Deep Learning Assignment 2

Yun-shao Sung* and Chung-Ling Yao†

Abstract. This is the report for deep learning assignment 2

1. More Backpropagation.

1.1. Backpropagation through a DAG of modules. Given each node is sigmoid layers, and the second layer is $O_{min} = min(i_1, i_2)$ and $O_{max} = max(i_1, i_2)$. Therefore, we can rewrite y as:

$$y = min(\frac{1}{1+e^{-x_1}}, \frac{1}{1+e^{-x_2}}) + max(\frac{1}{1+e^{-x_1}}, \frac{1}{1+e^{-x_2}}) \tag{1.1}$$

which can also rewrite as:

$$y = \frac{1}{1+e^{-x_1}} + \frac{1}{1+e^{-x_2}} \tag{1.2}$$

as we taking the deritive of E respect to x_i and with chain rule applied:

$$\frac{\partial E}{\partial x_i} = \frac{\partial E}{\partial y}\frac{e^{-x_i}}{(1+e^{x_i})^2} \tag{1.3}$$

1.2. Batch Normalization.

1.2.1. Calculate $\frac{\partial E}{\partial x_k}$. Let us concentrate on a particular activation x_k, which is a mini-batch β of size m

$$\beta = \{x_{1...m}\} \tag{1.4}$$

mean of this mini-batch:

$$\mu_\beta = \frac{1}{m}\sum_{i=1}^{m} x_i \tag{1.5}$$

variance of this mini-batch:

$$\sigma_\beta^2 = \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_\beta)^2 \tag{1.6}$$

By definition:

$$y_i = \frac{x_i - \mu_\beta}{\sqrt{\sigma_\beta^2}} \tag{1.7}$$

Since σ_β^2 and μ_β are function of x_i, we have to calculate $\frac{\partial E}{\partial \sigma_\beta^2}$ and $\frac{\partial E}{\partial \mu_\beta}$

$$\frac{\partial E}{\partial \sigma_\beta^2} = \sum_{i=1}^{m}\frac{\partial E}{\partial y_i}\cdot\frac{\partial y_i}{\partial \sigma_\beta^2} = \sum_{i=1}^{m}\frac{\partial E}{\partial y_i}\cdot(x_i - \mu_\beta)\cdot\frac{-1}{2}(\sigma_\beta^2)^{\frac{3}{2}} \tag{1.8}$$

*yss265@nyu.edu
†cly264@nyu.edu

$$\frac{\partial E}{\partial \mu_\beta} = \sum_{i=1}^{m} \frac{\partial E}{\partial y_i} \cdot \frac{\partial y_i}{\partial \mu_\beta} + \frac{\partial E}{\partial \sigma_\beta^2} \cdot \frac{\partial \sigma_\beta^2}{\partial \mu_\beta} = (\sum_{i=1}^{m} \frac{\partial E}{\partial y_i} \cdot \frac{-1}{\sqrt{\sigma_\beta^2}}) + \frac{\partial E}{\partial \sigma_\beta^2} \cdot \frac{\sum_{i=1}^{m} -2(x_i - \mu_\beta)}{m} \tag{1.9}$$

$$\frac{\partial E}{\partial x_i} = \frac{\partial E}{\partial y_i}\frac{\partial y_i}{\partial x_i} + \frac{\partial E}{\partial \sigma_\beta^2}\frac{\partial \sigma_\beta^2}{\partial x_i} + \frac{\partial E}{\partial \mu_\beta}\frac{\partial \mu_\beta}{\partial x_i} = \frac{\partial E}{\partial y_i} \cdot \frac{1}{\sqrt{\sigma_\beta^2}} + \frac{\partial E}{\partial \sigma_\beta^2} \cdot \frac{2(x_i - \mu_\beta)}{m} + \frac{\partial E}{\partial \mu_\beta} \cdot \frac{1}{m} \tag{1.10}$$

1.2.2. Shift variable. Let ε be a learnable shift variable, we can define the formula as: forward pass:

$$y_k = \frac{x_k - E(x_k)}{\sigma(x_k)} + \varepsilon_k \tag{1.11}$$

backward pass:

$$\frac{\partial E}{\partial x_i} = \frac{\partial E}{\partial y_i}\frac{\partial y_i}{\partial x_i} + \frac{\partial E}{\partial \sigma_\beta^2}\frac{\partial \sigma_\beta^2}{\partial x_i} + \frac{\partial E}{\partial \mu_\beta}\frac{\partial \mu_\beta}{\partial x_i} = \frac{\partial E}{\partial y_i} \cdot \frac{1}{\sqrt{\sigma_\beta^2}} + \frac{\partial E}{\partial \sigma_\beta^2} \cdot \frac{2(x_i - \mu_\beta)}{m} + \frac{\partial E}{\partial \mu_\beta} \cdot \frac{1}{m} \tag{1.12}$$

$$\frac{\partial E}{\partial \varepsilon} = \sum_{i=1}^{m} \frac{\partial E}{\partial y_i} \cdot \frac{\partial y_i}{\partial \varepsilon} = \sum_{i=1}^{m} \frac{\partial E}{\partial y_i} \cdot \frac{\partial}{\partial \varepsilon}(\frac{x_i - E(x_k)}{\sigma(x_k)} + \varepsilon_k) = \sum_{i=1}^{m} \frac{\partial E}{\partial y_i} \tag{1.13}$$

2. STL-10: semi-supervised image recognition.

2.1. Method: Surrogate Class. We were trying to create surrogate training data from unlabeled images. As the reference paper mentioned, we ramdonly choose 4000 unlabel images and obtain patches from different images at varying positions and scales. The tansformation $\{T_\alpha | \alpha \in A\}$ parameterized by vectors α and A is a set of random possible tranformation. Therefore, each tranformation T_α is a combination of each transformation below:

1. Rotation with random degree in range $[-20, 20]$
2. Translate random (2.0)horizontally and vertically in range $[0, 0.1]$ of patch size
3. Scale with random degree in range $[0.7, 1.4]$ of patch size

The procedures mentioned above are applied to unlabel images after normalization. Figure 3.1 showes the example surrogate figures. According to the reference paper, they construct the convolution neuron network for either 64c-64c-128 or 64c5-128c5-256c5-512f, which letter c represent number of convolutional filters and f represent the number of fully connected units. As we randomly selected 4000 unlabel images and create 100 surrogate set per image, we got total 400000 patches which belongs to 4000 class. The techinical difficulity for this method is the time complexity because the reference paper took 4 days to train the second model, and when we did face the issue when contruct and train the model. Therefore, we realized this model may not suitable for this assignment and decided to move to other model.

2.2. Method: Kmeans Centroids. We implemented Kmeans methods as one of our model, since based on previous paper that they can reach very good performance based on this simple model. The idea can divided into two part: first for getting centroids, and the second is perform feature mapping and perform classification. Regarding to the centroids identification, we did the following three steps:

1. Randomly select certain amount of unlabeled training images, and extract random patches. The size of patch is 22x22
2. Apply pre-processing to patches, including normalization and whitening
3. Learn feature-mapping using unsupervised method, and here we use kmeans due to the its good performance in the reference paper

After centroids are identified, the steps for the second stage is as followed:
1. Extract patches from input images. The size of patches is 22x22, and the gap between patches is 2, and therefore we can get 16 patches from each of imput figure. Then we perforem feature mapping as the following equation:

$$f_k(x) = max\{0, \mu(z) - z_k\} \tag{2.1}$$

where $z_k = \|x - c^{(k)}\|$ and $\mu(z)$ is the mean of the elements of z
2. Pool features together over region and perform 4 quadrants summation to reduce the number of feature values, and therefore we will get the feature in the size of $4k$ per figure, and k is the number of centroids.
3. Perform classification based on the feature vector, and here we used 1-vs-N SVM.
Figure 3.2 showes the process of the centroids identification part, and figure 2.1 shows the feature mapping and classification part

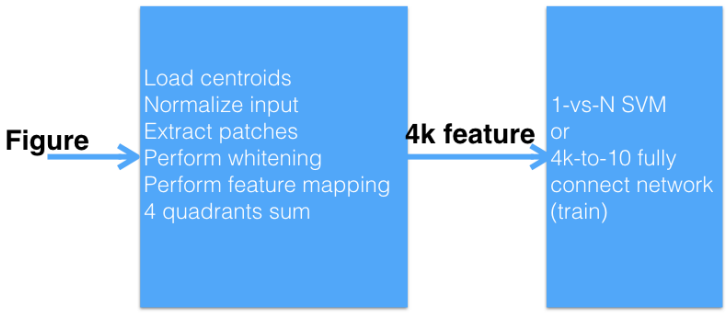

Figure 2.1. *Kmeans model*

2.2.1. Experiment: Kmeans Centroids. Here is the test

numIter	Training Accuracy (%)	Test Accuracy (%)
500	96.71	99.04
1000	99.04	99.41
10000	99.34	99.40
20000	99.53	99.47
50000	99.53	99.47
300000	99.53	99.47

Table 2.1
The training and test accuracy of the original model

2.3. Method: Pseudolabels. We started with the trained baseline model. For every 5 epochs, use the current model to predict 1000 unlabeled figures. We collected 100 figures out of 1000 having largest confidence (the largest prediction probabilities) and used the predicted classes as the real labels of these figures. Then we merged these new "labeled" figures into the training set and used them to train the model. We started with 4000 training set and terminated at 8800 training set. The training process was interrupted by "Out of Memory" for several times, so the training result is splited into several pictures. From the training result, we found this method didin't help to improve the validation accuracy.

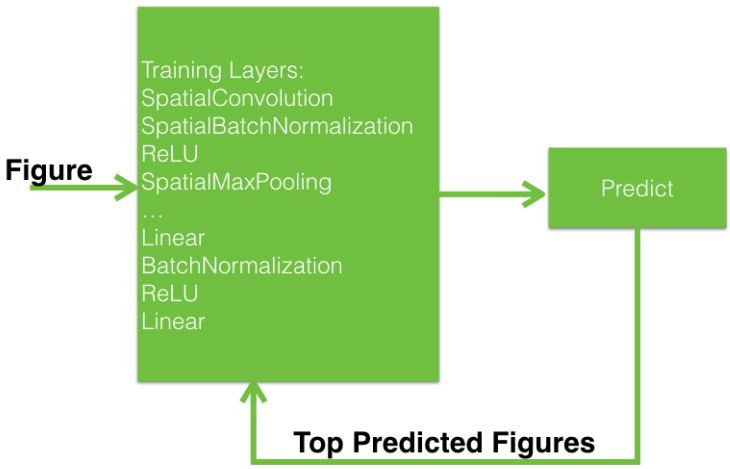

Figure 2.2. *Pseudolabels model*

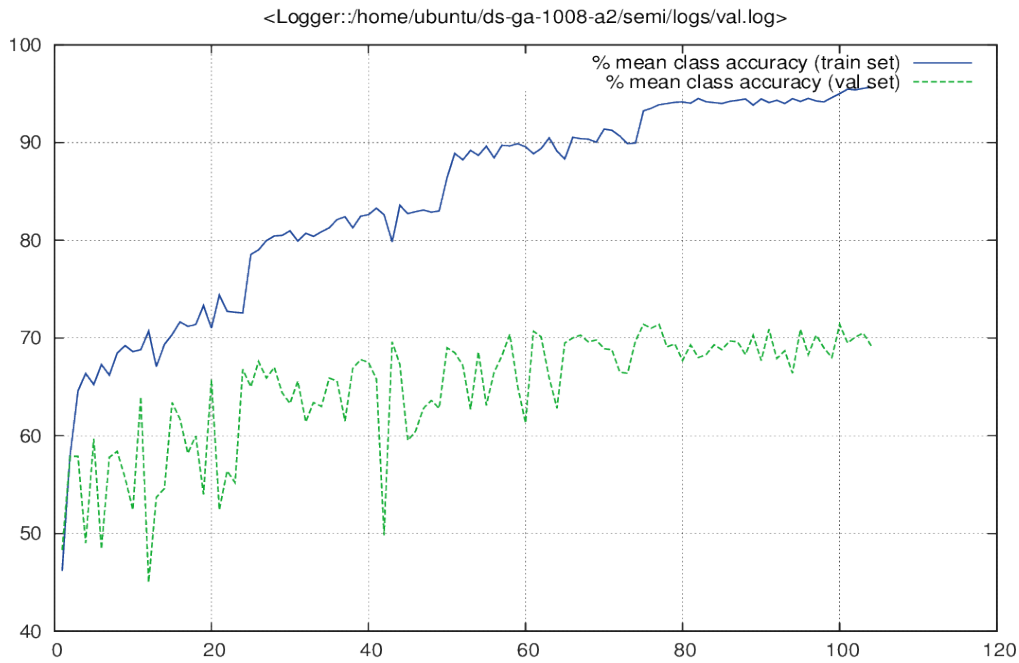

Figure 2.3. *Pseudolabels training result: training set start from 5000 figures and end at 7100.*

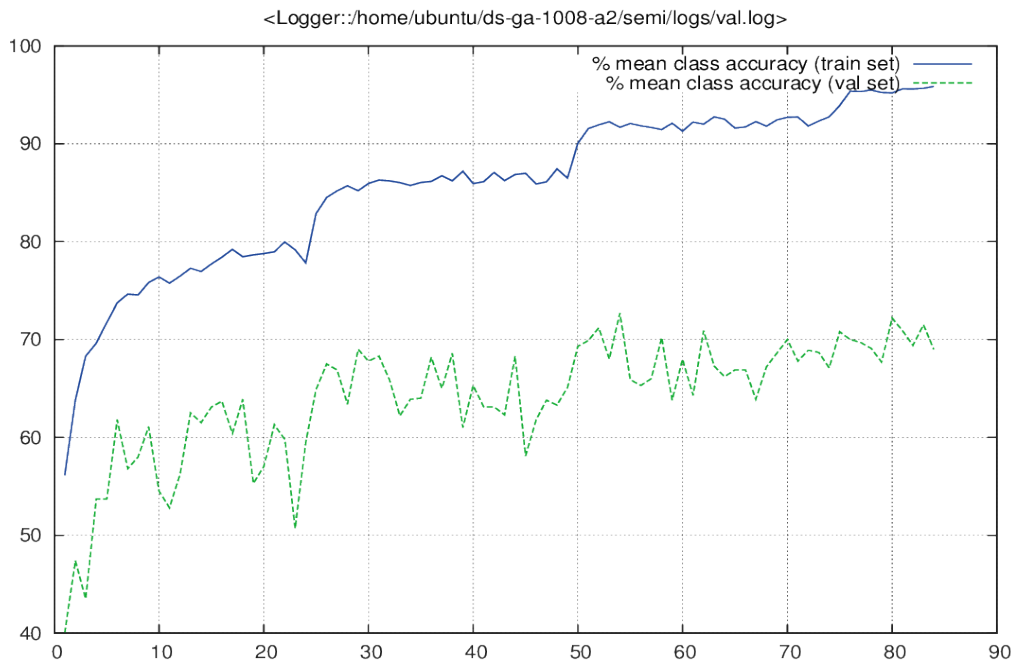

Figure 2.4. *Pseudolabels training result: training set start from 7100 figures and end at 8800.*

2.4. Method: Kmeans Centroids + Pseudolabels.

3. Visualization.

3.1. Visualizing filters and augmentations. Initially we were also trying to creat surrogate data set from 4000 unlabel figures, each figure will produced 100 surrogate figures and therefore we got 4000 class of 400000 figures in total. The initial figre is in the size of 3x96x96, and we created the surrogate figure by the size of 3x32x32 and each will subject in a random degree of rotation between -20 to 20, vertical and horizontal translate between 0 to 0.1, and scale in the range between 0.7 to 1.4. Figure 3.1 is the visualization of the surrogate figures.

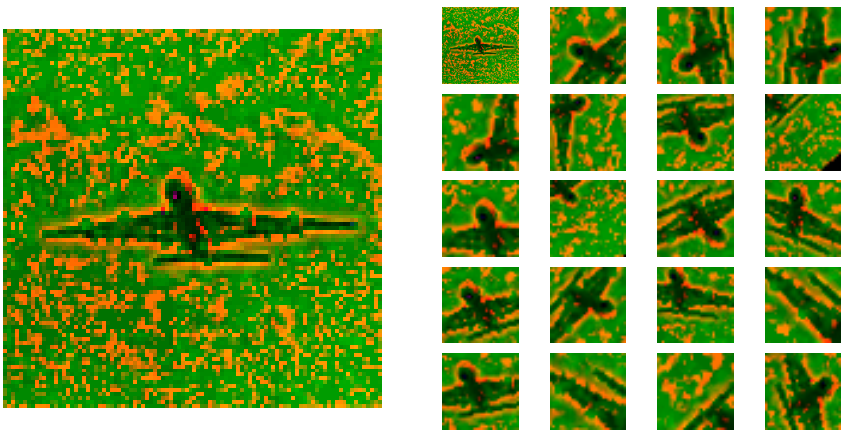

Figure 3.1. *Figures of surrogate set*

Then we implemented the kmeans method. According to the paper, they randomly extracted patches from unlabeled set of data, and performed kmean to find the centroids. The size of patch we extracted is 22x22 and we extracted 16 patches from total of 20000 unlabel figures. Number of centroids we obtained is 1600 because it produced good accuracy as the paper mentioned. As we can see from figure 3.2, most of the centroids are in very sharp of edge and color blobs.

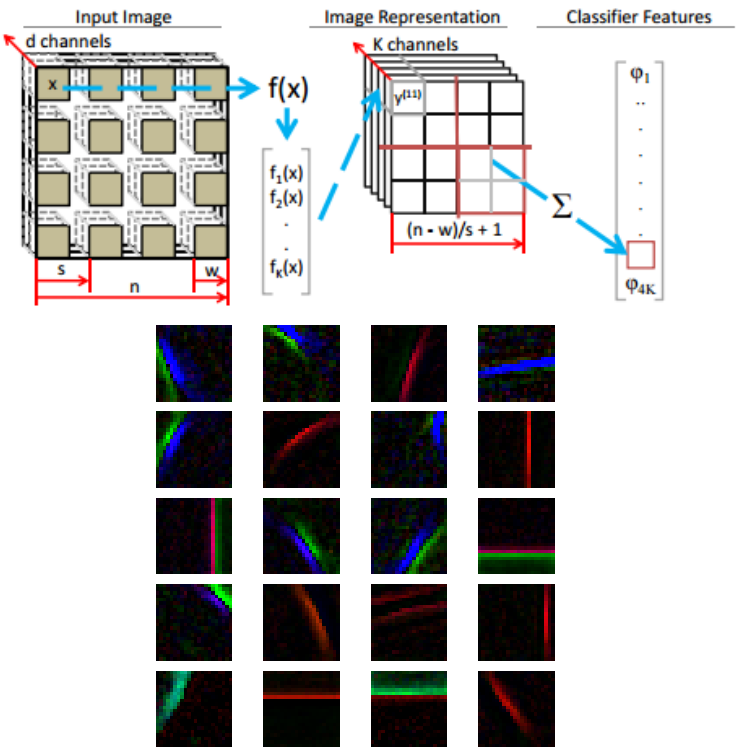

Figure 3.2. *Figure of centroids*

3.2. t-SNE. Here we took all the images from val.t7b, which contains 1000 figures for total and 100 figures in each of class. To generate t-SNE embedding, we used only the first channel of each of the image, which the dimension is 1x96x96, and feed it into manifold.embedding.tsne. Then we will get the mapping result for each of the image onto the 2D space, and we can plot the figures based on the mapped coordinate.

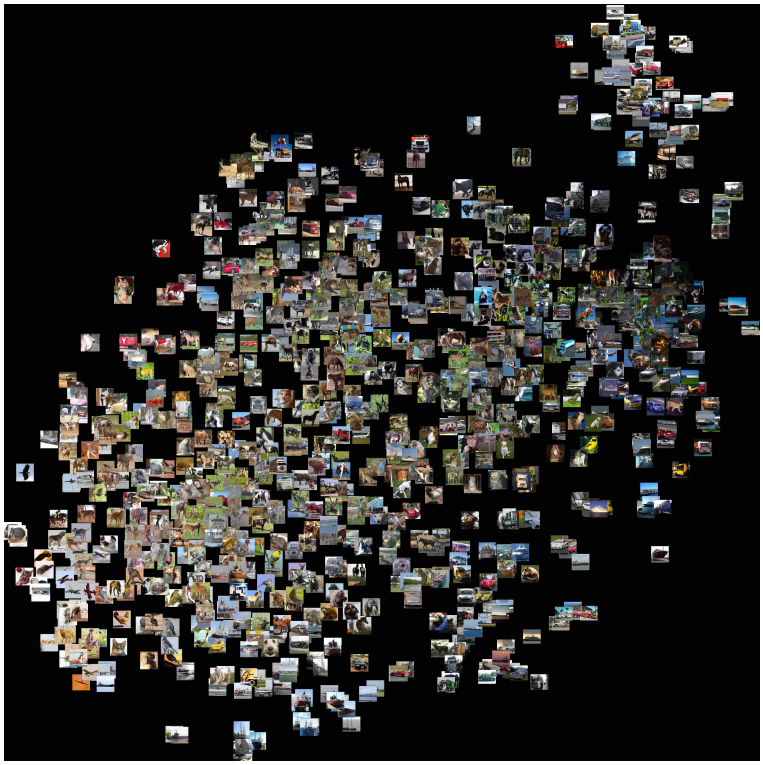

Figure 3.3. *The training and test accuracy of 3-layer versus 2-layer model*