

# 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:

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

which can also rewrite as:

$$y = \frac{1}{1 + e^{-x_1}} + \frac{1}{1 + e^{-x_2}} \quad (1.2)$$

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

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

## 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  $\beta$  of size  $m$

$$\beta = \{x_{1\dots m}\} \quad (1.4)$$

mean of this mini-batch:

$$\mu_\beta = \frac{1}{m} \sum_{i=1}^m x_i \quad (1.5)$$

variance of this mini-batch:

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

By definition:

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

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

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

---

\*yss265@nyu.edu

†cly264@nyu.edu

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

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

**1.2.2. Shift variable.** Let  $\varepsilon$  be a learnable shift variable, we can define the formula as: forward pass:

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

backward pass:

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

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

## 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 randomly choose 4000 unlabeled images and obtain patches from different images at varying positions and scales. The transformation  $\{T_\alpha | \alpha \in A\}$  parameterized by vectors  $\alpha$  and  $A$  is a set of random possible transformations. Therefore, each transformation  $T_\alpha$  is a combination of each transformation below:

1. Rotation with random degree in range  $[-10, 10]$
2. Translate random horizontally and vertically in range  $[0, 0.1]$  of patch size
3. Scale with random degree in range  $[1, 1.4]$  of patch size
4. Randomly decide to perform hflip

The procedures mentioned above are applied to unlabeled images after normalization. Figure 3.1 shows the example surrogate figures. According to the reference paper, they constructed the convolutional neural network for either 64c-64c-128 or 64c5-128c5-256c5-512f, where letter  $c$  represent number of convolutional filters and  $f$  represent the number of fully connected units. As we randomly selected 4000 unlabeled images and created 100 surrogate sets per image, we got a total of 400000 patches which belongs to 4000 classes. The technical difficulty 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 constructing and training the model. Therefore, we realized this model may not be suitable for this assignment and decided to move to other models.

---

**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:

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

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 demonstrate the procedures mentioned above, and also shows the example of centroids during the processes. As we can see from Table 2.1, when the number of 1600 centroids is fixed and changing the number of 1-vs-N SVM iterations, the accuracy is improving but not good enough. We are thinking there are few points need to take care more delicatly, for example the way of normalization, whitening, and 4 quadrants for dimension reduction.

numIter	Training Accuracy (%)	Test Accuracy (%)
500	20.3	16.1
1000	22.4	18.1
10000	26.2	18.5

Table 2.1

The training and test accuracy of kmeans implementation based on reference paper

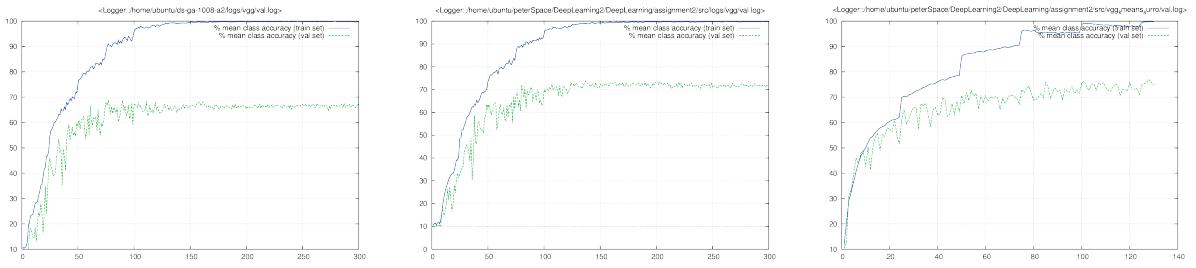
**2.3. Method: VGG+Kmeans+Surrogate.** Here we decide not to following the exact methoded mentioned in the reference papers, but to combine their high level ideas together. For example, the idea of kmeans is to find the centroids which will be a good representative for the small patches. Also, the idea of surrogate is to increase the size of training set by randomly transformation and therefore may give training model more different angles to learn the figure and better to predict the unlabel figures. For the left figure at figure 2.1, as we initially running the original vgg model, the training set is keep improving but not validation set after

epoch 50. This suggests the training set may found a point that is good to optimize toward it but seems it's just a local minimum as we can see it not able to improve the validation set. As the training process is pretty much all about updating the appropriate weighting, we think assigning a proper initial weighting will help the optimization away from local but more toward global minimum. Therefore, as centroids from the kmeans method is a good representation of patches, we decide to assign the centroids to the weighting of the first convolution layer in vgg model. The first layer performs SpatialConvolution from 3 input plane to 64 output plane with 3x3 patch size, and therefore the weighting of this layer is in the size of 64x3x3x3. To match the size of this layer, we used almost the same method mentioned above, but to randomly select 10 patches of size 3x3 from each of label figures, and run kmeans for 64 centroids. After those 64 centroids are found, we assigned the value to the first SpatialConvolution layer before the training. As we can see from the second figure in figure 2.1, generally the value for the validation set is lifting up from 68% in original vgg model to 72%.

However, we can still see the phenomenon that the training set keep improving but not the validation set, and we think the kmeans initialization indeed gave it a good start to train, but now the limitation is still the size of training set cannot reflect to the unlabeled data to give the proper prediction. Therefore, we decide to create 10 surrogate figures with the transformation mentioned above from each of the 4000 label figures, and therefore now we get 44000 training set with the original figure and surrogate figure included. The training curve become very interesting. First of all we can see the prediction for validation set sometimes even outperform the training set, and also we can see the prediction gap between training and validation set are much close than the other models. However, limitation can still be observed after epoch 50 that training still improving but not the validation set, suggesting surrogate indeed provide improvement but addition new figures may still required for model to learn from something new and potentially will need to include the method of Pseudolabels.

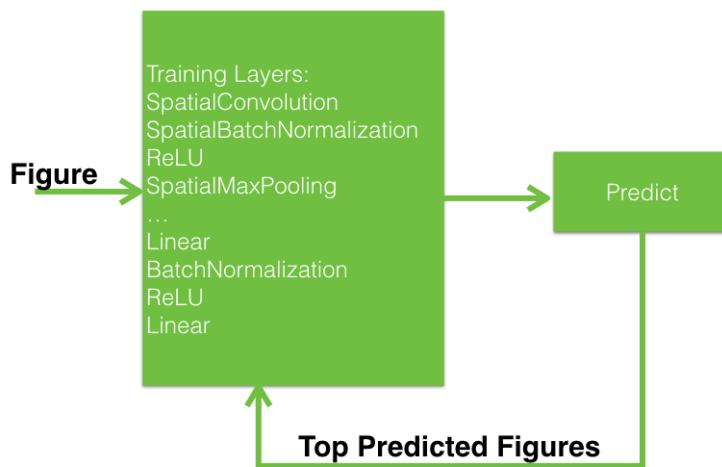
Method	Best Validation Accuracy (%)
vgg	68.6
vgg+kmeans	73.7
vgg+kmeans+surrogate	76.9

**Table 2.2**  
*Accuracy of different model*

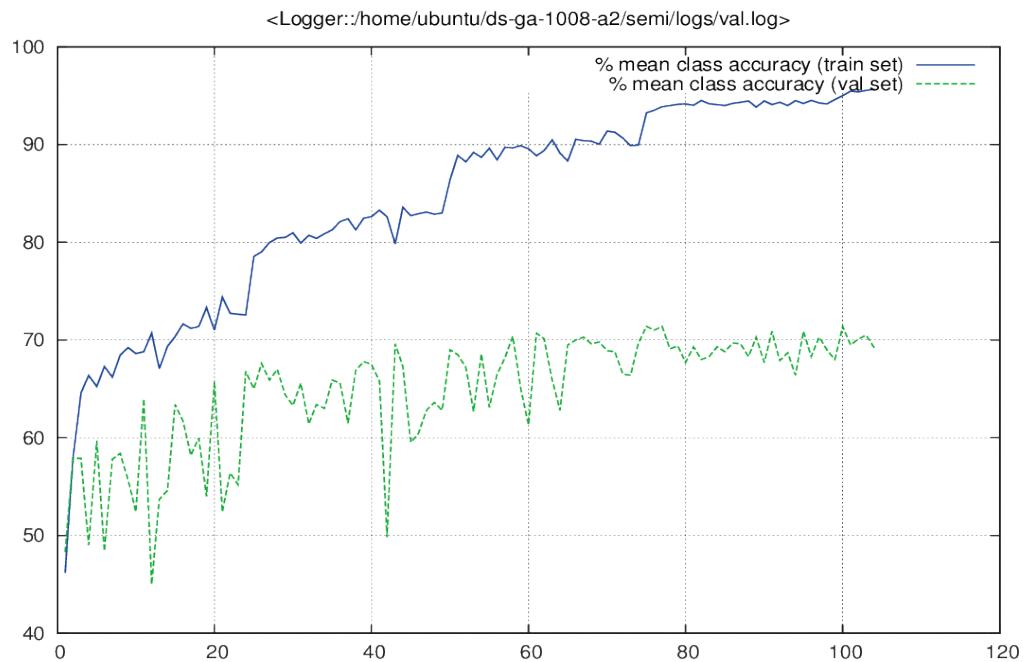


**Figure 2.1.** Training curve for original vgg, vgg+kmeans, and vgg+kmeans+surrogate

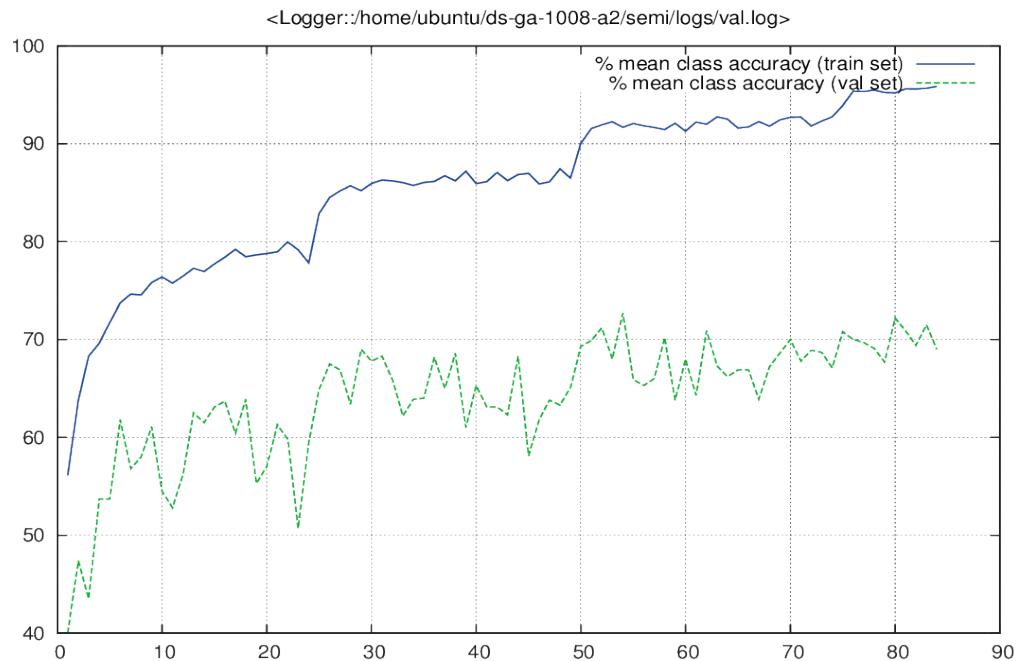
**2.4. Method: Pseudolabels.** We started with the trained baseline model. For every 5 epochs, use the current model to predict 1000 unlabeled figures. We collected 100 figures out of 1000 having largest confidence (the largest prediction probabilities) and used the predicted classes as the real labels 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 split into several pictures. From the training result, we found this method didn'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.



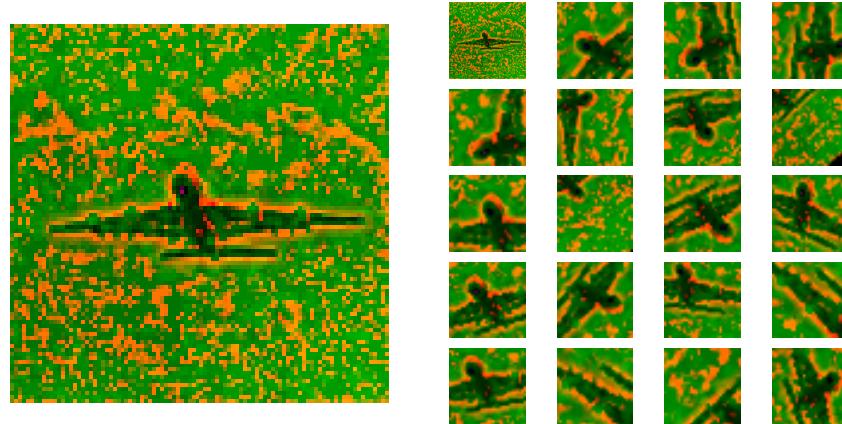
**Figure 2.4.** Pseudolabels training result: training set start from 7100 figures and end at 8800.

---

## 2.5. Method: Kmeans Centroids + Pseudolabels.

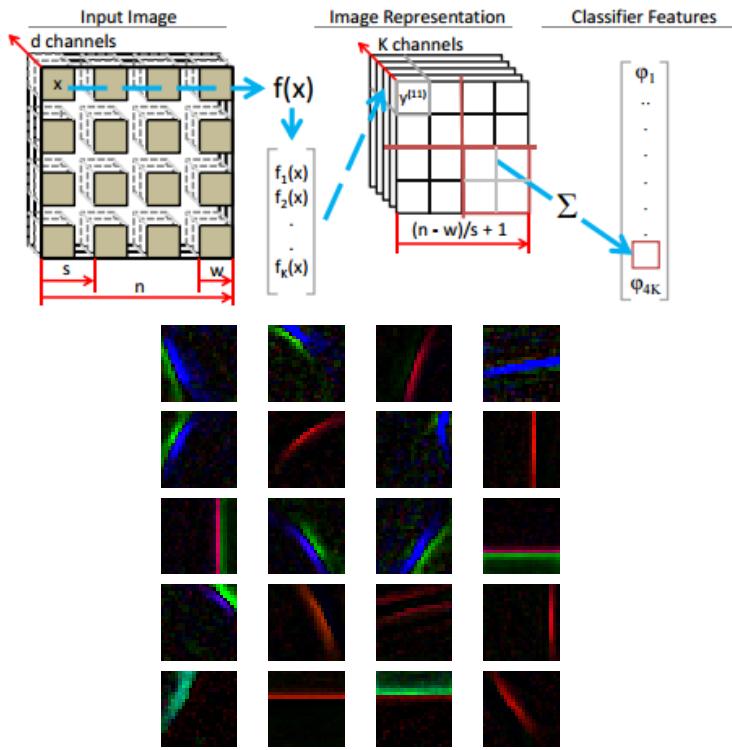
### 3. Visualization.

**3.1. Visualizing filters and augmentations.** Initially we were also trying to create surrogate data set from 4000 unlabeled figures, each figure will produce 100 surrogate figures and therefore we got 4000 class of 400000 figures in total. The initial figure 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 unlabeled 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  $1 \times 96 \times 96$ , 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.** t-SNE Figure of original, layer before full-connected network, and layer after full-connected network