Semi-Supervised Training of Convolutional Neural Networks

Kevin Duarte, Yang Zhang, and Boqing Gong
University of Central Florida
Center for Research in Computer Vision
kevin_duarte@knights.ucf.edu, yangzhang4065@gmail.com, bgong@crcv.ucf.edu

July 28, 2016

Abstract

In this paper we discuss a method for semi-supervised training of CNNs. By using auto-encoders to extract features from unlabeled images, we can train CNNs to accurately classify images with only a small set of labeled images. We show our method's results on a shallow CNN using the CIFAR-10 dataset, and some preliminary results on a VGG-16 network using the STL-10 dataset.

1 Introduction

Over the years, deep learning has been heavily used in the field of computer vision. Convolutional neural networks (CNNs) are a powerful deep learning tool which have a wide variety of applications in image and video analysis, classification, and segmentation. Since the addition of CNNs, image classification accuracy has improved immensely when compared to classical computer vision approaches. Due to this, effectively training CNNs has been an important problem that is being studied in the machine learning and computer vision communities.

With millions of images being uploaded to social networking sites daily, data-based learning algorithms, like CNNs, are able to reach their full potential. However, dealing with vast amounts of images introduces new problems, such as the time it requires to sort and label images. For this reason, using unlabeled data in the training of CNNs is extremely useful. Several techniques such as unsupervised and semi-supervised training have been used to take advantage of this large amount of unlabeled data. Unsupervised training uses only unlabeled data, while semi-supervised learning uses a large amount of unlabeled data in conjunction with a small set of labeled data. In this paper, we will use semi-supervised techniques to train CNNs for image classification.

1.1 Related Work

Using both unlabeled and labeled data to classify images is by no means a novel concept in machine learning. Semi-supervised learning can be seen in Transductive SVMs [1], where unlabeled data is used to bolster the margins of SVM classification. A simpler algorithm is self-training [5] where labeled data is used to improve upon predictions made by an unsupervised classifier. Furthermore, deep generative models [2] have been shown to be competitive semi-supervised learning algorithms. One of the most common semi-supervised approaches are graph-based methods that construct a graph connecting similar observations in labeled and unlabeled data [8].

Semi-supervised training of neural networks have also shown some promising results. One approach is training a feed-forward classifier and having an additional penalty from unsupervised embedding of the data [6]. The auto-encoder approach being used in the paper can be traced back to the semi-supervised learning of text documents [3]. Another method is the Manifold Tangent Classifier [4] which trains an auto-encoder to learn the manifold where the data resides and uses it to train a classifier.

2 Method

Our method of semi-supervised training begins with unsupervised feature learning using an auto-encoder. We train a fully convolutional auto-encoder to compress and decompress images. By doing this, the auto-encoder learns features necessary for image reconstruction. The features produced by the auto-encoder are then used in a CNN which, using a small set of labeled images, is trained to classify the images.

To measure the loss of the auto-encoder's reconstruction, we use the $L2$ loss

$$L = \sum (x - \bar{x})^2$$

where x is the original image and $\bar{x}$ is the image reconstruction. The used loss in the classification CNN is categorical cross-entropy.

In this paper, we also implement a stacked what-where auto-encoder (SWWAE)[7]. The main differences between this and a regular convolutional auto-encoder is the use of what-where pooling/unpooling and the addition of intermediate reconstruction losses. Unlike conventional max-pooling and upscaling layers, the what-where pooling saves the location of the maximum value, and sends it to the unpooling layer. When unpooling occurs, the values are placed in the saved location. This has been shown to greatly aid in the reconstruction of the input image. The addition intermediate reconstruction losses aids in the feature learning by acting as a regularizer for the auto-encoder. These losses are also L2 losses, but instead of using the original image and the image reconstruction, they use the feature maps in the encoder and decoder of the auto-encoder.

3 Experiments

We do experiments on two different network architectures of varying depth. We will represent the architectures with the following notation: $(512)3c - 2p - (512)3c - 2up - (16)3c - 128fc - 10fc$, where $(512)3c$ denotes a convolutional layer with 512 feature maps and a square kernel of size 3, $2p$ is a 2 x 2 max-pooling layer, $2up$ is a 2 x 2 unpooling (or upscaling) layer, an $128fc$ denotes a fully connected layer with 128 hidden units. Unless stated otherwise, convolutional layers have ReLU activations.

Table 1: CIFAR-10 accuracy results on the 10,000 testing images

CNN	Accuracy
Semi-supervised trained with 10,000 images	55.51%
Randomly initialized trained with 10,000 images	48.47%
Randomly initialized trained with 50,000 images	55.79%

3.1 Shallow Auto-encoder

We begin with creating a shallow auto-encoder that deconstructs and reconstructs small, 32 x 32, RGB images. This auto-encoder's architecture is $(256)3c - (128)3c - 2p - (64)3c - (3)3c - (64)3c - 2up - (128)3c - (256)3c - (3)3c$, where the final layer has a sigmoid activation. It was trained on 50,000 training images from the CIFAR-10 dataset and the only data preprocessing that occurred was the normalization of the input images. Since this auto-encoder was shallow, and it only compresses the images to one quarter of their original size, the reconstruction was very similar to the original image. The average reconstruction loss, per image, was 1.12645.

The classification CNN has the same structure as the auto-encoder's encoder, ended with two fully connected layers: $(256)3c - (128)3c - 2p - (64)3c - (3)3c - 256fc - 10fc$. We train this classification network on 10,000 of the labeled training images using the weights from the auto-encoder's encoder. To test how effective the semi-supervised training is, we trained two networks that have the same structure with randomly initialized weights; one was trained on the same 10,000 labeled images, while the other was trained on all 50,000 training images. The results are shown in table 1. For this network, the semi-supervised training reaches an accuracy similar to fully supervised training with the same amount of images.

3.2 Deep Auto-encoder

With successful results using the shallow auto-encoder, we design a deeper auto-encoder that can handle larger images. The auto-encoder contains the convolutional and pooling layers of the VGG-16 network with a decoder to reconstruct the images. The decoder's structure is $(512)3c - (512)3c - (512)3c - 2up - (512)3c - (512)3c -$

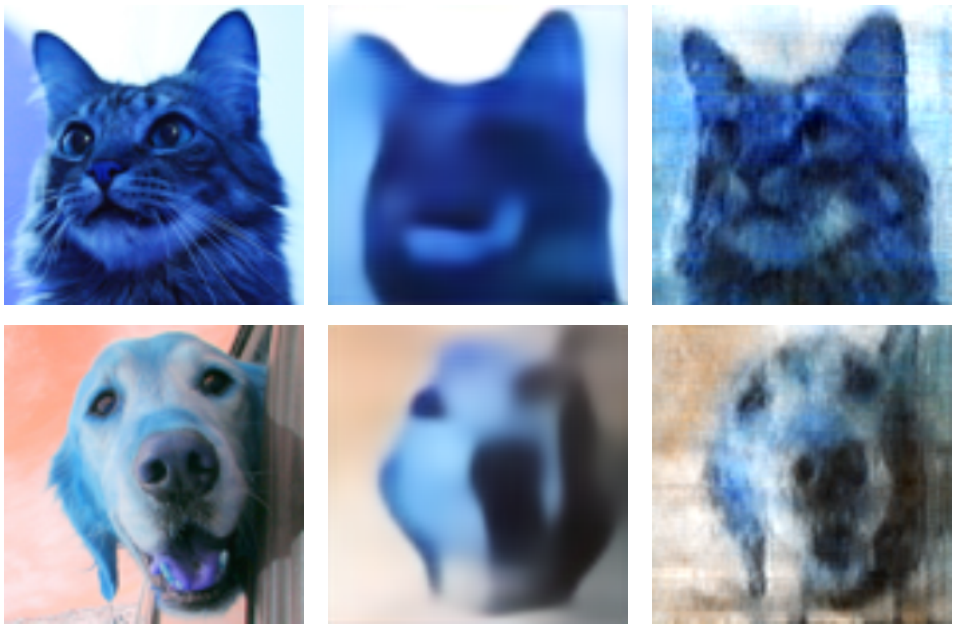


Figure 1: Comparison of image reconstruction on test images from the STL-10 dataset. The left is the original image, center is the conventional auto-encoder reconstruction, and the right is the SWWAE reconstruction.

$(512)3c - 2up - (256)3c - (256)3c - (256)3c - 2up - (128)3c - (128)3c - 2up - (64)3c - (64)3c - (3)3c$, and the final layer has a tanh activation function. Instead of using 224 x 224 images like the standard VGG-16 network, this auto-encoder uses 96 x 96 images.

We train two auto-encoders with this structure. The first is a conventional auto-encoder with max-pooling and upscaling; the second is a what-where auto-encoder that calculates the intermediate loss between the pooling layer and the corresponding convolutional layer prior to the unpooling. We use the 100,000 unlabeled images in the STL-10 dataset to train these auto-encoders. For image preprocessing, we normalize the images and center the mean of each channel. A comparison of the image reconstruction results for both auto-encoders are shown in figure 1. It can be seen that the reconstruction of the SWWAE is more visually similar to the original image, than that of the conventional auto-encoder.

The CNN used to classify the images has the same structure as a VGG-16 network, but uses 96 x 96 image inputs. We train the CNN on the STL-10 dataset's 5,000 labeled training images. For the randomly initialized classification CNN, we get a test accuracy of about 60%. Due to issues with time and technology, we only have preliminary results for the CNNs that use the auto-encoders' weights. The accuracy of these are $55 - 59\%$. In the coming months we will try and improve upon these result, so that the semi-supervised training has a noticeable impact on the classification accuracy.

4 Conclusion

We have shown that using auto-encoders for feature learning and training a classification CNN with the auto-encoders' weights can enhance accuracy in classification tasks for smaller networks. Although, we have not completed our experiments on deeper networks, we show that SWWAEs produce better image reconstructions when compared to conventional auto-encoders. The semi-supervised training methods used in this paper can be used to greatly improve convolutional neural network accuracy.

References

[1] Thorsten Joachims. Transductive inference for text classification using support vector machines. In *ICML*, volume 99, pages 200–209, 1999. 1

[2] Diederik P Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. Semi-supervised learning with deep generative models. In *Advances in Neural Information Processing Systems*, pages 3581–3589, 2014. 1

[3] Marc'Aurelio Ranzato and Martin Szummer. Semi-supervised learning of compact document representations with deep networks. In *Proceedings of the 25th international conference on Machine learning*, pages 792–799. ACM, 2008. 1

[4] Salah Rifai, Yann N Dauphin, Pascal Vincent, Yoshua Bengio, and Xavier Muller. The manifold tangent classifier. In *Advances in Neural Information Processing Systems*, pages 2294–2302, 2011. 1

[5] Chuck Rosenberg, Martial Hebert, and Henry Schneiderman. Semi-supervised self-training of object detection models. 2005. 1

[6] Jason Weston, Frédéric Ratle, Hossein Mobahi, and Ronan Collobert. Deep learning via semi-supervised embedding. In *Neural Networks: Tricks of the Trade*, pages 639–655. Springer, 2012. 1

[7] Junbo Zhao, Michael Mathieu, Ross Goroshin, and Yann Lecun. Stacked what-where auto-encoders. *arXiv preprint arXiv:1506.02351*, 2015. 2

[8] Xiaojin Zhu, Zoubin Ghahramani, John Lafferty, et al. Semi-supervised learning using gaussian fields and harmonic functions. In *ICML*, volume 3, pages 912–919, 2003. 1