

Machine Learning 2016

homework 3 – CIFAR-10 Semi-supervised Learning

翁丞世, R04945028, mob5566[at]gmail.com

Supervised Learning

In supervised learning, I have try the **fully connected network**, the **convolutional network** and the **all convolutional net** from J. T. Springenberg “Striving for Simplicity: The All Convolutional Net”.

The performance of the fully connected network and convolutional network are about **40% accuracy** on public set, but the all convolutional net trained with 50 epochs can achieve **67% accuracy** on public set. Therefore, I choose the all convolutional net as my supervised learning method. The architecture of all convolutional net is shown in Figure 1.

3×3 conv. 96 ReLU
3×3 conv. 96 ReLU
3×3 conv. 96 ReLU
3×3 max-pooling stride 2
3×3 conv. 192 ReLU
3×3 conv. 192 ReLU
3×3 conv. 192 ReLU
3×3 max-pooling stride 2
3×3 conv. 192 ReLU
1×1 conv. 192 ReLU
1×1 conv. 10 ReLU
global meanpool

Figure 1. The all convolutional net input with RGB 32x32

Semi-supervised Learning – Self-training

The self-training model is referred by teacher assist at the homework 3 announcement. I use a **convolutional net** as the basic model to **adaptively learn a classifier to label the unlabeled data**. For each iteration, I add the unlabeled data with **confidence higher than a given threshold** to the labeled data, then train the new labeled data with new CNN model until the total number of iterations reached or all the unlabeled data are labeled.

After the self-training is done, I use the all the labeled data to train a CNN model 100 epochs **with additional drop out layers** to reduce the overfitting condition. The final CNN can obtain 58% accuracy on public set.

The all convolutional net can't be used in self-training **due to out of memory**.

Semi-supervised Learning – Ladder Network

I surveyed the semi-supervised learning model on the internet, and found “Semi-Supervised Learning with Ladder Networks” by A. Rasmus to be my implementation target.

The **algorithm** and the **architecture** of **ladder network** are shown in **Figure 2**. The ladder network consists of **an encoder** with corrupted layers by Gaussian noise, **a decoder** from encoder by a specific de-noise function, and **a clean classifier** with the same weights of encoder.

For the simplicity, I use the τ network mentioned in the paper with all convolutional network that the decoder has only the topmost layer.

Algorithm 1 Calculation of the output \hat{y} and cost function C of the Ladder network

Require: $\mathbf{x}(n)$ # Corrupted encoder and classifier $\tilde{\mathbf{h}}^{(0)} \leftarrow \tilde{\mathbf{z}}^{(0)} \leftarrow \mathbf{x}(n) + \text{noise}$ for $l = 1$ to L do $\tilde{\mathbf{z}}^{(l)} \leftarrow \text{batchnorm}(\mathbf{W}^{(l)} \tilde{\mathbf{h}}^{(l-1)}) + \text{noise}$ $\tilde{\mathbf{h}}^{(l)} \leftarrow \text{activation}(\gamma^{(l)} \odot (\tilde{\mathbf{z}}^{(l)} + \beta^{(l)}))$ end for $P(\tilde{\mathbf{y}} \mathbf{x}) \leftarrow \tilde{\mathbf{h}}^{(L)}$ # Clean encoder (for denoising targets) $\mathbf{h}^{(0)} \leftarrow \mathbf{z}^{(0)} \leftarrow \mathbf{x}(n)$ for $l = 1$ to L do $\mathbf{z}_{\text{pre}}^{(l)} \leftarrow \mathbf{W}^{(l)} \mathbf{h}^{(l-1)}$ $\mu^{(l)} \leftarrow \text{batchmean}(\mathbf{z}_{\text{pre}}^{(l)})$ $\sigma^{(l)} \leftarrow \text{batchstd}(\mathbf{z}_{\text{pre}}^{(l)})$ $\mathbf{z}^{(l)} \leftarrow \text{batchnorm}(\mathbf{z}_{\text{pre}}^{(l)})$ $\mathbf{h}^{(l)} \leftarrow \text{activation}(\gamma^{(l)} \odot (\mathbf{z}^{(l)} + \beta^{(l)}))$ end for	# Final classification: $P(\mathbf{y} \mathbf{x}) \leftarrow \mathbf{h}^{(L)}$ # Decoder and denoising for $l = L$ to 0 do if $l = L$ then $\mathbf{u}^{(L)} \leftarrow \text{batchnorm}(\tilde{\mathbf{h}}^{(L)})$ else $\mathbf{u}^{(l)} \leftarrow \text{batchnorm}(\mathbf{V}^{(l+1)} \hat{\mathbf{z}}^{(l+1)})$ end if $\forall i : \hat{z}_i^{(l)} \leftarrow g(\tilde{z}_i^{(l)}, u_i^{(l)})$ # Eq. (2) $\forall i : \hat{z}_{i,\text{BN}}^{(l)} \leftarrow \frac{\tilde{z}_i^{(l)} - \mu^{(l)}}{\sigma_i^{(l)}}$ end for # Cost function C for training: $C \leftarrow 0$ if $t(n)$ then $C \leftarrow -\log P(\tilde{\mathbf{y}} = t(n) \mathbf{x}(n))$ end if $C \leftarrow C + \sum_{l=0}^L \lambda_l \left\ \mathbf{z}^{(l)} - \hat{\mathbf{z}}_{\text{BN}}^{(l)} \right\ ^2$ # Eq. (3)
--	--

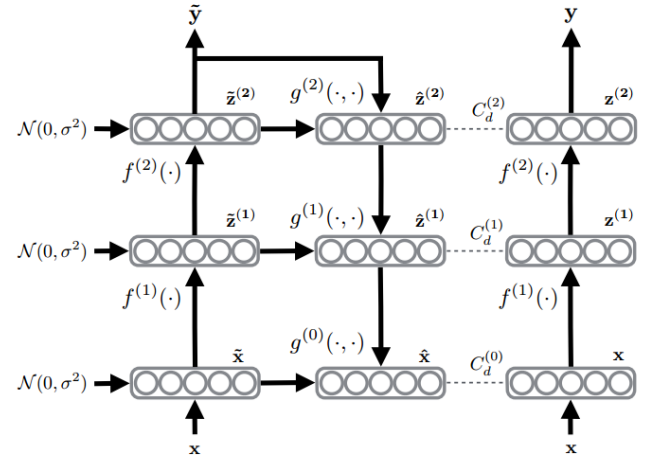


Figure 2. The algorithm and architecture of ladder network can be implement with any kind of neural network.

The performance of τ network is about **61% accuracy** on public set which is worse than all convolutional net with supervised learning. I think it is due to my **awful implementation** of ladder network, then the result is not as expected.

Discussion and Comparison

The power of the supervised learning is **limited** the model by the **little amount of data**, most **semi-supervised learning method** is based on the concept of **regularization** such as **pre-training**, **auto-encoder**, and **generative encoder model** to reduce the overfitting of lack of labeled data.

But I did not implement the ladder network well that even the all convolutional net outperforms 6% accuracy better than it. The time-consuming training is run out of my time that I have no time to fine tune the ladder model. Finally, I choose all convolutional net as my best model.