMs following 2D convolutions [Ng15; Li18] and various temporal fusion strategies [Che17; Dib17; Kar17; Bil18] emerged as competitive candidates. However, object interaction features were left unexplored for the most part until recently where researchers used a self-attention mechanism to develop non-local neural networks [Wan18]. Their use of a "non-local operation" is able to model long-range interactions between objects in space and time, where high-level latent feature vectors built on top of a series of convolution blocks represent the objects. But their network is applied to short snippets cropped from the original videos. As such, they fail to explore truly modeling long-term trajectory features from information across the full lengths of videos.

In this paper, we explore the novel idea of introducing the non-local operation from [Wan18] into an LSTM module to create a Relational LSTM. We hypothesize that introduction of our "relational LSTM" block into a two-stream architecture would aid in the modeling of features that capture object interactions, while retaining the property of LSTM to model long-term trajectory information. Our main contributions can be summarized as follows:

- We develop a novel Relational-LSTM module that models object interaction features and can be inserted into existing diverse architectures as a plug-in module.
- We incorporate the Relational-LSTM module into a two-stream, two-branch architecture to perform video action recognition.
- We show experimentally through ablation studies that the introduction of our module leads to clear and unquestionable gains in performance.
- Our architecture should be considered the new state-of-the-art for video action recognition among LSTM-based architectures, beating the current best architecture by 1.2% on UCF-101 and 5.2% on HMDB-51.
- Our architecture performs comparably to the top-tier of state-of-the-art architectures overall on UCF-101 and HBDM-51 datasets.
- Our architecture outperforms state-of-the-art methods with comparable schema on the large-scale Charades dataset by 3.2%.

The rest of the paper is organized as follows. In section 2.2, we discuss related work. Section 2.3 describes our Relational LSTM module. Experimental results and their analysis are presented in Section 2.5. Finally, the conclusions and potential for future work are discussed in Section 2.6.

2.2 Related Work

2.2.1 Deep learning for appearance and short-term motion features

The success of 2D ConvNets did not immediately follow for video tasks, where hand-crafted features of Improved Dense Trajectories dominated. It was not until the work of [Sim14a] that deep learning approaches started to show comparable performance.

Two-stream ConvNets: In [Sim14a], the authors employ a two-stream architecture with 2D ConvNets to learn appearance and motion features to aid classification. They show that 2D ConvNets are by themselves capable of capturing short-term motion features with densely stacked optical flow fields as inputs. They average the predictions from a single RGB image and a stack of 10 consecutive optical flow fields after feeding them through two separate 2D ConvNets with identical structure. Based on this two-stream architecture, Wang $et\ al.$ [Wan16a] propose the averaging pooling operator to aggregate multiple frame-level predictions into a video-level prediction to model long-range temporal structure over the entire video.

3D ConvNets: When viewing a video as a sequential stack of RGB images, it is natural to think of extending 2D convolution to the temporal dimension to model spatio-temporal features in videos. In early stages, Ji $et\ al.$ [Ji13] have attempted to replace pre-computed complex hand-crafted features with 3D ConvNets, but their network is still quite shallow with only three convolutional layers. Following their work, Tran $et\ al.$ [Tra15] further exploit 3D ConvNets' properties under various video datasets and experimentally show that 3D ConvNets are competent for learning appearance and short-motion features. Inflated 3D ConvNet (I3D) proposed by Carreira and Zisserman [Car17] makes full use of successful pre-trained image classification architectures by inflating all the filters and pooling kernels from 2D to 3D and achieves state of the art performance on UCF-101 [Soo12] and HMDB-51 [Kue11] datasets. Their competitive performance drove research in this direction. The authors in [Xie18] show that factorizing 3D convolution of I3D into a 2D spatial convolution and a 1D temporal convolution, analogous to spatial factorization in Inception-v2 [Sze16], yields slightly better accuracy.

2.2.2 Sequence modeling for long-term trajectory features

The aforementioned methods successfully model appearance and short-term motion features. However, with regard to long-term trajectory information, they either used unsuitable inputs (frames not spanning long temporal range), or they used the 3D convolution operation or optical flow inputs, which capture only *local* temporal properties. Alternatively, diverse methods have attempted to encode *long-term* trajectory information. Some authors [Wan11; Wan13; Pen14] make use of dense point trajectories by tracking densely sampled points using optical flow fields. Subsequently, this hand-crafted shallow video representation was replaced by deep representations learned from neural networks. The basic idea is to sequentially aggregate frame-level feature representations extracted from either 2D ConvNets or 3D ConvNets so that long-term trajectory information is

encoded into the deep video-level representations. This could be done using either recurrent neural networks (RNNs) such as LSTMs or temporal feature fusion methods such as temporal pooling. Other methods to aggregate temporal information include the use of long-term feature banks [Wu19], multi-scale temporal convolutions [Hus19] and grammar models [Qi18b].

RNN-based architectures: The sequential modeling ability of LSTMs makes them appealing to use for capturing long-range temporal dynamics in videos. In [Bac11], the authors propose applying an LSTM to high-level feature vectors extracted from 3D ConvNets, but they only apply it on short video snippets of 9 frames. Ng $et\ al.$ [Ng15] add five stacked LSTM layers before the last fully connected layer of the two-stream ConvNets [Sim14a] and slightly improve the performance. Wang $et\ al.$ [Wan16b] design a hierarchical attention network, which is implemented by skipping time steps in higher layers of the stacked LSTM layers. Additionally, the authors in [Wan19c] employ ConvLSTM along with an attention mechanism [Xu15] to automatically focus on sequential saliency regions from high-level appearance feature maps.

Temporal feature fusion architectures: Temporal feature pooling [Ng15; Bil18] is the most popular temporal feature fusion method, which usually uses either max-pooling or average-pooling over the temporal dimension to aggregate frame-level features. Furthermore, the authors [Kar17; Bil18] propose adaptive temporal feature pooling by simultaneously learning an importance score for each frame and use it as weight in the pooling process. Cherian $et\ al.$ [Che17] propose generalized ranking pooling, which projects all frame-level features together into a low-dimensional subspace and use an SVM classifier on the subspace representation. The subspace is parameterized by several orthonormal hyperplanes and is designed to have a property of preserving the temporal order of video frames. Besides temporal pooling, Diba $et\ al.$ [Dib17] introduce a temporal linear encoding method, where they first aggregate frame-level features using element-wise multiplication and then project it to a lower dimensional feature space using bilinear model.

2.2.3 Non-local operation for object interaction features

Both the convolution and recurrent operations compute spatial and temporal features respectively in a primarily *local* fashion. Long-range dependencies are then modeled through applying these local operations repeatedly, often accompanied by downsampling, to propagate signals across space and time domains. The non-local operation [Wan18] is one hypothesized solution to handle the remaining problem of object interaction modeling.

Relation reasoning Exploring object interactions is equivalent to reason the relations of objects. Recently, the authors in [San17] propose Relation Network (RN), which is a neural network module primarily for relation reasoning. It uses one MLP layer on top of a batch of feature vector pairs to learn pairwise relations, where each instance in the batch is a pair of feature vectors at two particular positions in the input feature maps. They use this RN module on the visual question-and-answer problem and achieve super-human level performance. Following their work, Zhou $et\ al.$[Zho18a] explore its usage on temporal relation reasoning in videos, which is implemented by sparsely sampling frames from videos and employing RN module to learn the causal relations

among frames. A similar work for exploring object relations is proposed by [Hu18], where they use 'Scaled Dot-Product Attention' [Vas17] to compute object relations. Another way to learn object interactions is using Graph Parsing Neural Network (GPNN) [Qi18c], where a graph is built on latent features and the edge weights of the graph are learnt through training by message passing.

Non-local operation The non-local operation can be considered as a general form of 'Scaled Dot-Product Attention', as mentioned in [Wan18]. The key idea of non-local operation is that the output features of a position are computed as a weighted sum of the features from all positions in the input feature maps, which allows contributions from features in distant positions. The non-local idea originates from [Bua05] for image denoising, where the estimated value in pixel i is computed as the weighted average of all the pixels in the image. In [Wan18], they leverage it to design a non-local block for a neural network, which can be used as a plug-in module inserted into diverse neural network architectures.

2.3 Relational LSTM Module

Considering the importance of object interaction features in videos, we propose a Relational LSTM module, which not only inherits the sequential modeling ability from LSTM but also incorporates spatial relation reasoning and temporal relation reasoning through a non-local operation. More specifically, we generalize the non-local operation in [Wan18] to compute spatial relations among input features at a single snippet, and to compute temporal relations between input features and past learned features at previous time steps. Meanwhile, because of the use of LSTM, we create video-level feature representations by using selected snippets from the whole video, which inherently encodes long-term trajectory information in our representations. As is common in the deep learning community, the objects we refer to in our method are instance-level, which are represented by high-level feature vectors built on top of a series of convolution blocks.

Non-local operation: We first review the non-local operation defined in [Wan18]. Given input feature maps $\boldsymbol{X} \in \mathbb{R}^{N \times C}$, where N represents the number of positions in $\boldsymbol{X}$, and C represents the number of dimensions of feature vector at each position. If we represent $\boldsymbol{X}$ as $\{\boldsymbol{x}_i\}_{i=1}^{N}$, the output $\boldsymbol{z}_i$ at i-th position of response feature maps $\boldsymbol{Z} \in \mathbb{R}^{N \times C}$ is a weighted sum over all input feature vectors, as shown in Equation **2.1**.

$$\begin{aligned} \boldsymbol{z}_i &= \sum_{j=1}^{N} \omega_{ij} g(\boldsymbol{x}_j) \\ \omega_{ij} &= \frac{f(\boldsymbol{x}_i, \boldsymbol{x}_j)}{\mathscr{C}(\boldsymbol{X})} \end{aligned} \tag{2.1}$$

where $i, j, k \in \mathbb{R}^N$ are position indices, $f(\boldsymbol{x}_i, \boldsymbol{x}_k)$ represents compatibility function computing the similarity between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, and $g(\boldsymbol{x}_j)$ is a unary function computing a representation of the input feature vector at j-th position. $\mathscr{C}(\boldsymbol{X})$ is the normalization factor, usually denoted as $\mathscr{C}(\boldsymbol{X}) = \sum_{k=1}^{N} f(\boldsymbol{x}_i, \boldsymbol{x}_k)$ or $\mathscr{C}(\boldsymbol{X}) = N$.

The non-local operation in Equation **2.1** considers all positions in $\boldsymbol{X}$ regardless of positional distance, which is in sharp contrast to a convolutional operation of considering absolutely local neighborhood. This non-local operation explores the relations of features globally, which can be adopted to learn object interaction features far from each other in a spatio-temporal layout.

Generalized non-local operation: First of all, we generalize the non-local operation to compute relations between any two feature maps $\boldsymbol{X} \in \mathbb{R}^{N \times C_X}$, $\boldsymbol{Y} \in \mathbb{R}^{N \times C_Y}$. Given feature maps $\boldsymbol{Y}$, we compute the output $\boldsymbol{Z} \in \mathbb{R}^{N \times C_Z}$ with respect to $\boldsymbol{X}$ as shown in Eq. **2.2**

$$\begin{aligned} \boldsymbol{z}_i &= \sum_{j=1}^{N} \omega_{ij} g(\boldsymbol{y}_j) \\ \omega_{ij} &= \frac{f(\boldsymbol{x}_i, \boldsymbol{y}_j)}{\sum_{k=1}^{N} f(\boldsymbol{x}_i, \boldsymbol{y}_k)} \end{aligned} \tag{2.2}$$

This general form of the non-local operation can also be interpreted using the attention mechanism in [Vas17]. Given a query feature vector $\boldsymbol{x}_i$ and a set of input feature vectors $\{\boldsymbol{y}_i\}_{i=1}^{N}$, the output $\boldsymbol{z}_i$ is computed as an attentional weighted sum of all input feature vectors, where the attentional weights are computed by a compatibility function of the query feature vector and input feature vectors. In the rest of the paper, we abbreviate this general form as $\boldsymbol{Z} = r(\boldsymbol{X}, \boldsymbol{Y})$.

Relational LSTM: Given a video V, we first divide it into T segments $\{S_1, S_2, ..., S_T\}$ of equal duration, and randomly sample one short snippet K_t from its corresponding segment S_t. Suppose $\boldsymbol{X}_t \in \mathbb{R}^{H \times W \times C}$ for $t = 1, ..., T$ represents the high-level feature maps obtained after feeding K_t through some Convolution layers of a pre-trained CNN model, where C is the number of feature maps, and H and W are the spatial height and width of each feature map. After extracting $\{X_1, ..., X_T\}$ from a convolution layer, a temporal aggregation function is required to encode these snippet-level feature maps into video-level feature maps. Inspired by the deep learning architecture LSTMs, which not only inherit the sequential modeling ability from vanilla RNNs but can also capture long-term dependencies through the memory cell mechanism, we employ LSTM-based architectures to this end.

Usually, given feature maps X_t which preserve spatial layout, ConvLSTM [Xin15; Wan19b] or Bidirectional ConvLSTM [Son18] are the natural choices as the aggregation function because it encodes spatial information through its convolutional operations. However, the experimental results in [Sun17] have shown that ConvLSTM did not perform well on this recognition task, and it could not capture crucial object interaction features because of its convolution operations, which are applied to a *local* receptive field. We introduce the generalized non-local operation into LSTM architecture and present a new module called Relational LSTM. It differs from regular LSTM and ConvLSTM [Xin15] in the aspect that the general form of non-local operation is used in both input-to-state transitions and state-to-state transitions. The key equations of Relational LSTM are shown in Equation 2.3.

$$
\begin{aligned}
\boldsymbol{i}_t &= \sigma[r_{ix}(\boldsymbol{X}_t, \boldsymbol{X}_t) + r_{ih}(\boldsymbol{X}_t, \boldsymbol{H}_{t-1})] \\
\boldsymbol{f}_t &= \sigma[r_{fx}(\boldsymbol{X}_t, \boldsymbol{X}_t) + r_{fh}(\boldsymbol{X}_t, \boldsymbol{H}_{t-1})] \\
\boldsymbol{o}_t &= \sigma[r_{ox}(\boldsymbol{X}_t, \boldsymbol{X}_t) + r_{oh}(\boldsymbol{X}_t, \boldsymbol{H}_{t-1})] \\
\boldsymbol{g}_t &= \tanh[r_{gx}(\boldsymbol{X}_t, \boldsymbol{X}_t) + r_{gh}(\boldsymbol{X}_t, \boldsymbol{H}_{t-1})] \\
\boldsymbol{C}_t &= \boldsymbol{f}_t \circ \boldsymbol{C}_{t-1} + \boldsymbol{i}_t \circ \boldsymbol{g}_t \\
\boldsymbol{H}_t &= \boldsymbol{o}_t \circ \tanh(\boldsymbol{C}_t)
\end{aligned}
\tag{2.3}
$$

Here inputs $\boldsymbol{X}_t$, memory cell $\boldsymbol{C}_t$, hidden state $\boldsymbol{H}_t$, and gates $\boldsymbol{i}_t, \boldsymbol{f}_t, \boldsymbol{o}_t, \boldsymbol{g}_t$ have same functionalities as traditional LSTM. σ represents the logistic sigmoid non-linear activation function and tanh represents the hyperbolic tangent non-linear activation function.

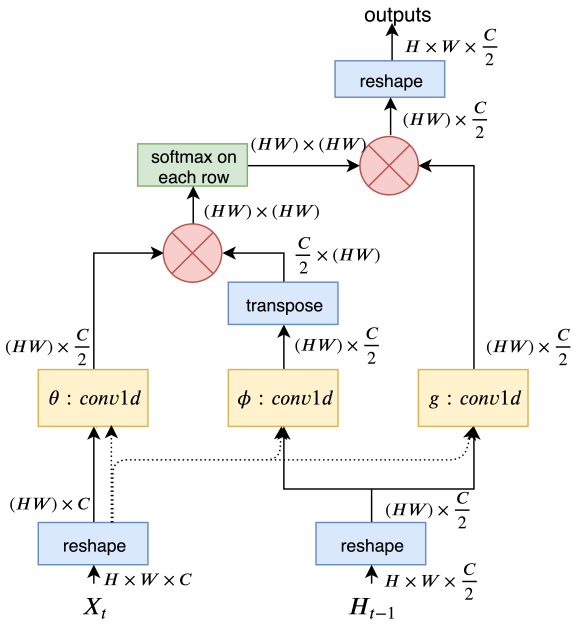


Figure 2.1 Generalized non-local operations $r(\boldsymbol{X}_t, \boldsymbol{X}_t)$ and $r(\boldsymbol{X}_t, \boldsymbol{H}_{t-1})$ in Relational LSTM. "$conv1d$" denotes 1D convolutional operation, "⊗" denotes matrix multiplication. Because the only difference between $r(\boldsymbol{X}_t, \boldsymbol{X}_t)$ and $r(\boldsymbol{X}_t, \boldsymbol{H}_{t-1})$ is the inputs, we use dashed arrow for inputs of $r(\boldsymbol{X}_t, \boldsymbol{X}_t)$, and use solid arrow for inputs of $r(\boldsymbol{X}_t, \boldsymbol{H}_{t-1})$.

We disentangle the mixed spatial-temporal relational reasoning. To ensure spatial relational reasoning is used, we adopt $r(\boldsymbol{X}_t, \boldsymbol{X}_t)$ in input-to-state transitions to model the feature interactions in same feature maps regardless of their relative positional distance. Regarding temporal relational reasoning, to model feature interactions of X_t with H_{t-1} which stores all important information from preceding feature maps, we adopt $r(\boldsymbol{X}_t, \boldsymbol{H}_{t-1})$ in state-to-state transitions. The detailed implementations of $r(\boldsymbol{X}_t, \boldsymbol{X}_t)$ and $r(\boldsymbol{X}_t, \boldsymbol{H}_{t-1})$ are shown in Fig. 2.1. In our implementations, we adopt the shape of $\boldsymbol{X}_t$ as $H \times W \times C$, and set $\boldsymbol{H}_{t-1}$ as $H \times W \times \frac{C}{2}$ to reduce memory cost. Given $\boldsymbol{X}_t$

and $\boldsymbol{H}_{t-1}$ as inputs of $r(\boldsymbol{X}_t, \boldsymbol{H}_{t-1})$, we first reshape $\boldsymbol{X}_t$ to $(HW) \times C$ and $\boldsymbol{H}_{t-1}$ to $(HW) \times \frac{C}{2}$. Then we apply the generalized non-local operation defined in Eq. **2.2** on them. We employ Embedded Gaussian function as $f(\boldsymbol{x}_i, \boldsymbol{y}_j)$ (shown in Equation 2.4) :

$$f(\boldsymbol{x}_i, \boldsymbol{y}_j) = e^{\theta(\boldsymbol{x}_i)^T \phi(\boldsymbol{y}_j)} \tag{2.4}$$

where $\theta(\boldsymbol{x}_i) = W_\theta x_i$ and $\phi(\boldsymbol{y}_j) = W_\phi y_j$ are two linear embedding function (implemented by 1D convolutional layer in Fig. 2.1). We consider $g(\boldsymbol{y}_j)$ also in the form of a linear embedding function expressed as $g(\boldsymbol{y}_j) = W_g \boldsymbol{y}_j$. The normalization function in the generalized non-local operation is implemented as a softmax layer. It is worth noting that even though we flatten the spatial layout when feeding $\boldsymbol{X}_t$ and $\boldsymbol{H}_{t-1}$ to the Relational LSTM block, we still preserve their relative positional information through Relational LSTM block and obtain output hidden feature maps H_t of shape as $(HW) \times \frac{C}{2}$, so that we can reshape H_t back to a shape of $H \times W \times \frac{C}{2}$. As for $r(\boldsymbol{X}_t, \boldsymbol{X}_t)$, it is implemented in the same way as mentioned above except that the inputs to it are both $\boldsymbol{X}_t$ (presented as dashed arrow in Fig 2.1).

2.4 Network Architecture

We build our architecture on top of "two-stream ConvNets" [Sim14a], where a spatial stream operates on RGB images and a temporal stream operates on sequences of 10 optical flow fields, and their prediction scores are fused by weighted averaging. Our network architecture of each stream is shown in Fig. 2.2. Obviously, the backbone network for image-level feature extraction plays an important role in our architecture and our architecture is compatible with most deep feature extractor networks, e.g. VGG-16 [Sim15], ResNet [He16a], BN-Inception [Li18], etc. However, our goal here is to evaluate the effectiveness of our proposed Relational LSTM module and two-branch network architecture, we fix the backbone network consistently through all our experiments. The ResNet-101-v2 [He16b] has been employed as the backbone network of our architecture. The detailed building blocks of ResNet-101-v2 are shown in Table 2.1. We extract the feature maps $\{X_1, ..., X_T\}$ after ResNet-101-v2 conv5_2 layer, and split our network into two branches. Two primary factors are determining our decision of the place for extracting the feature maps: 1. obtain relatively high-level features to represent objects; 2. have some convolutional block after the Relational LSTM module to further explore relative positional information.

Local branch: We design this branch for learning information that can be learned via local operations. The information can either be appearance features with RGB images as inputs, or short-motion features with optical flow fields as inputs. Specifically, we continue adopting ResNet-101-v2 architecture (conv5_3, global spatial average pooling) on every individual short snippet to obtain snippet-level feature representations, then integrating them to a video-level representation by temporal average pooling.

Non-local branch: We design this branch for learning object interactions in space and time, taking into account long-range temporal dependencies. The Relational LSTM module in this branch

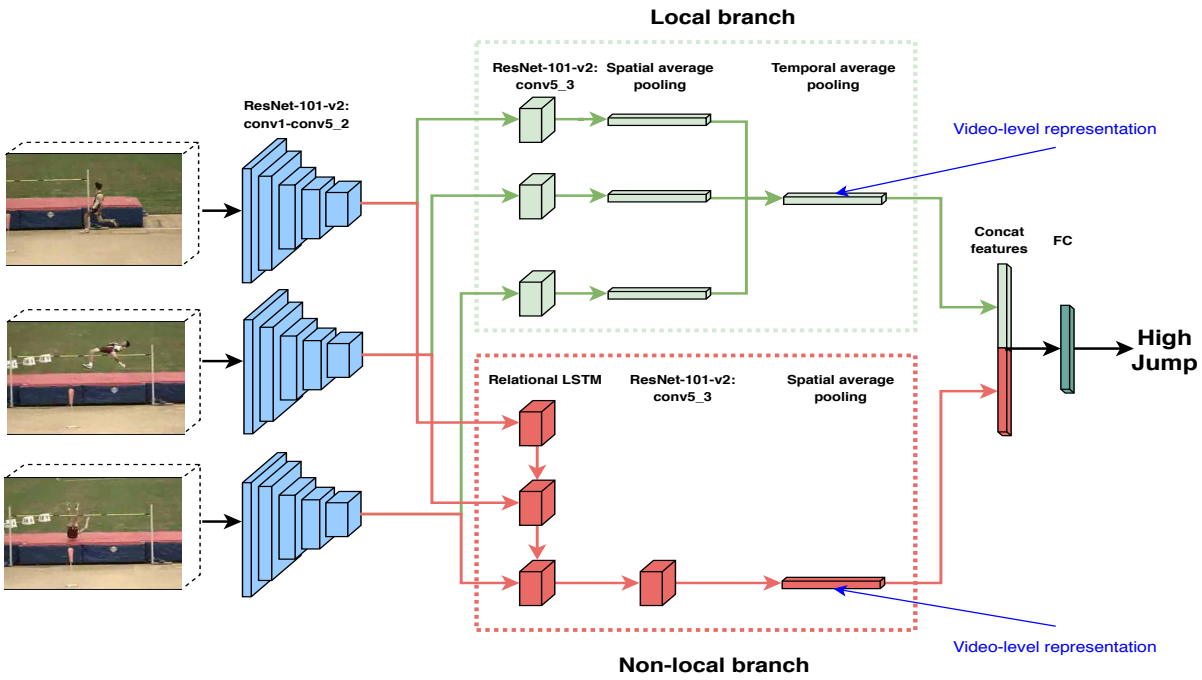


Figure 2.2 Network architecture. Only spatial stream is shown; the temporal stream is identical in structure. Given an input video, we divide it into T segments (T=3 in this case for succinct illustration) of equal duration, and randomly sample one short snippet from its corresponding segment. The short snippet is either a single RGB image (spatial stream) or a sequence of 10 optical flow fields (temporal stream). After feeding the selected short snippets through ResNet-101-v2 conv1 layer to ResNet-101-v2 conv5_2 layer, we have two separate branches. The local branch (green dotted box) captures local appearance and short-term motion features from snippets, and generates a video-level feature representation using temporal average pooling. The non-local branch (red dotted box) is for relation reasoning using Relation LSTM module. Eventually, we obtain an overall video-level feature representation by concatenating feature vectors generated by the two branches, and add one Fully Connected layer and optimize using a standard softmax with cross entropy classification loss.

can not only perform spatial relation reasoning and temporal relation reasoning among those snippets, but also obtain a video-level feature representation natively. As the spatial layout is preserved through the Relational LSTM module, and we further explore relative positional information by adding one residual block (ResNet-101-v2 conv5_3 layer) after Relational LSTM module. Specifically, in our design of Relational LSTM module, we add one batch normalization (BN) layer just before the first reshaping operator, and one 1×1 convolutional layer after the outputs to increase the number of feature maps from $H \times W \times \frac{C}{2}$ to $H \times W \times C$. We initialize H_0 of the Relational LSTM module with zeros, assuming that no information has been observed when the video starts.

Finally, we concatenate the video-level representations generated by the two branches as a complement to each other, and add one fully connected layer to perform classification by taking the argmax.

Table 2.1 ResNet-101-v2 architecture. Residual blocks are shown in brackets. The input is $224 \times 224 \times 3$, and downsampling is performed at conv3_1, conv4_1, conv5_1 with a stride of 2, 2.

layer name	101-layer	output size
conv1	7×7, 64, stride 2,2	112×112
pool1	3×3 max, stride 2,2	56×56
conv2_x	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	56×56
conv3_x	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	28×28
conv4_x	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 23$	14×14
conv5_x	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	7×7
global average pool, fc, softmax		1×1

2.5 Experiments

In this section, we first introduce the action recognition datasets we conduct experiments on and implementation details of our architecture including training and inferencing. Then, we study different aspects of our network on split 1 of UCF-101 dataset. Finally, we compare our architecture with state-of-the-art methods.

2.5.1 Datasets

We first conduct a series of experiments on two challenging video action recognition benchmark datasets, UCF-101 [Soo12] and HMDB-51 [Kue11]. The UCF-101 dataset consists of 13,320 short video clips with 101 action classes. The HMDB-51 dataset consists of 6,766 short video clips with 51 action classes. Both datasets contain video clips of typically 5-10 seconds and have more than 100 video clips in every action class. For both datasets, we use the provided evaluation schema and report the mean average accuracy over 3 training/testing splits. We also evaluate our architecture on the more complex and large-scale Charades dataset [Sig16]. The Charades dataset contains 9,848 videos of daily activities with 157 action classes. Unlike UCF-101 and HMDB-51 datasets, Charades dataset has longer video durations (around 30 seconds on average) and more than one action is performed concurrently or sequentially in each video. For evaluation on Charades, a single video is assigned to multiple class labels, and the standard mean avarage precision (mAP) measure is employed as the evaluation metric.

2.5.2 Implementation details

Training: We separately train the two streams of our architecture. The backbone ResNet-101-v2 is pre-trained on ImageNet. We employ the same data augmentations used in [Wan16a] e.g. horizontal flipping and multi-scale cropping. We adopt mini-batch stochastic gradient descent with a momentum of 0.9 and weight decay of 0.0005 as our optimizer. And we add a dropout layer right before the final fully connected layer. We train our model with Batch Normalization [Iof15]. Traditionally, batch mean and variance are adopted for training in Batch Normalization, and moving mean and moving variance are adopted for inferencing. However, because of the limited number of training instances and batch size, the estimates of mean and variance from each batch is highly variant from moving mean and moving average. It leads to severe over-fitting due to divergent distributions of data in training and inferencing stages after BN layer, as mentioned in [Wan16a]. Although completely freezing mean and variance parameters of all BN layers work well in practice, we found that adopting moving mean and moving variance in the training stage with a very small momentum (e.g. 0.001, 0.0005) to gradually update them makes more sense as they are progressively learning population mean and variance of the training dataset.

For the spatial stream, we initialize parameters for conv1-conv5_2 layers and conv5_3 layer in the local branch from pre-trained ResNet-101-v2 model, and initialize Relational LSTM module and conv5_3 layer in the non-local branch using Xavier-Glorot initialization [Glo10]. We set dropout rate to 0.8 and batch size to 24 (when T = 8). We train the model for 50 epochs. The learning rate starts at 0.0005 and is reduced by a factor of 10 after 35-th epoch and 45-th epoch.

For the temporal stream, we employ the cross-modality initialization strategy used by [Wan16a], where the parameters are initialized from our trained spatial stream model. We set dropout rate to 0.7 and batch size to 24 (when T=8). We train the model for 60 epochs. The learning rate starts at 0.0005 and is reduced by a factor of 10 after 45-th epoch and 55-th epoch.

Inference: For testing our architecture, we sample 1 RGB image or a sequence of 10 optical flow fields from the same position of each segment ($T = 8$ segments by default) to form one testing group. Meanwhile, we generate 5 crops (4 corners and 1 center) of 224×224 from the sampled images in the group. And we generate 4 groups by sampling from 4 positions with equal temporal spacing. So the final prediction scores for each stream are obtained by averaging over the 20 testing examples. We fuse the prediction scores of the two streams by weight averaging and the fusion is conducted before softmax normalization. Our empirical experiments suggest that the weight should be chosen around 0.5 for each stream in general.

2.5.3 Experimental evaluation on Relational LSTM network

In this experiment, we investigate how well standalone Relational LSTM module can perform on this task. We exclude local branch from our architecture and name the rest Relational LSTM network, and conduct experiments on it using split 1 of UCF-101 dataset. The spatial stream is initialized from pre-trained ResNet-101-v2 on ImageNet, and the temporal stream is initialized from the temporal

stream of Temporal Segment Networks (TSN) proposed in [Wan16a]. We compare its performance with our implementation of TSN using ResNet-101-v2 as backbone in Table 2.2. From the table, we observe that Relational LSTM network itself achieves a similar overall performance as TSN. Combining the two models yields obvious increases in both spatial stream and temporal streams, indicating that TSN and our Relational LSTM network are complementary to some extent. This complementary behavior is to be expected as the Relational LSTM network is designed to focus more on object interaction and long-term motion features, whereas TSN emphasizes learning appearance and short-motion features. Inspired by this complementary behavior, we designed the two-branch network architecture, where local branch aims to explore more on local features (e.g., appearance and short-motion features) by spatial-temporal pooling operations, and non-local branch aims to explore more on non-local features (e.g., object interactions and long-term motion features) through our proposed Relational LSTM module. It is worth noting that our local branch captures appearance/short-motion features in a different way compared to TSN. Our local branch aggregates snippet-level feature representation into video-level representation before FC layer, whereas TSN aggregates multiple snippet-level predictions after FC layer into a video-level predictions.

Table 2.2 Performance of Relational LSTM network, TSN and the ensemble of Relational LSTM network and TSN on UCF-101 (split 1). Relational LSTM network uses 8 segments ($T = 8$) in this experiment. We choose the best fusion weights in late fusion, 0.5 for spatial stream in Relation LSTM network, and 0.35 for spatial stream in TSN, and average weight for each stream in the ensemble.

Methods	Spatial	Temporal	Two-stream
TSN	87.0%	88.5%	93.8%
R-LSTM	86.9%	87.5%	93.8%
TSN + R-LSTM	88.6%	89.1%	94.2%

2.5.4 Ablation studies

In this section, we explore different aspects of our architecture. The experiments are all conducted on split 1 of UCF-101 dataset.

The number of input video segments: The most important parameter influencing our model performance is the number of input video segments. We mutate our architecture with a different number of input video segments from $T = 6$ to $T = 12$. The results are shown in Table 2.3. From the table, we see that there is a large increase (1.5%) in the spatial stream from $T = 6$ to $T = 8$, which implies that more segments in a video can provide richer information serving as a better video representation. However, on continuing to increase the number of segments, the performance takes a hit. This could partly owe to the naïve temporal pooling operation in the local branch causing a damping of the correct action signal. Therefore, we set $T = 8$ for the rest of our experiments.

Table 2.3 Performance of our architecture with different number of input video segments on UCF-101 (split 1).

Number of segments	Spatial	Temporal	Two-stream
6	87.2%	87.9%	94.2%
8	**88.7%**	**88.1%**	**94.4%**
10	88.1%	87.5%	94.1%
12	87.4%	87.4%	93.5%

Effect of introducing non-local branch: Since there are two branches in our architecture, we were compelled to quantify the benefit of introducing each branch into our architecture. In this experiment, we compare the performance of excluding non-local branch vs. excluding local branch vs. including both branches in our architecture. The results are shown in Table 2.4, and we find that there are improvements in each of the individual streams as well as overall. These small improvements on a large-scale dataset such as UCF-101 indicate that the contributions of the non-local branch in our architecture are significant and crucial.

Table 2.4 Performance comparison of our architecture with only local branch vs. non-local branch vs. two-branch on UCF-101 (split 1).

Methods	Spatial	Temporal	Two-stream
Local branch only	87.4%	87.3%	94.0%
Non-local branch only (R-LSTM)	86.9%	87.5%	93.8%
Two-branch	**88.7%**	**88.1%**	**94.4%**

Effect of adding Relational LSTM block to different positions of ResNet-101-v2: In this experiment, we aim to evaluate the impact of adding the Relational LSTM block to different positions of ResNet-101-v2. For the sake of still providing high-level features for Relational LSTM block and limiting computational cost, we mutate our architecture with Relation LSTM block added after conv5_1 layer, conv5_2 layer and conv5_3 layer of ResNet-101-v2. The results are shown in Table 2.5. From the table, we can see that there is performance increase when adding Relational LSTM block after conv5_2 layer instead of conv5_1 layer, indicating inputs of higher-level features to the Relational LSTM block would help it better learn long-term temporal dynamics and object interaction features. However, the performance decreases when adding Relational LSTM block after conv5_3 layer (right before spatial average pooling layer) instead of conv5_2 layer. One possible explanation is that the positional information preserved through Relational LSTM block is not further explored when the block is added right before a spatial average pooling layer.

Table 2.5 Performance comparison of our architecture with adding Relational LSTM block to different positions of ResNet-101-v2 backbone on UCF-101 (split 1).

Positions for adding Relational LSTM block	Spatial	Temporal	Two-stream
After conv5_1 layer	87.5%	87.7%	94.0%
After conv5_2 layer	**88.7%**	**88.1%**	**94.4%**
After conv5_3 layer	87.7%	87.5%	93.8%

2.5.5 Comparison with related work

Comparison with Temporal Segment Networks: First, we compare our architecture with TSN on split 1 of UCF-101 dataset. The reason we compare with TSN is that our local branch is most similar to TSN, the only difference being that our local branch aggregates snippet-level features before FC layer, whereas TSN aggregates snippet-level prediction scores after FC layer. The performance of our implementation of TSN is shown in Table 2.2, and the performance of our architecture is shown in Table 2.4. We observe that there is a 0.6% increase in overall performance. In Figure 2.3, we show the 10 classes of UCF-101 with the largest improvements in our architecture over TSN. Most of those classes involve object interaction features, e.g. PizzaTossing, Archery, implying that the introduction of non-local branch reaps benefits towards relation reasoning. We also visualize some instances of UCF101 (split 1) test data in Figure 2.4 to show the importance of learning object interactions and long-term motion features. These examples involve object interactions and cannot be easily recognized using solely appearance features and short-motion features. For example, the instance about JavelinThrow has been misclassified as LongJump by TSN as both classes have the 'running' sub-action in the early stages, whereas our architecture classifies its label correctly by a large margin. The possible reason is that our architecture learns about the interactions between the person and the javelin through time, but TSN erroneously focuses more on the running person.

Table 2.6 Comparison with LSTM-based state-of-the-art architectures on UCF-101 and HMDB-51 datasets. The performance accuracy is reported over all three splits.

Methods	UCF-101	HMDB-51
Two-Stream+LSTM [Ng15]	88.6%	-
VideoLSTM [Li18]	89.2%	56.4%
HAN [Wan16b]	92.7%	64.3%
L^2STM [Sun17]	93.6%	66.2%
R-LSTM(ours)	94.2%	-
Two-branch(ours)	**94.8%**	**71.4%**

Comparison with LSTM-based state-of-the-art methods: As our method belongs to the family of LSTM-based methods, we compare our method with other LSTM-based methods over all three

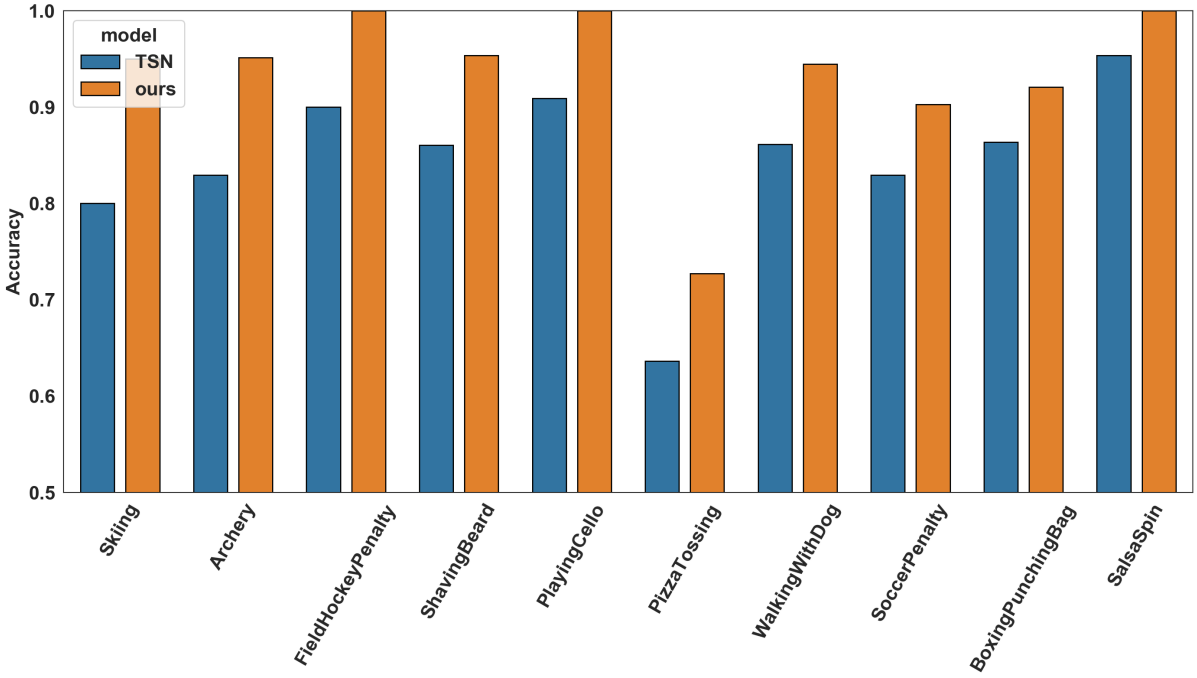


Figure 2.3 10 classes of UCF-101 (split 1) with largest improvements from TSN to our two-branch architecture.

splits of UCF-101 and HMDB-51, as shown in Table 2.6. From the table, we observe that our method outperforms other LSTM-based methods by a large margin. Even with the standalone Relational LSTM network (non-local branch), over three splits of UCF-101, we achieve 94.2% accuracy, which convincingly outperforms other LSTM-based methods. To the best of our knowledge, we achieve the best performance among all LSTM-based methods.

Comparison with non-LSTM-based state-of-the-art methods Before our work, LSTM-based methods have fallen behind non-LSTM-based methods in performance for a long period. So we provide comparisons of our method with current non-LSTM-based state-of-the-art methods over all three splits of UCF-101 and HMDB-51 datasets. We report these results in table 2.7. From the table, we can observe that we obtain performance comparable to the top tier of existing state-of-the-art methods. Moreover, our method is simple and the resulting networks very easy to train.

2.5.6 Experimental evaluation on large-scale Charades dataset

To further evaluate the effectiveness of our method, we conduct experiments on it using the large-scale and complex Charades dataset [Sig16]. The Charades dataset demands more spatial and temporal relation reasoning by asking the actors to perform a sequence of actions involving interactions with objects in specific scenes. Following the same training and evaluation schema [Sig16], we train our model on the actions extracted from 7,985 training videos and perform testing on the 1,863 validation videos. To perform inference on the entire video as a multi-label classification task,

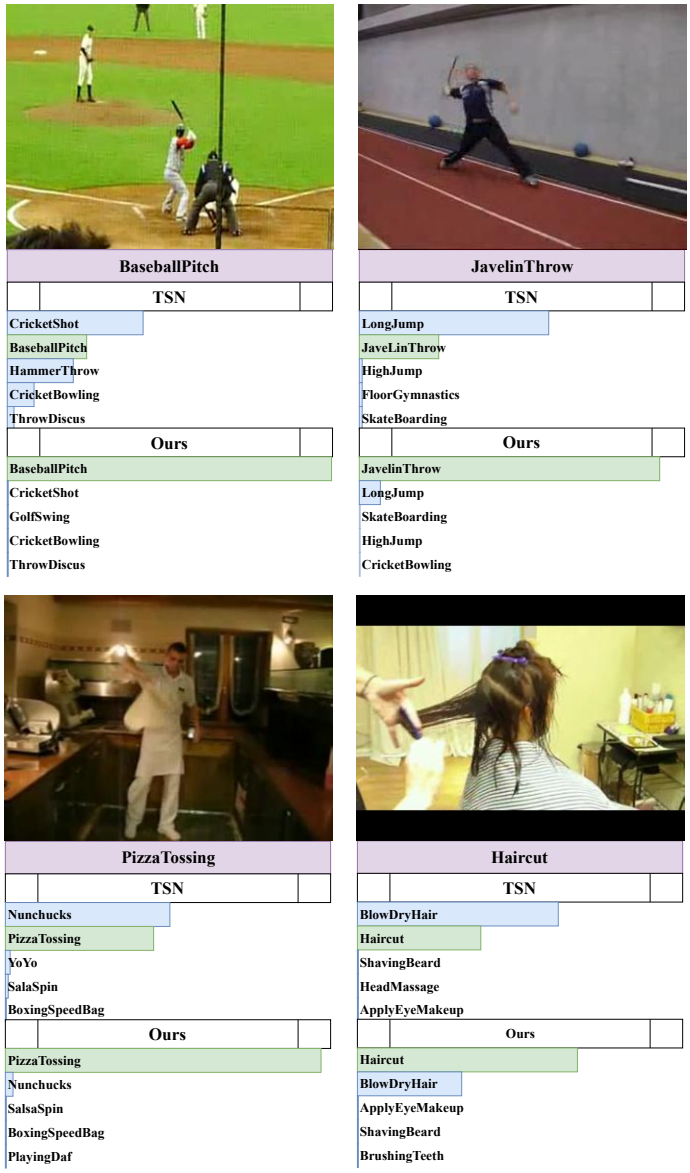


Figure 2.4 Visual comparison. A visual comparison of top-5 predictions between TSN with our architecture on some instances of UCF-101 (split 1) test data. The text within the purple bar indicates the ground truth label of the instance, and that within the green/blue bars indicates correct/incorrect predictions. Lengths of the bars correspond to prediction probabilities.

we sample 10 clips of 256 frames with equal temporal spacing from the video and evaluate each clip separately in the same way as mentioned in Section 2.5.2. The predicted logits before softmax normalization are considered as the prediction scores for each clip and the video-level prediction scores are obtained by aggregating clip-level prediction scores with max pooling. Following most recent works [Wu18; Zho18a], we perform experiments with only RGB images as inputs (spatial stream) for this dataset. For a fair and consistent comparison with state-of-the-art methods, we compare our model performance with methods that use 2D ConvNets as the backbone, as shown in

Table 2.7 Comparison with non-LSTM-based state-of-the-art methods on UCF-101 and HMDB-51 datasets. The performance accuracy is reported over all three splits. For a fair comparison, we only consider models that are pre-trained on ImageNet. We consistently set 0.45 for spatial and 0.55 for temporal stream in late fusion over the three splits of UCF-101 dataset and set 0.33 for spatial and 0.67 for temporal stream in late fusion over the three splits of HMDB-51 dataset.

Methods	UCF-101	HMDB-51
iDT [Wan13]	86.4%	61.7%
Two-Stream [Sim14a]	88.0%	59.4%
KVMDF [Zhu16]	93.1%	63.3%
ST-ResNet [Fei16]	93.4%	66.4%
Two-Stream I3D [Car17]	93.4%	66.4%
TSN [Wan16a]	94.0%	68.5%
ST-Multiplier [Fei17]	94.2%	68.9%
ST-Pyramid Network [Wan17]	94.6%	68.9%
MiCT-Net [Zho18b]	94.7%	70.5%
CoViAR [Wu18]	94.9%	70.2%
Ours	**94.8%**	**71.4%**

Table 2.8. As we can see from the table, our method outperforms other state-of-the-art methods by a large margin with only sparsely sampled RGB images as inputs. Furthermore, the performance difference between TSN and our method can validate the capability of our method for learning long-term temporal features and object interaction features.

2.6 Conclusions

In this paper, we present a novel Relational LSTM module which we embed into a two-branch architecture for relation reasoning across space and time between objects in videos. It complements

Table 2.8 Comparison with the state-of-the-art methods on Charades dataset. The classification **mAP** is reported as evaluation metric. For a fair comparison, only methods using 2D ConvNets backbone are reported. "*" indicates our re-implementation of the method.

Methods	Modality	Charades
Two-Stream [Sim14a]	RGB + flow	18.6%
Two-Stream + LSTM [Sig17]	RGB + flow	17.8%
Asyn-TF [Sig17]	RGB + flow	22.4%
CoViAR [Wu18]	RGB	21.9%
MultiScale TRN [Zho18a]	RGB	25.2%
TSN* [Wan16a]	RGB	25.6%
Two-branch(ours)	RGB	**28.8%**

most existing action recognition methods for their lack of relation reasoning and learns video-level representations implicitly for modeling long-term trajectory features. In our experiments, we validate the contributions of introducing Relational LSTM module, and demonstrate the performance of our architecture on three challenging action recognition datasets. Before our work, LSTM-based methods lagged behind non-LSTM-based methods in performance, especially those that use I3D convolutions [Car17]. Our method establishes state-of-the-art results among LSTM-based competitors and even enjoys performance comparable to non-LSTM-based counterparts on UCF-101 and HMDB-51 datasets. Moreover, our method achieves higher performance compared to the state-of-the-art methods that use 2D ConvNet backbones on the large-scale and complex Charades dataset.

CHAPTER

3

LOCAL CLUSTERING WITH MEAN TEACHER FOR SEMI-SUPERVISED LEARNING

The Mean Teacher (MT) model of Tarvainen and Valpola has shown favorable performance on several semi-supervised benchmark datasets. MT maintains a teacher model's weights as the exponential moving average of a student model's weights and minimizes the divergence between their probability predictions under diverse perturbations of the inputs. However, MT is known to suffer from confirmation bias, that is, reinforcing incorrect teacher model predictions.

In this chapter, we propose a simple yet effective method called Local Clustering (LC) to mitigate the effect of confirmation bias. In MT, each data point is considered independent of other points during training; however, data points are likely to be close to each other in feature space if they share similar features. Motivated by this, we cluster data points locally by minimizing the pairwise distance between neighboring data points in feature space. Combined with a standard classification cross-entropy objective on labeled data points, the misclassified unlabeled data points are pulled towards high-density regions of their correct class with the help of their neighbors, thus improving model performance.

We demonstrate on semi-supervised benchmark datasets SVHN and CIFAR-10 that adding our LC loss to MT yields significant improvements compared to MT and performance comparable to the state of the art in semi-supervised learning.

3.1 Introduction

In recent years, deep neural networks have achieved great success in many supervised machine learning tasks such as image classification [Kri12], object detection [He17], and video action recognition [Sim14a]. However, these efforts involved training deep networks on large amounts of labeled data, and such successes have not followed when only a small amount of labeled data is available. Human annotations are costly to obtain and often pose their own unique challenges [Bar12]. Herein lies the motivation for semi-supervised learning (SSL) techniques, which allow substantial amounts of cheaply available unlabeled data to supplement small amounts of expensive labeled data in training a machine learning model.

Among the various approaches to SSL with deep networks, consistency-based methods [Lai16; Tar17; Miy18] have set state of the art performance on multiple benchmark datasets [Net11; Den09]. The primary goal of consistency-based methods is to encourage consistent probability predictions for the same data under either (1) different noise conditions or (2) different network parameterizations. In other words, consistency-based methods enforce smoothness around each data point locally in output space. This class of methods has roots in Ladder Networks [Val15], a deep unsupervised learning method with an auto-encoder-like architecture. The model aims to learn abstract, invariant features in higher layers of the network that are robust to various kinds of noise added to either the input or intermediate representations. The Γ-model [Ras15] extends Ladder Networks to the semi-supervised setting by training the model using a multi-objective loss comprised of a supervised classification loss and an unsupervised consistency loss.

Following Γ-model, the Π-model [Lai16] and Mean Teacher model [Tar17] improve upon its performance by designing better quality teachers, which generate better learning objectives for their students. Taking a different perspective, VAT [Miy18] attempts to improve upon the Γ-model by selectively adding perturbations to inputs in *adversarial* directions, ones that can most quickly get the student and teacher predictions to deviate from each other.

However, a well-known problem with these recent consistency-based methods is confirmation bias [Tar17]. In requiring a student to be consistent with a teacher in terms of prediction, consistency-based methods assume that the teacher's predicted probabilities (targets for the student) are satisfactory. Confirmation bias in these approaches is caused by sub-optimal targets provided by the teacher model on unlabeled data, causing the student to reinforce these sub-optimal predictions. This problem is especially severe at the early stages of training, where the predictions from the teacher for many unlabeled data points can be inaccurate, causing those unlabel data points to be trapped in low-density regions in feature space, and those errors carry forward throughout the training. More importantly, the model is unable to learn discriminative features (class-specific knowledge) from those unlabeled data if they lie in low-density regions in feature space.

In this work, we propose a local clustering method to address the above limitation. We choose Mean Teacher [Tar17] as our baseline and add a novel regularizer to it. Motivated by widely used *local consistency assumption*[Zho04] in SSL, where nearby samples are likely to have the same label, we

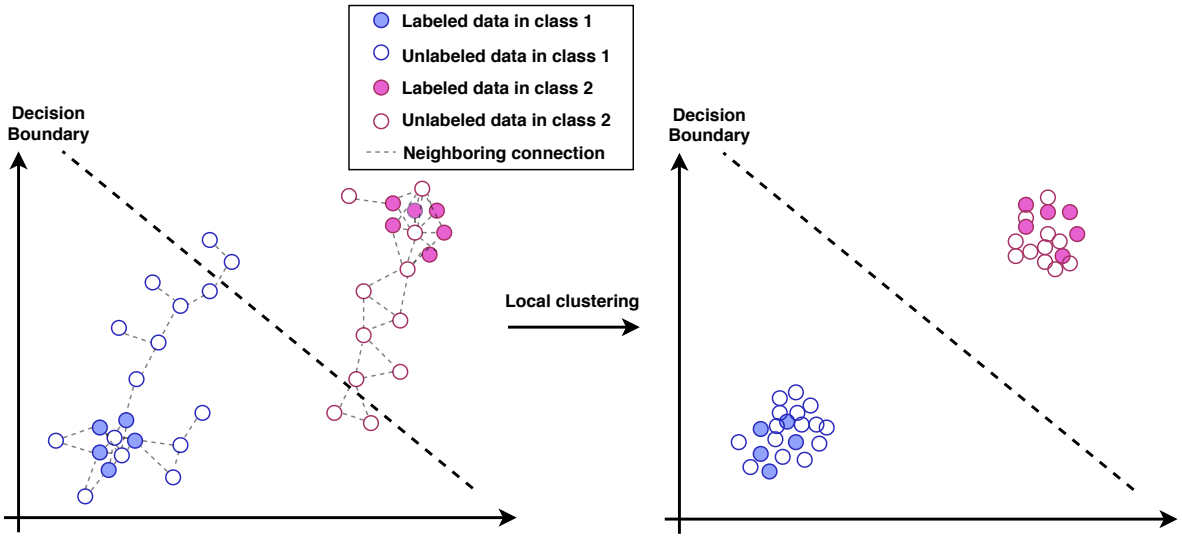


Figure 3.1 An illustration of the intuition behind Local Clustering in feature space. Each point represents the intermediate learned representation of one data sample. Best viewed in color in electronic form.

consider the vicinity of each unlabeled data sample in feature space and enhance the connections between neighboring samples. We propose the regularizer loss term that eventually pulls unlabeled data samples towards high-density regions with the assistance of neighboring samples. This is done by penalizing the distance between neighboring samples' feature representations. The intuition is that the weight updates thence caused would affect the learned features to help the model pick up on new cues from those unlabeled data samples, hopefully in the form of class-specific discriminative features. This intuition is illustrated in Figure 3.1.

Empirically, our approach substantially improves accuracy over Mean Teacher and achieves comparable performance with state of the art methods on benchmark datasets SVHN and CIFAR-10 following identical neural network architecture and evaluation rules.

3.2 Related Work

Recent deep SSL methods can be broadly categorized into deep clustering methods, deep generative methods, and consistency-based methods. We provide a short review of the first two and a detailed review of the more relevant consistency-based methods.

3.2.1 Deep Clustering methods

In general, deep clustering methods consist of two stages, where the first stage is a feature extractor network for learning deep feature embeddings from input data and the second stage is a clustering algorithm such as k-means or graph-based clustering on extracted feature embeddings guided by partial labeled information. Hsu and Kira [Hsu15] proposed a method to train the two stages jointly in an end-to-end manner. Instead of employing class labels for supervision, they define weak

labels in the form of similar/dissimilar data pairs, so that the clustering strategy is to minimize pairwise distances between similar pairs and maximize pairwise distances between dissimilar pairs. Following their work, Shukla *et al.* [Shu18] proposed ClusterNet, an auto-encoder based framework composed of both the aforementioned pairwise constraint clustering and k-means based clustering, where the latter additionally penalizes high intra-cluster variance. Another recent approach [Hae17] employs pairwise relations from a completely different perspective, where they define feature similarities as associations and form association cycles as a two-step walk starting and ending from labeled samples via an unlabeled sample in feature space. Accordingly, their objective is to encourage consistent association cycles that end at a sample of the same class as the starting sample and to penalize inconsistent ones. Apart from these methods, Kam-nitsas *et al.* [Kam18] propose a compact latent space clustering approach in the spirit of graph-based approaches. They build the graph directly in feature space instead of obtaining a pre-constructed graph from input space as the more general graph-based approaches, and form single and compact clusters for each class by encouraging balanced intra-cluster transition probabilities and zero inter-cluster transition probabilities.

3.2.2 Deep Generative methods

Deep generative methods have also been widely explored in recent years. Generative methods in SSL aim at estimating the joint distribution over data and class labels $p(\mathbf{x}, y)$, and compute a posterior distribution $p(y|\mathbf{x})$ via Bayes theorem. Two popular techniques with deep generative approaches in SSL are using variational auto-encoders (VAE) [Kin13; Rez14] and Generative Adversarial Networks (GANs) [Goo14b].

In VAE-based approaches[Kin14c; Rez15], the authors propose semi-supervised learning using VAEs by assuming that the data $\mathbf{x}$ are generated from the latent class variable y and a continuous latent variable $\mathbf{z}$. Consequently, the SSL problem is interpreted as a latent class missing variable problem. They construct a deep generative model and optimize it using variational inference to estimate the joint distribution $p(\mathbf{x}, \mathbf{z}, y)$ for labeled data and $p(\mathbf{x}, \mathbf{z})$ for unlabeled data. In these works, the continuous latent variable $\mathbf{z}$ is chosen as a diagonal Gaussian distribution. Maaløe *et al.* [Maa16] further introduce an auxiliary variable which helps generalize $\mathbf{z}$ to a non-Gaussian distribution and thus improve model performance.

In GANs [Goo14b], an adversarial game is set up between discriminator and generator networks. The objective of the generator is to generate fake samples that cannot be distinguished from real ones by the discriminator, which is tasked with telling them apart. Salimans *et al.* [Sal16] pioneered the extension of GANs to SSL. They propose a semi-GAN approach, where the objective of the generator is adjusted to match the first moment of the real data distribution in feature space, and the discriminator is a $(K+1)$-head classifier with the extra class referring to fake samples from the generator. Following their work, Dai *et al.* [Dai17] propose a complement generator to address the limitations of the feature matching objective in semi-GAN. They theoretically show that a preferred generator for GAN-based SSL is to encourage generating diverse fake samples in low-density regions

of the feature space, so that the real samples are pushed towards separable high-density regions and hence the discriminator is able to establish correct classification decision boundaries. The Localized GAN (LGAN) proposed by Qi $et\ al.$ [Qi18a] improves semi-GAN by making the discriminator resistant to local perturbations, where the perturbations are various fake samples produced by a local generator in the neighborhood of real samples on a data manifold.

Since our method builds on a consistency-based method, a detailed discussion for that category is provided in Section 3.3.1.

3.3 Preliminaries

Given a general SSL problem, let $\mathscr{D}_l := \{(x_i, y_i)\}_{i=1}^{n_l}$ represent a set of labeled samples, and $\mathscr{D}_u := \{x_i\}_{i=1}^{n_u}$ represent a set of unlabeled data samples. Typically, the number of labeled samples is much smaller than the number of unlabeled samples, $n_l \ll n_u$. The objective of SSL is to learn a mapping function $f(\mathbf{x};\theta) : \mathscr{X} \rightarrow \mathscr{Y}$, from the input space $\mathscr{X}$ to the label space $\mathscr{Y}$, where $\mathscr{X} \in \mathbb{R}^d$, $\mathscr{Y} \in \{1, 2, ..., K\}$ and K is the total number of classes. In our work, the mapping function $f(\mathbf{x};\theta)$ is chosen to be represented with a deep neural network. We can further decompose this as $f(\mathbf{x};\theta) = h(g(\mathbf{x};\theta_g);\theta_h)$, where $\mathbf{z} = g(\mathbf{x};\theta_g)$ is a feature extractor network mapping from input space $\mathscr{X}$ to latent space $\mathscr{Z}$, and $\mathbf{y} = h(\mathbf{z};\theta_h)$ is a classification network mapping from latent space $\mathscr{Z}$ to label space $\mathscr{Y}$. Consistency-based methods [Lai16; Tar17; Miy18] are the best performers among various classes of SSL methods and Mean Teacher is widely considered a state of the art baseline among consistency-based methods. We next briefly introduce consistency-based methods and the Mean Teacher model.

3.3.1 Review of Consistency-based methods

Consistency-based methods, also called consistency regularizers, encourage consistent probability predictions under small changes to either the inputs or the parameters of the model. Typically, the perturbations are represented in the form of input augmentations, dropout regularization [Sri14] or adversarial noise [Goo14b]. Given two random perturbations ξ' and ξ'' to input x_i, the general form of the consistency loss term can be formulated as the difference between a student model $f(\mathbf{x};\theta')$ and a teacher model $f(\mathbf{x};\theta'')$:

$$\mathscr{L}_{cons} = \mathop{\mathbb{E}}_{\{x_i\}_{i=1}^{n_l+n_u}} D[f(x_i,\xi';\theta'), f(x_i,\xi'';\theta'')] \tag{3.1}$$

where $D[\cdot,\cdot]$ measures the difference between the probability predictions of $f(\mathbf{x};\theta')$ and $f(\mathbf{x};\theta'')$, usually chosen to be Mean Squared Error or KL divergence. A theoretical analysis of the consistency loss in [Ath18] has shown that it improves the model generalization ability by penalizing the Jacobian norm and the Hessian eigenvalues of the predicted outputs with respect to inputs. It can be viewed as a regularization term leveraging both labeled and unlabeled data. The total loss for this class of methods integrates the consistency loss $\mathscr{L}_{cons}$ with the cross entropy loss $\mathscr{L}_{ce}$ defined on labeled

samples, expressed as

$$\mathscr{L} = \mathscr{L}_{ce} + \lambda \mathscr{L}_{cons} \tag{3.2}$$

where the coefficient λ is a hyperparameter that controls the relative importance of the consistency loss. In particular, several SSL approaches have been developed based on the idea of enforcing consistency including Mean Teacher.

Mean Teacher [Tar17]: In Mean Teacher, the weights θ'' of the teacher model are maintained through training as the exponential moving average (EMA) of the weights of the student model, formulated as

$$\theta''_t = \alpha\theta'_{t-1} + (1-\alpha)\theta'_t \tag{3.3}$$

where t indexes training iteration and the coefficient α is a smoothing hyperparameter. The main idea is to form a better teacher model, which gradually aggregates information from the student model in an EMA fashion. Eventually, a better teacher model can generate more stable probability predictions which serve as higher quality targets to guide the learning process of the student model. In [Tar17], the authors predicted that other methods which further improve the quality of targets would follow, and we believe that our local clustering is one such method.

Other consistency-based methods: In Π model [Lai16], the student model itself also serves as the teacher model, interpreted as $\theta' = \theta''$. Since two random perturbations ξ' and ξ'' are applied at each training iteration, the probability predictions of the same network from differently perturbed versions of the same input x_i could still be different. That difference can be considered as the inconsistency error to be minimized by the consistency objective. Temporal Ensembling [Lai16] utilizes the EMA of probability predictions of the student model as a teacher model's predictions, hence mitigating the high variation of teacher model predictions from one training iteration to the next. VAT [Miy18] and VAdD [Par18] impose adversarial perturbations to either the inputs or intermediate feature vectors in directions that would potentially maximize the difference in predictions between the student model and the teacher model.

3.4 Our method

Although the Mean Teacher model performs well among SSL methods empirically, the confirmation bias [Tar17] issue caused by inaccurate learning objectives generated from the teacher model still exists. The incorrect learning targets can trap unlabeled data samples in low-density regions or enforce them into high-density regions of incorrect class in feature space, preventing the learning of class-specific knowledge from them. Nevertheless, this issue can be addressed if the data samples can take cues from nearby data samples in feature space.

Our idea stems from two widely used assumptions in the SSL domain:

Clustering assumption [Cha03]: Samples are likely to have the same class label if there is a path

connecting them passing through regions of high density only.

Local consistency assumption [Zho04]: Nearby samples are likely to have the same label. Samples on the same structure (typically, a manifold) are likely to have the same label.

From *local consistency assumption*, each unlabeled data point and its neighbors are likely coming from the same class. Hence, the main idea behind our method overcoming confirmation bias is to employ the help of neighboring points around every unlabeled data point to pull those misclassified unlabeled data to the high-density regions of their correct class in feature space. This is done by penalizing the magnitude of pairwise Euclidean distances between feature representations of data points and their neighbors. More specifically, we dynamically build a graph for each batch of training data (including both labeled and unlabeled data points) in feature space, and move labeled data points and their neighboring unlabeled data points closer to each other, while also moving neighboring unlabeled data points towards each other. Since labeled data are affected by class information through the supervised loss term, they implicitly lie in high-density regions of the feature space. Unlabeled data points are gradually pulled towards labeled data points of their correct class, and in consequence high-density regions, either by a direct connection or a path. Figure 3.1 illustrates this idea. In this context, the hope is that moving unlabeled data points towards a high-density region of their correct class would help the model discover class-specific features from unlabeled data to enable better generalization to unseen test data.

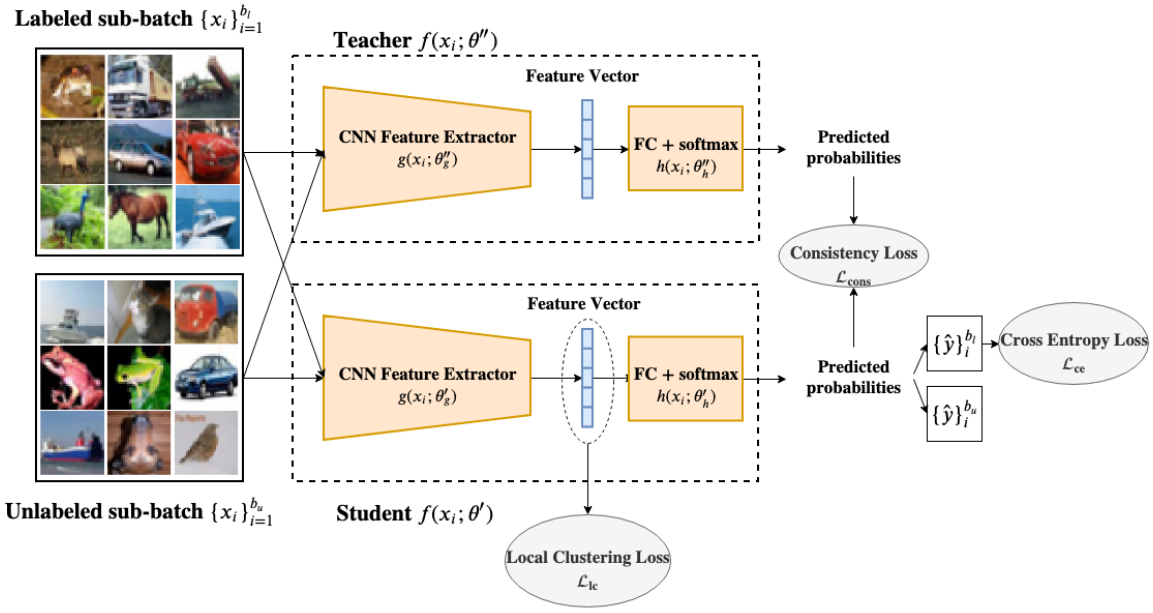


Figure 3.2 Network architecture. "FC" represents a fully connected neural network layer. Best viewed in color in electronic form.

3.4.1 Local Clustering

Since our goal is to get help from neighboring samples, our question boils down to finding neighboring data samples. In traditional graph-based SSL methods [Zhu03], neighboring data samples are found by pre-defining an adjacency matrix in input space on some distance measure (e.g., Euclidean distance or Cosine distance), but this is infeasible for capturing the perceptual similarity between images. As an alternative, we find neighboring data samples by searching in an intermediate learned representation space. At each training iteration, we sample a sub-batch of labeled data from $\mathscr{D}_l$ of size b_l and a sub-batch of unlabeled data from $\mathscr{D}_u$ of size b_u, and obtain their latent feature vectors by feeding them through the feature extractor network $\mathbf{z} = g(\mathbf{x}; \theta'_g)$ of the student model, as shown in 3.2. Then we perform local clustering by computing pairwise feature distances using a distance metric in feature space. In our work, we employ Euclidean distance as the distance function and empirically find it to perform well.

More formally, we formulate pairwise distance relationships as a weighted graph in feature space, and w_{ij} is the weight of an edge between samples x_i and x_j as

$$w_{ij} = \begin{cases} exp\left(-\frac{||z_i - z_j||^2}{\epsilon}\right) & \text{if } ||z_i - z_j||^2 \leq \epsilon \\ 0 & \text{otherwise} \end{cases} \tag{3.4}$$

where $z_i = g(x_i; \theta'_g)$ and $z_j = g(x_j; \theta'_g)$ are the latent feature vectors of x_i and x_j, ϵ is a cut-off distance threshold and $||.||$ is the L2-norm. Samples x_i and x_j are considered as neighbors if $w_{i,j} > 0$. There are two main considerations we take into account when computing edge weights. Firstly, if two samples x_i and x_j are too far from each other, they are not considered as neighbors. This motivates a cut-off threshold ϵ, which deems pairs with an Euclidean distance greater than ϵ as non-neighbors; secondly, the similarity between neighboring samples declines as their distance increases, so we use the negative exponential of their Euclidean distance as the distance decay function to represent their mutual effectiveness.

Similar to those deep clustering methods [Hsu15; Shu18], we minimize pairwise distance for every neighboring sample pair in feature space. We formulate a local clustering loss $\mathscr{L}_{lc}$ as

$$\begin{aligned} \mathscr{L}_{lc} = & \mathop{\mathbb{E}}_{\{x_i\}_{i=1}^{n_l}, \{x_j\}_{j=1}^{n_u}} [w_{ij} ||g(x_i; \theta'_g) - g(x_j; \theta'_g)||^2] + \\ & \mathop{\mathbb{E}}_{\{x_m\}_{m=1}^{n_u}, \{x_n\}_{n=1}^{n_u}} [w_{mn} ||g(x_m; \theta'_g) - g(x_n; \theta'_g)||^2] \end{aligned} \tag{3.5}$$

Notice that we minimize pairwise distance between labeled samples and their neighboring unlabeled samples, as well as between neighboring unlabeled samples. It is not necessary to minimize pairwise distance between neighboring labeled samples, owing to the fact that they are already supervised sufficiently by the cross entropy loss term $\mathscr{L}_{ce}$.

Consequently, our total loss function is formulated as

$$\mathscr{L} = \mathscr{L}_{ce} + \lambda_1 \mathscr{L}_{cons} + \lambda_2 \mathscr{L}_{lc} \tag{3.6}$$

where the coefficients λ_1, λ_2 are hyperparameters controlling the importance of the losses $\mathscr{L}_{cons}$ and $\mathscr{L}_{lc}$. A figure illustration of our method is presented in Figure 3.2. Note that the computational cost introduced by local clustering regularizer is negligible since the pairwise Euclidean distances are only computed in low-dimensional feature space within a mini-batch. The pseudocode of our algorithm is presented in Algorithm 1.

It is worth pointing out the difference between our method and one other recent method, SNTG [Luo18]. In SNTG, the authors are motivated to enforce smoothness across neighboring samples in feature space, while our motivation is to more directly affect the learned representations of unlabeled samples to pull them towards a high-density region of their correct class with the help of neighboring samples. SNTG constructs a teacher graph from the predicted outputs of the teacher model and uses it to guide the clustering, while our method builds a weighted graph directly from the feature representations of the student model in feature space. As we know, the predicted outputs generated by the teacher model could be inaccurate (which is why performance suffers from confirmation bias), so the clustering process in SNTG could be misguided. Moreover, we present the empirical comparison of our method versus SNTG in Section 3.5.2 as well.

3.5 Experiments

In this section, we conduct extensive experiments to evaluate the performance of our proposed method and ablation studies to validate hyperparameter choices.

3.5.1 Experimental setup

We quantitatively evaluate our proposed method on two widely used benchmark datsets: SVHN and CIFAR-10. Both datasets contain RGB images of size 32×32 and have 10 classes. To show that our method consistently improves the performance of Mean Teacher (MT) method [Tar17], we first implement MT as the baseline, and extend it by incorporating our local clustering (LC) method. When training our LC with MT, we incorporate the LC objective after the training of MT is close to convergence.

For a fair comparison with the state of the art methods, we employ an identical 13-layer ConvNet as used in previous works [Lai16; Tar17; Miy18]. We train the models with a batch of 32 labeled samples and 128 unlabeled samples. We employ stochastic gradient descent (SGD) with Nesterov momentum as our optimizer. Following the practice of [Tar17], we too have a ramp-up phase for both consistency loss and local clustering loss where their corresponding coefficients are increased from 0 to the final values in the first few epochs right after incorporating them. The training details for each dataset are explained in the following:

SVHN: We apply random translation to augment the training data of SVHN. When training MT model, we follow the same training schema as in MT paper [Tar17], where we ramp up the consistency loss coefficient λ_1 from 0 to its maximum value 100.0 in the first 5 epochs. We adopt the same sigmoid-shaped function $e^{-5(1-x)^2}$ [Tar17] as our ramp-up function, where $x \in [0,1]$. We set the EMA coefficient of α to 0.995. We train the MT model for 300 epochs with a learning rate of 0.05 and linear decay the learning rate from 0.05 to 0 in another 200 epochs afterward. When training our LC with MT, we first train MT model with the same hyperparameter settings as described above for the first 300 epochs. Then we incorporate our LC loss with a ramp-up phase of 50 epochs to increase our LC loss coefficient λ_2 from 0 to its maximum value 20.0. We employ the same ramp-up function as used in MT. We train the model with a learning rate of 0.05 for the first 400 epochs and linear decay the learning rate from 0.05 to 0 in another 200 epochs. Also, we set the cut-off threshold of ϵ to 50.0 through all the training experiments on SVHN.

CIFAR-10: We augment CIFAR-10 training data with both random translation and horizontal flips. When training MT model on CIFAR-10, we also follow the same training schema as we train MT on SVHN. We ramp up the consistency loss coefficient λ_1 from 0 to its maximum value 100.0 in the first 5 epochs. We set the EMA coefficient of α to 0.99. We train the MT model for 600 epochs (on CIFAR-10 with 2,000 labeled samples) or 800 epochs (on CIFAR-10 with 4,000 labeled samples) with a learning rate of 0.05, while linearly decaying learning rates from 0.05 to 0 in the last 200 epochs afterward. When training our LC with MT, we also first train MT with the same hyperparameter settings for 600 epochs. Then we incorporate our LC loss with a ramp-up phase of 100 epochs for λ_2 from 0 to its maximum value 10.0, using the same ramp-up function as in MT. We train the model with a learning rate of 0.05 for 800 epochs (on CIFAR-10 with 2,000 labeled samples) or 1,000 epochs (on CIFAR-10 with 4,000 labeled samples) and then decay learning rate from 0.05 to 0 in the last 200 epochs afterward. Moreover, the cut-off threshold of ϵ is set to 40.0 (on CIFAR-10 with 2,000 labeled samples) or 50.0 (on CIFAR-10 with 4,000 labeled samples) through all the training experiments.

3.5.2 Results

SVHN: The SVHN dataset consists of 73,257 training samples and 26,032 test samples. We train the models on SVHN training images with 500 and 1,000 randomly labeled samples respectively, and evaluate the performances on the corresponding test data. We follow the same evaluation standard used in state-of-the-art approaches (comparing mean of test errors). Table 3.1 shows that incorporating our LC loss on MT improves the performance of the model (the mean MT+LC error is over 2 standard deviations lower than the mean MT error rate) in both 500 and 1,000 labeled samples settings and achieves the best test performances among state of the art methods. For a fair comparison purpose, we only compare with methods that employ the same 13-layer ConvNet as network architecture in the table. Some recent works (e.g. MixMatch[Ber19a], ADA-Net[Wan19a]) that adopt deep ResNet model as network architecture are not included.

Table 3.1 Error rate percentage comparison with the state of the art methods on SVHN and CIFAR-10 over 10 runs. "*" indicates our re-implementation of Mean Teacher and "LC" denotes our local clustering method. Only methods that employ 13-layer ConvNet as their network architecture are reported for a fair comparison purpose.

Method	SVHN		CIFAR-10	
	$n_l = 500$	$n_l = 1000$	$n_l = 2,000$	$n_l = 4,000$
semi-GAN [Sal16]	18.44 ± 4.80	8.11 ± 1.30	19.61 ± 2.09	18.63 ± 2.32
Bad GAN [Dai17]	-	7.42 ± 0.65	-	14.41 ± 0.30
Local GAN [Qi18a]	5.48 ± 0.29	4.73 ±0.29	-	14.23 ± 0.27
Π model [Lai16]	6.65 ± 0.53	4.82 ± 0.17	-	12.36 ± 0.31
TempEns [Lai16]	5.12 ± 0.13	4.42 ± 0.16	-	12.16 ± 0.31
Mean Teacher [Tar17]	4.18 ± 0.27	3.95 ± 0.19	15.73 ± 0.31	12.31 ± 0.28
VAdD [Par18]	-	4.16 ± 0.08	-	11.32 ± 0.11
VAT + EntMin [Miy18]	-	3.86 ± 0.11	-	10.55 ± 0.05
TempEns + SNTG [Luo18]	4.46 ± 0.26	3.98 ± 0.21	13.64 ± 0.32	10.93 ± 0.14
MT + SNTG [Luo18]	3.99 ± 0.24	3.86 ± 0.27	-	-
MT*	3.91 ± 0.11	3.80 ± 0.09	12.37 ± 0.29	9.93 ± 0.16
MT + LC **(ours)**	**3.54 ± 0.17**	**3.35 ± 0.09**	**11.56 ± 0.31**	**9.26 ± 0.16**

Furthermore, we visualize the test error as a function of training epoch on SVHN with 500 labeled samples in Figure 3.3 (left). The figure clearly shows that there is a substantial reduction on error rate in both student and teacher models right after the LC objective is incorporated starting from 300 epochs, demonstrating that the performance improvement is owing to the introduction of LC loss.

CIFAR-10: The CIFAR-10 dataset consists of 50,000 training samples and 10,000 test samples. Similarly, we train the models on CIFAR-10 training images with 2,000 and 4,000 randomly labeled samples, and evaluate them on CIFAR-10 test data. As seen in Table 3.1, LC with MT substantially improves test accuracy of the MT model, and outperforms all state of the art methods. Meanwhile, the test error curves on CIFAR-10 with 2,000 examples in Figure 3.3 (right) reaffirms that the improvement is consistent regardless of the dataset used.

3.5.3 Ablation studies

In this section, we study the effects of the two new hyperparameters introduced by our LC objective (ϵ and λ_2). Note that for these experiments, all other hyperparameters are kept fixed to values used in our implementation of the baseline MT model. We conduct all these experiments on SVHN with 500 labeled samples.

Effect of cut-off threshold ϵ and : One essential hyperparameter introduced by our LC loss is the

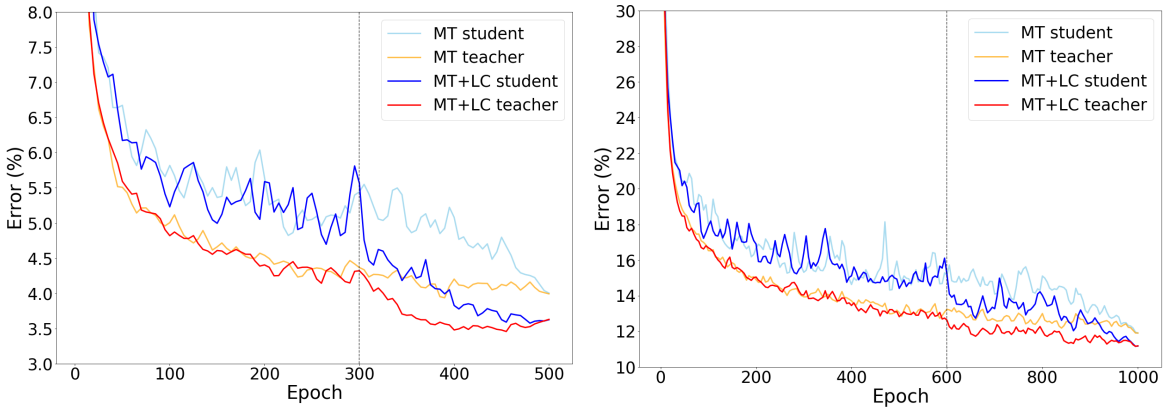


Figure 3.3 Smoothed test error curves of MT and MT+LC on SVHN (left) with 500 labeled samples, and CIFAR-10 (right) with 2,000 labeled samples. The LC objective is incorporated into training from 300 epochs on SVHN and 600 epochs on CIFAR-10. Best viewed in color in electronic form.

cut-off threshold ϵ. If ϵ is too small, almost no neighbors are found around each data point. To the contrary, distant points are considered as neighbors when ϵ is too large. We vary ϵ in a broad range of values and the results are shown in Figure 3.4. Note that the model with $\epsilon = 0$ is equivalent to the MT model as no neighbors would be considered in this scenario. From the figure, we observe that there is a decreasing trend on error rate when increasing ϵ from 0 to 50, which implies LC works better with more neighbors taken into account. However, there is an error rate increase if ϵ is increased further, indicating too large ϵ would degrade model performance. In our experiments, we also find that the LC method would fail for $\epsilon > 500$, leading to a scenario we term "distribution collapse", where the entire dataset is mapped to a single point by the feature extractor network. Nevertheless, all explored values of ϵ in a wide range that do not lead to distribution collapse improve upon the MT model.

Effect of LC loss weight λ_2: We also evaluate model performance on different λ_2 values, as shown in Figure 3.5. Similarly, the model with $\lambda_2 = 0$ is purely a MT model. The figure shows that the error rate decreases gradually as λ_2 increases, and arrives at a steady state when $\lambda_2 \geq 10$. The model sees a distribution collapse failure case similar to $\epsilon > 500$ when $\lambda_2 > 50$, owing to the fact that LC loss dominates the learning process. Similarly, all explored values of λ_2 in a wide range that do not lead to distribution collapse improve upon the MT model.

3.5.4 Visualization

To better understand the effect of LC loss, we visualize the intermediate learned representations of MT and MT + LC on CIFAR-10 training data with t-Distributed Stochastic Neighbor Embedding (t-SNE [Maa08]) in Figure 3.6. We train the models on CIFAR-10 with 2,000 labeled samples, extract and project the intermediate learned representations from the layer on which LC loss was added $z \in \mathbb{R}^{128}$ for all training data into 2-dimensional feature space. From the figure, we observe that the

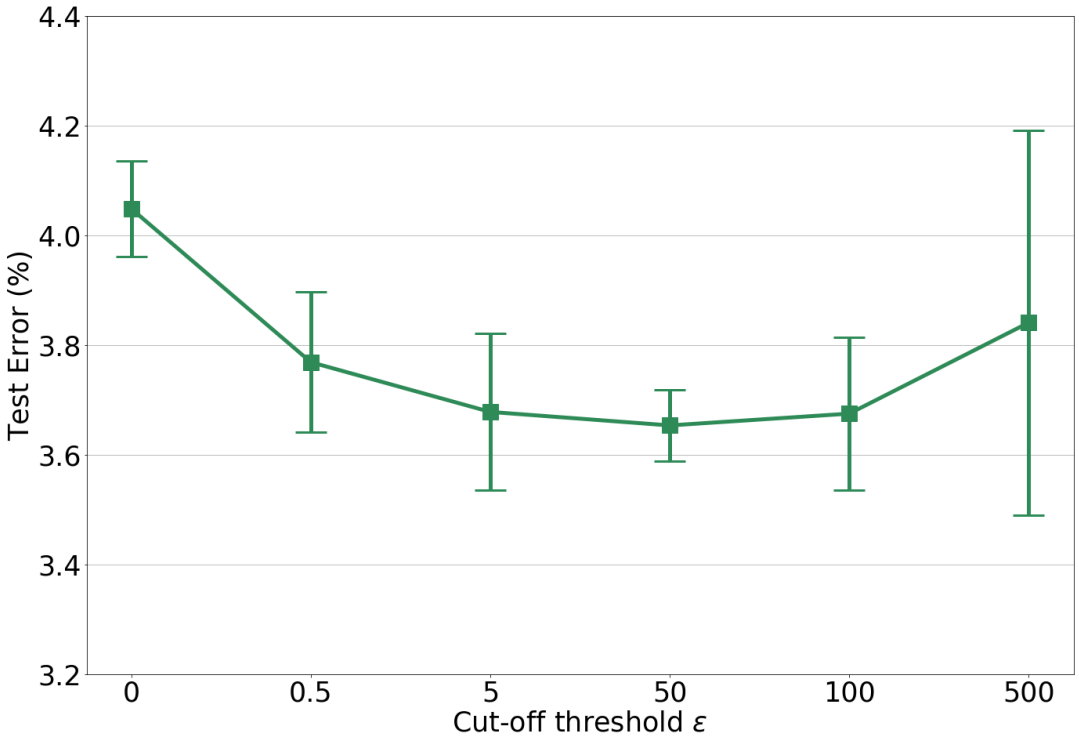


Figure 3.4 Test errors of LC with MT with different cut-off thresholds on SVHN with 500 labeled samples over 5 runs.

labeled data of the different blobs (which also correspond roughly to predicted class labels) are more separated in MT + LC (right) compared to MT (left). Additionally, LC causes the space to warp such that the labeled points take a majority of unlabeled points of the same class to a high-density region that hugs the edge of its cluster. An additional insight here is that this visualization shows counter to intuition that distribution alignment between labeled and unlabeled points is not necessary for higher classification accuracy.

We also visualize the intermediate learned representations of MT and MT + LC on CIFAR-10 test data with t-SNE in Figure 3.7. From the figure, it is undeniable that the learned representations of different classes are more distinctively separated in MT + LC (right), while they are mixed in MT (left), validating that the LC loss helps learn more class-specific knowledge.

3.6 Conclusions

In this work, we have proposed a new method called Local Clustering to tackle the confirmation bias issue in Mean Teacher method. In particular, it considers the correlations between nearby data points in feature space and hence corrects misclassified data points by pushing them to the high-density region of their ground truth class with the help of neighboring points. In our experiments, we validate the effectiveness of our method with MT on two benchmark datasets SVHN and CIFAR-10, and achieve performance comparable to state of the art among semi-supervised methods.

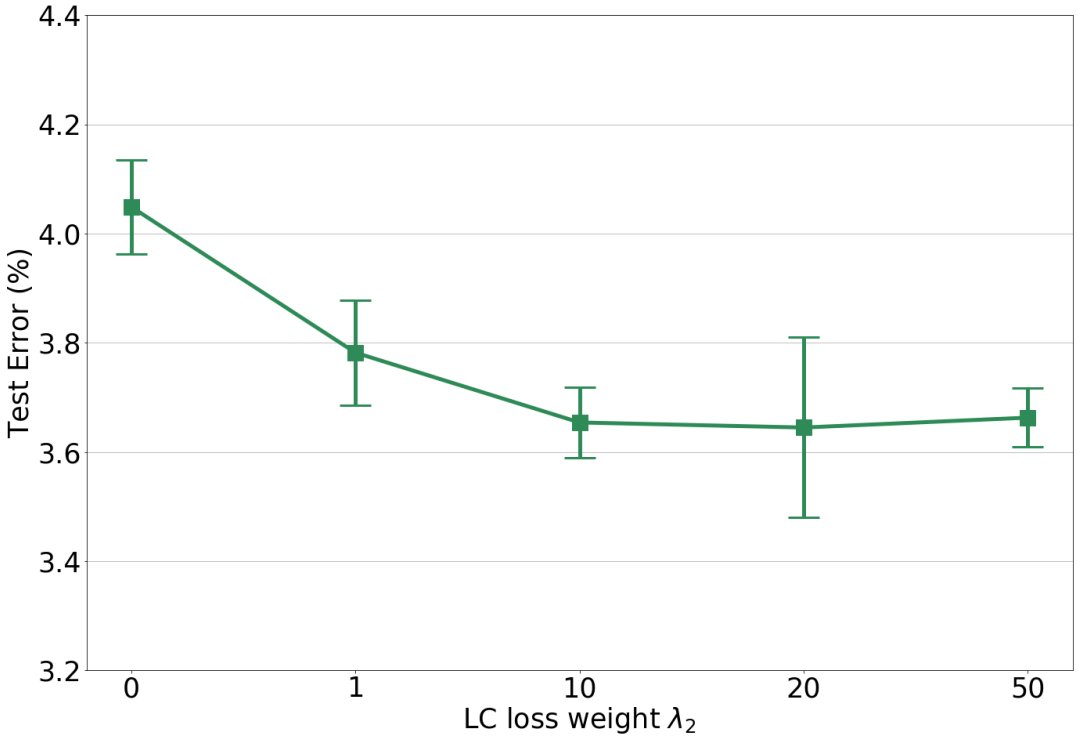
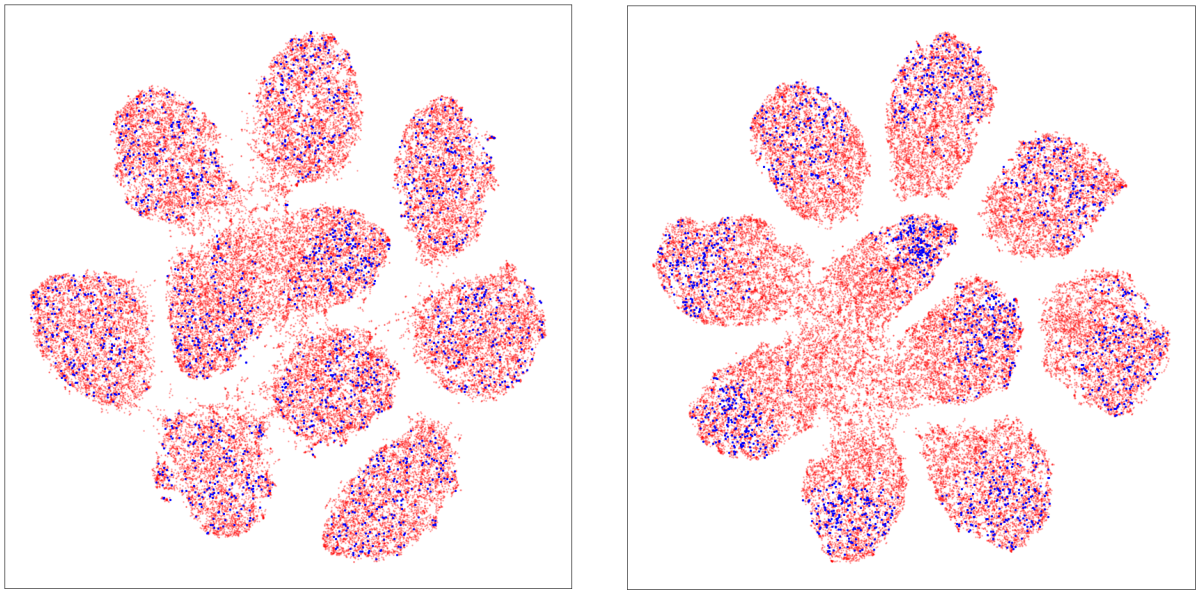


Figure 3.5 Test errors of LC with MT with different LC loss weights on SVHN with 500 labeled samples over 5 runs.



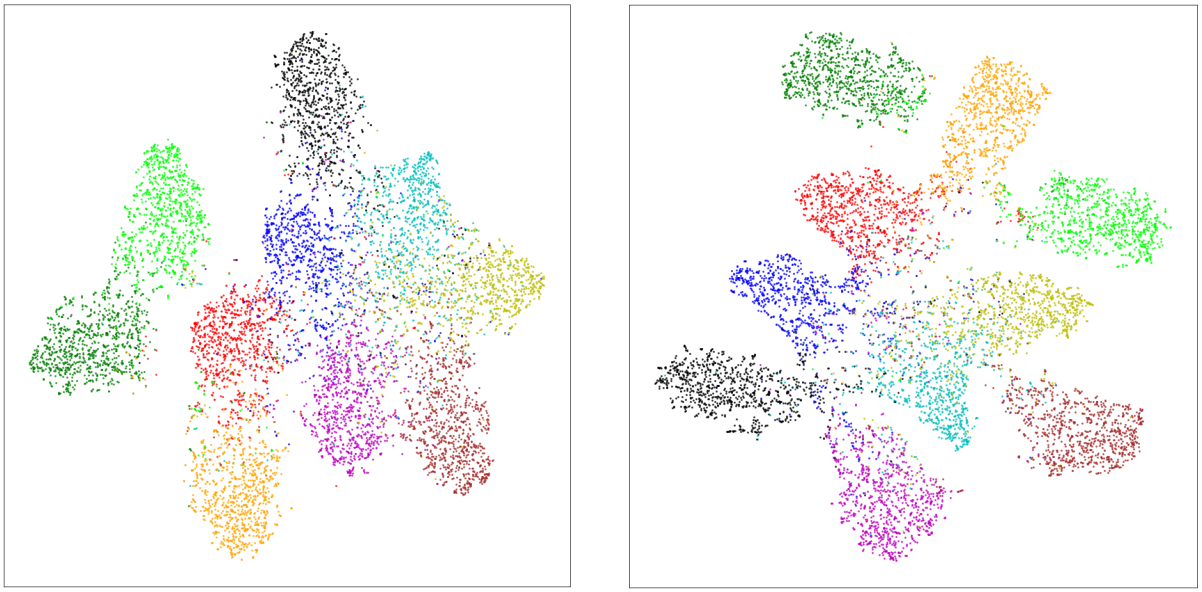
(a) MT, Training Data

(b) MT + LC, Training Data

Figure 3.6 t-SNE Visualization of CIFAR-10 training data features obtained by MT (left) and MT + LC (right). The models are trained on CIFAR-10 with 2,000 labeled samples. Unlabeled samples are in red and labeled samples are in blue. Best viewed in color in electronic form.

Algorithm 1 LC with MT training process

Require: x_i = training input sample
Require: y_i = sample class label
Require: $r_1(t)$ = consistency loss coefficient ramp-up function
Require: $r_2(t)$ = local clustering loss coefficient ramp-up function
Require: b_l = number of labeled samples in each batch
Require: b_u = number of unlabeled samples in each batch
Require: $g(\mathbf{x};\theta'_g)$ = feature extractor network of student model with parameters θ'_g
Require: $h(\mathbf{x};\theta'_h)$ = classification network of student model with parameters θ'_h
Require: $f(\mathbf{x};\theta')$ = student model with parameters θ'
Require: $f(\mathbf{x};\theta'')$ = teacher model with parameters θ''
Require: λ_1 = consistency loss coefficient
Require: λ_2 = local clustering loss coefficient
Require: ξ' = one random perturbation applied to x_i
Require: ξ'' = another random perturbation applied to x_i
for t in [1, *num_epochs*] **do**
  for $\{x_i\}_{i=1}^{b_l}$ and $\{x_i\}_{i=1}^{b_u}$ sampled from training data **do**
    Sample ξ'
    $z'_i = g(x_i, \xi'; \theta'_g)$
    for i in [1, b_l], j in [1, b_u] **do**
      Compute w_{ij} using Eq.(4)
    end for
    for m in [1, b_u], n in [1, b_u] **do**
      Compute w_{mn} using Eq.(4)
    end for
    $\hat{y}'_i = h(z'_i; \theta'_h)$
    Sample ξ''
    $\hat{y}''_i = f(x_i, \xi''; \theta'')$
    $\mathscr{L}_{ce} = -\frac{1}{|b_l|}\sum_{i=1}^{b_l} y_i log(\hat{y}'_i)$
    $\mathscr{L}_{cons} = \frac{1}{|b_l+b_u|}\sum_{i=1}^{b_l+b_u} ||\hat{y}'_i - \hat{y}''_i||^2$
    $\mathscr{L}_{lc} = \frac{1}{\sum_{i=1,j=1}^{b_l,b_u} \mathbb{1}_{[w_{ij}>0]}}\sum_{i=1,j=1}^{b_l,b_u} w_{ij}||z'_i - z'_j||^2 + \frac{1}{\sum_{m=1,n=1}^{b_u,b_u} \mathbb{1}_{[w_{mn}>0]}}\sum_{m=1,n=1}^{b_u,b_u} w_{mn}||z'_m - z'_n||^2$
    $\mathscr{L} = \mathscr{L}_{ce} + r_1(t)\lambda_1\mathscr{L}_{cons} + r_2(t)\lambda_2\mathscr{L}_{lc}$
    update θ' by SGD
    update θ'' using Eq.(3)
  end for
end for
Return θ', θ''



(a) MT, Test Data

(b) MT + LC, Test Data

Figure 3.7 t-SNE Visualization of CIFAR-10 test data features obtained by MT (left) and MT + LC (right). The models are trained on CIFAR-10 with 2,000 labeled samples. Each color denotes a ground truth class. Best viewed in color in electronic form.

CHAPTER

4

CONSISTENCY REGULARIZATION WITH GENERATIVE ADVERSARIAL NETWORKS FOR SEMI-SUPERVISED LEARNING

Generative Adversarial Networks (GANs) based semi-supervised learning (SSL) approaches are shown to improve classification performance by utilizing a large number of unlabeled samples in conjunction with limited labeled samples. However, their performance still lags behind the state-of-the-art non-GAN based SSL approaches. One main reason we identify is the lack of consistency in class probability predictions on the same image under local perturbations. This problem was addressed in the past in a generic setting using the label consistency regularization, which enforces the class probability predictions for an input image to be unchanged under various semantic-preserving perturbations.

In this work, we incorporate the consistency regularization in the vanilla semi-GAN to address this critical limitation. In particular, we present a new composite consistency regularization method which, in spirit, combines two well-known consistency-based techniques – Mean Teacher and Interpolation Consistency Training. We demonstrate the efficacy of our approach on two SSL image classification benchmark datasets, SVHN and CIFAR-10. Our experiments show that this new composite consistency regularization based semi-GAN significantly improves its performance and achieves new state-of-the-art performance among GAN-based SSL approaches.

4.1 Introduction

In recent years, we have witnessed great achievements of deep neural networks on supervised image classification tasks [Sim14b; He16a; Hua17]. However, these achievements are partially owing to training deep neural networks on large-scale well-annotated image classification datasets, e.g., ImageNet [Den09]. Obtaining such datasets with large amounts of labeled data is often prohibitive due to time, cost, expertise, and privacy restrictions. Semi-supervised learning (SSL) presents an alternative, where models are leveraged to learn representations from plentiful unlabeled data as well, and hence reduces the heavy dependence on labeled data.

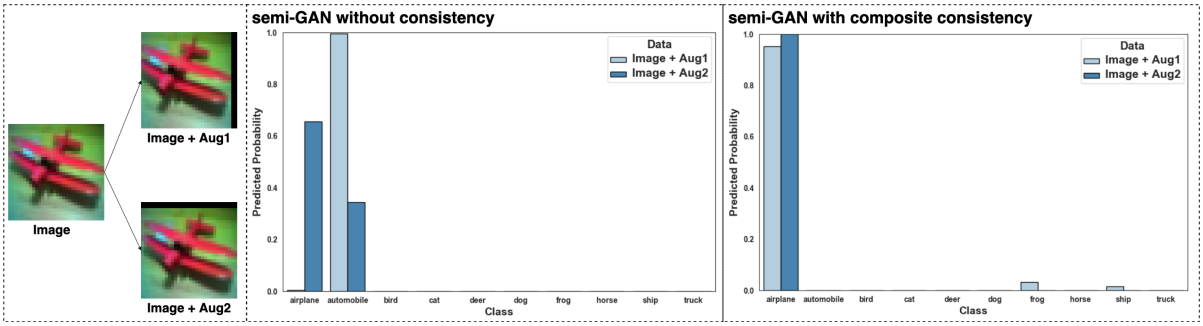


Figure 4.1 Research motivation. The example above shows that when an image is augmented with two different data augmentations (i.e. image shifting) and fed into a well-trained discriminator of semi-GAN, it suffers from inconsistent predictions issue, while this issue can be resolved if combined with our composite consistency. This example is conducted on CIFAR-10 dataset.

Deep generative models (DGMs) [Kin14b; Rez14; Goo14a] have emerged as an advanced framework for learning data representations in an unsupervised manner. In particular, Generative Adversarial Networks (GANs) [Goo14a] have demonstrated their capability for modeling data distributions and generating visually realistic images. GANs set up an adversarial game between a generator network and a discriminator network, where the generator is tasked to trick the discriminator with generated samples, whereas the discriminator is tasked to tell apart real and generated samples. Semi-GAN [Sal16] pioneers the extension of GANs to the SSL domain, where the discriminator employs a (K+1)-class objective with the extra class referring to the fake samples from the generator.

We first observe that semi-GAN suffers from inconsistent predictions by conducting experiments on the CIFAR-10 dataset, in which each unlabeled image is augmented with two different data augmentations and fed into a well-trained discriminator of semi-GAN. Figure 4.1 depicts such input images on which vanilla semi-GAN's discriminator is inconsistent, and our composite consistency is able to resolve. Although many approaches [Dai17; Qi18a; Dum16; Lec18] have been developed to improve the performance of semi-GAN, regularizing semi-GAN with consistency techniques has barely been explored in the literature. Consistency regularization specifies that the classifier should always make consistent predictions for an unlabeled data sample even under semantic-preserving

perturbations. It follows from the popular *smoothness assumption* [Cha09] in SSL: if two points in a high-density region of data manifold are close, then so should be the corresponding outputs. Based on this intuition, we hypothesize that the discriminator of semi-GAN should also be granted with this consistency property so that it would be resilient to semantic-preserving augmentations and focus more on the semantic differences between classes.

We propose to integrate consistency regularization into semi-GAN. Since both Mean Teacher (MT) [Tar17] and Interpolation Consistency Training (ICT) [Ver19] perform well among consistency-based approaches, we explore both of them in this work. Besides, we find that MT consistency and ICT consistency are complementary to each other, so we propose composite consistency by combining the two into one unified framework. In summary, we have made the following contributions:

- We propose integrating consistency regularization into the discriminator of semi-GAN so that the discriminator would make consistent predictions for data samples with local perturbations, improving the model's performance for SSL classification tasks.
- We propose a new consistency-based technique called composite consistency by combining the Mean Teacher and Interpolation Consistency Training consistency techniques, and empirically show that it produces the best results among these three consistency-based techniques.

4.2 Preliminaries

In a general SSL problem, we are given a small set of labeled samples $(\mathbf{x}_l, y_l)$ and a large set of unlabeled samples $\mathbf{x}_u$, where every $\mathbf{x} \in \mathbb{R}^d$ is an input data sample of d-dimension and $y \in \{1, 2, ..., K\}$ is one of K class labels. The objective of SSL is to learn a classifier $D(y|\mathbf{x};\theta) : \mathscr{X} \rightarrow \mathscr{Y}$, mapping from the input space $\mathscr{X}$ to the label space $\mathscr{Y}$, parameterized by θ. In deep SSL approaches, $D(y|\mathbf{x};\theta)$ is chosen to be represented by a deep neural network.

4.2.1 Review of semi-GAN

In a Generative Adversarial Network (GAN), an adversarial two-player game is set up between discriminator and generator networks. The objective of the generator $G(\mathbf{z};\delta)$ is to transform a random vector $\mathbf{z}$ into a fake sample that cannot be distinguished from real samples by the discriminator. The discriminator is a binary classifier tasked to judge whether a sample is real or fake. Salimans *et al*. [Sal16] pioneered the extension of GANs to SSL by proposing the first GAN-based SSL approach named as semi-GAN. In semi-GAN [Sal16], the discriminator is adjusted into a $(K+1)$-head classifier, where the first K are real classes originated from the dataset and the $(K+1)$-th class is the fake class referring to generated samples. The objective function for the discriminator is formulated as:

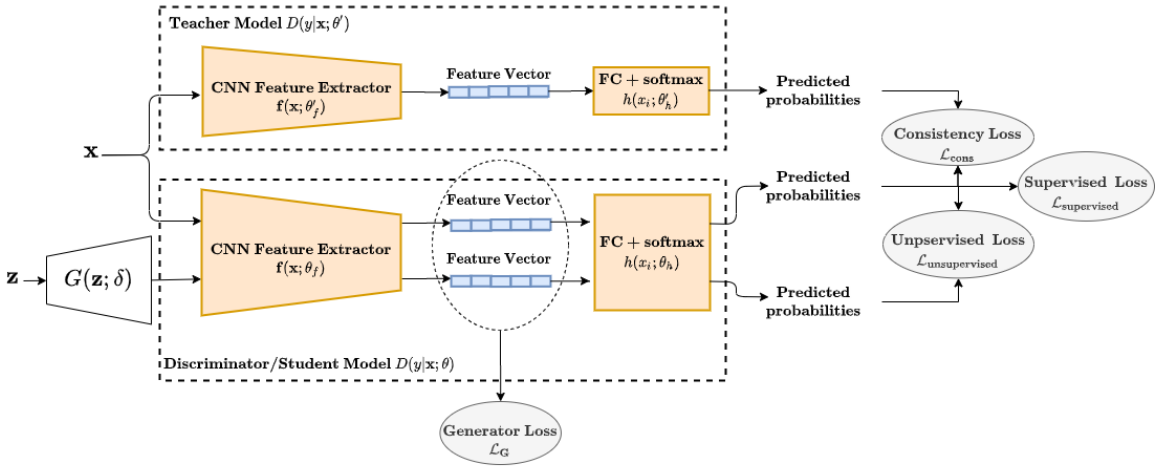


Figure 4.2 Semi-GAN with Consistency Regularization. The model that integrates consistency regularization into semi-GAN. The discriminator of the semi-GAN is also treated as the student model for the consistency regularization, and the consistency loss is enforced as the prediction difference between the student and teacher models for real data. "FC" represents a fully connected layer.

$$\begin{aligned}\mathscr{L}_D = &-\mathbb{E}_{p(\mathbf{x}_l,y_l)}[\log D(y_l|\mathbf{x}_l;\theta)] \\ &-\mathbb{E}_{p(\mathbf{z})}[\log D(y=K+1|G(\mathbf{z};\delta);\theta)] \\ &-\mathbb{E}_{p(\mathbf{x})}[\log(1-D(y=K+1|\mathbf{x};\theta))]\end{aligned} \tag{4.1}$$

The above objective function consists of three terms. The first term is the standard supervised loss $\mathscr{L}_{\text{supervised}}$ that maximizes the log-likelihood that a labeled data sample is classified correctly into its ground-truth class. The second and third terms constitute the unsupervised loss $\mathscr{L}_{\text{unsupervised}}$ that classifies real samples $\mathbf{x}$ as non-fake ($y < K+1$) and generated samples $G(\mathbf{z})$ as fake ($y = K+1$).

On the other hand, the authors of [Sal16] propose a feature matching loss as the objective function for the generator, where the functionality is to minimize the discrepancy of the first moment between real and generated data distributions in feature space, represented as:

$$\mathscr{L}_G = ||\mathbb{E}_{p(\mathbf{x})}\mathbf{f}(\mathbf{x};\theta_f) - \mathbb{E}_{p(\mathbf{z})}\mathbf{f}(G(\mathbf{z};\delta);\theta_f)||_2^2 \tag{4.2}$$

where $\mathbf{f}$ is an intermediate layer from the discriminator D, and θ_f is a subset of θ, including all the parameters up to that intermediate layer of the discriminator. In practice, feature matching loss has exhibited excellent performance for SSL tasks and has been broadly employed by GAN-based SSL approaches [Sal16; Dai17; Qi18a].

4.2.2 Review of consistency regularization

Consistency regularization has been widely used in semi-supervised or unsupervised learning approaches [Val15; Lai16; Tar17; Miy18]. The intuition behind it is that the classifier should always make consistent predictions that are invariant to small perturbations added to either inputs or intermediate representations for both labeled and unlabeled data. Typical perturbations are represented in the form of input augmentations, dropout regularization [Sri14], or adversarial noise [Goo14b]. To enforce consistency, the Γ-model [Ras15] evaluates each data input with and without perturbation, and minimizes the discrepancy between the two predictions. In this case, the classifier can be considered as assuming two parallel roles, one as a student model for regular learning and the other as a teacher model for generating learning targets. Since there are no ground truth labels for unlabeled data, the learning targets generated by the teacher model can be incorrect, and some recent works [Lai16; Tar17] have been focusing on improving the quality of the teacher model to generate better learning targets for the student model.

More formally, the consistency loss term is defined as the divergence of the predictions between the student model and the teacher model, formulated as

$$\mathscr{L}_{cons} = \mathbb{E}_{p(\mathbf{x})} d[D(y|\mathbf{x};\theta,\xi), D(y|\mathbf{x};\theta',\xi')] \tag{4.3}$$

where $D(y|\mathbf{x};\theta,\xi)$ is the student model with parameters θ and random perturbation ξ, and $D(y|\mathbf{x};\theta',\xi')$ is the teacher model with parameters θ' and random perturbation ξ'. $d[\cdot,\cdot]$ measures the divergence between the two predictions, usually chosen to be Euclidean distance or Kullback-Leibler divergence.

4.3 Methodology

To address the prediction inconsistency of semi-GAN [Sal16], we integrate consistency regularization into semi-GAN, encouraging it to produce consistent predictions under small perturbations. In other words, the consistency regularization serves as an additional auxiliary loss term to the discriminator. Hence the new objective function for the discriminator is formulated as

$$\begin{aligned}\mathscr{L}_D = &-\mathbb{E}_{p(\mathbf{x}_l,y_l)}[\log D(y_l|\mathbf{x}_l;\theta,\xi)] \\ &-\mathbb{E}_{p(\mathbf{z})}[\log D(y=K+1|G(\mathbf{z};\delta);\theta)] \\ &-\mathbb{E}_{p(\mathbf{x})}[\log(1-D(y=K+1|\mathbf{x};\theta,\xi))] \\ &+\lambda_{cons}\mathbb{E}_{p(\mathbf{x})} d[D(y|\mathbf{x};\theta,\xi), D(y|\mathbf{x};\theta',\xi')]\end{aligned} \tag{4.4}$$

where the coefficient λ_{cons} is a hyper-parameter controlling the importance of the consistency loss. Figure 4.2 displays the new model architecture. As shown in the figure, the discriminator $D(y|\mathbf{x};\theta)$ in semi-GAN [Sal16] is also treated as the student model for the consistency regularization and the consistency loss is enforced as the prediction difference between the student and teacher

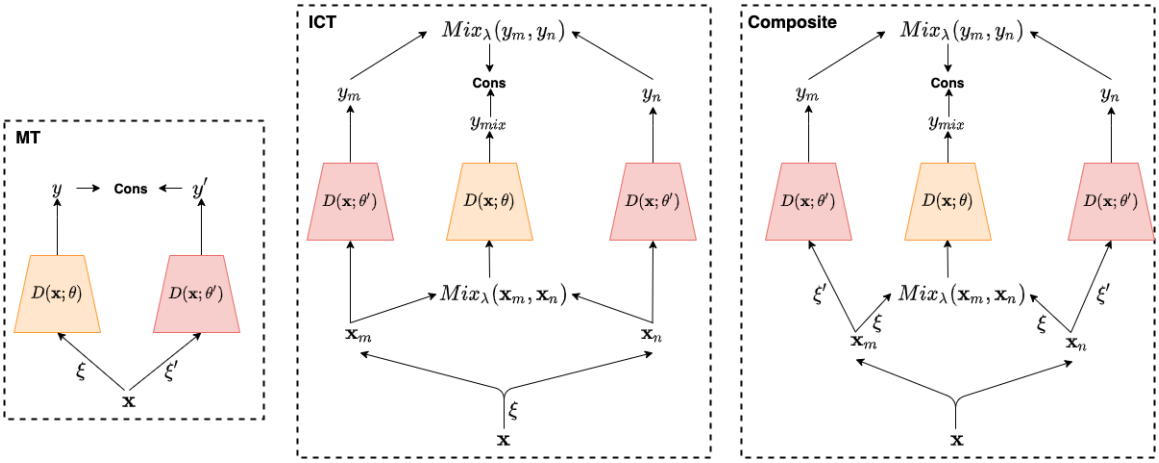


Figure 4.3 Consistency regularization illustration. Three types of consistency techniques: MT (left), ICT (middle) and Composite (right). In the figure, $\mathbf{x}_m$ and $\mathbf{x}_n$ are two shuffled versions of $\mathbf{x}$, while ξ and ξ' represent two random data augmentations.

models for real data. See Section 4.3.1 for more details on how the teacher model $D(y|\mathbf{x};\theta')$ is generated from the student model.

More specifically, we integrate two well-known consistency-based techniques called Mean Teacher (MT) [Tar17] and Interpolation Consistency Training (ICT) [Ver19] into semi-GAN. Furthermore, we propose the combination of both of these techniques as a new composite consistency-based technique and show that it enhances the robustness of the discriminator by taking advantage of both techniques. Figure 4.3 illustrates the ideas of these three consistency-based techniques.

4.3.1 MT Consistency

Rooted from Γ-model [Ras15], Mean Teacher [Tar17] naturally imposes consistency by adding random perturbations to the input of the model. As shown in Figure 4.3 (left), the input data are applied with certain types of augmentation (e.g., image shifting, flipping, etc.) randomly twice. After that, the two augmented inputs are fed into the student model and teacher model separately, and the consistency is achieved by minimizing the prediction difference between the student model and teacher model. One key aspect of MT is that it improves the quality of the learning targets from the teacher model by forming a better teacher model. Namely, the parameters θ' of the teacher model are maintained as an exponential moving average (EMA) of the parameters θ of the student model during training, formulated as:

$$\theta'_t = k\theta'_{t-1} + (1-k)\theta'_t \tag{4.5}$$

where t indexes the training step and the hyper-parameter k is the EMA decay coefficient. By aggregating information from the student model in an EMA manner at training time, a better teacher model can generate more stable predictions which serve as higher quality learning targets to

guide the learning of the student model. This way of generating the teacher model is also employed in ICT [Ver19] and eventually in our composite consistency-based technique.

4.3.2 ICT Consistency

Interpolation Consistency Training [Ver19] proposes a new type of consistency that encourages consistent predictions at interpolations of two data samples. The interpolation is the linear interpolation implemented using the MixUp operation [Zha18]. Given any two vectors u and v, the MixUp operation is defined as

$$Mix_\lambda(u,v) = \lambda \cdot u + (1-\lambda) \cdot v \tag{4.6}$$

where $\lambda \in [0,1]$ is a parameter randomly sampled from Beta distribution denoted as $\lambda \sim \text{Beta}(\alpha,\alpha)$, and α is a hyper-parameter controlling the sampling process. With the MixUp operation, given two randomly shuffled versions of the dataset $\mathbf{x}$ after data augmentation ξ represented as $\mathbf{x}_m$ and $\mathbf{x}_n$, the ICT consistency is computed as

$$\begin{aligned} \mathscr{L}_{ict\_cons} = \mathbb{E}_{p(\mathbf{x}_m,\mathbf{x}_n|\mathbf{x},\xi)} d[D(y_{mix}|Mix_\lambda(\mathbf{x}_m,\mathbf{x}_n);\theta), \\ Mix_\lambda(D(y_m|\mathbf{x}_m;\theta'), D(y_n|\mathbf{x}_n;\theta'))] \end{aligned} \tag{4.7}$$

where it encourages the predictions from the student model at interpolations of any two data samples (denoted as $D(y_{mix}|Mix_\lambda(\mathbf{x}_m,\mathbf{x}_n);\theta)$) to be consistent with the interpolations of the predictions from the teacher model on the two samples (denoted as $Mix_\lambda(D(y_m|\mathbf{x}_m;\theta'), D(y_n|\mathbf{x}_n;\theta'))$), shown in Figure 4.3 (middle).

4.3.3 Composite Consistency

Even though MT chooses to perturb data samples by certain types of data augmentations, while ICT chooses to perturb data samples from the perspective of data interpolations, they have some common characters. If we set $\lambda = 1$ in ICT, the interpolated sample $Mix_\lambda(\mathbf{x}_m,\mathbf{x}_n)$ is reduced to $\mathbf{x}_m$, hence the ICT consistency loss term is reduced to

$$\mathscr{L}_{ict\_cons} = \mathbb{E}_{p(\mathbf{x}_m,\mathbf{x}_n|\mathbf{x},\xi)} d[D(y_{mix}|\mathbf{x}_m;\theta), D(y_m|\mathbf{x}_m;\theta')] \tag{4.8}$$

This loss term is the same as MT consistency loss (seen Eq.4.3) except that the same data augmentation ξ is applied to the inputs of both student and teacher models. Accordingly, if two different data augmentations are applied to the inputs of the student and teacher models separately as MT, we can make ICT also robust to data augmentation perturbations, as shown in Figure 4.3 (right). In other words, we can combine these two consistency techniques so that the model would be robust to both data augmentation perturbations and data interpolation perturbations. We name

the combination of these two consistency techniques as composite consistency, and formulate the corresponding loss $\mathscr{L}_{comp\_cons}$ term as

$$\begin{aligned}\mathscr{L}_{comp\_cons} = \mathbb{E}_{p(\mathbf{x}_m,\mathbf{x}_n|\mathbf{x})} d[D(y_{mix}|Mix_\lambda(\mathbf{x}_m,\mathbf{x}_n);\theta,\xi),\\ Mix_\lambda(D(y_m|\mathbf{x}_m;\theta',\xi'),D(y_n|\mathbf{x}_n;\theta',\xi'))]\end{aligned} \tag{4.9}$$

4.4 Experiments

In this section, we present comprehensive experiments to evaluate the effectiveness of our proposed methods. The purpose of these experiments is to demonstrate the improvements achievable by incorporating consistency regularization, especially our proposed composite consistency, into semi-GAN. More specifically, we evaluate our approach with varying amounts of labeled samples on two benchmark SSL datasets and conduct ablation studies to analyze different aspects of our approach.

4.4.1 Datasets

Following common practice in GAN-based SSL approaches [Sal16; Dum16; Cho17; Qi18a; Dum16], we quantitatively evaluate our methods on two SSL benchmark datasets: SVHN and CIFAR-10. The SVHN dataset consists of 73,257 training images and 26,032 test images. Each image has a size of 32 $\times$ 32 centered with a street view house number (a digit from 0 to 9). So there are total 10 classes in the dataset. The CIFAR-10 dataset consists of 50,000 training images and 10,000 test images. Similarly, the CIFAR-10 dataset also has images of size 32 $\times$ 32 and 10 classes. The 10 classes represent some common objects in daily life, such as airplanes and birds.

4.4.2 Implementation Details

We utilize the same discriminator and generator network architectures as used in CT-GAN [Wei18]. See Appendix A.1 for more details of the network architectures. When training models on SVHN training data, we augment the images with random translation, where the image is randomly translated in both horizontal and vertical directions with a maximum of 2 pixels. For the CIFAR-10 dataset, we apply both random translation (in the same way as SVHN) and horizontal flips. For both datasets, we train the models with a batch size of 128 labeled samples and 128 unlabeled samples. We run the experiments with Adam Optimizer [Kin14a](set $\beta_1 = 0.5$, $\beta_2 = 0.999$), where the learning rate is set to be 3e-4 for the first 400 epochs and linearly decayed to 0 in another 200 epochs. Following the same training schema as in MT and ICT, we also employ the ramp-up phase for the consistency loss, where we increase consistency loss weight λ_{cons} from 0 to its final value in the first 200 epochs. We adopt the same sigmoid-shaped function $e^{-5(1-\gamma)^2}$ [Tar17] as our ramp-up function, where $\gamma \in [0,1]$. We set the EMA decay coefficient k to 0.99 and the parameter α in Beta(α,α) distribution to 0.1 through all our experiments.

4.4.3 Ablation study

In this section, we aim at analyzing our proposed method from different aspects empirically.

Effect of consistency loss weight λ_{cons}: The most important hyper-parameter influencing model performance is the consistency loss weight λ_{cons}. So we conduct an experiment using semi-GAN with composite consistency on CIFAR-10 with 4,000 labeled images. In this experiment, we train our model with a wide range of λ_{cons} values, and the results are shown in Figure 4.4. Note that the model with $\lambda_{cons} = 0$ is equivalent to a vanilla semi-GAN model. From the figure, we see that there is a sharp decrease in error rate as λ_{cons} increases from 0 to 10, implying composite consistency starts taking more effect, then it arrives into a relatively steady state, and the error rate slightly increases upon continuing to increase λ_{cons}, indicating λ_{cons} could span a good range of values and may degrade model performance if set too large.

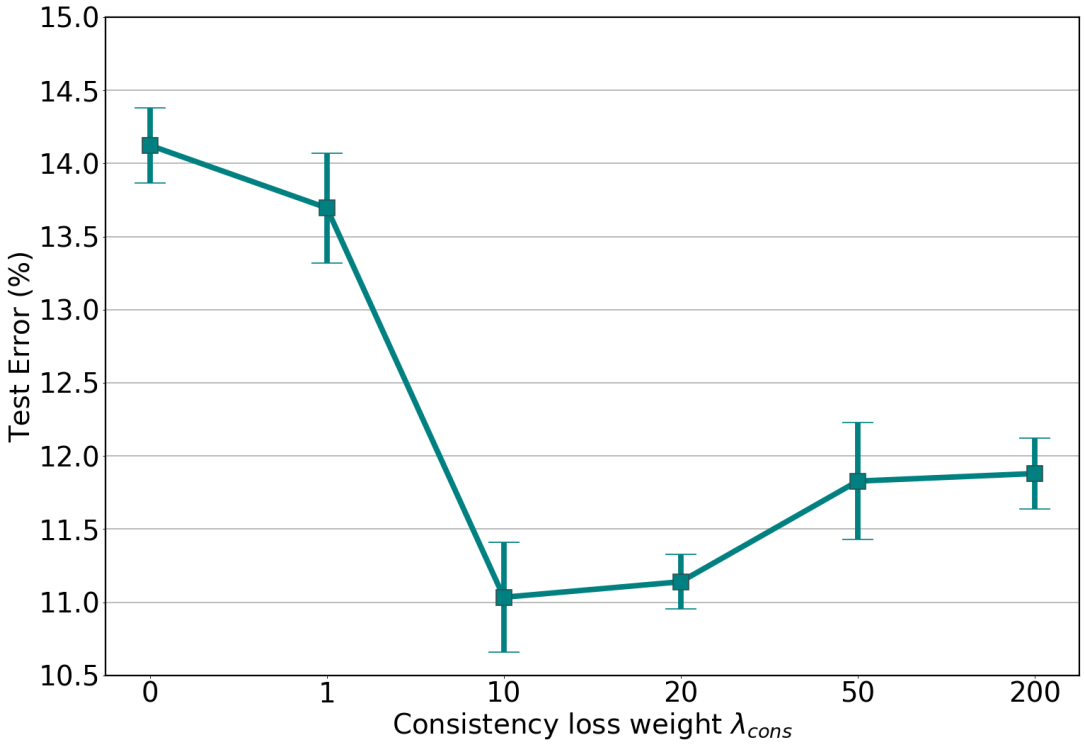


Figure 4.4 Test errors of semi-GAN with composite consistency on CIFAR-10 with 4,000 labeled samples over 5 runs.

Effect of different consistency techniques: As we present three consistency-based techniques in our methodology section, it is necessary to quantify the benefits of introducing these consistency-based techniques into semi-GAN. So we compare them empirically on CIFAR-10 with 1,000 and 4,000 labeled images, respectively. Table 4.1 shows the comparison results. From the table, we can see

that incorporating consistency regularization into semi-GAN consistently improves its performance, and semi-GAN with composite consistency yields better results than with MT consistency or ICT consistency.

Table 4.1 An ablation study on comparing the effects of different consistency techniques with semi-GAN. The experiments are conducted on CIFAR-10 with 1,000 and 4,000 labeled samples over 5 runs and error rate percentage is reported as the evaluation criteria. "CC" is short for our proposed composite consistency.

Models	Error rate (%)	
	CIFAR-10 $n_l = 1,000$	CIFAR-10 $n_l = 4,000$
semi-GAN	17.27 ± 0.83	14.12 ± 0.29
semi-GAN + MT	15.28 ± 1.03	12.08 ± 0.27
semi-GAN + ICT	15.11 ± 0.86	11.66 ± 0.50
semi-GAN + CC	14.36 ± 0.35	11.03 ± 0.42

On the other hand, we have also conduct experiments on MT and ICT alone to demonstrate that semi-GAN with consistency regularization would outperform consistency-based approaches themselves. Under the same experimental settings as we describe in Section 4.4.2, we exclude semi-GAN from the framework and evaluate the performance of either MT or ICT alone on CIFAR-10 with 4,000 labeled images. The error rate of MT is 18.57% ± 0.43, whereas combining with semi-GAN yields a lower error rate of 12.08% ± 0.27. Also, the error rate of ICT is 18.16% ± 1.25, whereas combining with semi-GAN yields a lower error rate of 11.66% ± 0.50. Thus, it proves that semi-GAN and consistency regularization are complementary and could achieve better performance when combined. Meanwhile, we notice some performance differences between our MT and ICT re-implementations with the ones reported in original MT and ICT papers. We hypothesize that the performance difference is primarily caused by the network architecture difference. So we have experimented with our re-implementation of MT with the network architecture used in MT paper and obtained similar performance as the one reported in MT paper.

Effect of imposing consistency at different places of the discriminator: Although consistency has always been imposed at output space for consistent predictions in consistency-based approaches [Lai16; Tar17; Par18; Miy18], it could also be imposed at feature space to help the model learn high-level features invariant to diverse perturbations. Therefore, in this study, we aim at exploring the effect of imposing consistency at different places of the discriminator. More specifically, we choose to impose consistency with three different settings: 1) on the output layer of the discriminator for prediction consistency; 2) on the intermediate layer of the discriminator (the layer right before FC + softmax as seen in Figure 4.2) for feature consistency; 3) on both the output layer

and the intermediate layer of the discriminator for prediction and feature consistencies. When imposing feature consistency, we perform hyper-parameter search for its consistency weight over the values in {0.01, 0.1, 1.0, 10, 100} and report the results with the optimal hyper-parameter value. We conduct experiments on CIFAR-10 dataset with 1,000 and 4,000 labeled images, respectively. The results are shown in Table 4.2. From the table, we can observe that incorporating consistency in both output space and feature space yields the best performance among the three, implying both feature consistency and prediction consistency can benefit the classification task. Thus, we would impose consistency on both the output layer and the intermediate layer of the discriminator in our final evaluation on benchmark datasets.

Table 4.2 An ablation study comparing the effects of imposing consistency at different places of the discriminator. The experiments are conducted using semi-GAN with composite consistency. We evaluate the methods on CIFAR-10 dataset with 1,000 and 4,000 labeled images over 5 runs.

Consistency type	Error rate (%)	
	CIFAR-10 $n_l = 1,000$	CIFAR-10 $n_l = 4,000$
Prediction	14.36 ± 0.35	11.03 ± 0.42
Feature	16.19 ± 0.80	13.19 ± 0.50
Prediction & Feature	14.14 ± 0.23	10.69 ± 0.49

Table 4.3 Error rate percentage comparison with GAN-based approaches on CIFAR-10 over 5 runs. "*" indicates our re-implementation of the method. "CC" is short for our proposed composite consistency.

Models	CIFAR-10		
	$n_l = 1,000$	$n_l = 2,000$	$n_l = 4,000$
CatGAN [Spr16]	-	-	19.58 ± 0.46
semi-GAN [Sal16]	21.83 ± 2.01	19.61 ± 2.09	18.63 ± 2.32
Bad GAN [Dai17]	-	-	14.41 ± 0.30
CLS-GAN [Qi19]	-	-	17.30 ± 0.50
Triple-GAN [Cho17]	-	-	16.99 ± 0.36
Local GAN [Qi18a]	17.44 ± 0.25	-	14.23 ± 0.27
ALI [Dum16]	19.98 ± 0.89	19.09 ± 0.44	17.99 ± 1.62
Manifold Regularization [Lec18]	16.37 ± 0.42	15.25 ± 0.35	14.34 ± 0.17
semi-GAN*	17.27 ± 0.83	15.36 ± 0.74	14.12 ± 0.29
semi-GAN + CC **(ours)**	$\mathbf{14.14 \pm 0.23}$	$\mathbf{12.11 \pm 0.46}$	$\mathbf{10.69 \pm 0.49}$

Table 4.4 Error rate percentage comparison with GAN-based approaches on SVHN over 5 runs. "*" indicates our re-implementation of the method. "CC" is short for our proposed composite consistency. Note that CatGAN [Spr16] did not conduct experiments on SVHN dataset.

Models	SVHN	
	$n_l = 500$	$n_l = 1{,}000$
semi-GAN [Sal16]	18.44 ± 4.80	8.11 ± 1.30
Bad GAN [Dai17]	-	7.42 ± 0.65
CLS-GAN [Qi19]	-	5.98 ± 0.27
Triple-GAN [Cho17]	-	5.77 ± 0.17
Local GAN [Qi18a]	5.48 ± 0.29	4.73 ± 0.29
ALI [Dum16]	-	7.41 ± 0.65
Manifold Regularization [Lec18]	5.67 ± 0.11	4.63 ± 0.11
semi-GAN*	6.66 ± 0.58	5.36 ± 0.31
semi-GAN + CC **(ours)**	$\mathbf{3.79 \pm 0.23}$	$\mathbf{3.64 \pm 0.08}$

4.4.4 Results

Following the standard evaluation schema as per other GAN-based approaches [Sal16; Dum16; Cho17; Qi18a; Dum16], we train the models on SVHN training data with 500 and 1,000 randomly labeled images respectively and evaluate model classification performance on the corresponding test data. For CIFAR-10, we train the models on CIFAR-10 training data with 1,000, 2,000, and 4,000 randomly labeled images and then evaluate them on CIFAR-10 test data. The results are provided in Table 4.3 and Table 4.4. For both datasets, semi-GAN with composite consistency outperforms semi-GAN by a large margin and sets new state-of-the-art among GAN-based SSL approaches.

It is worth mentioning that we cannot form a direct comparison between our approach with non-GAN-based SSL approaches due to network architecture differences. As a sanity check, we have experimented with the CNN-13 architecture adopted in recent consistency-based SSL approaches [Lai16; Tar17; Miy18; Ver19] as our discriminator, but encountered mode collapse issue [Goo14a] during training in multiple trials. We suspect it is because the discriminator too easily overpowers the generator in this setting.

4.4.5 Visualization

Furthermore, we visualize the learned feature embeddings of semi-GAN model and semi-GAN + CC on both CIFAR-10 test data and SVHN test data with t-SNE [Maa08], as shown in Figure 4.5. We train models on CIFAR-10 with 4,000 labeled images and SVHN with 1,000 labeled images respectively, and project the feature embeddings ($\mathbf{f}(\mathbf{x}) \in \mathbb{R}^{128}$) into 2-D space using t-SNE, where the feature embeddings are obtained from the layer right before final FC + softmax layer. From the figure, we can easily observe that the feature embeddings of our semi-GAN + CC models are more concentrated within each class and more separable between classes on both CIFAR-10 and SVHN

test data, while they are more mixed in semi-GAN models. This visualization further validates that composite consistency helps semi-GAN with the classification task.

4.5 Related Work

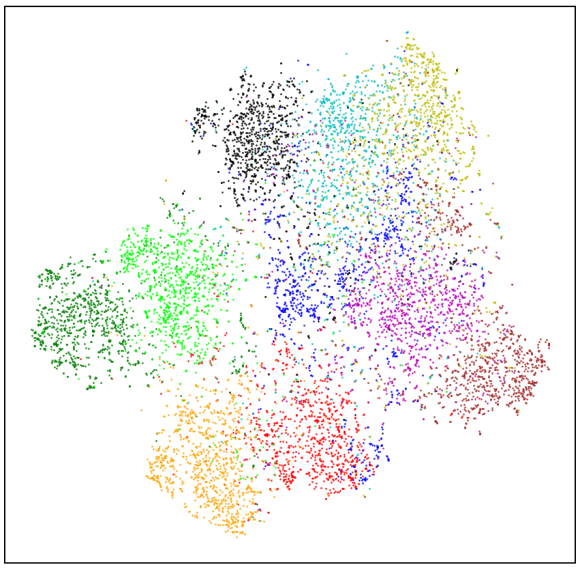
A large variety of approaches have been developed lately in deep SSL literature. In this section, we only focus on reviewing the most relevant GAN-based SSL approaches and provide a brief review of other categories of deep SSL approaches.

GAN-based SSL approaches: Following semi-GAN [Sal16] work, Qi *et al*. [Qi18a] propose Local-GAN to improve the robustness of the discriminator to locally noised samples, which are generated by a local generator at the neighborhood of real samples on real data manifold. Likewise, the authors in [Dai17] have proposed a complement generator to address the drawbacks in the feature matching objective of semi-GAN. They show both theoretically and empirically that a preferred generator should generate complement samples in low-density regions of the feature space, so that real samples are pushed to separable high-density regions and hence the discriminator can learn correct class decision boundaries. Taking perspective from information theory, CatGAN [Spr16] adapts the real/fake adversary formulation of the standard GAN to the adversary on the level of confidence in class predictions, where the discriminator is encouraged to predict real samples into one of the K classes with high confidence and to predict fake samples into all of the K classes with low confidence, and the generator is designated to perform in the opposite. Similarly, the CLS-GAN [Qi19] designs a new loss function for the discriminator with the assumption that the prediction error of real samples should always be smaller than that of fake ones by a desired margin, and further regularizes this loss with Lipschitz continuity on the density of real samples. Apart from them, Li *et al*. [Cho17] design a Triple-GAN consisting of three networks, including a discriminator, a classifier, and a generator. Here, the discriminator is responsible for distinguishing real image-label pairs from fake ones, which are generated by either the classifier or the generator using conditional generation.

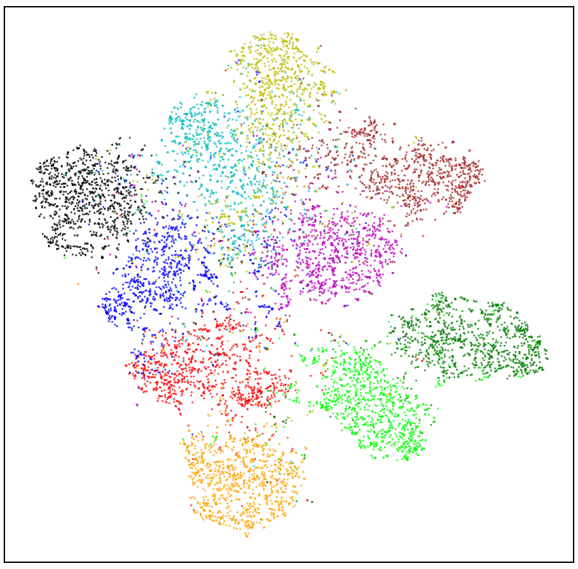
Other deep SSL categories: Besides GAN-based approaches, Variational Auto-Encoders (VAEs) [Kin14b; Rez14] have also been explored at early dates in deep generative models (DGMs) domain. VAE-based SSL approaches [Kin14c; Rez15] treat class label as an additional latent variable and learn data distribution by optimizing the lower bound of data likelihood using a stochastic variational inference mechanism. Aside from DGMs, graph-based approaches [Atw16; Kip17] have also been developed with deep neural networks, which smooth the label information on a pre-constructed similarity graph using some variants of label propagation mechanism [Ben06]. Differing from graph-based approaches, deep clustering approaches [Hsu15; Hae17; Kam18] build the graph directly in feature space instead of obtaining a pre-constructed graph from input space and perform clustering on the graph guided by partial labeled information. Furthermore, some recent advances [Wan19a; Ber19b] focus on the idea of distribution alignment, attempting to reduce the empirical distribution mismatch between labeled and unlabeled data caused by sampling bias.

4.6 Conclusions

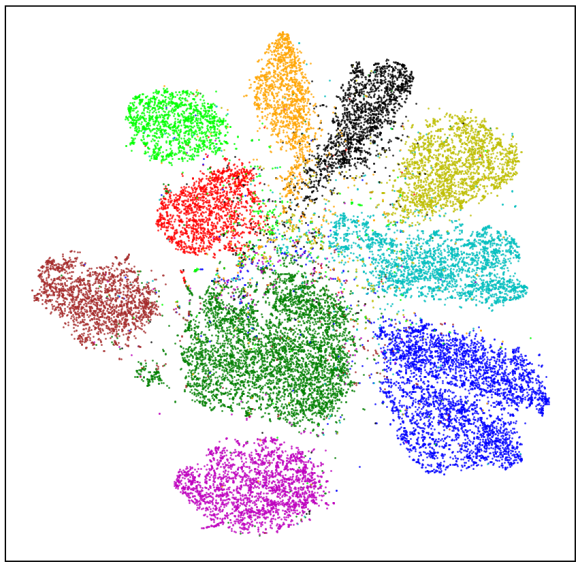
In this work, we propose to integrate consistency regularization into semi-GAN for fixing its prediction inconsistency issue. In particular, we present a simple but effective composite consistency into semi-GAN. It takes advantages of both MT consistency and ICT consistency, and can help the model resilient to both data augmentation perturbations and data interpolation perturbations. Empirically, we evaluate the effectiveness of semi-GAN with composite consistency on two benchmark datasets SVHN and CIFAR-10, and yield lowest error rates among GAN-based SSL approaches.



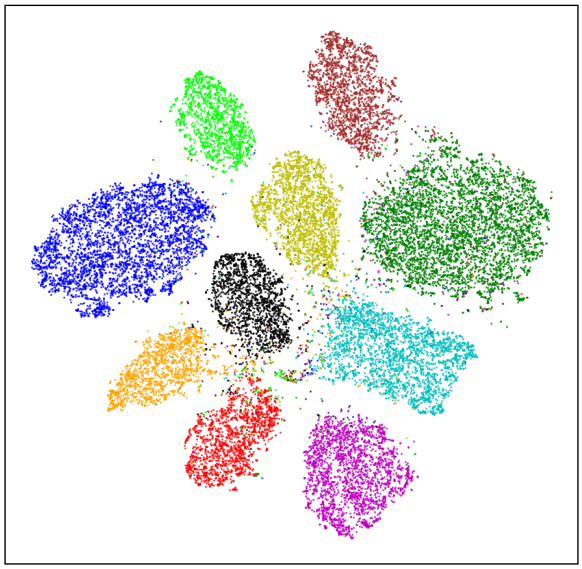
(a) CIFAR-10,semi-GAN



(b) CIFAR-10,semi-GAN+CC



(c) SVHN,semi-GAN



(d) SVHN,semi-GAN+CC

Figure 4.5 (a, b) are feature embeddings (models trained on CIFAR-10 with 4,000 labeled images) of CIFAR-10 test data visualized by t-SNE. And (c, d) are feature embeddings (models trained on SVHN with 1,000 labeled images) of SVHN test data visualized by t-SNE. Each color denotes a ground truth class. “CC” is short for our proposed composite consistency. Best viewed in color in electronic form.

CHAPTER

5

CONCLUSIONS

In this dissertation, we have made several research contributions towards deep semi-supervised video action recognition.

First, we developed a novel two-branch neural network architecture for the action recognition task, which serves as a unified framework that can jointly model appearance features, short-term motion features, long-term trajectory features, and object-interaction features. For capturing both long-term trajectory features and object interaction features, we developed a new variant of LSTM, namely Relational LSTM, to address the challenges of relation reasoning across space and time between objects. Even though we focused on the action recognition task in this dissertation, our Relation LSTM can be inserted into various network architectures for other tasks. Besides, we only explored the application of our Relational LSTM in two-stream ConvNets. In the future, we plan to explore the possibility of applying it on 3D ConvNets.

As noted earlier, generating ground-truth data for video action recognition is a costly and time-consuming process. To address this challenge, we developed two new deep semi-supervised algorithms. In the first approach, we developed a novel local clustering method to address the confirmation bias issue in the existing Mean Teacher method. This approach considers the neighboring data samples in feature space and pulls unlabeled data samples towards high-density regions with the assistance of neighboring samples. Although we only focused on applying our local clustering method on the Mean Teacher method to semi-supervised classification in this dissertation, the local clustering loss can be treated as a general regularization technique that might provide an interesting foundation for existing methods and enable new applications.

Finally, we developed a novel composite consistency regularizer to address the prediction inconsistency issue in semi-GAN framework. As the composite consistency with semi-GAN is shown

to be effective on real images, we propose to study the effect of enforcing composite consistency on the images produced by the generator. Additionally, while we adopt standard data augmentations (e.g., image shifting and flipping) to input images in this method, we are also interested in replacing standard data augmentations with stronger forms of augmentation (i.e., AutoAugment [Cub19], RandAugment [Cub20]) which may further improve model's classification performance.

BIBLIOGRAPHY

[Ath18] Athiwaratkun, B. et al. “There are many consistent explanations of unlabeled data: Why you should average” (2018).

[Atw16] Atwood, J. & Towsley, D. “Diffusion-convolutional neural networks”. *Advances in neural information processing systems*. 2016, pp. 1993–2001.

[Bac11] Baccouche, M. et al. “Sequential deep learning for human action recognition”. *International Workshop on Human Behavior Understanding*. Springer. 2011, pp. 29–39.

[Bar12] Barriuso, A. & Torralba, A. “Notes on image annotation”. *arXiv preprint arXiv:1210.3448* (2012).

[Ben06] Bengio, Y. et al. “11 label propagation and quadratic criterion” (2006).

[Ber19a] Berthelot, D. et al. “Mixmatch: A holistic approach to semi-supervised learning”. *arXiv preprint arXiv:1905.02249* (2019).

[Ber19b] Berthelot, D. et al. “ReMixMatch: Semi-Supervised Learning with Distribution Alignment and Augmentation Anchoring”. *arXiv preprint arXiv:1911.09785* (2019).

[Bil18] Bilen, H et al. “Action recognition with dynamic image networks”. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **40**.12 (2018), pp. 2799–2813.

[Bua05] Buades, A. et al. “A non-local algorithm for image denoising”. *2005 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Vol. 2. IEEE. 2005, pp. 60–65.

[Car17] Carreira, J. & Zisserman, A. “Quo vadis, action recognition? a new model and the kinetics dataset”. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. 2017, pp. 4724–4733.

[Cha03] Chapelle, O. et al. “Cluster kernels for semi-supervised learning”. *Advances in neural information processing systems*. 2003, pp. 601–608.

[Cha09] Chapelle, O. et al. “Semi-supervised learning (chapelle, o. et al., eds.; 2006)[book reviews]”. *IEEE Transactions on Neural Networks* **20**.3 (2009), pp. 542–542.

[Che17] Cherian, A. et al. “Generalized Rank Pooling for Activity Recognition”. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. 2017, pp. 1581–1590.

[Cho17] Chongxuan, L. et al. “Triple generative adversarial nets”. *Advances in neural information processing systems*. 2017, pp. 4088–4098.

[Cub19] Cubuk, E. D. et al. “Autoaugment: Learning augmentation strategies from data”. *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2019, pp. 113–123.

[Cub20] Cubuk, E. D. et al. “Randaugment: Practical automated data augmentation with a reduced search space”. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*. 2020, pp. 702–703.

[Dai17] Dai, Z. et al. “Good semi-supervised learning that requires a bad gan”. *Advances in neural information processing systems*. 2017, pp. 6510–6520.

[Den09] Deng, J. et al. “Imagenet: A large-scale hierarchical image database”. *2009 IEEE conference on computer vision and pattern recognition*. Ieee. 2009, pp. 248–255.

[Dib17] Diba, A. et al. “Deep Temporal Linear Encoding Networks”. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. 2017, pp. 1541–1550.

[Dum16] Dumoulin, V. et al. “Adversarially learned inference”. *arXiv preprint arXiv:1606.00704* (2016).

[Fei16] Feichtenhofer, C. et al. “Spatiotemporal residual networks for video action recognition”. *Advances in Neural Information Processing Systems (NIPS)*. 2016, pp. 3468–3476.

[Fei17] Feichtenhofer, C. et al. “Spatiotemporal multiplier networks for video action recognition”. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. 2017, pp. 7445–7454.

[Glo10] Glorot, X. & Bengio, Y. “Understanding the difficulty of training deep feedforward neural networks”. *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. 2010, pp. 249–256.

[Goo14a] Goodfellow, I. et al. “Generative adversarial nets”. *Advances in neural information processing systems*. 2014, pp. 2672–2680.

[Goo14b] Goodfellow, I. J. et al. “Explaining and harnessing adversarial examples”. *arXiv preprint arXiv:1412.6572* (2014).

[Hae17] Haeusser, P. et al. “Learning by Association–A Versatile Semi-Supervised Training Method for Neural Networks”. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 89–98.

[He16a] He, K. et al. “Deep residual learning for image recognition”. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 770–778.

[He16b] He, K. et al. “Identity mappings in deep residual networks”. *European Conference on Computer Vision (ECCV)*. Springer. 2016, pp. 630–645.

[He17] He, K. et al. “Mask r-cnn”. *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 2961–2969.

[Hsu15] Hsu, Y.-C. & Kira, Z. “Neural network-based clustering using pairwise constraints”. *arXiv preprint arXiv:1511.06321* (2015).

[Hu18] Hu, H. et al. “Relation networks for object detection”. *Computer Vision and Pattern Recognition (CVPR)*. Vol. 2. 3. 2018.

[Hua17] Huang, G. et al. “Densely connected convolutional networks”. *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 4700–4708.

[Hus19] Hussein, N. et al. “Timeception for Complex Action Recognition”. *2019 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 254–263.

[Iof15] Ioffe, S. & Szegedy, C. “Batch normalization: Accelerating deep network training by reducing internal covariate shift”. *International Conference on Machine Learning (ICML)*. 2015, pp. 448–456.

[Ji13] Ji, S. et al. “3D convolutional neural networks for human action recognition”. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **35**.1 (2013), pp. 221–231.

[Kai96] Kaiser, P. K. & Boynton, R. M. “Human color vision” (1996).

[Kam18] Kamnitsas, K. et al. “Semi-supervised learning via compact latent space clustering”. *arXiv preprint arXiv:1806.02679* (2018).

[Kar17] Kar, A. et al. “AdaScan: Adaptive Scan Pooling in Deep Convolutional Neural Networks for Human Action Recognition in Videos”. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. 2017, pp. 5699–5708.

[Kin13] Kingma, D. P. “Fast gradient-based inference with continuous latent variable models in auxiliary form”. *arXiv preprint arXiv:1306.0733* (2013).

[Kin14a] Kingma, D. P. & Ba, J. “Adam: A method for stochastic optimization”. *arXiv preprint arXiv:1412.6980* (2014).

[Kin14b] Kingma, D. P. & Welling, M. “Auto-Encoding Variational Bayes”. *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*. Ed. by Bengio, Y. & LeCun, Y. 2014.

[Kin14c] Kingma, D. P. et al. “Semi-supervised learning with deep generative models”. *Advances in neural information processing systems*. 2014, pp. 3581–3589.

[Kip17] Kipf, T. N. & Welling, M. “Semi-Supervised Classification with Graph Convolutional Networks”. *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

[Kri12] Krizhevsky, A. et al. “Imagenet classification with deep convolutional neural networks”. *Advances in neural information processing systems*. 2012, pp. 1097–1105.

[Kue11] Kuehne, H. et al. “HMDB: a large video database for human motion recognition”. *2011 IEEE International Conference on Computer Vision (ICCV)*. IEEE. 2011, pp. 2556–2563.

[Lai16] Laine, S. & Aila, T. “Temporal ensembling for semi-supervised learning”. *arXiv preprint arXiv:1610.02242* (2016).

[Lec18] Lecouat, B. et al. “Manifold regularization with gans for semi-supervised learning”. *arXiv preprint arXiv:1807.04307* (2018).

[Li18] Li, Z. et al. “VideoLSTM convolves, attends and flows for action recognition”. *Computer Vision and Image Understanding* **166**.C (2018), pp. 41–50.

[Lin08] Lin, W. et al. “Human activity recognition for video surveillance”. *Circuits and Systems, 2008. ISCAS 2008. IEEE International Symposium on*. IEEE. 2008, pp. 2737–2740.

[Luo18] Luo, Y. et al. “Smooth neighbors on teacher graphs for semi-supervised learning”. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 8896–8905.

[Maa16] Maaløe, L. et al. “Auxiliary deep generative models”. *arXiv preprint arXiv:1602.05473* (2016).

[Maa08] Maaten, L. v. d. & Hinton, G. “Visualizing data using t-SNE”. *Journal of machine learning research* **9**.Nov (2008), pp. 2579–2605.

[Miy18] Miyato, T. et al. “Virtual adversarial training: a regularization method for supervised and semi-supervised learning”. *IEEE transactions on pattern analysis and machine intelligence* (2018).

[Net11] Netzer, Y. et al. “Reading digits in natural images with unsupervised feature learning” (2011).

[Ng15] Ng, J. Y.-H. et al. “Beyond short snippets: Deep networks for video classification”. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. 2015, pp. 4694–4702.

[Par18] Park, S. et al. “Adversarial dropout for supervised and semi-supervised learning”. *Thirty-Second AAAI Conference on Artificial Intelligence*. 2018.

[Pen14] Peng, X. et al. “Action recognition with stacked fisher vectors”. *European Conference on Computer Vision (ECCV)*. Springer. 2014, pp. 581–595.

[Qi19] Qi, G.-J. “Loss-sensitive generative adversarial networks on lipschitz densities”. *International Journal of Computer Vision* (2019), pp. 1–23.

[Qi18a] Qi, G.-J. et al. “Global versus localized generative adversarial nets”. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 1517–1525.

[Qi18b] Qi, S. et al. “Generalized Earley Parser: Bridging Symbolic Grammars and Sequence Data for Future Prediction”. *International Conference on Machine Learning (ICML)*. 2018, pp. 4168–4176.

[Qi18c] Qi, S. et al. “Learning human-object interactions by graph parsing neural networks”. *European Conference on Computer Vision (ECCV)*. 2018, pp. 401–417.

[Ram19] Ramachandra, B. & Jones, M. “Street Scene: A new dataset and evaluation protocol for video anomaly detection”. *arXiv preprint arXiv:1902.05872* (2019).

[Ras15] Rasmus, A. et al. “Semi-supervised learning with ladder networks”. *Advances in Neural Information Processing Systems*. 2015, pp. 3546–3554.

[Rat13] Ratanpara, T. V. & Bhatt, M. S. “A novel approach to retrieve video song using continuty of audio segments from Bollywood movies” (2013).

[Rez15] Rezende, D. J. & Mohamed, S. “Variational inference with normalizing flows”. *arXiv preprint arXiv:1505.05770* (2015).

[Rez14] Rezende, D. J. et al. “Stochastic backpropagation and approximate inference in deep generative models”. *arXiv preprint arXiv:1401.4082* (2014).

[Sal16] Salimans, T. et al. “Improved techniques for training gans”. *Advances in neural information processing systems*. 2016, pp. 2234–2242.

[San17] Santoro, A. et al. “A simple neural network module for relational reasoning”. *Advances in Neural Information Processing Systems (NIPS)*. 2017, pp. 4974–4983.

[Shu18] Shukla, A. et al. “Semi-Supervised Clustering with Neural Networks.” 2018.

[Sig16] Sigurdsson, G. A. et al. “Hollywood in homes: Crowdsourcing data collection for activity understanding”. *European Conference on Computer Vision (ECCV)*. Springer. 2016, pp. 510–526.

[Sig17] Sigurdsson, G. A. et al. “Asynchronous temporal fields for action recognition”. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 585–594.

[Sim14a] Simonyan, K. & Zisserman, A. “Two-stream convolutional networks for action recognition in videos”. *Advances in neural information processing systems*. 2014, pp. 568–576.

[Sim14b] Simonyan, K. & Zisserman, A. “Very deep convolutional networks for large-scale image recognition”. *arXiv preprint arXiv:1409.1556* (2014).

[Sim15] Simonyan, K. & Zisserman, A. “Very Deep Convolutional Networks for Large-Scale Image Recognition”. *International Conference on Learning Representations (ICLR)*. 2015.

[Son18] Song, H. et al. “Pyramid dilated deeper convlstm for video salient object detection”. *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 715–731.

[Soo14] Soomro, K. & Zamir, A. R. “Action recognition in realistic sports videos”. *Computer Vision in Sports*. Springer, 2014, pp. 181–208.

[Soo12] Soomro, K. et al. “A dataset of 101 human action classes from videos in the wild”. *Center for Research in Computer Vision* (2012).

[Spr16] Springenberg, J. T. “UNSUPERVISED AND SEMI-SUPERVISED LEARNING WITH CATEGORICAL GENERATIVE ADVERSARIAL NETWORKS”. *stat* **1050** (2016), p. 30.

[Sri14] Srivastava, N. et al. “Dropout: A simple way to prevent neural networks from overfitting”. *The Journal of Machine Learning Research* **15**.1 (2014), pp. 1929–1958.

[Sun17] Sun, L. et al. “Lattice Long Short-Term Memory for Human Action Recognition.” *2017 IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 2166–2175.

[Sze16] Szegedy, C. et al. “Rethinking the Inception Architecture for Computer Vision”. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. 2016, pp. 2818–2826.

[Tar17] Tarvainen, A. & Valpola, H. “Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results”. *Advances in neural information processing systems*. 2017, pp. 1195–1204.

[Tra15] Tran, D. et al. “Learning Spatiotemporal Features with 3D Convolutional Networks”. *2015 IEEE International Conference on Computer Vision (ICCV)*. IEEE. 2015, pp. 4489–4497.

[Val15] Valpola, H. “From neural PCA to deep unsupervised learning”. *Advances in Independent Component Analysis and Learning Machines*. Elsevier, 2015, pp. 143–171.

[Van13] Vantigodi, S. & Babu, R. V. “Real-time human action recognition from motion capture data”. *Computer Vision, Pattern Recognition, Image Processing and Graphics (NCVPRIPG), 2013 Fourth National Conference on*. IEEE. 2013, pp. 1–4.

[Vas17] Vaswani, A. et al. “Attention is all you need”. *Advances in Neural Information Processing Systems (NIPS)*. 2017, pp. 6000–6010.

[Ver19] Verma, V. et al. “Interpolation consistency training for semi-supervised learning”. *arXiv preprint arXiv:1903.03825* (2019).

[Wan13] Wang, H. & Schmid, C. “Action recognition with improved trajectories”. *2013 IEEE International Conference on Computer Vision (ICCV)*. IEEE. 2013, pp. 3551–3558.

[Wan11] Wang, H. et al. “Action recognition by dense trajectories”. *2011 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. 2011, pp. 3169–3176.

[Wan16a] Wang, L. et al. “Temporal segment networks: Towards good practices for deep action recognition”. *European Conference on Computer Vision (ECCV)*. Springer. 2016, pp. 20–36.

[Wan19a] Wang, Q. et al. “Semi-supervised learning by augmented distribution alignment”. *Proceedings of the IEEE International Conference on Computer Vision*. 2019, pp. 1466–1475.

[Wan19b] Wang, W. et al. “Learning unsupervised video object segmentation through visual attention”. *2019 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 3064–3074.

[Wan19c] Wang, W. et al. “Revisiting video saliency prediction in the deep learning era”. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2019).

[Wan18] Wang, X. et al. “Non-local Neural Networks”. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. 2018, pp. 7794–7803.

[Wan16b] Wang, Y. et al. “Hierarchical attention network for action recognition in videos”. *arXiv preprint arXiv:1607.06416* (2016).

[Wan17] Wang, Y. et al. “Spatiotemporal Pyramid Network for Video Action Recognition”. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. 2017, pp. 2097–2106.

[Wei18] Wei, X. et al. “Improving the improved training of wasserstein gans: A consistency term and its dual effect”. *arXiv preprint arXiv:1803.01541* (2018).

[Wu18] Wu, C.-Y. et al. “Compressed video action recognition”. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 6026–6035.

[Wu19] Wu, C.-Y. et al. “Long-term feature banks for detailed video understanding”. *2019 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 284–293.

[Xie18] Xie, S. et al. “Rethinking spatiotemporal feature learning: Speed-accuracy trade-offs in video classification”. *European Conference on Computer Vision (ECCV)*. 2018, pp. 305–321.

[Xin15] Xingjian, S. et al. “Convolutional LSTM network: A machine learning approach for precipitation nowcasting”. *Advances in Neural Information Processing Systems (NIPS)*. 2015, pp. 802–810.

[Xu15] Xu, K. et al. “Show, attend and tell: Neural image caption generation with visual attention”. *International Conference on Machine Learning (ICML)*. 2015, pp. 2048–2057.

[Yu18] Yu, H. et al. “Fine-grained video captioning for sports narrative”. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 6006–6015.

[Zha18] Zhang, H. et al. “mixup: Beyond Empirical Risk Minimization”. *International Conference on Learning Representations*. 2018.

[Zho18a] Zhou, B. et al. “Temporal relational reasoning in videos”. *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 803–818.

[Zho04] Zhou, D. et al. “Learning with local and global consistency”. *Advances in neural information processing systems*. 2004, pp. 321–328.

[Zho18b] Zhou, Y. et al. “MiCT: Mixed 3D/2D Convolutional Tube for Human Action Recognition”. *2018 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018, pp. 449–458.

[Zhu16] Zhu, W. et al. “A key volume mining deep framework for action recognition”. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 1991–1999.

[Zhu03] Zhu, X. et al. “Semi-supervised learning using gaussian fields and harmonic functions”. *Proceedings of the 20th International conference on Machine learning (ICML-03)*. 2003, pp. 912–919.

APPENDIX

APPENDIX

A

CONSISTENCY REGULARIZATION WITH GENERATIVE ADVERSARIAL NETWORKS FOR SEMI-SUPERVISED LEARNING

A.1 Network Architectures

Table A.1 and Table A.2 present the network architectures used in all our experiments. They are identical as the network architectures used in CT-GAN [Wei18] except that we remove the first dropout layer of the discriminator as we find it slightly downgrades the classification performance.

Table A.1 The discriminator network architecture used in our experiments.

Discriminator D
Input: 32 × 32 RGB image
3 × 3 conv, 128, Pad=1, Stride=1, WeightNorm, lReLU(0.2)
3 × 3 conv, 128, Pad=1, Stride=1, WeightNorm, lReLU(0.2)
3 × 3 conv, 128, Pad=1, Stride=2, WeightNorm, lReLU(0.2)
Dropout: p =0.5
3 × 3 conv, 256, Pad=1, Stride=1 WeightNorm, lReLU(0.2)
3 × 3 conv, 256, Pad=1, Stride=1, WeightNorm, lReLU(0.2)
3 × 3 conv, 256, Pad=1, Stride=2, WeightNorm, lReLU(0.2)
Dropout: p =0.5
3 × 3 conv, 512, Pad=0, Stride=1, WeightNorm, lReLU(0.2)
1 × 1 conv, 256, Pad=0, Stride=1, WeightNorm, lReLU(0.2)
1 × 1 conv, 128, Pad=0, Stride=1, WeightNorm, lReLU(0.2)
Global AveragePool
MLP 10, WeightNorm, Softmax

Table A.2 The generator network architecture used in our experiments.

Generator G
Input: $\mathbf{z} \sim \mathbf{U}(0,1)$ of 100 dimension
MLP 8192, BatchNorm, ReLU
Reshape to 512 × 4 × 4
5 × 5 deconv, 256, InputPad=2, Stride=2,
OutputPad=1, BatchNorm, ReLU
5 × 5 deconv, 128, InputPad=2, Stride=2,
OutputPad=1, BatchNorm, ReLU
5 × 5 deconv, 3, InputPad=2, Stride=2,
OutputPad=1, WeightNorm, Tanh