UNIVERSITY OF TECHNOLOGY SYDNEY
Faculty of Engineering and Information Technology

Regularization in Deep Neural Networks

by

Guoliang Kang

A Thesis Submitted
in Partial Fulfillment of the
Requirements for the Degree

Doctor of Philosophy

Sydney, Australia

2019

Certificate of Authorship/Originality

I certify that the work in this thesis has not been previously submitted for a degree nor has it been submitted as a part of the requirements for other degree except as fully acknowledged within the text.

I also certify that this thesis has been written by me. Any help that I have received in my research and in the preparation of the thesis itself has been fully acknowledged. In addition, I certify that all information sources and literature used are quoted in the thesis. This research is supported by the Australian Government Research Training Program.

Guoliang Kang

Feb. 2019

Production Note:
Signature removed
prior to publication.

ABSTRACT

Regularization in Deep Neural Networks

by

Guoliang Kang

Recent years have witnessed the great success of deep learning. As the deep architecture becomes larger and deeper, it is easy to overfit to relatively small amount of data. Regularization has proved to be an effective way to reduce overfitting in traditional statistical learning area. In the context of deep learning, some special design is required to regularize their training process. Generally, we firstly proposed a new regularization technique named "Shakeout" to improve the generalization ability of deep neural networks beyond Dropout, via introducing a combination of L_0, L_1, and L_2 regularization effect into the network training. Then we considered the unsupervised domain adaptation setting where the source domain data is labeled and the target domain data is unlabeled. We proposed "deep adversarial attention alignment" to regularize the behavior of the convolutional layers. Such regularization reduces the domain shift existing at the start in the convolutional layers which has been ignored by previous works and leads to superior adaptation results.

Dissertation directed by Professor Yi Yang

Center of AI, School of Software

Acknowledgements

First and foremost, I am tremendously grateful for my supervisor Yi Yang for his continuous support and guidance throughout my PhD, and for providing me the freedom to work on a variety of problems. I am grateful for Prof. Dacheng Tao, who has ever supervised me and provided me support. I am grateful for my co-supervisor Jun Li for his beneficial suggestions for my research.

I am happy to collaborate with the previous postdoc in our team Liang Zheng. Thanks for his creative guidance and suggestions for my research and academic writing. I am happy to collaborate with many creative students in our team. I am grateful for the creative discussions with them and I really appreciate the kind and useful suggestions given by them.

Thanks for all the people that ever helped me and encouraged me.

Finally, this thesis is dedicated to my parents Zhongwen Kang, Fenglan Zhang, and my wife Mingyue You, for all the years of love and support. They are always the source of my power and the reason I insist on pursuing my dream.

Guoliang Kang

Sydney, 2019.

List of Publications

Journal Papers

J-1. **G. Kang**, J. Li, and D. Tao, "Shakeout: A new approach to regularized deep neural network training", IEEE transactions on pattern analysis and machine intelligence, vol. 40, no. 5, pp. 12451258, 2018.

Conference Papers

C-1. **G. Kang**, J. Li, and D. Tao, "Shakeout: A new regularized deep neural networktraining scheme," in AAAI, 2016.

C-2. **G. Kang**, L. Zheng, Y. Yan, and Y. Yang, "Deep Adversarial Attention Alignment for Unsupervised Domain Adaptation: the Benefit of Target Expectation Maximization", in ECCV, 2018

Contents

List of Figures

Chapter 1

Introduction

1.1 Background

The attraction of machine learning is that it can make predictions for the unseen inputs, based on the collected (annotated) training data. The main focus of machine learning community is to reduce the prediction error rate on the new inputs, via minimize the training error. However, one of the main obstacles to realize this is that large amounts of in-domain annotated training data is expensive to obtain. Consequently, the model which performs well on the training data, cannot give satisfactory estimates for the test data. There exist a noticeable gap between model's performance on training data and that on test data, *i.e.* the model's generalization performance is poor.

Recent years have witnessed the rising of deep learning. The success of deep learning can be largely attributed to the access to large amounts of annotated data, *e.g.* ImageNet [18]. However, in the scenario where the data is limited or scarce, millions of parameters make deep architectures easy to overfit to the training data.

Regularization has long been an effective technique to reduce the risk of overfitting and has been widely adopted in traditional statistical learning, *e.g.* ridge loss [42], lasso, ElasticNet regularization, group lasso [35],*etc.* However, when it turns to deep neural networks, we should take some new thoughts about the way to regularize their training. 1) should it be applied on the weights or the hierachical representations? 2) is it differentiable and can it facilitate the SGD training? 3) how about its efficiency? 4) $\cdots$

In this thesis, we consider two settings.

1) Supervised learning: During training, we have access to the training data with annotations, but no access to the test data. We assume the underlying distributions of training data and test data are the same. When the training data is not abundant (it is always the case), the model is easy to overfit to the training data, and thus performs worse on the test data.

2) Unsupervised domain adaptation: Under this setting, we have data from two domains, *i.e.* the source domain data with annotations, and the target domain data without annotations. The assumption is that the data distributions of two domains are different, *i.e.* the domain shift exists between the source and target domains. Another assumption is that the tasks for both domains are the same, *e.g.* for classification, they share the same set of underlying categories. We exploit both the labeled source data and unlabeled target data to train our model and make predictions on target domain data. Without any regularization, the model's performance degenerates dramatically on the target data, due to its overfitting to the source data.

From a unified view, if we treat the data upon which we would like to make predictions as the *target* domain data, and the other data (usually labeled) as the *source* domain data, the comparison of different settings is summarized in Table 1.1.

For both settings, the regularization techniques can be employed to improve the model's generalization performance on the target data, no matter whether the target data subjects to the same distribution with the source data or not. For the first setting, we study the effect of regularization on improving the model's "*identical domain generalization*" performance, while for the second one, we aim to adopt regularization to boost the model's "*cross-domain generalization*" ability.

For supervised learning, we improve the generalization performance beyond Dropout,

Setting	source data	source labels	target data	target labels	same distributions
Supervised	yes	yes	no	no	yes
UDA	yes	yes	yes	no	no
Semi-supervised	yes	yes	yes	no	yes
Domain Generalization	yes	yes	no	no	no

Table 1.1 : Comparison of different settings in a unified view. The last column illustrates whether the source and target data distributions are the same. The "source/target data" here means the samples without labels. The "yes" or "no" denotes if a particular kind of data is available during training. In this thesis, we focus on discussing the first two settings, *i.e.* supervised learning and UDA.

by proposing a new regularized deep neural network training method named "Shakeout". Instead of randomly discarding units as Dropout does at the training stage, Shakeout randomly chooses to enhance or reverse each unit's contribution to the next layer. This minor modification of Dropout has the statistical trait: the regularizer induced by Shakeout adaptively combines L_0, L_1 and L_2 regularization terms. Our classification experiments with representative deep architectures on image datasets MNIST, CIFAR-10 and ImageNet show that Shakeout deals with over-fitting effectively and outperforms Dropout. We empirically demonstrate that Shakeout leads to sparser weights under both unsupervised and supervised settings. Shakeout also leads to the grouping effect of the input units in a layer. Considering the weights in reflecting the importance of connections, Shakeout is superior to Dropout, which is valuable for the deep model compression. Moreover, we demonstrate that Shakeout can effectively reduce the instability of the training process of the deep architecture.

In UDA, we make two contributions using the convolutional neural network (CNN). First, our approach transfers knowledge in all the convolutional layers

through attention alignment. Most previous methods align high-level representations, *e.g.* activations of the fully connected (FC) layers. In these methods, however, the convolutional layers which underpin critical lowlevel domain knowledge cannot be updated directly towards reducing domain discrepancy. Specifically, we assume that the discriminative regions in an image are relatively invariant to image style changes. Based on this assumption, we propose an attention alignment scheme on all the target convolutional layers to uncover the knowledge shared by the source domain. Second, we estimate the posterior label distribution of the unlabeled data for target network training. Previous methods, which iteratively update the pseudo labels by the target network and refine the target network by the updated pseudo labels, are vulnerable to label estimation errors. Instead, our approach uses category distribution to calculate the cross-entropy loss for training, thereby ameliorating the error accumulation of the estimated labels. The two contributions allow our approach to outperform the state-of-the-art methods by +2.6 on the Office-31 dataset.

1.2 Thesis Organization

This thesis is organized as follows:

- *Chapter 2*: This chapter presents a survey of various techniques applied to regularize the training of deep neural networks, including the ones which can be applied generally in various applications, and those which are designed for the specific scenarios.

- *Chapter 3*: We introduce a new regularize technique named "Shakeout" to improve the generalization performance of deep neural networks beyond Dropout. Compared to Dropout which can be viewed as implicitly introducing L_2 regularization on the network weights, Shakeout additionally introduces L_0 and L_1

regularization effects. Consequently, the learned network weights are sparser than those learned by Dropout. Shakeout can be adopted in various deep architectures and various applications. This chapter is based on our work [49] and [50].

- *Chapter 4*: This chapter deals with the model overfitting to one specific domain. Suppose we have two domains, *i.e.* the source domain which is labeled, and the target domain which is unlabeled. And they share the same task (*e.g.* the underlying categories are the same for such two domains). Due to the domain shift, the model purely trained with labeled source data may overfit to source domain and thus performs worse on the target. A new regularization technique was proposed to reduce the domain shift starting from the convolutional layers for the visual domain adaptation. This chapter is based on our work [51].

- *Chapter 5*: A brief summary of the thesis contents and its contributions are given in the final chapter. Recommendation for future works is given as well.

Chapter 2

Literature Survey

Regularization is the technique that aims to reduce the error rate of the model on the test data, rather than on the training data [9]. In this chapter, we will give a brief review about the widely adopted regularization techniques for the conventional supervised learning problem and the unsupervised domain adaptation scenario.

2.1 Regularization for Supervised Learning

Deep neural networks have shown their success in a wide variety of applications. The representative power of the network becomes stronger as the architecture gets deeper [8]. However, millions of parameters make deep neural networks easily overfit. Regularization [23, 105] is an effective way to obtain a model that generalizes well. There exist many approaches to regularize the training of deep neural networks, like weight decay [68], early stopping [73], *etc.* The categories of widely adopted regularization techniques are summarized as follows.

2.1.1 Data Augmentation

More data enables the data-fitting model to generalize better. Although in practice, we have limited data, we can create fake data and add it to the training set. For classification, a reasonable assumption is that the classifier should be invariant to various input transformations which will not alter the semantic meaning of data. The transformations, despite their simplicity, contribute a lot to the success of deep architectures in various scenarios, *e.g.* in the ImageNet competition, the image transformations, *e.g.* cropping, flipping, *etc.* are widely adopted [55, 86, 94, 36, 37].

Injecting noise into the input or the hidden units of layers can also be viewed as the way to perform data augmentation. It not only improves the generalization performance of a deep neural network, but also enables the network to be more robust to such kind of noise. Plenty of works to regularize the training of deep neural networks fall into this category [89, 106, 58, 69].

2.1.2 Model Ensemble.

It adopts model averaging in which several separately trained models vote on the output given a test sample. The voting procedure is robust to prediction errors made by individual classifiers. Many methods implicitly implement model ensemble, such as dropout [89], stochastic depth [44] and swapout [87]. Stochastic depth averages architectures with various depths through randomly skipping layers. Swapout samples from abundant set of architectures with dropout and stochastic depth as its special case.

2.1.3 Weight Tying or Sharing

In some scenario, from the domain knowledge and the model architecture, a prior that there should be dependencies between model weights can be applied to regularize the training of the model.

In practice, two ways are usually adopted to depict the dependencies between model weights [27]. One is weight tying, where we explicitly impose a regularization term to penalize the norm-based distance between the weights. Another way is weight sharing, where the weights of one model equals to those of another.

A typical deep architecture to employ weight sharing is the convolutional neural network [56], where the kernels (weights) applied to perform convolution operation over different spatial locations of an image or a feature map are shared.

2.1.4 Adversarial Training

Adversarial training is the training on adversarial examples constructed from the training set. The adversarial examples are those intentionally constructed by adopting an optimization procedure to search for an input similar to the original one but leading to different output. Adversarial training can be treated as a regularization technique because it can reduce the error rate on the original test set [96, 30]. Adversarial examples also provide a way to deal with semi-supervised learning. Miyato *et al.* [67] proposed virtual adversarial examples to encourage the classifier to be robust to the small changes anywhere along the manifold where the unlabeled data lie.

2.1.5 Teacher-student Framework

The teacher-student framework is widely adopted to achieve a better target (student) network. In such framework, a teacher network is firstly trained. And the student network is trained under the "supervision" of the teacher. The architecture of the teacher and the student could be different, *e.g.* in knowledge distillation [39], the training of the student network is regularized by the teacher which is much larger, to uncover the knowledge encoded in the teacher network. The "supervison" (*i.e.* the regularization) can be applied either to the outputs of the network ([39]) or to the abstract representations of multi-layers ([115, 71]).

2.1.6 Dropout

Dropout, proposed by [41], is an efficient and effective way to regularize the training of deep neural networks. It is easy to implement: at each iteration, a subset of units of each layer is randomly chosen to be zeroed out. Many subsequent works were devised to improve the performance of Dropout [106, 5, 58]. The underlying reason why Dropout improves performance has also attracted the interest

of many researchers. Evidence has shown that Dropout may work because of its good approximation to model averaging and regularization on the network weights [89, 108, 6]. Srivastava [89] and Warde-Farley [108] exhibited through experiments that the weight scaling approximation is an accurate alternative for the geometric mean over all possible sub-networks. Gal et al. [24] claimed that training the deep neural network with Dropout is equivalent to performing variational inference in a deep Gaussian Process. Dropout can also be regarded as a way of adding noise into the neural network. By marginalizing the noise, Srivastava [89] proved for linear regression that the deterministic version of Dropout is equivalent to adding an adaptive L_2 regularization on the weights. Furthermore, Wager [105] extended the conclusion to generalized linear models (GLMs) using a quadratic approximation to the induced regularizer. The inductive bias of Dropout was studied by Helmbold et al. [38] to illustrate the properties of the regularizer induced by Dropout further.

2.2 Regularization for Unsupervised Domain Adaptation

Unsupervised domain adaptation (UDA) makes predictions for the target data, when only source annotations are available. The model trained with annotated source data only is easy to overfit to the source domain. And due to the domain shift, the model may perform worse on the target data. Thus the training of the model needs to be regularized to reduce the adverse influence of the domain shift.

Plenty of deep adaptation methods have been proposed to deal with UDA [101, 62, 63, 64, 25, 79, 81]. A popular way among these methods is to minimize the discrepancy between source and target domain, via implicitly or explicitly imposing regularization penalizing the domain discrepancy during training. We will focus on reviewing the methods developed along this line.

2.2.1 Explicitly Inducing Regularization Term

Tzeng *et al.* [101] propose a kind of domain confusion loss to encourage the network to learn both semantically meaningful and domain invariant representations. Similarly, Long *et al.* [62] minimize the MMD distance of the fully-connected activations between source and target domain while sharing the convolutional features. JAN [63] penalizes the JMMD over multiple fully-connected layers to minimize the domain discrepancy coming from both the data distribution and the label distribution. DSN [14] explicitly models domain-specific features to help improve networks' ability to learn domain-invariant features. Associative domain adaptation (ADA) [32] reinforces associations across domains directly in embedding space to extract statistically domain-invariant and class discriminative features. The Deep CORAL [91] aims to learn a nonlinear transformation that aligns correlations of the activations of FC layers across domains, which extends the shallow CORAL [90] method to deep architectures. This idea is similarly to DAN [62] and JAN [63], except that instead of MMD, the CORAL loss (expressed by the distance between the covariances) is used to minimize discrepancy between the domains.

In contrast to the above methods, Rozantsev *et al.* [76] independently trains two models, *i.e.* the source model and the target model which will be adopted for the target data predictions. The source model is trained with the cross-entropy loss on the labeled source data. Besides considering the MMD distance between the FC layers of the source and target models, an extra regularization term is imposed to ensure the weights of these two models remain linearly related.

Other than the cross-entropy loss on the labeled source domain data, all of these additional losses built upon both domain data can be regarded as the regularization terms to reduce the domain discrepancy and encourage domain-invariant representation learning.

Besides the above regularization techniques, the entropy regularization [64, 80] and the label smooth regularization (LSR) [19] are often adopted in UDA to deal with the class imbalance and reduce overfitting.

2.2.2 Implicitly Imposing Regularization

The methods that minimize the domain discrepancy in adversarial way fall into this category [25, 80, 81, 112, 82, 17, 61]. For example, Ganin *et al.* [25] enabled the network to learn domain invariant representations in an adversarial way by adding a domain classifier and back-propagating inverse gradients. Adversarial Dropout Regularization (ADR) [80] and Maximum Classifier Discrepancy (MCD) [81] were proposed to train a deep neural network in adversarial way to avoid generating non-discriminative features lying in the region near the decision boundary. Pei et al.[72] take the class information into account while measuring the domain discrepancy in adversarial way.

Chapter 3

Regularization for Supervised Learning

3.1 Introduction

Deep neural networks have recently achieved impressive success in a number of machine learning and pattern recognition tasks and been under intensive research [37, 93, 26, 92, 118, 110, 48]. Hierarchical neural networks have been known for decades, and there are a number of essential factors contributing to its recent rising, such as the availability of big data and powerful computational resources. However, arguably the most important contributor to the success of deep neural network is the discovery of efficient training approaches [40, 8, 7, 103, 104].

A particular interesting advance in the training techniques is the invention of Dropout [41]. At the operational level, Dropout adjusts the network evaluation step (feed-forward) at the training stage, where a portion of units are randomly discarded. The effect of this simple trick is impressive. Dropout enhances the generalization performance of neural networks considerably, and is behind many record-holders of widely recognized benchmarks [55, 93, 114]. The success has attracted much research attention, and found applications in a wider range of problems [105, 15, 102]. Theoretical research from the viewpoint of statistical learning has pointed out the connections between Dropout and model regularization, which is the de facto recipe of reducing over-fitting for complex models in practical machine learning. For example, Wager et al. [105] showed that for a generalized linear model (GLM), Dropout implicitly imposes an adaptive L_2 regularizer of the network weights through an estimation of the inverse diagonal Fisher information matrix.

Sparsity is of vital importance in deep learning. It is straightforward that through removing unimportant weights, deep neural networks perform prediction faster. Additionally, it is expected to obtain better generalization performance and reduce the number of examples needed in the training stage [57]. Recently much evidence has shown that the accuracy of a trained deep neural network will not be severely affected by removing a majority of connections and many researchers focus on the deep model compression task [16, 34, 33, 20, 4, 39]. One effective way of compression is to train a neural network, prune the connections and fine-tune the weights iteratively [34, 33]. However, if we can cut the connections naturally via imposing sparsity-inducing penalties in the training process of a deep neural network, the work-flow will be greatly simplified.

In this chapter, we propose a new regularized deep neural network training approach: Shakeout, which is easy to implement: randomly choosing to enhance or reverse each unit's contribution to the next layer in the training stage. Note that Dropout can be considered as a special "flat" case of our approach: randomly keeping (enhance factor is 1) or discarding (reverse factor is 0) each unit's contribution to the next layer. Shakeout enriches the regularization effect. In theory, we prove that it adaptively combines L_0, L_1 and L_2 regularization terms. L_0 and L_1 regularization terms are known as sparsity-inducing penalties. The combination of sparsity-inducing penalty and L_2 penalty of the model parameters has shown to be effective in statistical learning: the Elastic Net [122] has the desirable properties of producing sparse models while maintaining the grouping effect of the weights of the model. Because of the randomly "shaking" process and the regularization characteristic pushing network weights to zero, our new approach is named "Shakeout".

As discussed above, it is expected to obtain much sparser weights using Shakeout than using Dropout because of the combination of L_0 and L_1 regularization terms induced in the training stage. We apply Shakeout on one-hidden-layer autoencoder

and obtain much sparser weights than that resulted by Dropout. To show the regularization effect on the classification tasks, we conduct the experiments on image datasets including MNIST, CIFAR-10 and ImageNet with the representative deep neural network architectures. In our experiments we find that by using Shakeout, the trained deep neural networks always outperform those by using Dropout, especially when the data is scarce. Besides the fact that Shakeout leads to much sparser weights, we also empirically find that it groups the input units of a layer. Due to the induced L_0 and L_1 regularization terms, Shakeout can result in the weights reflecting the importance of the connections between units, which is meaningful for conducting compression. Moreover, we demonstrate that Shakeout can effectively reduce the instability of the training process of the deep architecture.

In the rest of the chapter, we give a review about the related work in Section 2. Section 3 presents Shakeout in detail, along with theoretical analysis of the regularization effect induced by Shakeout. In Section 4, we first demonstrate the regularization effect of Shakeout on the autoencoder model. The classification experiments on MNIST , CIFAR-10 and ImageNet illustrate that Shakeout outperforms Dropout considering the generalization performance, the regularization effect on the weights, and the stabilization effect on the training process of the deep architecture. Finally, we give some recommendations for the practitioners to make full use of Shakeout.

3.2 Related Work

Deep neural networks have shown their success in a wide variety of applications. One of the key factors contributes to this success is the creation of powerful training techniques. The representative power of the network becomes stronger as the architecture gets deeper [8]. However, millions of parameters make deep neural networks easily over-fit. Regularization [23, 105] is an effective way to obtain a model that generalizes well. There exist many approaches to regularize the training of deep

neural networks, like weight decay [68], early stopping [73], etc. Shakeout belongs to the family of regularized training techniques.

Among these regularization techniques, our work is closely related to Dropout [41]. Many subsequent works were devised to improve the performance of Dropout [106, 5, 58]. The underlying reason why Dropout improves performance has also attracted the interest of many researchers. Evidence has shown that Dropout may work because of its good approximation to model averaging and regularization on the network weights [89, 108, 6]. Srivastava [89] and Warde-Farley [108] exhibited through experiments that the weight scaling approximation is an accurate alternative for the geometric mean over all possible sub-networks. Gal et al. [24] claimed that training the deep neural network with Dropout is equivalent to performing variational inference in a deep Gaussian Process. Dropout can also be regarded as a way of adding noise into the neural network. By marginalizing the noise, Srivastava [89] proved for linear regression that the deterministic version of Dropout is equivalent to adding an adaptive L_2 regularization on the weights. Furthermore, Wager [105] extended the conclusion to generalized linear models (GLMs) using a quadratic approximation to the induced regularizer. The inductive bias of Dropout was studied by Helmbold et al. [38] to illustrate the properties of the regularizer induced by Dropout further. In terms of implicitly inducing regularizer of the network weights, Shakeout can be viewed as a generalization of Dropout. It enriches the regularization effect of Dropout, i.e. besides the L_2 regularization term, it also induces the L_0 and L_1 regularization terms, which may lead to sparse weights of the model.

Due to the implicitly induced L_0 and L_1 regularization terms, Shakeout is also related to sparsity-inducing approaches. Olshausen et al. [70] introduced the concept of sparsity in computational neuroscience and proposed the sparse coding method in the visual system. In machine learning, the sparsity constraint enables a model

to capture the implicit statistical data structure, performs feature selection and regularization, compresses the data at a low loss of the accuracy, and helps us to better understand our models and explain the obtained results. Sparsity is one of the key factors underlying many successful deep neural network architectures [56, 94, 95, 93] and training algorithms [12][29]. A Convolutional neural network is much sparser than the fully-connected one, which results from the concept of local receptive field [56]. Sparsity has been a design principle and motivation for Inception-series models [94, 95, 93]. Besides working as the heuristic principle of designing a deep architecture, sparsity often works as a penalty induced to regularize the training process of a deep neural network. There exist two kinds of sparsity penalties in deep neural networks, which lead to the activity sparsity [12][29] and the connectivity sparsity [97] respectively. The difference between Shakeout and these sparsity-inducing approaches is that for Shakeout, the sparsity is induced through simple stochastic operations rather than manually designed architectures or explicit norm-based penalties. This implicit way enables Shakeout to be easily optimized by stochastic gradient descent (SGD) − the representative approach for the optimization of a deep neural network.

3.3 Method

Shakeout applies on the weights in a linear module. The linear module, i.e. weighted sum,

$$\theta = \sum_{j=1}^{p} w_j x_j \tag{3.1}$$

is arguably the most widely adopted component in data models. For example, the variables x_1, x_2, ..., x_p can be input attributes of a model, e.g. the extracted features for a GLM, or the intermediate outputs of earlier processing steps, e.g. the activations of the hidden units in a multilayer artificial neural network. Shakeout

randomly modifies the computation in Eq. (3.1). Specifically, Shakeout can be realized by randomly modifying the weights

$$\textbf{\textit{Step 1}}\text{: Draw } r_j\text{, where } \begin{cases} P(r_j = 0) & = \tau \\ P(r_j = \frac{1}{1-\tau}) & = 1-\tau \end{cases}.$$

$$\textbf{\textit{Step 2}}\text{: Adjust the weight according to } r_j\text{,}$$

$$\begin{cases} \tilde{w}_j \leftarrow -cs_j, & \text{if } r_j = 0 \qquad \text{(A)} \\ \tilde{w}_j \leftarrow (w_j + c\tau s_j)/(1-\tau) & \text{otherwise} \qquad \text{(B)} \end{cases}$$

where $s_j = \text{sgn}(w_j)$ takes ± 1 depending on the sign of w_j or takes 0 if $w_j = 0$. As shown above, Shakeout chooses (randomly by drawing r) between two fundamentally different ways to modify the weights. Modification (A) is to set the weights to constant magnitudes, despite their original values except for the signs (to be opposite to the original ones). Modification (B) updates the weights by a factor $(1-\tau)^{-1}$ and a bias depending on the signs. Note both (A) and (B) preserve zero values of the weights, i.e. if $w_j = 0$ then $\tilde{w}_j = 0$ with probability 1. Let $\tilde{\theta} = \tilde{\boldsymbol{w}}^T \boldsymbol{x}$, and Shakeout leaves θ unbiased, i.e. $\mathbb{E}[\tilde{\theta}] = \theta$. The hyper-parameters $\tau \in (0, 1)$ and $c \in (0, +\infty)$ configure the property of Shakeout.

Shakeout is naturally connected to the widely adopted operation of Dropout [41, 89]. We will show that Shakeout has regularization effect on model training similar to but beyond what is induced by Dropout. From an operational point of view, Fig. 3.1 compares Shakeout and Dropout. Note that Shakeout includes Dropout as a special case when the hyper-parameter c in Shakeout is set to zero.

When applied at the training stage, Shakeout alters the objective − the quantity to be minimized − by adjusting the weights. In particular, we will show that Shakeout (with expectation over the random switch) induces a regularization term effectively penalizing the magnitudes of the weights and leading to sparse weights. Shakeout is an approach designed for helping model training, when the models are

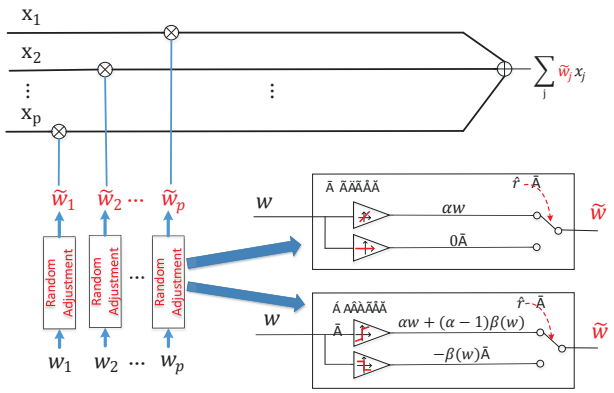


Figure 3.1 : Comparison between Shakeout and Dropout operations. This figure shows how Shakeout and Dropout are applied to the weights in a linear module. In the original linear module, the output is the summation of the inputs $\boldsymbol{x}$ weighted by $\boldsymbol{w}$, while for Dropout and Shakeout, the weights $\boldsymbol{w}$ are first randomly modified. In detail, a random switch $\hat{r}$ controls how each w is modified. The manipulation of w is illustrated within the amplifier icons (the red curves, best seen with colors). The coefficients are $\alpha = 1/(1-\tau)$ and $\beta(w) = cs(w)$, where $s(w)$ extracts the sign of w and $c > 0$, $\tau \in [0, 1]$. Note the sign of $\beta(w)$ is always the same as that of w. The magnitudes of coefficients α and $\beta(w)$ are determined by the Shakeout hyper-parameters τ and c. Dropout can be viewed as a special case of Shakeout when $c = 0$ because $\beta(w)$ is zero at this circumstance.

trained and deployed, one should relieve the disturbance to allow the model work with its full capacity, i.e. we adopt the resulting network without any modification of the weights at the test stage.

3.3.1 Regularization Effect of Shakeout

Shakeout randomly modifies the weights in a linear module, and thus can be regarded as injecting noise into each variable x_j, i.e. x_j is randomly scaled by γ_j: $\tilde{x}_j = \gamma_j x_j$. Note that $\gamma_j = r_j + \frac{c(r_j - 1)}{|w_j|}$, the modification of x_j is actually determined by the random switch r_j. Shakeout randomly chooses to enhance (i.e.

when $r_j = \frac{1}{1-\tau}$, $\gamma_j > \frac{1}{1-\tau}$) or reverse (i.e. when $r_j = 0$, $\gamma_j < 0$) each original variable x_j's contribution to the output at the training stage (see Fig. 3.1). However, the expectation of $\tilde{x}_j$ over the noise remains unbiased, i.e. $\mathbb{E}_{r_j}[\tilde{x}_j] = x_j$.

It is well-known that injecting artificial noise into the input features will regularize the training objective [105, 75, 11], i.e. $\mathbb{E}_{\boldsymbol{r}}[\ell(\boldsymbol{w}, \tilde{\boldsymbol{x}}, y)] = \ell(\boldsymbol{w}, \boldsymbol{x}, y) + \pi(\boldsymbol{w})$, where $\tilde{\boldsymbol{x}}$ is the feature vector randomly modified by the noise induced by $\boldsymbol{r}$. The regularization term $\pi(\boldsymbol{w})$ is determined by the characteristic of the noise. For example, Wager et al.[105] showed that Dropout, corresponding to inducing blackout noise to the features, helps introduce an adaptive L_2 penalty on $\boldsymbol{w}$. In this section we illustrate how Shakeout helps regularize model parameters $\boldsymbol{w}$ using an example of GLMs.

Formally, a GLM is a probabilistic model of predicting target y given features $\boldsymbol{x} = [x_1, \ldots, x_p]$, in terms of the weighted sum in Eq. (3.1):

$$\begin{aligned} P(y|\boldsymbol{x}, \boldsymbol{w}) &= h(y)g(\theta)e^{\theta y} \\ \theta &= \boldsymbol{w}^T\boldsymbol{x} \end{aligned} \tag{3.2}$$

With different $h(\cdot)$ and $g(\cdot)$ functions, GLM can be specialized to various useful models or modules, such as logistic regression model or a layer in a feed-forward neural network. However, roughly speaking, the essence of a GLM is similar to that of a standard linear model which aims to find weights $w_1, \ldots, w_p$ so that $\theta = \boldsymbol{w}^T\boldsymbol{x}$ aligns with y (functions $h(\cdot)$ and $g(\cdot)$ are independent of $\boldsymbol{w}$ and y respectively). The loss function of a GLM with respect to $\boldsymbol{w}$ is defined as

$$l(\boldsymbol{w}, \boldsymbol{x}, y) = -\theta y + A(\theta) \tag{3.3}$$

$$A(\theta) = -\ln[g(\theta)] \tag{3.4}$$

The loss (3.3) is the negative logarithm of probability (3.2), where we keep only terms relevant to $\boldsymbol{w}$.

Let the loss with Shakeout be

$$l_{\text{sko}}(\boldsymbol{w}, \boldsymbol{x}, y, \boldsymbol{r}) := l(\boldsymbol{w}, \tilde{\boldsymbol{x}}, y) \tag{3.5}$$

where $\boldsymbol{r} = [r_1, \ldots, r_p]^T$, and $\tilde{\boldsymbol{x}} = [\tilde{x}_1, \ldots, \tilde{x}_p]^T$ represents the features randomly modified with $\boldsymbol{r}$.

Taking expectation over $\boldsymbol{r}$, the loss with Shakeout becomes

$$\mathbb{E}_{\boldsymbol{r}}[l_{\text{sko}}(\boldsymbol{w}, \boldsymbol{x}, y, \boldsymbol{r})] = l(\boldsymbol{w}, \boldsymbol{x}, y) + \pi(\boldsymbol{w})$$

where

$$\begin{aligned} \pi(\boldsymbol{w}) &= \mathbb{E}_{\boldsymbol{r}}[A(\tilde{\theta}) - A(\theta)] \\ &= \sum_{k=1}^{\infty} \frac{1}{k!} A^{(k)}(\theta) \mathbb{E}[(\tilde{\theta} - \theta)^k] \end{aligned} \tag{3.6}$$

is named *Shakeout regularizer*. Note that if $A(\theta)$ is k-th order derivable, let the $k^{'}$ order derivative $A^{(k^{'})}(\theta) = 0$ where $k^{'} > k$, to make the denotation simple.

Theorem 1. *Let $q_j = x_j(w_j + cs_j)$, $\theta_{j-} = \theta - q_j$ and $\theta_{j+} = \theta + \frac{\tau}{1-\tau} q_j$, then Shakeout regularizer $\pi(\boldsymbol{w})$ is*

$$\pi(\boldsymbol{w}) = \tau \sum_{j=1}^{p} A(\theta_{j-}) + (1 - \tau) \sum_{j=1}^{p} A(\theta_{j+}) - pA(\theta) \tag{3.7}$$

Proof. Note that $\tilde{\theta} - \theta = \sum_{j=1}^{p} q_j(r_j - 1)$, then for Eq. (3.6)

$$\mathbb{E}[(\tilde{\theta} - \theta)^k] = \sum_{j_1=1}^{p} \sum_{j_2=1}^{p} \cdots \sum_{j_k=1}^{p} \prod_{m=1}^{k} q_{j_m} \mathbb{E}[\prod_{m=1}^{k} (r_{j_m} - 1)]$$

Because arbitrary two random variables $r_{j_{m_1}}$, $r_{j_{m_2}}$ are independent unless $j_{m_1} = j_{m_2}$ and $\forall r_{j_m}$, $\mathbb{E}[r_{j_m} - 1] = 0$, then

$$\begin{aligned} \mathbb{E}[(\tilde{\theta} - \theta)^k] &= \sum_{j=1}^{p} q_j^k \mathbb{E}[(r_j - 1)^k] \\ &= \tau \sum_{j=1}^{p} (-q_j)^k + (1 - \tau) \sum_{j=1}^{p} (\frac{\tau}{1 - \tau} q_j)^k \end{aligned}$$

Then

$$\begin{aligned}\pi(\boldsymbol{w}) &= \tau\sum_{j=1}^{p}\sum_{k=1}^{\infty}\frac{1}{k!}A^{(k)}(\theta)(-q_j)^k \\ &\quad +(1-\tau)\sum_{j=1}^{p}\sum_{k=1}^{\infty}\frac{1}{k!}A^{(k)}(\theta)(\frac{\tau}{1-\tau}q_j)^k\end{aligned}$$

Further, let $\theta_{j-} = \theta - q_j$, $\theta_{j+} = \theta + \frac{\tau}{1-\tau}q_j$, $\pi(\boldsymbol{w})$ becomes

$$\pi(\boldsymbol{w}) = \tau\sum_{j=1}^{p}A(\theta_{j-}) + (1-\tau)\sum_{j=1}^{p}A(\theta_{j+}) - pA(\theta)$$

The theorem is proved. □

We illustrate several properties of Shakeout regularizer based on Eq. (3.7). The proof of the following propositions can be found in the appendices.

Proposition 1. $\pi(\mathbf{0}) = 0$

Proposition 2. *If $A(\theta)$ is convex, $\pi(\boldsymbol{w}) \geq 0$.*

Proof. Because $A(\theta)$ is convex, then

$$\tau\sum_{j=1}^{p}A(\theta_{j-}) + (1-\tau)\sum_{j=1}^{p}A(\theta_{j+}) \geq \sum_{j=1}^{p}A(\tau\theta_{j-} + (1-\tau)\theta_{j+}) = pA(\theta)$$

The proposition is proved. □

Proposition 3. *Suppose $\exists j$, $x_j w_j \neq 0$. If $A(\theta)$ is convex, $\pi(\boldsymbol{w})$ monotonically increases with τ. If $A^{''}(\theta) > 0$, $\pi(\boldsymbol{w})$ monotonically increases with c.*

Proof. The gradient of $\pi(\boldsymbol{w})$ with respect to τ is

$$\frac{\partial\pi(\boldsymbol{w})}{\partial\tau} = \sum_{j=1}^{p}[A(\theta_{j-}) - A(\theta_{j+})] + \sum_{j=1}^{p}A^{'}(\theta_{j+})\frac{q_j}{1-\tau}$$

Due to $A(\theta)$ is convex

i) For $x_j w_j > 0$, $q_j > 0$, $\theta_{j-} < \theta < \theta_{j+}$, then

$$\frac{A(\theta_{j-}) - A(\theta_{j+})}{\theta_{j-} - \theta_{j+}} = \frac{A(\theta_{j-}) - A(\theta_{j+})}{-\frac{q_j}{1-\tau}} < A^{'}(\theta_{j+})$$

Thus

$$A(\theta_{j-}) - A(\theta_{j+}) + \frac{q_j}{1-\tau}A^{'}(\theta_{j+}) > 0$$

ii) For $x_j w_j < 0$, $q_j < 0$, $\theta_{j+} < \theta < \theta_{j-}$, then

$$\frac{A(\theta_{j-}) - A(\theta_{j+})}{\theta_{j-} - \theta_{j+}} = \frac{A(\theta_{j-}) - A(\theta_{j+})}{-\frac{q_j}{1-\tau}} > A^{'}(\theta_{j+})$$

Thus

$$A(\theta_{j-}) - A(\theta_{j+}) + \frac{q_j}{1-\tau}A^{'}(\theta_{j+}) > 0$$

iii) For $x_j w_j = 0$, $q_j = 0$, $\theta_{j+} = \theta = \theta_{j-}$, then

$$A(\theta_{j-}) - A(\theta_{j+}) + \frac{q_j}{1-\tau}A^{'}(\theta_{j+}) = 0$$

Because $\exists j$, $x_j w_j \neq 0$, $\frac{\partial \pi(\boldsymbol{w})}{\partial \tau} > 0$ always holds.

The gradient of $\pi(\boldsymbol{w})$ with respect to c is

$$\frac{\partial \pi(\boldsymbol{w})}{\partial c} = \tau \sum_{j=1}^{p} [x_j s_j (A^{'}(\theta_{j+}) - A^{'}(\theta_{j-}))]$$

Because $A^{''}(\theta) > 0$, so

i) For $x_j w_j > 0$, $q_j > 0$, $\theta_{j-} < \theta < \theta_{j+}$, then $A^{'}(\theta_{j+}) > A^{'}(\theta_{j-})$, $\frac{\partial \pi(\boldsymbol{w})}{\partial c} > 0$;

ii) For $x_j w_j < 0$, $q_j < 0$, $\theta_{j+} < \theta < \theta_{j-}$, then $A^{'}(\theta_{j+}) < A^{'}(\theta_{j-})$, $\frac{\partial \pi(\boldsymbol{w})}{\partial c} > 0$;

iii) For $x_j w_j = 0$, $q_j = 0$, $\theta_{j+} = \theta = \theta_{j-}$, then $A^{'}(\theta_{j+}) = A^{'}(\theta_{j-})$, $\frac{\partial \pi(\boldsymbol{w})}{\partial c} = 0$.

Because $\exists j$, $x_j w_j \neq 0$, $\frac{\partial \pi(\boldsymbol{w})}{\partial c} > 0$ always holds.

The proposition is proved. □

Proposition 3 implies that the hyper-parameters τ and c relate to the strength of the regularization effect. It is reasonable because higher τ or c means the noise injected into the features $\boldsymbol{x}$ has larger variance.

Proposition 4. *Suppose i)* $\forall j \neq j^{'}$, $x_j w_j = 0$, *and ii)* $x_{j^{'}} \neq 0$.

Then

i) if $A^{''}(\theta) > 0$, $\begin{cases} \frac{\partial \pi(\boldsymbol{w})}{\partial w_{j'}} > 0, & when\ w_{j'} > 0 \\ \frac{\partial \pi(\boldsymbol{w})}{\partial w_{j'}} < 0, & when\ w_{j'} < 0 \end{cases}$

ii) if $\lim_{|\theta| \to \infty} A^{''}(\theta) = 0$, $\lim_{|w_{j'}| \to \infty} \frac{\partial \pi(\boldsymbol{w})}{\partial w_{j'}} = 0$

Proof. For denotation simplicity, we use x, w to represent $x_{j'}$ and $w_{j'}$, respectively. At this circumstance, Shakeout regularizer becomes

$$\pi(\boldsymbol{w}) = \tau A(-cxs) + (1-\tau)A(\frac{1}{1-\tau}xw + \frac{c\tau}{1-\tau}xs) - A(xw)$$

The gradient with respect to w ($w \neq 0$) is

$$\frac{\partial \pi(\boldsymbol{w})}{\partial w} = x(A^{'}(\frac{1}{1-\tau}xw + \frac{c\tau}{1-\tau}xs) - A^{'}(xw))$$

i) If $A^{''}(\theta) > 0$, suppose $xw > 0$, $\frac{1}{1-\tau}xw + \frac{c\tau}{1-\tau}xs > xw$,

a) If $x > 0$, $w > 0$, $\frac{\partial \pi(\boldsymbol{w})}{\partial w} > 0$

b) If $x < 0$, $w < 0$, $\frac{\partial \pi(\boldsymbol{w})}{\partial w} < 0$

Similarly when $xw < 0$

a) If $x > 0$, $w < 0$, $\frac{\partial \pi(\boldsymbol{w})}{\partial w} < 0$

b) If $x < 0$, $w > 0$, $\frac{\partial \pi(\boldsymbol{w})}{\partial w} > 0$

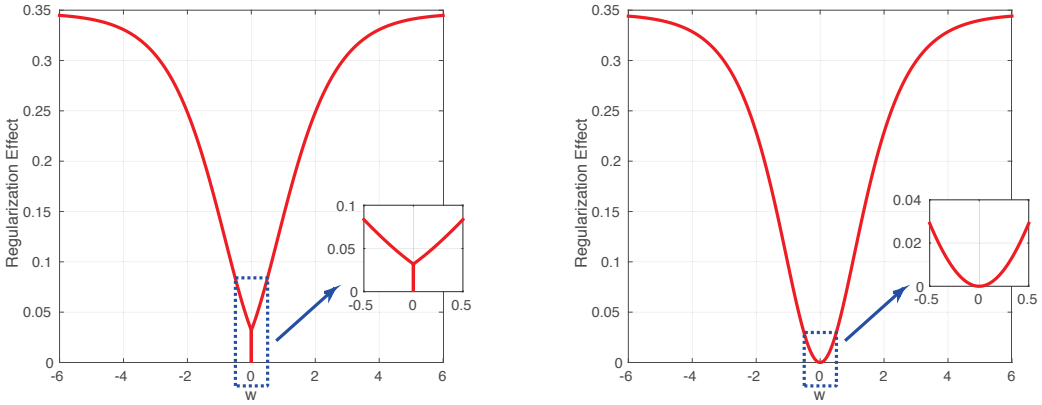
So despite the sgn of x, there always exists

$$\begin{cases} \frac{\partial \pi(\boldsymbol{w})}{\partial w} > 0, & \text{when } w > 0 \\ \frac{\partial \pi(\boldsymbol{w})}{\partial w} < 0, & \text{when } w < 0 \end{cases}$$

ii) If $\lim_{|\theta| \to \infty} A^{''}(\theta) = 0$, $\lim_{|w| \to \infty}(A^{'}(\frac{1}{1-\tau}xw + \frac{c\tau}{1-\tau}xs) - A^{'}(xw)) = 0$ Thus

$$\lim_{|w| \to \infty} \frac{\partial \pi(\boldsymbol{w})}{\partial w} = 0$$

The proposition is proved. □



(a) Shakeout: $(\tau, c) = (0.3, 0.78)$ (b) Dropout: $\tau = 0.5$

Figure 3.2 : Regularization effect as a function of a single weight when other weights are fixed to zeros for logistic regression model. The corresponding feature x is fixed at 1.

Proposition 4 implies that under certain conditions, starting from a zero weight vector, Shakeout regularizer penalizes the magnitude of $w_{j'}$ and its regularization effect is bounded by a constant value. For example, for logistic regression, $\pi(\boldsymbol{w}) \leq \tau \ln(1+\exp(c|x_{j'}|))$, which is illustrated in Fig. 3.2. This bounded property has been proved to be useful: capped-norm [47] is more robust to outliers than the traditional L_1 or L_2 norm.

Based on the Eq. (3.7), the specific formulas for the representative GLM models can be derived:

i) Linear regression: $A(\theta) = \frac{1}{2}\theta^2$, then

$$\pi(\boldsymbol{w}) = \frac{\tau}{2(1-\tau)} \left\| \boldsymbol{x} \circ (\boldsymbol{w} + c\boldsymbol{s}) \right\|_2^2$$

where $\circ$ denotes the element-wise product and the $\left\| \boldsymbol{x} \circ (\boldsymbol{w} + c\boldsymbol{s}) \right\|_2^2$ term can be decomposed into the summation of three components

$$\sum_{j=1}^{p} x_j^2 w_j^2 + 2c \sum_{j=1}^{p} x_j^2 |w_j| + c^2 \sum_{j=1}^{p} x_j^2 \mathbf{1}_{w_j \neq 0}[w_j] \tag{3.8}$$

where $\mathbf{1}_{w_j \neq 0}[w_j]$ is an indicator function which satisfies $\mathbf{1}_{w_j \neq 0}[w_j] = \begin{cases} 1 & w_j \neq 0 \\ 0 & w_j = 0 \end{cases}$.
This decomposition implies that Shakeout regularizer penalizes the combination of L_0-norm, L_1-norm and L_2-norm of the weights after scaling them with the square of corresponding features. The L_0 and L_1 regularization terms can lead to sparse weights.

ii) Logistic regression: $A(\theta) = \ln(1 + \exp(\theta))$, then

$$\pi(\boldsymbol{w}) = \sum_{j=1}^{p} \ln(\frac{(1 + \exp(\theta_{j-}))^{\tau}(1 + \exp(\theta_{j+}))^{1-\tau}}{1 + \exp(\theta)}) \tag{3.9}$$

Fig. 3.3 illustrates the contour of Shakeout regularizer based on Eq. (3.9) in the 2D weight space. On the whole, the contour of Shakeout regularizer indicates that the regularizer combines L_0, L_1 and L_2 regularization terms. As c goes to zero, the contour around $w = 0$ becomes less sharper, which implies hyper-parameter c relates to the strength of L_0 and L_1 components. When $c = 0$, Shakeout degenerates to Dropout, the contour of which implies Dropout regularizer consists of L_2 regularization term.

The difference between Shakeout and Dropout regularizers is also illustrated in Fig. 3.2. We set $\tau = 0.3$, $c = 0.78$ for Shakeout, and $\tau = 0.5$ for Dropout to make the bounds of the regularization effects of two regularizers the same. In this one dimension circumstance, the main difference is that at $w = 0$ (see the enlarged snapshot for comparison), Shakeout regularizer is sharp and discontinuous while Dropout regularizer is smooth. Thus compared to Dropout, Shakeout may lead to much sparser weights of the model.

To simplify the analysis and prove the intuition we have observed in Fig. 3.3 about the properties of Shakeout regularizer, we quadratically approximate Shakeout

regularizer of Eq. (3.7) by

$$\pi_{approx}(\boldsymbol{w}) = \frac{\tau}{2(1-\tau)} A^{''}(\theta) \left\| \boldsymbol{x} \circ (\boldsymbol{w} + c\boldsymbol{s}) \right\|_2^2 \tag{3.10}$$

The $\left\| \boldsymbol{x} \circ (\boldsymbol{w} + c\boldsymbol{s}) \right\|_2^2$, already shown in Eq. (3.8), consists of the combination of L_0, L_1, L_2 regularization terms. It tends to penalize the weight whose corresponding feature's magnitude is large. Meanwhile, the weights whose corresponding features are always zeros are less penalized. The term $A^{''}(\theta)$ is proportional to the variance of prediction y given $\boldsymbol{x}$ and $\boldsymbol{w}$. Penalizing $A^{''}(\theta)$ encourages the weights to move towards making the model be more "confident" about its predication, i.e. be more discriminative.

Generally speaking, Shakeout regularizer adaptively combines L_0, L_1 and L_2 regularization terms, the property of which matches what we have observed in Fig. 3.3. It prefers penalizing the weights who have large magnitudes and encourages the weights to move towards making the model more discriminative. Moreover, the weights whose corresponding features are always zeros are less penalized. The L_0 and L_1 components can induce sparse weights.

Last but not the least, we want to emphasize that when $\tau = 0$, the noise is eliminated and the model becomes a standard GLM. Moreover, Dropout can be viewed as the special case of Shakeout when $c = 0$, and a higher value of τ means a stronger L_2 regularization effect imposed on the weights. Generally, when τ is fixed ($\tau \neq 0$), a higher value of c means a stronger effect of the L_0 and L_1 components imposed and leads to much sparser weights of the model. We will verify this property in our experiment section later.

3.3.2 Shakeout in Multilayer Neural Networks

It has been illustrated that Shakeout regularizes the weights in linear modules. Linear module is the basic component of multilayer neural networks. That is, the

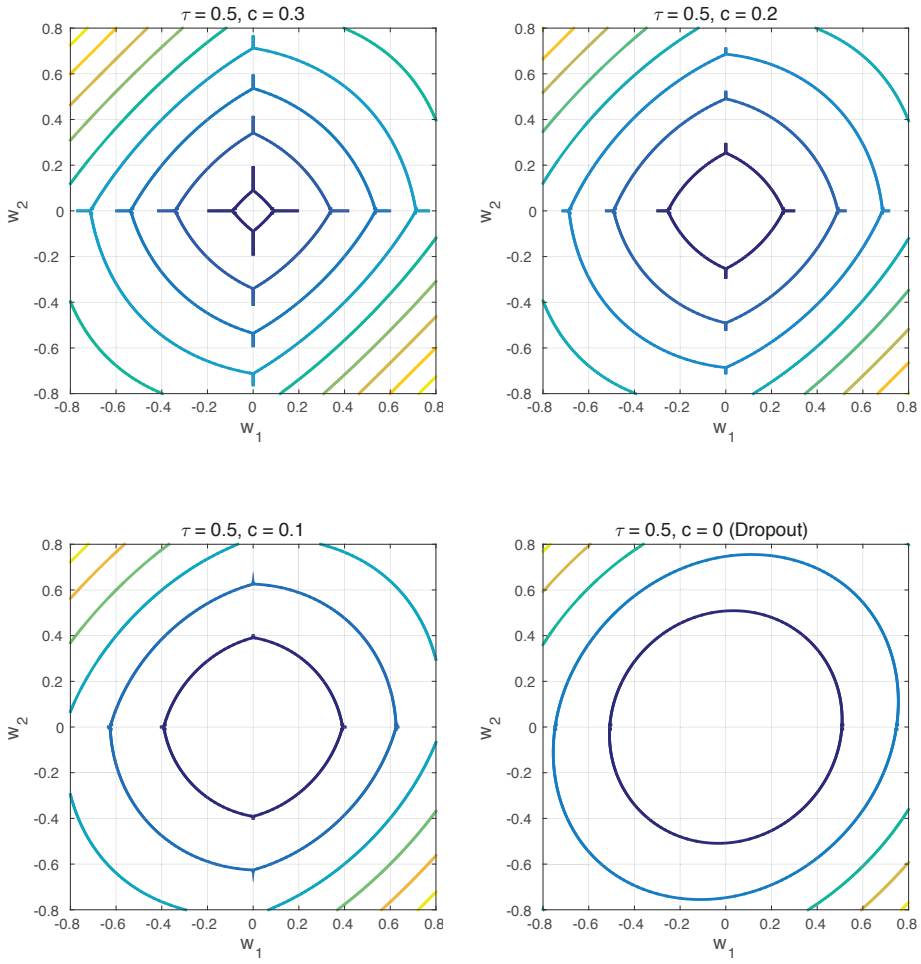


Figure 3.3 : The contour plots of the regularization effect induced by Shakeout in 2D weight space with input $\boldsymbol{x} = [1, 1]^T$. Note that Dropout is a special case of Shakeout with $c = 0$.

linear operations connect the outputs of two successive layers. Thus Shakeout is readily applicable to the training of multilayer neural networks.

Considering the forward computation from layer l to layer $l+1$, for a fully-connected layer, the Shakeout forward computation is as follows

$$u_i = \sum_j x_j[r_j W_{ij} + c(r_j - 1)S_{ij}] + b_i \tag{3.11}$$

$$x_i^{'} = f(u_i) \tag{3.12}$$

where i denotes the index of the output unit of layer $l+1$, and j denotes the index of the output unit of layer l. The output unit of a layer is represented by x. The weight of the connection between unit x_j and unit $x_i^{'}$ is represented as W_{ij}. The bias for the i-th unit is denoted by b_i. The S_{ij} is the sign of corresponding weight W_{ij}. After Shakeout operation, the linear combination u_i is sent to the activation function $f(\cdot)$ to obtain the corresponding output $x_i^{'}$. Note that the weights W_{ij} that connect to the same input unit x_j are controlled by the same random variable r_j.

During back-propagation, we should compute the gradients with respect to each unit to propagate the error. In Shakeout, $\frac{\partial u_i}{\partial x_j}$ takes the form

$$\frac{\partial u_i}{\partial x_j} = r_j(W_{ij} + cS_{ij}) - cS_{ij} \tag{3.13}$$

And the weights are updated following

$$\frac{\partial u_i}{\partial W_{ij}} = x_j(r_j + c(r_j - 1)\frac{dS_{ij}}{dW_{ij}}) \tag{3.14}$$

where $\frac{dS_{ij}}{dW_{ij}}$ represents the derivative of a sgn function. Because the sgn function is not continuous at zero and thus the derivative is not defined, we approximate this derivative with $\frac{d\tanh(W_{ij})}{dW_{ij}}$. Empirically we find that this approximation works well.

Note that the forward-backward computations with Shakeout can be easily extended to the convolutional layer. For a convolutional layer, the Shakeout feed-

forward process can be formalized as

$$\mathbf{U}_i = \sum_j (\mathbf{X}_j \circ \mathbf{R}_j) * \mathbf{W}_{ij} + c(\mathbf{X}_j \circ (\mathbf{R}_j - 1)) * \mathbf{S}_{ij} + b_i \tag{3.15}$$

$$\mathbf{X}_i^{'} = f(\mathbf{U}_i) \tag{3.16}$$

where $\mathbf{X}_j$ represents the j-th feature map. $\mathbf{R}_j$ is the j-th random mask which has the same spatial structure (i.e. the same height and width) as the corresponding feature map $\mathbf{X}_j$. $\mathbf{W}_{ij}$ denotes the kernel connecting $\mathbf{X}_j$ and $\mathbf{U}_i$. And $\mathbf{S}_{ij}$ is set as $\text{sgn}(\mathbf{W}_{ij})$. The symbol * denotes the convolution operation. And the symbol $\circ$ means element-wise product.

Correspondingly, during the back-propagation process, the gradient with respect to a unit of the layer on which Shakeout is applied takes the form

$$\begin{aligned}\frac{\partial \mathbf{U}_i(a,b)}{\partial \mathbf{X}_j(a-a^{'},b-b^{'})} = \mathbf{R}_j(a-a^{'},b-b^{'})(\mathbf{W}_{ij}(a^{'},b^{'}) + \\ c\mathbf{S}_{ij}(a^{'},b^{'})) - c\mathbf{S}_{ij}(a^{'},b^{'})\end{aligned} \tag{3.17}$$

where (a,b) means the position of a unit in the output feature map of a layer, and $(a^{'},b^{'})$ represents the position of a weight in the corresponding kernel.

The weights are updated following

$$\begin{aligned}\frac{\partial \mathbf{U}_i(a,b)}{\partial \mathbf{W}_{ij}(a^{'},b^{'})} = \mathbf{X}_j(a-a^{'},b-b^{'})(\mathbf{R}_j(a-a^{'},b-b^{'}) \\ + c(\mathbf{R}_j(a-a^{'},b-b^{'})-1)\frac{d\mathbf{S}_{ij}(a^{'},b^{'})}{d\mathbf{W}_{ij}(a^{'},b^{'})})\end{aligned} \tag{3.18}$$

3.4 Experiments

In this section, we report empirical evaluations of Shakeout in training deep neural networks on representative datasets. The experiments are performed on three kinds of image datasets: the hand-written image dataset MNIST [56], the CIFAR-10 image dataset [54] and the ImageNet-2012 dataset [77]. MNIST consists of

60,000+10,000 (training+test) 28×28 images of hand-written digits. CIFAR-10 contains 50,000+10,000 (training+test) 32×32 images of 10 object classes. ImageNet-2012 consists of 1,281,167+50,000+150,000 (training+validation+test) variable-resolution images of 1000 object classes. We first demonstrate that Shakeout leads to sparse models as our theoretical analysis implies under the unsupervised setting. Then we show that for the classification task, the sparse models have desirable generalization performances. Further, we illustrate the regularization effect of Shakeout on the weights in the classification task. Moreover, the effect of Shakeout on stabilizing the training processes of the deep architectures is demonstrated. Finally, we give some practical recommendations of Shakeout. All the experiments are implemented based on the modifications of *Caffe* library [46]. Our code is released on the github: https://github.com/kgl-prml/shakeout-for-caffe.

3.4.1 Shakeout and Weight Sparsity

Since Shakeout implicitly imposes L_0 penalty and L_1 penalty of the weights, we expect the weights of neural networks learned by Shakeout contain more zeros than those learned by the standard back-propagation (BP) [111] or Dropout [41]. In this experiment, we employ an autoencoder model for the MNIST hand-written data, train the model using standard BP, Dropout and Shakeout, respectively, and compare the degree of sparsity of the weights of the learned encoders. For the purpose of demonstration, we employ the simple autoencoder with one hidden layer of 256 units; Dropout and Shakeout are applied on the input pixels.

To verify the regularization effect, we compare the weights of the four autoencoders trained under different settings which correspond to standard BP, Dropout ($\tau = 0.5$) and Shakeout ($\tau = 0.5$, $c = \{1, 10\}$). All the training methods aim to produce hidden units which can capture good visual features of the handwritten digits. The statistical traits of these different resulting weights are shown in Fig.

3.4. Moreover, Fig. 3.5 shows the features captured by each hidden unit of the autoencoders.

As shown in the Fig. 3.4, the probability density of weights around the zero obtained by standard BP training is quite small compared to the one obtained either by Dropout or Shakeout. This indicates the strong regularization effect induced by Dropout and Shakeout. Furthermore, the sparsity level of weights obtained from training by Shakeout is much higher than the one obtained from training by Dropout. Using the same τ, increasing c makes the weights much sparser, which is consistent with the characteristics of L_0 penalty and L_1 penalty induced by Shakeout. Intuitively, we can find that due to the induced L_2 regularization, the distribution of weights obtained from training by the Dropout is like a Gaussian, while the one obtained from training by Shakeout is more like a Laplacian because of the additionally induced L_1 regularization. Fig. 3.5 shows that features captured by the hidden units via standard BP training are not directly interpretable, corresponding to insignificant variants in the training data. Both Dropout and Shakeout suppress irrelevant weights by their regularization effects, where Shakeout produces much sparser and more global features thanks to the combination of L_0, L_1 and L_2 regularization terms.

The autoencoder trained by Dropout or Shakeout can be viewed as the denosing autoencoder, where Dropout or Shakeout injects special kind of noise into the inputs. Under this unsupervised setting, the denoising criterion (i.e. minimizing the error between imaginary images reconstructed from the noisy inputs and the real images without noise) is to guide the learning of useful high level feature representations [103, 104]. To verify that Shakeout helps learn better feature representations, we adopt the hidden layer activations as features to train SVM classifiers, and the classification accuracies on test set for standard BP, Dropout and Shakeout are 95.34%, 96.41% and 96.48%, respectively. We can see that Shakeout leads to much

sparser weights without defeating the main objective.

Gaussian Dropout has similar effect on the model training as standard Dropout [89], which multiplies the activation of each unit by a Gaussian variable with mean 1 and variance σ^2. The relationship between σ^2 and τ is that $\sigma^2 = \frac{\tau}{1-\tau}$. The distribution of the weights trained by Gaussian Dropout ($\sigma^2 = 1$, i.e. $\tau = 0.5$) is illustrated in Fig. 3.4. From Fig. 3.4, we find no notable statistical difference between two kinds of Dropout implementations which all exhibit a kind of L_2 regularization effect on the weights. The classification performances of SVM classifiers on test set based on the hidden layer activations as extracted features for both kinds of Dropout implementations are quite similar (i.e. 96.41% and 96.43% for standard and Gaussian Dropout respectively). Due to these observations, we conduct the following classification experiments using standard Dropout as a representative implementation (of Dropout) for comparison.

3.4.2 Classification Experiments

Sparse models often indicate lower complexity and better generalization performance [98, 122, 70, 113]. To verify the effect of L_0 and L_1 regularization terms induced by Shakeout on the model performance, we apply Shakeout, along with Dropout and standard BP, on training representative deep neural networks for classification tasks. In all of our classification experiments, the hyper-parameters τ and c in Shakeout, and the hyper-parameter τ in Dropout are determined by validation.

MNIST

We train two different neural networks, a shallow fully-connected one and a deep convolutional one. For the fully-connected neural network, a big hidden layer size is adopted with its value at 4096. The non-linear activation unit adopted is the rectifier linear unit (ReLU). The deep convolutional neural network employed

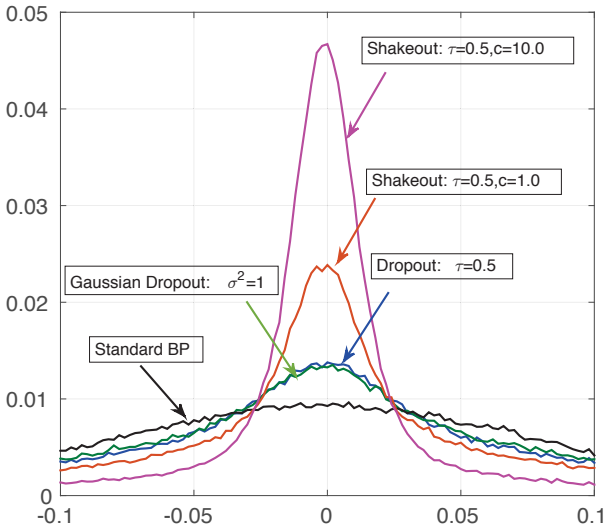
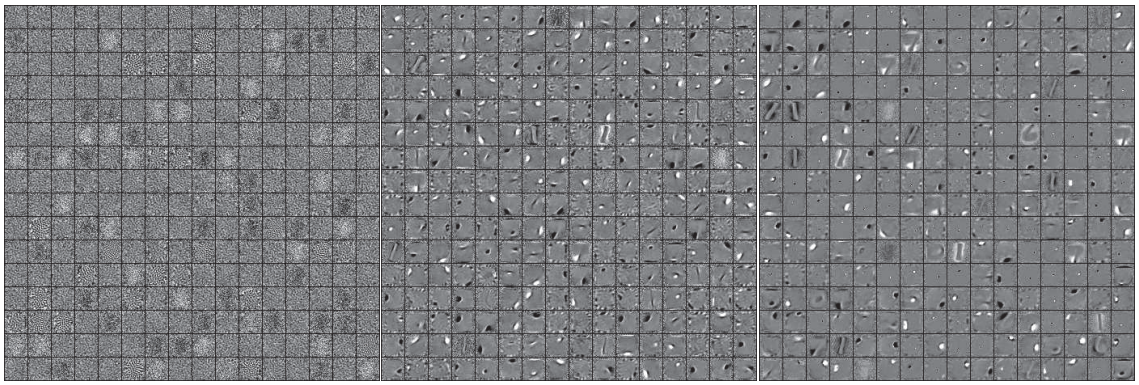


Figure 3.4 : Distributions of the weights of the autoencoder models learned by different training approaches. Each curve in the figure shows the frequencies of the weights of an autoencoder taking particular values, i.e. the empirical population densities of the weights. The five curves correspond to five autoencoders learned by standard back-propagation, Dropout ($\tau = 0.5$), Gaussian Dropout ($\sigma^2 = 1$) and Shakeout ($\tau = 0.5$, $c = \{1, 10\}$). The sparsity of the weights obtained via Shakeout can be seen by comparing the curves.



(a) standard BP (b) Dropout: $\tau = 0.5$ (c) Shakeout: $\tau = 0.5$, $c = 0.5$

Figure 3.5 : Features captured by the hidden units of the autoencoder models learned by different training methods. The features captured by a hidden unit are represented by a group of weights that connect the image pixels with this corresponding hidden unit. One image patch in a sub-graph corresponds to the features captured by one hidden unit.

is based on the modifications of the LeNet [56], which contains two convolutional layers and two fully-connected layers. The detailed architecture information of this convolutional neural network is described in Tab. 3.1. We separate 10,000 training samples from original training dataset for validation. The results are shown in Tab. 3.2 and Tab. 3.3. Dropout and Shakeout are applied on the hidden units of the fully-connected layer. The table compares the errors of the networks trained by standard back-propagation, Dropout and Shakeout. The mean and standard deviation of the classification errors are obtained by 5 runs of the experiment and are shown in percentage. We can see from the results that when the training data is not sufficient enough, due to over-fitting, all the models perform worse. However, the models trained by Dropout and Shakeout consistently perform better than the one trained by standard BP. Moreover, when the training data is scarce, Shakeout leads to superior model performance compared to the Dropout. Fig. 3.6 shows the

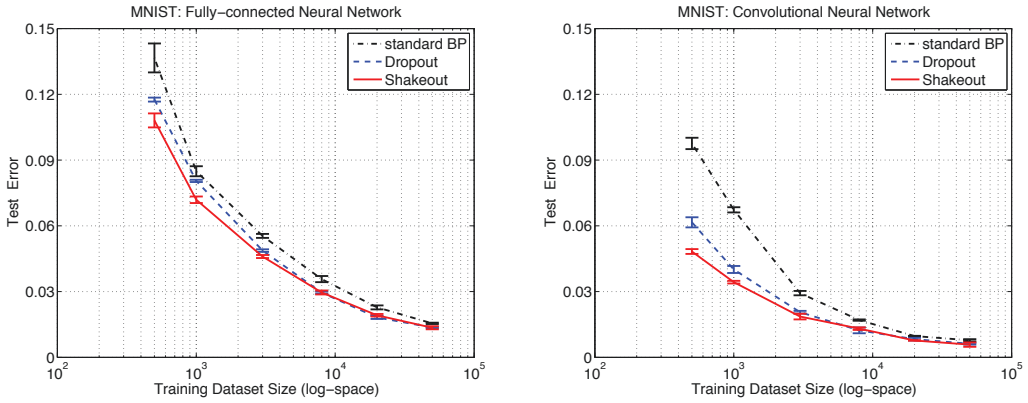
Layer	1	2	3	4
Type	conv.	conv.	FC	FC
Channels	20	50	500	10
Filter size	5×5	5×5	-	-
Conv. stride	1	1	-	-
Pooling type	max	max	-	-
Pooling size	2×2	2×2	-	-
Pooling stride	2	2	-	-
Non-linear	ReLU	ReLU	ReLU	Softmax

Table 3.1 : The architecture of convolutional neural network adopted for MNIST classification experiment

Size	std-BP	Dropout	Shakeout
500	13.66±0.66	11.76±0.09	**10.81**±0.32
1000	8.49±0.23	8.05±0.05	**7.19**±0.15
3000	5.54±0.09	4.87±0.06	**4.60**±0.07
8000	3.57±0.14	**2.95**±0.05	2.96±0.09
20000	2.28±0.09	**1.82**±0.07	1.92±0.06
50000	1.55±0.03	1.36±0.03	**1.35**±0.07

Table 3.2 : Classification on MNIST using training sets of different sizes: fully-connected neural network

results in a more intuitive way.



(a) Fully-connected neural network (b) Convolutional neural network

Figure 3.6 : Classification of two kinds of neural networks on MNIST using training sets of different sizes. The curves show the performances of the models trained by standard BP, and those by Dropout and Shakeout applied on the hidden units of the fully-connected layer.

CIFAR-10

We use the simple convolutional network feature extractor described in cuda-convnet (layers-80sec.cfg) [53]. We apply Dropout and Shakeout on the first fully-connected layer. We call this architecture "AlexFastNet" for the convenience of description. In this experiment, 10,000 colour images are separated from the training dataset for validation and no data augmentation is utilized. The per-pixel mean computed over the training set is subtracted from each image. We first train for 100 epochs with an initial learning rate of 0.001 and then another 50 epochs with the learning rate of 0.0001. The mean and standard deviation of the classification errors are obtained by 5 runs of the experiment and are shown in percentage. As shown in Tab. 3.4, the performances of models trained by Dropout and Shakeout are consistently superior to the one trained by standard BP. Furthermore, the model trained by Shakeout also outperforms the one trained by Dropout when the training data is scarce. Fig. 3.7 shows the results in a more intuitive way.

Size	std-BP	Dropout	Shakeout
500	9.76±0.26	6.16±0.23	**4.83**±0.11
1000	6.73±0.12	4.01±0.16	**3.43**±0.06
3000	2.93±0.10	2.06±0.06	**1.86**±0.13
8000	1.70±0.03	**1.23**±0.13	1.31±0.06
20000	0.97±0.01	0.83±0.06	**0.77**±0.001
50000	0.78±0.05	0.62±0.04	**0.58**±0.10

Table 3.3 : Classification on MNIST using training sets of different sizes: convolutional neural network

Size	std-BP	Dropout	Shakeout
300	68.26±0.57	65.34±0.75	**63.71**±0.28
700	59.78±0.24	56.04±0.22	**54.66**±0.22
2000	50.73±0.29	46.24±0.49	**44.39**±0.41
5500	41.41±0.52	36.01±0.13	**34.54**±0.31
15000	32.53±0.25	27.28±0.26	**26.53**±0.17
40000	24.48±0.23	**20.50**±0.32	20.56±0.12

Table 3.4 : Classification on CIFAR-10 using training sets of different sizes: Alex-FastNet

To test the performance of Shakeout on a much deeper architecture, we also conduct experiments based on the Wide Residual Network (WRN) [114]. The configuration of WRN adopted is WRN-16-4, which means WRN has 16 layers in total and the number of feature maps for the convolutional layer of each residual block is 4 times as the corresponding original one [37]. Because the complexity is much higher

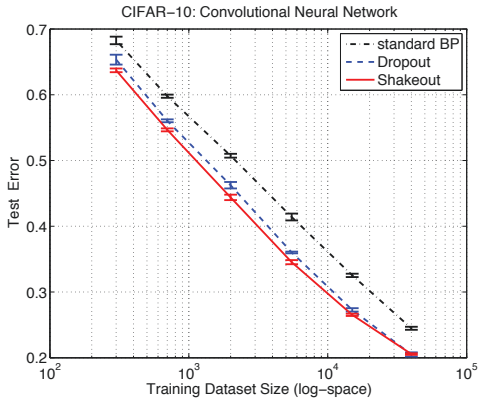
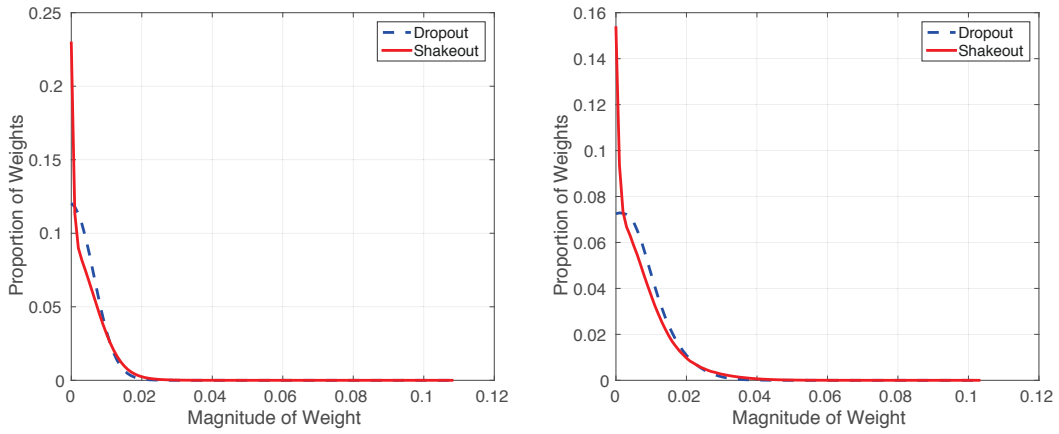


Figure 3.7 : Classification on CIFAR-10 using training sets of different sizes. The curves show the performances of the models trained by standard BP, and those by Dropout and Shakeout applied on the hidden units of the fully-connected layer.

than that of "AlexFastNet", the experiments are performed on relatively larger training sets with sizes of 15000, 40000, 50000. Dropout and Shakeout are applied on the second convolutional layer of each residual block, following the protocol in [114]. All the training starts from the same initial weights. Batch Normalization is applied the same way as [114] to promote the optimization. No data-augmentation or data pre-processing is adopted. All the other hyper-parameters other than τ and c are set the same as [114]. The results are listed in Tab. 3.5. For the training of CIFAR-10 with 50000 training samples, we adopt the same hyper-parameters as those chosen in the training with training set size at 40000. From Tab. 3.5, we can arrive at the same conclusion as previous experiments, i.e. the performances of the models trained by Dropout and Shakeout are consistently superior to the one trained by standard BP. Moreover, Shakeout outperforms Dropout when the data is scarce.

Size	std-BP	Dropout	Shakeout
15000	20.95	15.05	**14.68**
40000	15.37	9.32	**9.01**
50000	14.39	8.03	**7.97**

Table 3.5 : Classification on CIFAR-10 using training sets of different sizes: WRN-16-4



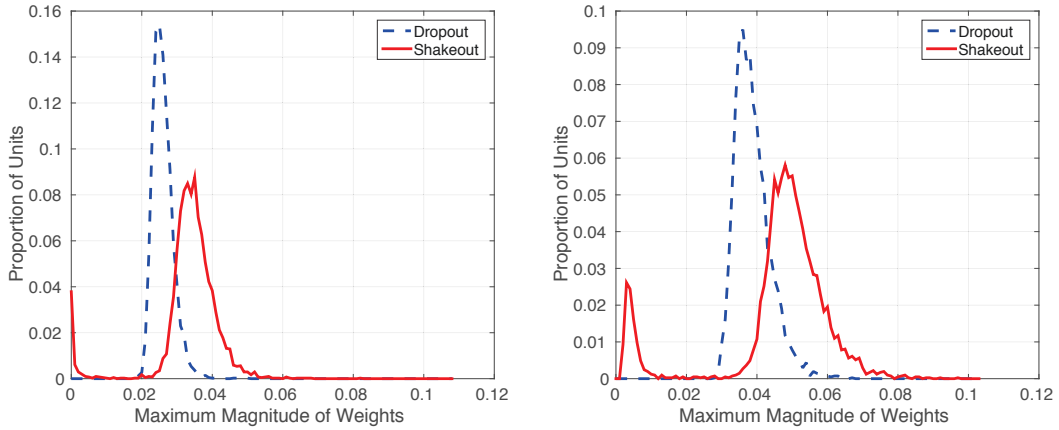
(a) AlexNet FC7 layer

(b) AlexNet FC8 layer

Figure 3.8 : Comparison of the distributions of the magnitude of weights trained by Dropout and Shakeout. The experiments are conducted using AlexNet on ImageNet-2012 dataset. Shakeout or Dropout is applied on the last two fully-connected layers, i.e. FC7 layer and FC8 layer.

Regularization Effect on the Weights

Shakeout is a different way to regularize the training process of deep neural networks from Dropout. For a GLM model, we have proved that the regularizer induced by Shakeout adaptively combines L_0, L_1 and L_1 regularization terms. In section 3.4.1, we have demonstrated that for a one-hidden layer autoencoder, it leads to much sparser weights of the model. In this section, we will illustrate the



(a) AlexNet FC7 layer

(b) AlexNet FC8 layer

Figure 3.9 : Distributions of the maximum magnitude of the weights connected to the same input unit of a layer. The maximum magnitude of the weights connected to one input unit can be regarded as a metric of the importance of that unit. The experiments are conducted using AlexNet on ImageNet-2012 dataset. For Shakeout, the units can be approximately separated into two groups and the one around zero is less important than the other, whereas for Dropout, the units are more concentrated.

regularization effect of Shakeout on the weights in the classification task and make a comparison to that of Dropout.

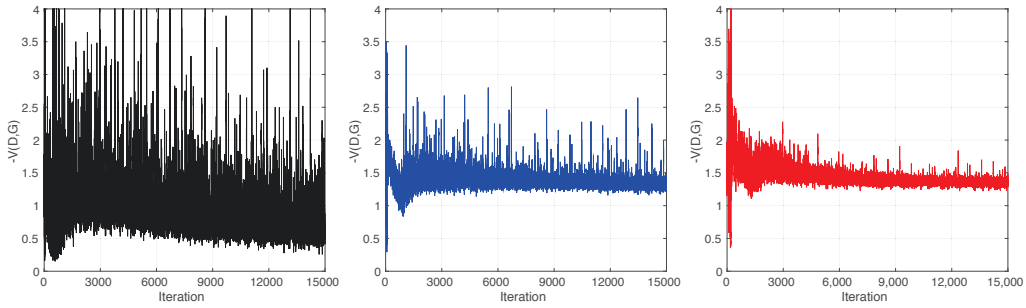
The results shown in this section are mainly based on the experiments conducted on ImageNet-2012 dataset using the representative deep architecture: AlexNet [55]. For AlexNet, we apply Dropout or Shakeout on layers FC7 and FC8 which are the last two fully-connected layers. We train the model from the scratch and obtain the comparable classification performances on validation set for Shakeout (top-1 error: 42.88%; top-5 error: 19.85%) and Dropout (top-1 error: 42.99%; top-5 error: 19.60%). The model is trained based on the same hyper-parameter settings provided by Shelhamer in *Caffe* [46] other than the hyper-parameters τ and c for Shakeout. The initial weights for training by Dropout and Shakeout are kept the same.

Fig. 3.8 illustrates the distributions of the magnitude of weight resulted by Shakeout and Dropout. It can be seen that the weights learned by Shakeout are much sparser than those learned by Dropout, due to the implicitly induced L_0 and L_1 components.

The regularizer induced by Shakeout not only contains L_0 and L_1 regularization terms but also contains L_2 regularization term, the combination of which is expected to discard a group of weights simultaneously. In Fig. 3.9, we use the maximum magnitude of the weights connected to one input unit of a layer to represent the importance of that unit for the subsequent output units. From Fig. 3.9, it can be seen that for Shakeout, the units can be approximately separated into two groups according to the maximum magnitudes of the connected weights and the group around zero can be discarded, whereas for Dropout, the units are concentrated. This implies that compared to Dropout which may encourage a "distributed code" for the features captured by the units of a layer, Shakeout tends to discard the useless features (or units) and award the important ones. This experiment result verifies the regularization properties of Shakeout and Dropout further.

As known to us, L_0 and L_1 regularization terms are related to performing feature selection [31, 107]. For a deep architecture, it is expected to obtain a set of weights using Shakeout suitable for reflecting the importance of connections between units. We perform the following experiment to verify this effect. After a model is trained, for the layer on which Dropout or Shakeout is applied, we sort the magnitudes of the weights increasingly. Then we prune the first $m\%$ of the sorted weights and evaluate the performance of the pruned model again. The pruning ratio m goes from 0 to 1. We calculate the relative accuracy loss (we write $R.A.L$ for simplification) at each pruning ratio $m^{'}$ as

$$R.A.L(m^{'}) = \frac{Accu.(m=0) - Accu.(m^{'})}{Accu.(m=0)}$$



(a) standard BP (b) Dropout (c) Shakeout

Figure 3.11 : The value of $-V(D,G)$ as a function of iteration for the training process of DCGAN. DCGANs are trained using standard BP, Dropout and Shakeout for comparison. Dropout or Shakeout is applied on the discriminator of GAN.

Fig. 3.10 shows the $R.A.L$ curves for Dropout and Shakeout based on the AlexNet model on ImageNet-2012 dataset. The models trained by Dropout and Shakeout are under the optimal hyper-parameter settings. Apparently, the relative accuracy loss for Dropout is more severe than that for Shakeout. For example, the largest margin of the relative accuracy losses between Dropout and Shakeout is 22.50%, which occurs at the weight pruning ratio $m = 96\%$. This result proves that considering the trained weights in reflecting the importance of connections, Shakeout is much better than Dropout, which benefits from the implicitly induced L_0 and L_1 regularization effect. This kind of property is useful for the popular compression task in deep learning area which aims to cut the connections or throw units of a deep neural network to a maximum extent without obvious loss of accuracy. The above experiments illustrate that Shakeout can play a considerable role in selecting important connections, which is meaningful for promoting the performance of a compression task. This is a potential subject for the future research.

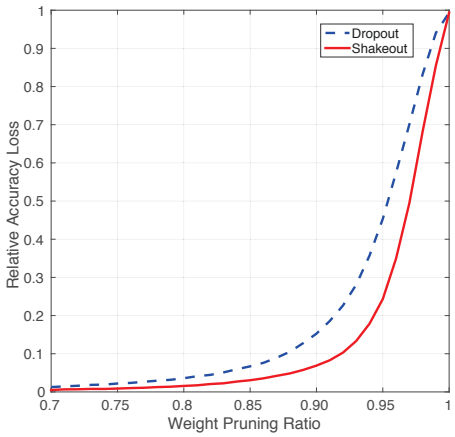


Figure 3.10 : Relative accuracy loss as a function of the weight pruning ratio for Dropout and Shakeout based on AlexNet architecture on ImageNet-2012. The relative accuracy loss for Dropout is much severe than that for Shakeout. The largest margin of the relative accuracy losses between Dropout and Shakeout is 22.50%, which occurs at the weight pruning ratio $m = 96\%$.

3.4.3 Stabilization Effect on the Training Process

In both research and production, it is always desirable to have a level of certainty about how a model's fitness to the data improves over optimization iterations, namely, to have a *stable* training process. In this section, we show that Shakeout helps reduce fluctuation in the improvement of model fitness during training.

The first experiment is on the family of Generative Adversarial Networks (GANs) [28], which is known to be instable in the training stage [74, 2, 3]. The purpose of the following tests is to demonstrate the Shakeout's capability of stabilizing the training process of neural networks in a general sense. GAN plays a min-max game between the generator G and the discriminator D over the expected log-likelihood of real data $\boldsymbol{x}$ and imaginary data $\hat{\boldsymbol{x}} = G(\boldsymbol{z})$ where $\boldsymbol{z}$ represents the random input

$$\min_G \max_D V(D, G) = \mathbb{E}[\log[D(\boldsymbol{x})] + \log[1 - D(G(\boldsymbol{z}))]] \tag{3.19}$$

The architecture that we adopt is DCGAN [74]. The numbers of feature maps of

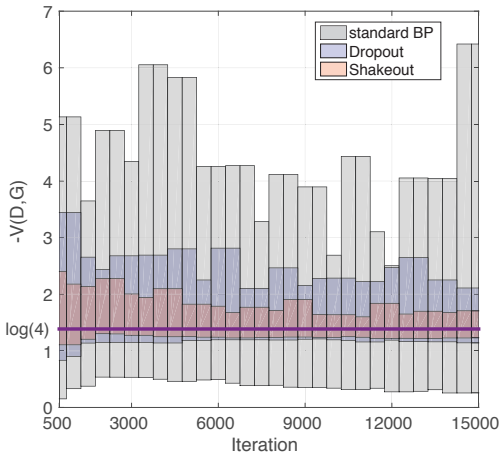


Figure 3.12 : The minimum and maximum values of $-V(D, G)$ within fixed length intervals moving from the start to the end of the training by standard BP, Dropout and Shakeout. The optimal value log(4) is obtained when the imaginary data distribution $P(\hat{\boldsymbol{x}})$ matches with the real data distribution $P(\boldsymbol{x})$.

the deconvolutional layers in the generator are 1024, 64 and 1 respectively, with the corresponding spatial sizes 7×7, 14×14 and 28×28. We train DCGANs on MNIST dataset using standard BP, Dropout and Shakeout. We follow the same experiment protocol described in [74] except for adopting Dropout or Shakeout on all layers of the discriminator. The values of $-V(D, G)$ during training are illustrated in Fig. 3.11. It can be seen that $-V(D, G)$ during training by standard BP oscillates greatly, while for Dropout and Shakeout, the training processes are much steadier. Compared with Dropout, the training process by Shakeout has fewer spikes and is smoother. Fig. 3.12 demonstrates the minimum and maximum values of $-V(D, G)$ within fixed length intervals moving from the start to the end of the training by standard BP, Dropout and Shakeout. It can be seen that the gaps between the minimum and maximum values of $-V(D, G)$ trained by Dropout and Shakeout are much smaller than that trained by standard BP, while that by Shakeout is the smallest, which implies the stability of the training process by Shakeout is the best.

The second experiment is based on Wide Residual Network architecture to perform the classification task. In the classification task, generalization performance is the main focus and thus, we compare the validation errors during the training processes by Dropout and Shakeout. Fig. 3.13 demonstrates the validation error as a function of the training epoch for Dropout and Shakeout on CIFAR-10 with 40000 training examples. The architecture adopted is WRN-16-4. The experiment settings are the same as those described in Section 3.4.2. Considering the generalization performance, the learning rate schedule adopted is the one optimized through validation to make the models obtain the best generalization performances. Under this schedule, we find that the validation error temporarily increases when lowering the learning rate at the early stage of training, which has been repeatedly observed by [114]. Nevertheless, it can be seen from Fig. 3.13 that the extent of error increase is less severe for Shakeout than Dropout. Moreover, Shakeout recovers much faster than Dropout does. At the final stage, both of the validation errors steadily decrease. Shakeout obtains comparable or even superior generalization performance to Dropout. In a word, Shakeout significantly stabilizes the entire training process with superior generalization performance.

3.4.4 Practical Recommendations

Selection of Hyper-parameters The most practical and popular way to perform hyper-parameter selection is to partition the training data into a training set and a validation set to evaluate the classification performance of different hyper-parameters on it. Due to the expensive cost of time for training a deep neural network, cross-validation is barely adopted. There exist many hyper-parameter selection methods in the domain of deep learning, such as the grid search, random search [10], Bayesian optimization methods [88], gradient-based hyper-parameter Optimization [66], etc. For applying Shakeout on a deep neural network, we need to

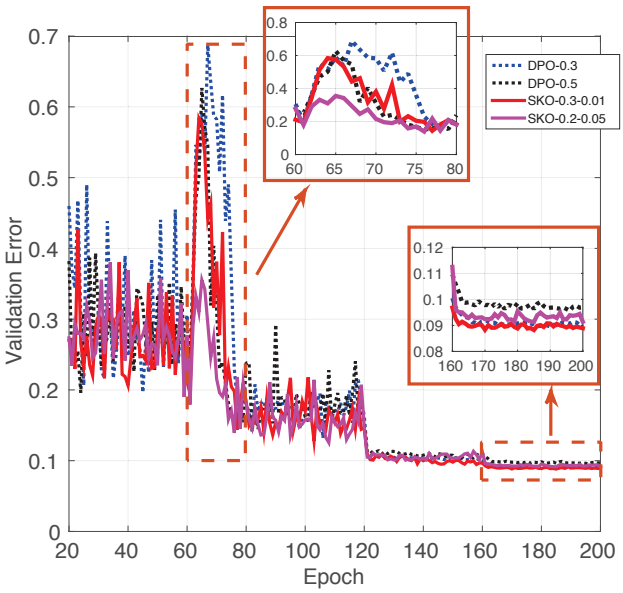


Figure 3.13 : Validation error as a function of training epoch for Dropout and Shakeout on CIFAR-10 with training set size at 40000. The architecture adopted is WRN-16-4. "DPO" and "SKO" represent "Dropout" and "Shakeout" respectively. The following two numbers denote the hyper-parameters τ and c respectively. The learning rate decays at epoch 60, 120, and 160. After the first decay of learning rate, the validation error increases greatly before the steady decrease (see the enlarged snapshot for training epochs from 60 to 80). It can be seen that the extent of error increase is less severe for Shakeout than Dropout. Moreover, Shakeout recovers much faster than Dropout does. At the final stage, both of the validation errors steadily decrease (see the enlarged snapshot for training epochs from 160 to 200). Shakeout obtains comparable or even superior generalization performance to Dropout.

decide two hyper-parameters τ and c. From the regularization perspective, we need to decide the most suitable strength of regularization effect to obtain an optimal trade-off between model bias and variance. We have pointed out that in a unified framework, Dropout is a special case of Shakeout when Shakeout hyper-parameter c is set to zero. Empirically we find that the optimal τ for Shakeout is not higher

than that for Dropout. After determining the optimal τ, keeping the order of magnitude of hyper parameter c the same as $\sqrt{\frac{1}{N}}$ (N represents the number of training samples) is an effective choice. If you want to obtain a model with much sparser weights but meanwhile with superior or comparable generalization performance to Dropout, a relatively lower τ and larger c for Shakeout always works.

Shakeout combined with Batch Normalization Batch Normalization [45] is the widely-adopted technique to promote the optimization of the training process for a deep neural network. In practice, combining Shakeout with Batch Normalization to train a deep architecture is a good choice. For example, we observe that the training of WRN-16-4 model on CIFAR-10 is slow to converge without using Batch Normalization in the training. Moreover, the generalization performance on the test set for Shakeout combined with Batch Normalization always outperforms that for standard BP with Batch Normalization consistently for quite a large margin, as illustrated in Tab. 3.5. These results imply the important role of Shakeout in reducing over-fitting of a deep neural network.

3.5 Conclusion

We have proposed Shakeout, which is a new regularized training approach for deep neural networks. The regularizer induced by Shakeout is proved to adaptively combine L_0, L_1 and L_2 regularization terms. Empirically we find that

1) Compared to Dropout, Shakeout can afford much larger models. Or to say, when the data is scarce, Shakeout outperforms Dropout with a large margin.

2) Shakeout can obtain much sparser weights than Dropout with superior or comparable generalization performance of the model. While for Dropout, if one wants to obtain the same level of sparsity as that obtained by Shakeout, the model may bear a significant loss of accuracy.

3) Some deep architectures in nature may result in the instability of the training process, such as GANs, however, Shakeout can reduce this instability effectively.

In future, we want to put emphasis on the inductive bias of Shakeout and attempt to apply Shakeout to the compression task.

Chapter 4

Regularization for Unsupervised Domain Adaptation

4.1 Introduction

This chapter focuses on unsupervised domain adaptation (UDA) for visual classification task. We aim to adapt the knowledge from a source network, trained by the source domain data, to the training of a target network, which will be used for making predications in the target domain. Note that in UDA the *target domain is unlabeled*. The increasing popularity of UDA arises from the fact that the performance of a model trained on one domain may degenerate heavily on another when their underlying data distributions are different.

In the community of UDA, many deep learning methods attempt to minimize the discrepancy across domains on the top layers, such as the fully connected layers, of the neural network via explicitly imposing penalty terms [101, 62, 63, 91] or in an adversarial way [25, 100, 99]. While the modifications at the fully connected layers can be back-propagated in principle, it may decay after a few layers, especially when gradient explosion or vanishing takes place. Consequently, the convolutional layers may be under-constrained. However, the domain discrepancy may emerge at the start from the convolutional layers, which makes any adjustment purely at the tail of the network less effective.

We investigate the domain discrepancy of the convolutional layers by visualizing their attention mechanisms. In essence, the attention mechanism is emphasized as a key ingredient for CNN, suggested by a number of studies [85, 116, 120, 83,

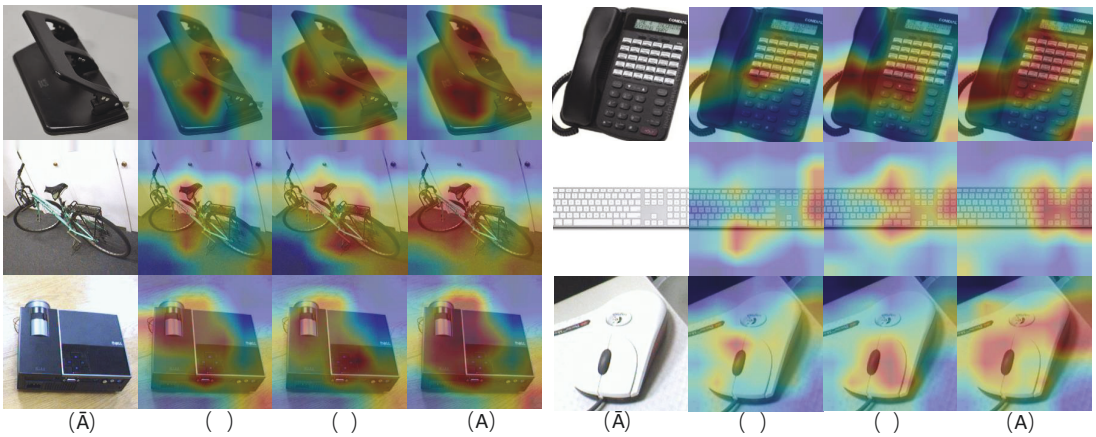


Figure 4.1 : Attention visualization of the last convolutional layer of ResNet-50. The original *target* input images are illustrated in (**a**). The corresponding attentions of the source network, the target network trained on labeled target data, and the target network adapted with adversarial attention alignment are shown in (**b**), (**c**), and (**d**) respectively.

115, 109, 117]. Zagoruyko *et al.* [115] find that the model performance is highly correlated with the attention mechanism: a stronger model always owns better aligned attention than a weaker one. From Fig. 4.1, suppose we have networks trained on labeled data from source and target domains respectively, we observe distinct attention patterns exhibited by the convolutional layers for the same target domain image. The attention mechanism degenerates when directly applying the source network to the target domain data, which may exert negative influence on the classification performance. Therefore, we expect the attention of the convolutional layers to be *invariant to the domain shift*.

Based on the above discussions, this chapter takes the domain discrepancy of the convolutional layers directly into account by aligning the attention of the target network with the source network. Our assumption is that no matter how domain varies, the discriminative parts of an image should be insensitive to the changes of image style. Previous discrepancy measures (*e.g.*, MMD [62] and JMMD [63]) which work

effectively on high-level *semantic* representations cannot be trivially transferred to measure the attention discrepancy of the convolutional layers where low-level *structure* information is critical. In this chapter, we propose using CycleGAN [121] to build the data correspondence across domains, *i.e.*, translating the data from one domain to another without modifying its underlying content. Then, for the paired samples (*e.g.* real source (or target) image and synthetic target (or source) image), we explicitly penalize the distances between attentions of the source and the target networks.

Additionally, we train our target network with real and synthetic data from both source and target domains. For source domain and its translated data, we impose the cross-entropy loss between the predictions and the ground-truth labels. For target domain and its translated source domain data, due to the lack of ground-truth labels, we make use of their underlying category distributions which provide insight into the target data. In a nutshell, we adopt the modified Expectation Maximization (EM) steps to maximize the likelihood of target domain images and update the model. Training iterations improve both the label posterior distribution estimation and the discriminative ability of the model.

Our contributions are summarized below,

- We propose a deep attention alignment method which allows the target network to mimic the attention of the source network. Taking advantage of the pairing nature of CycleGAN, no additional supervision is needed.
- We propose using EM algorithm to exploit the unlabeled target data to update the network. Several modifications are made to stabilize training and improve the adaptation performance.
- Our method outperforms the state-of-the-art in all the six transfer tasks,

achieving +2.6% improvement in average on the real-world domain adaptation dataset Office-31.

4.2 Related Work

Unsupervised domain adaptation. Various methods have been proposed for unsupervised domain adaptation [101, 62, 25, 63]. Many works try to make the representations at the tail of neural networks invariant across domains. Tzeng *et al.* [101] propose a kind of domain confusion loss to encourage the network to learn both semantically meaningful and domain invariant representations. Similarly, Long *et al.* [62] minimize the MMD distance of the fully-connected activations between source and target domain while sharing the convolutional features. Ganin *et al.* [25] enable the network to learn domain invariant representations in an adversarial way by adding a domain classifier and back-propagating inverse gradients. JAN [63] penalizes the JMMD over multiple fully-connected layers to minimize the domain discrepancy coming from both the data distribution and the label distribution. Further, JAN-A [63], as a variant of JAN, trains the network in an adversarial way with JMMD as the domain adversary. DSN [14] explicitly models domain-specific features to help improve networks' ability to learn domain-invariant features. Associative domain adaptation (ADA) [32] reinforces associations across domains directly in embedding space to extract statistically domain-invariant and class discriminative features. Few works pay attention to the domain shift coming from the convolutional layers. In this chapter, we notice that the attention mechanism cannot be preserved when directly applying the model trained on the source domain to the target domain. To alleviate this problem, we constrain the training of convolutional layers by imposing the attention alignment penalty across domains.

Attention of CNNs. There exist many ways to define and visualize the attention mechanisms learned by CNNs. Zeiler & Fergus [116] project certain features

back onto the image through a network called "deconvnet" which shares the same weights as the original feed-forward network. Simonyan *et al.* [85] propose using the gradient of the class score *w.r.t* the input image to visualize the CNN. Class activation maps (CAMs), proposed by [120], aim to visualize the class-discriminative image regions used by a CNN. Grad-CAM [83] combines gradient based attention method and CAM, enabling to obtain class-discriminative attention maps without modifying the original network structure as [120].

Zagoruyko *et al.* [115] define attention as a set of spatial maps indicating which area the network focuses on to perform a certain task. The attention maps can also be defined *w.r.t* various layers of the network so that they are able to capture both low-, mid-, and high-level representation information. They propose that attention mechanism should be a kind of knowledge transferred across different *network architectures*. Zaogruyko *et al.* [115] align the attention across different architectures for exactly the same image during the training process and aim to transfer the knowledge from a large model to a smaller one. Different to [115], our method aligns the attention across different *data domains* where images across domains are unpaired and aims to promote the model adaptation performance.

Unpaired image-to-image translation. Unpaired image-to-image translation aims to train a model to map image samples across domains, under the absence of pairing information. It can be realized through GAN to pair the real source (or target) and synthetic target (or source) images [60, 84, 121, 52, 59, 13, 43, 78]. Generating synthetic images can be beneficial for various vision tasks [65, 119, 22, 21]. In this chapter, we concentrate on maximizing the utility of given paired real and synthetic samples. And we choose CycleGAN [121] to perform such adversarial data pairing.

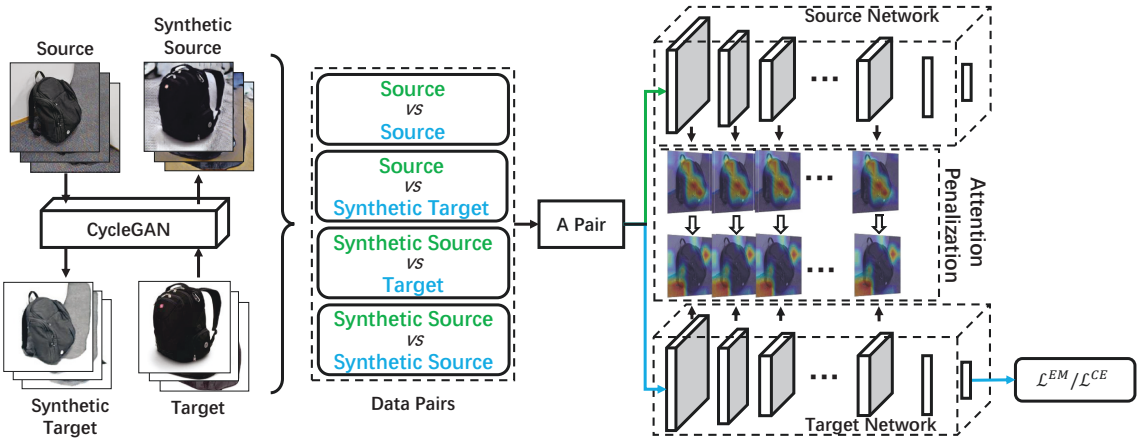


Figure 4.2 : The framework of deep adversarial attention alignment. We train a source network and fix it. The source network guides the attention alignment of the target network. The target network is trained with real and synthetic images from both domains. For labeled real source and synthetic target data, we update the network by computing the cross-entropy loss between the predictions and the ground-truth labels. For unlabeled real target and synthetic source images, we maximize the likelihood of the data with EM steps. The attention distance for a pair of images (as illustrated in the "Data Pairs" block) passing through the source network and the target network, respectively, is minimized.

4.3 Method

Our framework is illustrated in Fig. 4.2. We train a source CNN which guides the attention alignment of the target CNN whose convolutional layers have the same architecture as the source network. The target CNN is trained with a mixture of real and synthetic images from both source and target domains. For source and synthetic target domain data, we have ground-truth labels and use them to train the target network with cross-entropy loss. On the other hand, for the target and synthetic source domain data, due to the lack of ground-truth labels, we optimize the target network through an EM algorithm.

4.3.1 Adversarial Data Pairing

We use CycleGAN to translate the samples in the source domain S to those in the target domain T, and vice versa. The underlying assumption to obtain meaningful translation is that there exist some relationships between two domains. For unsupervised domain adaptation, the objects of interest across domains belong to the same set of category. So it is possible to use CycleGAN to map the sample in the source domain to that in the target domain while maintaining the underlying object-of-interest.

The Generative Adversarial Network (GAN) aims to generate synthetic images which are indistinguishable from real samples through an adversarial loss,

$$\mathcal{L}^{GAN}(G^{ST}, D^T, X^S, X^T) = \mathbb{E}_{x^T}[\log D^T(x^T)] + \mathbb{E}_{x^S}[1 - \log D^T(G^{ST}(x^S))], \quad (4.1)$$

where x^S and x^T are sampled from source domain S and target domain T, respectively. The generator G^{ST} mapping X^S to X^T strives to make its generated synthetic outputs $G^{ST}(x^S)$ indistinguishable from real target samples x^T for the domain discriminator D^T.

Because the training data across domains are unpaired, the translation from source domain to target domain is highly under-constrained. CycleGAN couples the adversarial training of this mapping with its inverse one, *i.e.* the mapping from S to T and that from T to S are learned concurrently. Moreover, it introduces a cycle consistency loss to regularize the training,

$$\mathcal{L}^{cyc}(G^{ST}, G^{TS}) = \mathbb{E}_{x^S}[\|G^{TS}(G^{ST}(x^S)) - x^S\|_1] + \mathbb{E}_{x^T}[\|G^{ST}(G^{TS}(x^T)) - x^T\|_1], \quad (4.2)$$

Formally, the full objective for CycleGAN is,

$$\mathcal{L}^{cyc}(G, F, D_X, D_Y) = \mathcal{L}^{GAN}(G^{ST}, D^T, X^S, X^T) + \mathcal{L}^{GAN}(G^{TS}, D^S, X^T, X^S) + \lambda \mathcal{L}^{cyc}(G^{ST}, G^{TS}), \quad (4.3)$$

(a) (b) (c) (d)

Figure 4.3 : Paired data across domains using CycleGAN. (**a**) and (**c**): real images sampled from source and target domain, respectively. (**b**): a synthetic target image paired with (**a**) through G^{ST}. (**d**): a synthetic source image paired with a real target image (**c**) through G^{TS}.

where the constant λ controls the strength of the cycle consistency loss. Through CycleGAN, we are able to translate an image in the source domain to that in the target domain in the context of our visual domain adaptation tasks (Fig. 4.3).

As illustrated in Fig. 4.1, the target model pays too much attention to the irrelevant background or less discriminative parts of the objects of interest. This attention misalignment will degenerate the model's performance. In this chapter, we propose to use the style-translated images as natural image correspondences to guide the attention mechanism of the target model to mimic that of the source model, to be detailed in Section 4.3.2.

4.3.2 Attention Alignment

Based on the paired images, we propose imposing the attention alignment penalty to reduce the discrepancy of attention maps across domains. Specifically, we represent attention as a function of spatial maps *w.r.t* each convolutional layer [115]. For the input x of a CNN, let the corresponding feature maps *w.r.t* layer l be represented by $F_l(x)$. Then, the attention map $A_l(x)$ *w.r.t* layer l is defined as

$$A_l(x) = \sum_c |F_{l,c}(x)|^2, \tag{4.4}$$

where $F_{l,c}(x)$ denotes the c-th channel of the feature maps. The operations in Eq. (4.4) are all element-wise. Alternative ways to represent the attention maps include $\sum_c |F_{l,c}|$, and $\max |F_{l,c}|$, *etc.* We adopt Eq. (4.4) to emphasize the salient parts of the feature maps.

We propose using the source network to guide the attention alignment of the target network, as illustrated in Fig. 4.2. We penalize the distance between the vectorized attention maps between the source and the target networks to minimize their discrepancy. In order to make the attention mechanism invariant to the domain shift, we train the target network with a mixture of real and synthetic data from both source and target domains.

Formally, the attention alignment penalty can be formulated as,

$$\begin{aligned}\mathcal{L}^{AT} = \sum_l \{ & \sum_i \| \frac{A_l^S(x_i^S)}{\|A_l^S(x_i^S)\|_2} - \frac{A_l^T(x_i^S)}{\|A_l^T(x_i^S)\|_2} \|_2 + \sum_j \| \frac{A_l^S(x_j^S)}{\|A_l^S(x_j^S)\|_2} - \frac{A_l^T(\tilde{x}_j^T)}{\|A_l^T(\tilde{x}_j^T)\|_2} \|_2 \\ & + \sum_m \| \frac{A_l^S(\tilde{x}_m^S)}{\|A_l^S(\tilde{x}_m^S)\|_2} - \frac{A_l^T(\tilde{x}_m^S)}{\|A_l^T(\tilde{x}_m^S)\|_2} \|_2 + \sum_n \| \frac{A_l^S(\tilde{x}_n^S)}{\|A_l^S(\tilde{x}_n^S)\|_2} - \frac{A_l^T(x_n^T)}{\|A_l^T(x_n^T)\|_2} \|_2 \}, \end{aligned} \quad (4.5)$$

where the subscript l denotes the layer and i, j denote the samples. The A_l^S and A_l^T represent the attention maps *w.r.t* layer l for the source network and the target network, respectively. x^S and x^T are real source and real target domain data, respectively. The synthetic target data $\tilde{x}_i^T$ and synthetic source data $\tilde{x}_n^S$ satisfy $\tilde{x}_i^T = G^{ST}(x_i^S)$ and $\tilde{x}_n^S = G^{TS}(x_n^T)$, respectively.

Through Eq. (4.5), the distances of attention maps for the paired images (*i.e.,* $(x_j^S, \tilde{x}_j^T)$ and $(x_n^T, \tilde{x}_n^S)$) are minimized. Moreover, we additionally penalize the attention maps of the same input (*i.e.,* x_i^S and $\tilde{x}_m^S$) passing through different networks. The attention alignment penalty $\mathcal{L}^{AT}$ allows the attention mechanism to be gradually adapted to the target domain, which makes the attention mechanism of the target network invariant to the domain shift.

Discussion. On minimizing the discrepancy across domains, our method shares

similar ideas with DAN [62] and JAN [63]. The difference is that our method works on the convolutional layers where the critical structure information is captured and aligned across domains; in comparison, DAN and JAN focus on the FC layers where high-level semantic information is considered. Another notable difference is that our method deals with the image-level differences through CycleGAN data pairing, whereas DAN and JAN consider the discrepancy of feature distributions.

In DAN and JAN, MMD and JMMD criteria are adopted respectively to measure the discrepancy of feature distributions across domains. Technically, MMD and JMMD can also be used as attention discrepancy measures. However, as to be shown in the experiment part, MMD and JMMD yield inferior performance to the L_2 distance enabled by adversarial data pairing in our method. The reason is that MMD and JMMD are distribution distance estimators: they map the attention maps to the Reproducing Kernel Hilbert Space (RKHS) and lose the structure information. So they are not suitable for measuring the attention discrepancy across domains.

4.3.3 Training with EM

To make full use of the available data (labeled and unlabeled), we train the target-domain model with a mixture of real and synthetic data from both source and target domains, as illustrated in Fig. 4.2. For the source and its translated synthetic target domain data, we compute the cross-entropy loss between the predictions and ground-truth labels to back-propagate the gradients through the target network. The cross-entropy loss for the source and corresponding synthetic target domain data can be formulated as follows,

$$\mathcal{L}^{CE} = -[\sum_{i} \log p_\theta(y_i^S|x_i^S) + \sum_{j} \log p_\theta(y_j^S|\tilde{x}_j^T)], \tag{4.6}$$

where $y^S \in \{1, 2, \cdots, K\}$ denotes the label for the source sample x^S and the translated synthetic target sample $\tilde{x}^T$. The probability $p_\theta(y|x)$ is represented by the

y-th output of the target network with parameters θ given the input image x. $\tilde{x}_j^T = G^{ST}(x_j^S)$.

For the unlabeled target data, due to the lack of labels, we employ the EM algorithm to optimize the target network. The EM algorithm can be split into two alternative steps: the (**E**)xpectation computation step and the expectation (**M**)aximization step. The objective is to maximize the log-likelihood of target data samples,

$$\sum_i \log p_\theta(x_i^T), \tag{4.7}$$

In image classification, our prior is that the target data samples belong to K different categories. We choose the underlying category $z_i \in \{1, 2, \cdots, K\}$ of each sample as the hidden variable, and the algorithm is depicted as follows (we omit the sample subscript and the target domain superscript for description simplicity).

***(i)* The Expectation step.** We first estimate $p_{\theta_{t-1}}(z|x)$ through,

$$p_{\theta_{t-1}}(z|x) = \frac{p_{\theta_{t-1}}(x|z)p(z)}{\sum_z p_{\theta_{t-1}}(x|z)p(z)}, \tag{4.8}$$

where the distribution $p_{\theta_{t-1}}(z|x)$ is modeled by the target network. θ_{t-1} is the parameters of the target-domain CNN at last training step $t-1$. We adopt the uniform distributions to depict $p(z)$ (*i.e.,* assuming the occurrence probabilities of all the categories are the same) and $p(x)$ (*i.e.,* assuming all possible image instantiations are distributed uniformly in the manifold of image gallery). In this manner, $p_{\theta_{t-1}}(z|x) = \alpha p_{\theta_{t-1}}(x|z)$ where α is a constant.

***(ii)* The Maximization step.** Based on the computed posterior $p_{\theta_{t-1}}(z|x)$, our objective is to update θ_t to improve the lower bound of Eq. (4.7),

$$\sum_z p_{\theta_{t-1}}(z|x) \log p_{\theta_t}(x|z) \tag{4.9}$$

Note that we omit $\sum_z p_{\theta_{t-1}}(z|x) \log p(z)$ because we assume $p(z)$ subjects to the uniform distribution which is irrelevant to θ_t. Also, because $p_\theta(z|x) = p_\theta(x|z)$, Eq.

(4.9) is equivalent to,

$$\sum_{z} p_{\theta_{t-1}}(z|x) \log p_{\theta_t}(z|x). \tag{4.10}$$

Moreover, we propose to improve the effectiveness and stability of the above EM steps through three aspects

A) Asynchronous update of $p(z|x)$. We adopt an independent network M^{post} to estimate $p(z|x)$ and update M^{post} asynchronously, *i.e.,* M^{post} synchronizes its parameters θ^{post} with the target network every N steps: $\theta_t^{post} = \theta_{\lfloor t/N \rfloor \times N}$. In this manner, we avoid the frequent update of $p(z|x)$ and make the training process much more stable.

B) Filtering the inaccurate estimates. Because the estimate of $p(z|x)$ is not accurate, we set a threshold p_t and discard the samples whose maximum value of $p(z|x)$ over z is lower than p_t.

C) Initializing the learning rate schedule after each update of M^{post}. To accelerate the target network adapting to the new update of the distribution $p(z|x)$, we choose to initialize the learning rate schedule after each update of M^{post}.

Note that for synthetic source data $\tilde{x}^S = G^{TS}(x^T)$, we can also apply the modified EM steps for training. Because G^{TS} is a definite mapping, we assume $p(z|\tilde{x}^S) = p(z|x^T)$.

To summarize, when using the EM algorithm to update the target network with target data and synthetic source data, we first compute the posterior $p(z|x^T)$ through network M^{post} which synchronizes with the target network every N steps. Then we minimize the loss,

$$\mathcal{L}^{EM} = -\{\sum_{i}\sum_{z_i} p_{\theta^{post}}(z_i|x_i^T) \log p_\theta(z_i|x_i^T) + \sum_{j}\sum_{z_j} p_{\theta^{post}}(z_j|x_j^T) \log p_\theta(z_j|\tilde{x}_j^S)\}. \tag{4.11}$$

In our experiment, we show that these modifications yield consistent improvement

over the basic EM algorithm.

4.3.4 Deep Adversarial Attention Alignment

Based on the above discussions, our full objective for training the target network can be formulated as,

$$\min_{\theta} \mathcal{L}^{full} = \mathcal{L}^{CE} + \mathcal{L}^{EM} + \beta \mathcal{L}^{AT} \tag{4.12}$$

where β determines the strength of the attention alignment penalty term $\mathcal{L}^{AT}$.

Discussion. Our approach mainly consists of two parts: attention alignment and EM training. On the one hand, attention alignment is crucial for the success of EM training. For EM training, there originally exists no constraint that the estimated hidden variable Z is assigned with the semantic meaning aligned with the ground-truth label, *i.e.* there may exist label shift or the data is clustered in an undesirable way. Training with labeled data (*e.g.* source and synthetic target data) and synchronizing θ^{post} with θ, the above issue can be alleviated. In addition, attention alignment further regularizes the training process by encouraging the network to focus on the desirable discriminative information.

On the other hand, EM benefits attention alignment by providing label distribution estimations for target data. EM approximately guides the attention of target network to fit the target domain statistics, while attention alignment regularizes the attention of target network to be not far from source network. These two seemingly adversarial counterparts cooperate to make the target network acquire the attention mechanism which is invariant to the domain shift.

Note that both parts are promoted by the use of adversarial data pairing which provides natural image correspondences to perform attention alignment. Thus our method is named "deep adversarial attention alignment".

4.4 Experiments

4.4.1 Setup

Datasets. We use the following two UDA datasets for image classification.

1) Digit datasets from **MNIST** [56] (60,000 training + 10,000 test images) to **MNIST-M** [25] (59,001 training + 90,001 test images). MNIST and MNIST-M are treated as the source domain and target domain, respectively. The images of MNIST-M are created by combining MNIST digits with the patches randomly extracted from color photos of BSDS500 [1] as their background.

2) **Office-31** is a standard benchmark for real-world domain adaptation tasks. It consists of 4,110 images subject to 31 categories. This dataset contains three distinct domains, 1) images which are collected from the Amazon website (**A**mazon domain), 2) web camera (**W**ebcam domain), and 3) digital SLR camera (**D**SLR domain) under different settings, respectively. The dataset is also imbalanced across domains, with 2,817 images in **A** domain, 795 images in **W** domain, and 498 images in **D** domain. We evaluate our algorithm for six transfer tasks across these three domains, including $\mathbf{A} \rightarrow \mathbf{W}$, $\mathbf{D} \rightarrow \mathbf{W}$, $\mathbf{W} \rightarrow \mathbf{D}$, $\mathbf{A} \rightarrow \mathbf{D}$, $\mathbf{D} \rightarrow \mathbf{A}$, and $\mathbf{W} \rightarrow \mathbf{A}$.

Competing methods. We compare our method with some representative and state-of-the-art approaches, including RevGrad [25], JAN [63], JAN-A [63], DSN [14] and ADA [32] which minimize domain discrepancy on the FC layers of CNN. We compare with the results of these methods reported in their published papers with identical evaluation setting. For the task MNIST $\rightarrow$ MNIST-M, we also compare with PixelDA [13], a state-of-the-art method on this task. Both CycleGAN and PixelDA transfer the source style to the target domain without modifying its content heavily. Therefore, PixelDA is an alternative way to generate paired images across domains and is compatible to our framework. We emphasize that a model capable of generating more genuine paired images will probably lead to higher accuracy using

our method. The investigation in this direction can be parallel and reaches beyond the scope of this chapter.

4.4.2 Implementation Details

MNIST → MNIST-M The source network is trained on the MNIST training set. When the source network is trained, it is fixed to guide the training of the target network. The target and the source network are made up of four convolutional layers, where the first three are for feature extraction and the last one acts as a classifier. We align the attention between the source and target network for the three convolutional layers. We adopt Adam to update our network and the initial learning rate is set to 0.001. For a mini-batch input data, we fix the proportions of real source data, synthetic target data, real target data and synthetic source data as 0.35, 0.15, 0.35, and 0.15, respectively, throughout the experiment. For EM training, we set the threshold $p_t = 1$ so that the network is learned with all the source and synthetic target data before the first update of M^{post}. We then set the threshold $p_t = 0.95$ afterwards.

Office-31 To make a fair comparison with the state-of-the-art domain adaptation methods [63], we adopt the ResNet-50 [36, 37] architecture to perform the adaptation tasks on Office-31 and we start from the model pre-trained on ImageNet [18]. We first fine-tune the model on the source domain data and fix it. The source model is then used to guide the attention alignment of the target network. The target network starts from the fine-tuned model and is gradually trained to adapt to the target domain data. We penalize the distances of the attention maps $w.r.t$ all the convolutional layers except for the first convolutional layer and the max-pooling layers. We follow the same learning rate schedule adopted in [63] throughout our experiment except that we initialize the learning rate schedule after each update of posterior estimation network M_{post} (see Section 4.3.3). For a mini-batch input data,

the proportions of real and synthetic data from both domains are set as the same with those in task MNIST $\rightarrow$ MNIST-M. For a mini-batch input data, we fix the proportions of real source data, synthetic target data, real target data and synthetic source data as 0.35, 0.15, 0.35, and 0.15 respectively, throughout our experiment. Threshold p_t for EM training is set as 0.95. We choose β through validation following the same protocol as [63].

In the experiment of Office-31, we do not penalize the distances between attention maps *w.r.t* the first convolutional layer and the max-pooling layers of ResNet-50, because 1) Attention of the first convolutional layer focuses on low-level details and is easily affected by noise. 2) The max-pooling layer does not have parameters (totally determined by the outputs of previous convolutional layer). So it is not necessary to additionally align its attention. 3) We empirically find that ignoring these layers when performing attention alignment brings no loss of accuracy but is more efficient in computation.

4.4.3 Evaluation

MNIST $\rightarrow$ MNIST-M. The classification results of transferring MNIST to MNIST-M are presented in Table 4.1. We arrive at four observations. First, our method outperforms a series of representative domain adaptation methods (*e.g.*, RevGrad, DSN, ADA) with a large margin, all of which minimize the domain discrepancy at the FC layers of neural networks. Moreover, we achieve competitive accuracy (95.6%) to the state-of-the-art result (98.2%) reported by PixelDA. Note that technically, PixelDA is compatible to our method, and can be adopted to improve the accuracy of our model. We will investigate this in the future. Second, we observe that the accuracy of the source network drops heavily when transferred to the target domain (from 99.3% on source test set to 45.6% on target test set), which implies the significant domain shift from MNIST to MNIST-M. Third, we can see

that the distribution of synthetic target data is much closer to real target data than real source data, by observing that training with synthetic target data improves the performance over the source network by about +30%. Finally, training with a mixture of source and synthetic target data is beneficial for learning domain invariant features, and improves the adaptation performance by +3.5% over the model trained with synthetic target data only.

Table 4.1 demonstrates that our EM training algorithm is an effective way to exploit unlabeled target domain data. Moreover, imposing the attention alignment penalty $\mathcal{L}^{AT}$ always leads to noticeable improvement.

Method	Train Data	Accuracy (%)	Method	Train Data	Accuracy (%)
RevGrad [25]	S+T	81.5	CNN	S	45.6
DSN [14]	S+T	83.2	CNN	T_f	75.0
ADA [32]	S+T	85.9	CNN	S+T_f	78.5
PixelDA [13]	S+T+T_f	**98.2**	CNN + $\mathcal{L}^{AT}$	S+T_f	85.7
Ours (wo $\mathcal{L}^{AT}$)	S+T_f+T+S_f	93.5	Ours (wo $\mathcal{L}^{AT}$)	S+T_f+T+S_f	93.5
Ours (w $\mathcal{L}^{AT}$)	S+T_f+T+S_f	**95.6**	Ours (w $\mathcal{L}^{AT}$)	S+T_f+T+S_f	**95.6**

Table 4.1 : Classification accuracy (%) for MNIST → MNIST-M. "CNN" denotes the source and target network (Section 4.4.2). The "S" and "T_f" represent labeled source data and synthetic target data, respectively. The "T" and "S_f" denote unlabeled target data and synthetic source data, respectively

Office-31. The classification results based on ResNet-50 are shown in Table 4.2. With identical evaluation setting, we compare our methods with previous transfer methods and variants of our method. We have three major conclusions.

First, from Table 4.2, it can be seen that our method outperforms the state of art in all the transfer tasks with a large margin. The improvement is larger on harder transfer tasks, where the source domain is substantially different from and

Method	Train Data	A → W	D → W	W → D	A → D	D → A	W → A	Average
ResNet-50	S	68.4 ± 0.2	96.7 ± 0.1	99.3 ± 0.1	68.9 ± 0.2	62.5 ± 0.3	60.7 ± 0.3	76.1
RevGrad [25]	S+T	82.0 ± 0.4	96.9 ± 0.2	99.1 ± 0.1	79.7 ± 0.4	68.2 ± 0.4	67.4 ± 0.5	82.2
JAN [63]	S+T	85.4 ± 0.3	97.4 ± 0.2	99.8 ± 0.2	84.7 ± 0.3	68.6 ± 0.3	70.0 ± 0.4	84.3
JAN-A [63]	S+T	86.0 ± 0.4	96.7 ± 0.3	99.7 ± 0.1	85.1 ± 0.4	69.2 ± 0.4	70.7 ± 0.5	84.6
ResNet-50	T_f	81.1 ± 0.2	98.5 ± 0.2	99.8 ± 0.0	83.3 ± 0.3	61.0 ± 0.2	60.2 ± 0.3	80.6
ResNet-50	S+T_f	81.9 ± 0.2	98.5 ± 0.2	99.8 ± 0.0	83.7 ± 0.3	66.5 ± 0.2	64.8 ± 0.3	82.5
Ours (wo $\mathcal{L}^{AT}$)	T_f+T	86.2 ± 0.2	**99.3** ± 0.1	**100** ± 0.0	86.5 ± 0.6	69.9 ± 0.6	70.2 ± 0.2	85.4
Ours (w $\mathcal{L}^{AT}$)	T_f+T	86.8 ± 0.2	**99.3** ± 0.1	**100** ± 0.0	87.2 ± 0.5	71.7 ± 0.5	71.8 ± 0.1	86.1
Ours (wo $\mathcal{L}^{AT}$)	S+T_f+T+S_f	**87.1** ± 0.3	**99.3** ± 0.1	**100** ± 0.0	87.1 ± 0.2	72.3 ± 0.2	72.2 ± 0.2	86.3
Ours (w $\mathcal{L}^{AT}$)	S+T_f+T+S_f	86.8 ± 0.2	**99.3** ± 0.1	**100** ± 0.0	**88.8** ± 0.4	**74.3** ± 0.2	**73.9** ± 0.2	**87.2**

Table 4.2 : Classification accuracy (%) on the Office-31 dataset based on ResNet-50

has much less data than the target domain, *e.g.* **D** → **A**, and **W** → **A**. Specifically, we improve over the state of art result by +2.6% on average, and by +5.1 % for the difficult transfer task **D** → **A**.

Second, we also compare our method with and without the adversarial attention alignment loss $\mathcal{L}^{AT}$. Although for easy transfer tasks, the performance of these two variants are comparable, when moving to much harder tasks, we observe obvious improvement brought by the adversarial attention alignment, *e.g.,* training with adversarial attention alignment outperforms that without attention alignment by +2% for the task **D** → **A**, and +1.7% for the task **W** → **A**. This implies that adversarial attention alignment helps reduce the discrepancy across domains and regularize the training of the target model.

Third, we validate that augmenting with synthetic target data to facilitate the target network training brings significant improvement of accuracy over source network. This indicates that the discrepancy between synthetic and real target data is much smaller. We also notice that in our method, the accuracy of the network

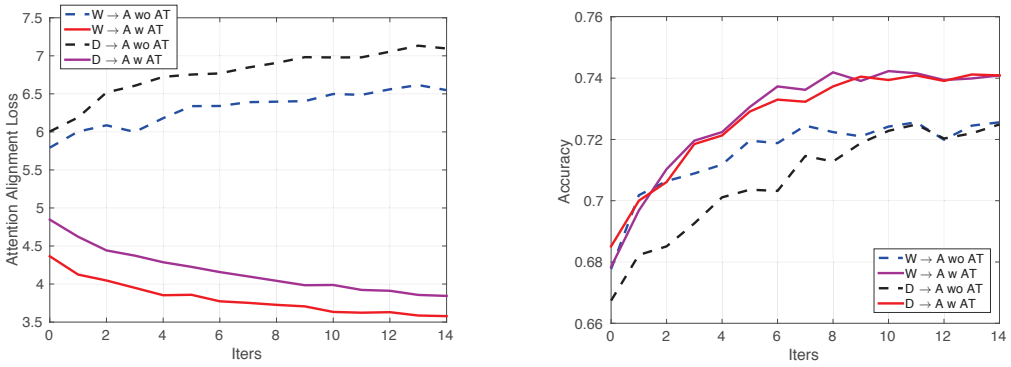


Figure 4.4 : Analysis of the training process (EM is implemented). **Left**: The trend of $\mathcal{L}^{AT}$ during training with and without imposing the $\mathcal{L}^{AT}$ penalty term. **Right**: The curves of test accuracy on the target domain. The results of tasks **W** $\rightarrow$ **A** and **D** $\rightarrow$ **A** are presented. The results for other tasks are similar. One iteration here represents one update of the network M^{post} (see Section 4.3.3).

Method	Train Data	A → W	A → D	D → A	W → A	Average
ResNet-50	S	68.4 ± 0.2	68.9 ± 0.2	62.5 ± 0.3	60.7 ± 0.3	65.1
EM-A	S+T_f+T+S_f	68.6 ± 0.3	73.5 ± 0.3	62.7 ± 0.3	52.8 ± 0.3	64.4
EM-A + $\mathcal{L}^{AT}$	S+T_f+T+S_f	80.4 ± 0.2	79.1 ± 0.2	66.4 ± 0.2	58.4 ± 0.2	71.1
EM-C	S+T_f+T+S_f	86.4 ± 0.3	87.0 ± 0.3	69.5 ± 0.3	71.4 ± 0.3	78.6
EM-C + $\mathcal{L}^{AT}$	S+T_f+T+S_f	86.2 ± 0.2	86.6 ± 0.3	71.8 ± 0.3	73.7 ± 0.2	79.6
EM-B	S+T_f+T+S_f	*very low*	*very low*	*very low*	*very low*	*very low*
EM-B + $\mathcal{L}^{AT}$	S+T_f+T+S_f	*very low*	*very low*	*very low*	*very low*	*very low*
Ours (wo $\mathcal{L}^{AT}$)	S+T_f+T+S_f	**87.1** ± 0.3	87.1 ± 0.2	72.3 ± 0.2	72.2 ± 0.2	79.7
Ours (w $\mathcal{L}^{AT}$)	S+T_f+T+S_f	86.8 ± 0.2	**88.8** ± 0.4	**74.3** ± 0.2	**73.9** ± 0.2	**80.9**

Table 4.3 : Variants of the EM algorithm with and without $\mathcal{L}^{AT}$. The EM algorithm without asynchronous update of M^{post} is denoted by EM-A, while that without filtering the noisy data is denoted by EM-B. EM-C represents EM training without initializing the learning rate schedule when M^{post} is updated

Measure	**A** $\rightarrow$ **W**	**A** $\rightarrow$ **D**	**D** $\rightarrow$ **A**	**W** $\rightarrow$ **A**	Average
L_1-norm	*very low*	*very low*	*very low*	*very low*	*very low*
MMD	84.7	84.1	66.2	64.5	74.9
JMMD	85.9	85.3	70.1	71.1	78.1
Ours	**86.8**	**88.8**	**74.3**	**73.9**	**80.9**

Table 4.4 : Comparison of different attention discrepancy measures on Office-31

trained with real and synthetic data from both domains is much better than the one purely trained with real and synthetic target data. This verifies the knowledge shared by the source domain can be sufficiently uncovered by our framework to improve the target network performance.

Fig. 4.4 illustrates how the attention alignment penalty $\mathcal{L}^{AT}$ changes during the training process with and without this penalty imposed. Without attention alignment, the discrepancy of the attention maps between the source and target network is significantly larger and increases as the training goes on. The improvement of accuracy brought by adding $\mathcal{L}^{AT}$ penalty to the objective can be attributed to the much smaller discrepancy of attention maps between the source and the target models, *i.e.,* better aligned attention mechanism. The testing accuracy curves on the target domain for tasks **D** $\rightarrow$ **A** and **D** $\rightarrow$ **A** are also drawn in Fig. 4.4. It can be seen that the test accuracy steadily increases and the model with $\mathcal{L}^{AT}$ converges much faster than that without any attention alignment.

Visualization of the attention maps of our method is provided in Fig. 4.1. We observe that through attention alignment, the attention maps of the target network adapt well to the target domain images, and are even better than those of the target

model trained on labeled target images.

4.4.4 Ablation Study

Table 4.3 compares the accuracy of different EM variants. We conduct ablation studies by removing one component from the system at a time (three components are considered which are defined in Section 4.3.3). For each variant of EM, we also evaluate the effect of imposing $\mathcal{L}^{AT}$ by comparing training with and without $\mathcal{L}^{AT}$. By comparing the performances of EM-A, EM-B, EM-C and full method we adopted, we find that the three modifications all contribute considerably to the system. Among them, filtering the noisy data is the most important factor. We also notice that for EM-A and EM-C, training along with $\mathcal{L}^{AT}$ always leads to a significant improvement, implying performing attention alignment is an effective way to improve the adaptation performance.

4.4.5 Comparing Different Attention Discrepancy Measures

In this section, we provide a method comparison in measuring the attention discrepancy across domains which is discussed in Section 4.3.2. We use the L_2 distance, and the compared methods include the L_1 distance, MMD [62] and JMMD [63]. Results are presented in Table 4.4.

We find that our method achieves the best results among the four measures. The L_1 distance fails in training a workable network because it is misled by the noise in the attention maps. Our method outperforms MMD/JMMD by a large margin, because our method preserves the structure information, as discussed in Section 4.3.2.

4.4.6 Impact of Hyper-parameters

We investigate the impact of p_t (*i.e.* filtering threshold in EM) and β (*i.e.* the strength of attention alignment penalty) on the classification accuracy of target

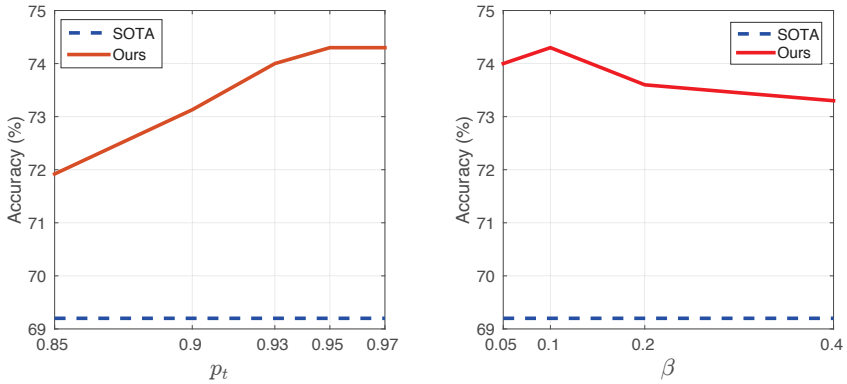


Figure 4.5 : The impact of hyper-parameters on the classification accuracy of target model. The results for task $\mathbf{D} \rightarrow \mathbf{A}$ on Office-31 are illustrated, with a comparison to the previous state-of-the-art (SOTA). The trends are similar for other tasks. **Left**: Accuracy vs. p_t. **Right**: Accuracy vs. β.

model, respectively. The results are shown in Fig. 4.5.

Within a range, a larger p_t leads to better accuracy, while with the growth of β, the accuracy of the model increases before the slightly decrease. For both p_t and β, we observe that within a wide range, the accuracy of our method outperforms the previous state-of-the-art method with a large margin, which implies the superiority of our method.

4.4.7 Comparison with Different Variants of Attention

We conduct experiment to verify the effectiveness of attention defined by Eq. (4) of the text. The comparison results are summarized in Table 4.5. Note that the attention mechanism defined in our method is the aggregation of feature maps along channels using L_2-norm, and L_1- and L_∞-norm aggregating methods are compared in Table 4.5. We also compare our method with directly aligning feature maps without any aggregation (denoted as "FM" in Table 4.5). We find that aligning the proposed attention performs much better than aligning other variants, which verifies the effectiveness of attention defined by Eq. (4).

Variant	**A** $\rightarrow$ **W**	**A** $\rightarrow$ **D**	**D** $\rightarrow$ **A**	**W** $\rightarrow$ **A**	Average
L_1	85.2	87.8	73.3	73.0	79.8
L_∞	86.4	87.2	73.1	73.2	80.0
FM	86.0	87.6	73.2	72.9	79.9
Ours (attention)	**86.8**	**88.8**	**74.3**	**73.9**	**80.9**

Table 4.5 : Comparison of aligning different representations on Office-31

4.5 Conclusion

In this chapter, we make two contributions to the community of UDA. First, from the *convolutional layers*, we propose to align the attention maps of the source network and target network to make the knowledge from source network better adapted to the target one. Second, from an *EM perspective*, we maximize the likelihood of unlabeled target data, which enables target network to leverage more training data for better domain adaptation. Both contributions benefit from the unsupervised image correspondences provided by CycleGAN. Experiment demonstrates that the two contributions both have positive effects on the system performance, and they cooperate together to achieve competitive or even state-of-the-art results on two benchmark datasets.

Chapter 5

Conclusion

In this thesis, we investigate the regularization techniques in deep learning. We contribute in two settings, one is the conventional supervised learning, and the other is the unsupervised domain adaptation.

For the first setting, we proposed a new regularization technique named "Shakeout" to improve the generalization performance beyond Dropout. Moreover, Shakeout introduces a combination of L_0, L_1, L_2 regularization effect upon the weights during the network training. Consequently, Shakeout leads to much sparser weights, compared to those learned through Dropout. This statistical trait is expected to benefit other applications, such as network compression.

In unsupervised domain adaptation, previous methods mainly consider the alignment across domains at the tail of the networks. However, we found that the discrepancy between the source and target domain emerges at the start from the convolutional layers, by observing the distinct attention patterns across domains. Based on this observation, we proposed to align the attention mechanism of the target network (student) with the source network (teacher) to explicitly regularize the behavior of the convolutional layers of the target network. Experiment results demonstrate that introducing such regularization improves the adaptation performance noticeably.

In future, we will apply the proposed regularization methods in other application scenarios. Moreover, it is also valuable to investigate how to effectively employ the regularization techniques in the semi-supervised and domain generalization problems. Finally, the theoretical innovation is also a promising direction.

Bibliography

[1] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik, "Contour detection and hierarchical image segmentation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 33, no. 5, pp. 898–916, 2011.

[2] M. Arjovsky and L. Bottou, "Towards principled methods for training generative adversarial networks," in *NIPS 2016 Workshop on Adversarial Training. In review for ICLR*, vol. 2016, 2017.

[3] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein gan," *arXiv preprint arXiv:1701.07875*, 2017.

[4] J. Ba and R. Caruana, "Do deep nets really need to be deep?" in *Advances in neural information processing systems*, 2014, pp. 2654–2662.

[5] J. Ba and B. Frey, "Adaptive dropout for training deep neural networks," in *Advances in Neural Information Processing Systems*, 2013, pp. 3084–3092.

[6] P. Baldi and P. J. Sadowski, "Understanding dropout," in *Advances in Neural Information Processing Systems*, 2013, pp. 2814–2822.

[7] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1798–1828, Aug 2013.

[8] Y. Bengio, "Learning deep architectures for AI," *Foundations and trends® in Machine Learning*, vol. 2, no. 1, pp. 1–127, 2009.

[9] Y. Bengio, I. J. Goodfellow, and A. Courville, "Deep learning," *An MIT Press book in preparation. Draft chapters available at http://www. iro. umontreal. ca/ bengioy/dlbook*, 2015.

[10] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *Journal of Machine Learning Research*, vol. 13, no. Feb, pp. 281–305, 2012.

[11] C. M. Bishop, "Training with noise is equivalent to tikhonov regularization," *Neural computation*, vol. 7, no. 1, pp. 108–116, 1995.

[12] Y.-l. Boureau, Y. L. Cun *et al.*, "Sparse feature learning for deep belief networks," in *Advances in neural information processing systems*, 2008, pp. 1185–1192.

[13] K. Bousmalis, N. Silberman, D. Dohan, D. Erhan, and D. Krishnan, "Unsupervised pixel-level domain adaptation with generative adversarial networks," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[14] K. Bousmalis, G. Trigeorgis, N. Silberman, D. Krishnan, and D. Erhan, "Domain separation networks," in *Advances in Neural Information Processing Systems*, 2016, pp. 343–351.

[15] N. Chen, J. Zhu, J. Chen, and B. Zhang, "Dropout training for support vector machines," in *Twenty-Eighth AAAI Conference on Artificial Intelligence*, 2014.

[16] W. Chen, J. T. Wilson, S. Tyree, K. Q. Weinberger, and Y. Chen, "Compressing neural networks with the hashing trick," in *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, 2015, pp. 2285–2294.

[17] Y. Chen, W. Li, C. Sakaridis, D. Dai, and L. Van Gool, "Domain adaptive faster r-cnn for object detection in the wild," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 3339–3348.

[18] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on.* IEEE, 2009, pp. 248–255.

[19] W. Deng, L. Zheng, G. Kang, Y. Yang, Q. Ye, and J. Jiao, "Image-image domain adaptation with preserved self-similarity and domain-dissimilarity for person reidentification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 1, no. 2, 2018, p. 6.

[20] M. Denil, B. Shakibi, L. Dinh, N. de Freitas *et al.*, "Predicting parameters in deep learning," in *Advances in Neural Information Processing Systems*, 2013, pp. 2148–2156.

[21] M. Ding and G. Fan, "Multilayer joint gait-pose manifolds for human gait motion modeling." *IEEE Trans. Cybernetics*, vol. 45, no. 11, pp. 2413–2424, 2015.

[22] X. Dong, Y. Yan, W. Ouyang, and Y. Yang, "Style aggregated network for facial landmark detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018, pp. 379–388.

[23] D. Erhan, Y. Bengio, A. Courville, P.-A. Manzagol, P. Vincent, and S. Bengio, "Why does unsupervised pre-training help deep learning?" *The Journal of Machine Learning Research*, vol. 11, pp. 625–660, 2010.

[24] Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," in *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, 2016, pp. 1050–1059.

[25] Y. Ganin and V. Lempitsky, "Unsupervised domain adaptation by backpropagation," in *International Conference on Machine Learning*, 2015, pp. 1180–1189.

[26] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Region-based convolutional networks for accurate object detection and segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 1, pp. 142–158, Jan 2016.

[27] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[28] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[29] I. J. Goodfellow, A. Courville, and Y. Bengio, "Spike-and-slab sparse coding for unsupervised feature discovery," *arXiv preprint arXiv:1201.3382*, 2012.

[30] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *CoRR*, vol. abs/1412.6572, 2014. [Online]. Available: http://arxiv.org/abs/1412.6572

[31] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *Journal of machine learning research*, vol. 3, no. Mar, pp. 1157–1182, 2003.

[32] P. Haeusser, T. Frerix, A. Mordvintsev, and D. Cremers, "Associative domain

adaptation," in *International Conference on Computer Vision (ICCV)*, vol. 2, no. 5, 2017, p. 6.

[33] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding," *CoRR, abs/1510.00149*, vol. 2, 2015.

[34] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connections for efficient neural network," in *Advances in Neural Information Processing Systems*, 2015, pp. 1135–1143.

[35] T. Hastie, R. Tibshirani, and M. Wainwright, *Statistical learning with sparsity: the lasso and generalizations*. CRC press, 2015.

[36] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[37] ——, "Identity mappings in deep residual networks," in *European Conference on Computer Vision*. Springer, 2016, pp. 630–645.

[38] D. P. Helmbold and P. M. Long, "On the inductive bias of dropout," *Journal of Machine Learning Research*, vol. 16, pp. 3403–3454, 2015.

[39] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[40] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[41] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *arXiv preprint arXiv:1207.0580*, 2012.

[42] A. E. Hoerl and R. W. Kennard, "Ridge regression: Biased estimation for nonorthogonal problems," *Technometrics*, vol. 12, no. 1, pp. 55–67, 1970.

[43] J. Hoffman, E. Tzeng, T. Park, J.-Y. Zhu, P. Isola, K. Saenko, A. A. Efros, and T. Darrell, "Cycada: Cycle-consistent adversarial domain adaptation," *arXiv preprint arXiv:1711.03213*, 2017.

[44] G. Huang, Y. Sun, Z. Liu, D. Sedra, and K. Q. Weinberger, "Deep networks with stochastic depth," in *European Conference on Computer Vision*. Springer, 2016, pp. 646–661.

[45] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, 2015, pp. 448–456.

[46] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," in *Proceedings of the ACM International Conference on Multimedia*. ACM, 2014, pp. 675–678.

[47] W. Jiang, F. Nie, and H. Huang, "Robust dictionary learning with capped l1-norm," in *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, 2015, pp. 3590–3596.

[48] X. Jin, C. Xu, J. Feng, Y. Wei, J. Xiong, and S. Yan, "Deep learning with s-shaped rectified linear activation units," *arXiv preprint arXiv:1512.07030*, 2015.

[49] G. Kang, J. Li, and D. Tao, "Shakeout: A new regularized deep neural network training scheme," in *Thirtieth AAAI Conference on Artificial Intelli-*

gence, 2016.

[50] ——, "Shakeout: A new approach to regularized deep neural network training," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 5, pp. 1245–1258, 2018.

[51] G. Kang, L. Zheng, Y. Yan, and Y. Yang, "Deep adversarial attention alignment for unsupervised domain adaptation: the benefit of target expectation maximization," *ECCV*, 2018.

[52] T. Kim, M. Cha, H. Kim, J. Lee, and J. Kim, "Learning to discover cross-domain relations with generative adversarial networks," in *International Conference on Machine Learning*, 2017.

[53] A. Krizhevsky, "cuda-convnet," 2012. [Online]. Available: https://code.google.com/p/cuda-convnet/

[54] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images. Technical report, University of Toronto," 2009.

[55] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[56] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[57] Y. LeCun, J. S. Denker, S. A. Solla, R. E. Howard, and L. D. Jackel, "Optimal brain damage." in *NIPs*, vol. 2, 1989, pp. 598–605.

[58] Z. Li, B. Gong, and T. Yang, "Improved dropout for shallow and deep learning," in *Advances In Neural Information Processing Systems*, 2016, pp. 2523–

2531.

[59] M.-Y. Liu, T. Breuel, and J. Kautz, "Unsupervised image-to-image translation networks," in *Advances in Neural Information Processing Systems*, 2017, pp. 700–708.

[60] M.-Y. Liu and O. Tuzel, "Coupled generative adversarial networks," in *Advances in neural information processing systems*, 2016, pp. 469–477.

[61] Y.-C. Liu, Y.-Y. Yeh, T.-C. Fu, S.-D. Wang, W.-C. Chiu, and Y.-C. F. Wang, "Detach and adapt: Learning cross-domain disentangled deep representation," *arXiv preprint arXiv:1705.01314*, 2017.

[62] M. Long, Y. Cao, J. Wang, and M. Jordan, "Learning transferable features with deep adaptation networks," in *International Conference on Machine Learning*, 2015, pp. 97–105.

[63] M. Long, J. Wang, and M. I. Jordan, "Deep transfer learning with joint adaptation networks," in *ICML*, 2017.

[64] M. Long, H. Zhu, J. Wang, and M. I. Jordan, "Unsupervised domain adaptation with residual transfer networks," in *Advances in Neural Information Processing Systems*, 2016, pp. 136–144.

[65] P. Luc, C. Couprie, S. Chintala, and J. Verbeek, "Semantic segmentation using adversarial networks," in *NIPS Workshop on Adversarial Training*, 2016.

[66] D. Maclaurin, D. Duvenaud, and R. P. Adams, "Gradient-based hyperparameter optimization through reversible learning," in *Proceedings of the 32nd International Conference on Machine Learning*, 2015.

[67] T. Miyato, S.-i. Maeda, S. Ishii, and M. Koyama, "Virtual adversarial training: a regularization method for supervised and semi-supervised learning," *IEEE*

transactions on pattern analysis and machine intelligence, 2018.

[68] J. Moody, S. Hanson, A. Krogh, and J. A. Hertz, "A simple weight decay can improve generalization," *Advances in neural information processing systems*, vol. 4, pp. 950–957, 1995.

[69] H. Noh, T. You, J. Mun, and B. Han, "Regularizing deep neural networks by noise: Its interpretation and optimization," in *Advances in Neural Information Processing Systems*, 2017, pp. 5109–5118.

[70] B. A. Olshausen and D. J. Field, "Sparse coding with an overcomplete basis set: A strategy employed by v1?" *Vision research*, vol. 37, no. 23, pp. 3311–3325, 1997.

[71] N. Passalis and A. Tefas, "Learning deep representations with probabilistic knowledge transfer," in *The European Conference on Computer Vision (ECCV)*, September 2018.

[72] Z. Pei, Z. Cao, M. Long, and J. Wang, "Multi-adversarial domain adaptation," in *AAAI Conference on Artificial Intelligence*, 2018.

[73] L. Prechelt, "Automatic early stopping using cross validation: quantifying the criteria," *Neural Networks*, vol. 11, no. 4, pp. 761–767, 1998.

[74] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv preprint arXiv:1511.06434*, 2015.

[75] S. Rifai, X. Glorot, Y. Bengio, and P. Vincent, "Adding noise to the input of a model trained with a regularized objective," *arXiv preprint arXiv:1104.3250*, 2011.

[76] A. Rozantsev, M. Salzmann, and P. Fua, "Beyond sharing weights for deep domain adaptation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2018.

[77] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.

[78] P. Russo, F. M. Carlucci, T. Tommasi, and B. Caputo, "From source to target and back: symmetric bi-directional adaptive gan," *arXiv preprint arXiv:1705.08824*, 2017.

[79] K. Saito, Y. Ushiku, and T. Harada, "Asymmetric tri-training for unsupervised domain adaptation," *arXiv preprint arXiv:1702.08400*, 2017.

[80] K. Saito, Y. Ushiku, T. Harada, and K. Saenko, "Adversarial dropout regularization," *arXiv preprint arXiv:1711.01575*, 2017.

[81] K. Saito, K. Watanabe, Y. Ushiku, and T. Harada, "Maximum classifier discrepancy for unsupervised domain adaptation," *arXiv preprint arXiv:1712.02560*, vol. 3, 2017.

[82] S. Sankaranarayanan, Y. Balaji, C. D. Castillo, and R. Chellappa, "Generate to adapt: Aligning domains using generative adversarial networks," *ArXiv e-prints, abs/1704.01705*, 2017.

[83] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-cam: Visual explanations from deep networks via gradient-based localization," in *ICCV*, 2017, pp. 618–626.

[84] A. Shrivastava, T. Pfister, O. Tuzel, J. Susskind, W. Wang, and R. Webb,

"Learning from simulated and unsupervised images through adversarial training," in *CVPR*, 2017.

[85] K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep inside convolutional networks: Visualising image classification models and saliency maps," *arXiv preprint arXiv:1312.6034*, 2013.

[86] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[87] S. Singh, D. Hoiem, and D. Forsyth, "Swapout: Learning an ensemble of deep architectures," in *Advances in neural information processing systems*, 2016, pp. 28–36.

[88] J. Snoek, H. Larochelle, and R. P. Adams, "Practical bayesian optimization of machine learning algorithms," in *Advances in neural information processing systems*, 2012, pp. 2951–2959.

[89] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[90] B. Sun, J. Feng, and K. Saenko, "Return of frustratingly easy domain adaptation." in *AAAI*, vol. 6, no. 7, 2016, p. 8.

[91] B. Sun and K. Saenko, "Deep coral: Correlation alignment for deep domain adaptation," in *Computer Vision–ECCV 2016 Workshops*. Springer, 2016, pp. 443–450.

[92] Y. Sun, X. Wang, and X. Tang, "Hybrid deep learning for face verification," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 10, pp. 1997–2009, Oct 2016.

[93] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning," in *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA.*, 2017, pp. 4278–4284.

[94] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1–9.

[95] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2818–2826.

[96] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," *arXiv preprint arXiv:1312.6199*, 2013.

[97] M. Thom and G. Palm, "Sparse activity and sparse connectivity in supervised learning," *Journal of Machine Learning Research*, vol. 14, no. Apr, pp. 1091–1143, 2013.

[98] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 267–288, 1996.

[99] E. Tzeng, J. Hoffman, T. Darrell, and K. Saenko, "Simultaneous deep transfer across domains and tasks," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 4068–4076.

[100] E. Tzeng, J. Hoffman, K. Saenko, and T. Darrell, "Adversarial discriminative domain adaptation," in *Computer Vision and Pattern Recognition (CVPR)*,

2017.

[101] E. Tzeng, J. Hoffman, N. Zhang, K. Saenko, and T. Darrell, "Deep domain confusion: Maximizing for domain invariance," *arXiv preprint arXiv:1412.3474*, 2014.

[102] L. Van Der Maaten, M. Chen, S. Tyree, and K. Q. Weinberger, "Learning with marginalized corrupted features." in *ICML (1)*, 2013, pp. 410–418.

[103] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proceedings of the 25th international conference on Machine learning*. ACM, 2008, pp. 1096–1103.

[104] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *The Journal of Machine Learning Research*, vol. 11, pp. 3371–3408, 2010.

[105] S. Wager, S. Wang, and P. S. Liang, "Dropout training as adaptive regularization," in *Advances in Neural Information Processing Systems*, 2013, pp. 351–359.

[106] L. Wan, M. Zeiler, S. Zhang, Y. L. Cun, and R. Fergus, "Regularization of neural networks using dropconnect," in *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, 2013, pp. 1058–1066.

[107] K. Wang, R. He, L. Wang, W. Wang, and T. Tan, "Joint feature selection and subspace learning for cross-modal retrieval," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 10, pp. 2010–2023, Oct 2016.

[108] D. Warde-Farley, I. J. Goodfellow, A. Courville, and Y. Bengio, "An empirical analysis of dropout in piecewise linear networks," *arXiv preprint arXiv:1312.6197*, 2013.

[109] Y. Wei, J. Feng, X. Liang, M.-M. Cheng, Y. Zhao, and S. Yan, "Object region mining with adversarial erasing: A simple classification to semantic segmentation approach," in *IEEE CVPR*, 2017.

[110] Y. Wei, W. Xia, M. Lin, J. Huang, B. Ni, J. Dong, Y. Zhao, and S. Yan, "Hcp: A flexible cnn framework for multi-label image classification," *IEEE transactions on pattern analysis and machine intelligence*, vol. 38, no. 9, pp. 1901–1907, 2016.

[111] D. R. G. H. R. Williams and G. Hinton, "Learning representations by back-propagating errors," *Nature*, pp. 323–533, 1986.

[112] R. Xu, Z. Chen, W. Zuo, J. Yan, and L. Lin, "Deep cocktail network: Multi-source unsupervised domain adaptation with category shift," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 3964–3973.

[113] L. Yuan, J. Liu, and J. Ye, "Efficient methods for overlapping group lasso," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 9, pp. 2104–2116, 2013.

[114] S. Zagoruyko and N. Komodakis, "Wide residual networks," in *Proceedings of the British Machine Vision Conference 2016, BMVC 2016, York, UK, September 19-22, 2016*, 2016.

[115] ——, "Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer," in *ICLR*, 2017.

[116] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *European conference on computer vision*. Springer, 2014, pp. 818–833.

[117] X. Zhang, Y. Wei, J. Feng, Y. Yang, and T. Huang, "Adversarial complementary learning for weakly supervised object localization," in *IEEE CVPR*, 2018.

[118] Y. Zheng, Y. J. Zhang, and H. Larochelle, "A deep and autoregressive approach for topic modeling of multimodal data," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 6, pp. 1056–1069, June 2016.

[119] Z. Zheng, L. Zheng, and Y. Yang, "Unlabeled samples generated by gan improve the person re-identification baseline in vitro," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017.

[120] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, "Learning deep features for discriminative localization," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2921–2929.

[121] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networkss," in *Computer Vision (ICCV), 2017 IEEE International Conference on*, 2017.

[122] H. Zou and T. Hastie, "Regularization and variable selection via the elastic net," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 67, no. 2, pp. 301–320, 2005.