Deep Learning Teaching Kit
Lab 2, Sample Solution 2

1 More Backpropagation

1.1 Backpropagation through a DAG of modules

According to the DAG,

$$y = max(\sigma(x_1), \sigma(x_2)) + min(\sigma(x_1), \sigma(x_2)) = \sigma(x_1) + \sigma(x_2)$$

Suppose the error passed down to y is $\frac{\partial E}{\partial y}$. Write the values for $\frac{\partial E}{\partial x_1}$ and $\frac{\partial E}{\partial x_2}$ in terms of $\frac{\partial E}{\partial y}$.

Answer:

$$\begin{aligned}\frac{\partial E}{\partial x_1} &= \frac{\partial E}{\partial y} \cdot \frac{\partial y}{\partial x_1} \\ &= \frac{\partial E}{\partial y} \cdot \frac{\partial[\sigma(x_1) + \sigma(x_2)]}{\partial x_1} \\ &= \frac{\partial E}{\partial y} \cdot [1 - \sigma(x_1)] \cdot \sigma(x_1)\end{aligned}$$

$$\begin{aligned}\frac{\partial E}{\partial x_2} &= \frac{\partial E}{\partial y} \cdot \frac{\partial y}{\partial x_2} \\ &= \frac{\partial E}{\partial y} \cdot \frac{\partial[\sigma(x_1) + \sigma(x_2)]}{\partial x_2} \\ &= \frac{\partial E}{\partial y} \cdot [1 - \sigma(x_2)] \cdot \sigma(x_2)\end{aligned}$$

1.1.1 Batch Normalization [3]

Let m represents for batch size. Let μ_B represents for $E(x_k)$ and σ_B represents for $\sigma(x_k)$, which are mean and standard deviation of input batch respectively. Then we have,

$$y_k = \frac{x_k - \mu_B}{\sqrt{\sigma_B^2}}$$

(i) Given $\frac{\partial E}{\partial y_k}$, write down $\frac{\partial E}{\partial x_k}$.

$$\begin{aligned}\frac{\partial E}{\partial x_k} &= \frac{\partial E}{\partial y_k} \cdot \frac{\partial y_k}{\partial x_k} + \frac{\partial E}{\partial \mu_B} \cdot \frac{\partial \mu_B}{\partial x_k} + \frac{\partial E}{\partial \sigma_B^2} \cdot \frac{\partial \sigma_B^2}{\partial x_k} \\ &= \frac{\partial E}{\partial y_k} \cdot \frac{1}{\sqrt{\sigma_B^2}} + \frac{\partial E}{\partial \mu_B} \cdot \frac{1}{m} + \frac{\partial E}{\partial \sigma_B^2} \cdot \frac{2(x_k - \mu_B)}{m}\end{aligned}$$

in which,

$$\frac{\partial E}{\partial \mu_B} = \sum_{k=1}^{m} \frac{\partial E}{\partial y_k} \cdot \frac{\partial y_k}{\partial \mu_B} + \frac{\partial E}{\partial \sigma_B^2} \cdot \frac{\partial \sigma_B^2}{\partial \mu_B}$$
$$= \sum_{k=1}^{m} \frac{\partial E}{\partial y_k} \cdot [-\frac{1}{\sqrt{\sigma_B^2}}] + \frac{\partial E}{\partial \sigma_B^2} \cdot \frac{\sum_{k=1}^{m} -2(x_k - \mu_B)}{m}$$

$$\frac{\partial E}{\partial \sigma_B^2} = \sum_{k=1}^{m} \frac{\partial E}{\partial y_k} \cdot \frac{\partial y_k}{\partial \sigma_B^2}$$
$$= \sum_{k=1}^{m} \frac{\partial E}{\partial y_k} \cdot (x_k - \mu_B) \cdot (-\frac{1}{2}) \cdot (\sigma_B^2)^{-\frac{3}{2}}$$

(ii) Introduce a shift variable ϵ, write down the forward and backward pass and derive $\frac{\partial E}{\partial \epsilon}$. We introduced a shift variable and got,

$$y_k = \frac{x_k - \mu_B}{\sqrt{\sigma_B^2}} + \epsilon$$

And,

$$\frac{\partial E}{\partial \epsilon} = \sum_{k=1}^{m} \frac{\partial E}{\partial y_k} \cdot \frac{\partial y_k}{\partial \epsilon} = \sum_{k=1}^{m} \frac{\partial E}{\partial y_k}$$

Then, the forward pass is,

$$y_k = \frac{x_k - \mu_B}{\sqrt{\sigma_B^2}} + \epsilon$$

And the backward pass is,

$$\frac{\partial E}{\partial \mu_B} = \sum_{k=1}^{m} \frac{\partial E}{\partial y_k} \cdot [-\frac{1}{\sqrt{\sigma_B^2}}] + \frac{\partial E}{\partial \sigma_B^2} \cdot \frac{\sum_{k=1}^{m} -2(x_k - \mu_B)}{m}$$

$$\frac{\partial E}{\partial \sigma_B^2} = \sum_{k=1}^{m} \frac{\partial E}{\partial y_k} \cdot (x_k - \mu_B) \cdot (-\frac{1}{2}) \cdot (\sigma_B^2)^{-\frac{3}{2}}$$

$$\frac{\partial E}{\partial x_k} = \frac{\partial E}{\partial y_k} \cdot \frac{1}{\sqrt{\sigma_B^2}} + \frac{\partial E}{\partial \mu_B} \cdot \frac{1}{m} + \frac{\partial E}{\partial \sigma_B^2} \cdot \frac{2(x_k - \mu_B)}{m}$$

$$\frac{\partial E}{\partial \epsilon} = \sum_{k=1}^{m} \frac{\partial E}{\partial y_k}$$

2 STL-10: semi-supervised image recognition

2.1 Data

The STL-10 dataset has 5000 labeled data samples and 100000 unlabeled data samples. We used 4000 labeled data samples for training and 1000 labeled data samples for validation. The extra 100000 unlabeled data samples are used for semi-supervised learning to improve the performances of purely supervised modeling.

- Training: $4,000$
- Validation: $1,000$
- Extra: $100,000$

2.2 Model Structure

Our convolution neural network architecture is composed of five stacked feature stages. Each stage contains a convolution module, followed by a normalization module, a RELU module and a max-pooling module. And the final stage is a 2-layer fully connected neural network to do classification.

2.3 Training

We implemented the model on Torch and trained on NVIDIA GPUs. As shown before, we split the training data into training and validation and we controlled the generalization of the model by referring its performances on validation data. We also used unsupervised approach on the extra data to improve the modeling performances. We trained our model using stochastic gradient descent with batch size 32. And training parameters are learning rate (1), learning rate decay (1e-7), weight decay (0.0005) and momentum (0.5). Those training parameters are chosen by adapting the suggestion from work [1].

2.4 Experiment and Analysis

2.4.1 Data Augmentation

Since our training dataset is limited, we considered data augmentation as an important part to improve modeling performance. We considered following augmentations [4]:

- flip: horizontal flipping;
- rotation: clockwise or anticlockwise rotation of the image by an image by an angle up to 15 degrees;
- scaling: multiplication of the patch scale by a factor between 1 and 1.4;
- translation: vertical or horizontal translation by a distance within 0.2 of the patch size.

Examples of transformed versions of images are shown in figure 3. As table 1 shows, the data augmentation has great effect on performance improvement. The baseline model, without augmentation, has an accuracy of 71.4% on validation set. After adding data augmentation, the accuracy is improved to 80.9% on validation set.

2.4.2 Unsupervised Pre-training

As work [2] shows, using k-means clustering centroids to initialize first-layer weights improves the performances on CIFAR-10 and NORB dataset. We applied this approach to STL-10 dataset. Specifically, our steps are:

- randomly crop 3*3 patches from unlabeled images (20 patches per image for 10000 images);
- run K-means and get 64 centroids;
- use K-means centroids to initialize the first layer weights;
- progressively initialize weights of layers by this way (we only did this up to the second layer).

The K-means centroids of 3*3 and 7*7 patches to initialize the first layer are shown in figure 1. Table 1 shows that there are some effects when using K-means pre-training. The pure supervised learning model (with augmentation) has an accuracy of 80.9%. After using K-means centroids to initialize the first layer, the accuracy is improved to 83.9%. And after convolutionally using it to additionally initialize the second layer, the accuracy is further improved to 84.4%.

2.5 Visualization

2.5.1 Visualizing filters and augmentations

(i) Visualizing first layer filters
We visualized the first layer filters in figure 2. Since the size of first layer filter is 3 by 3, features

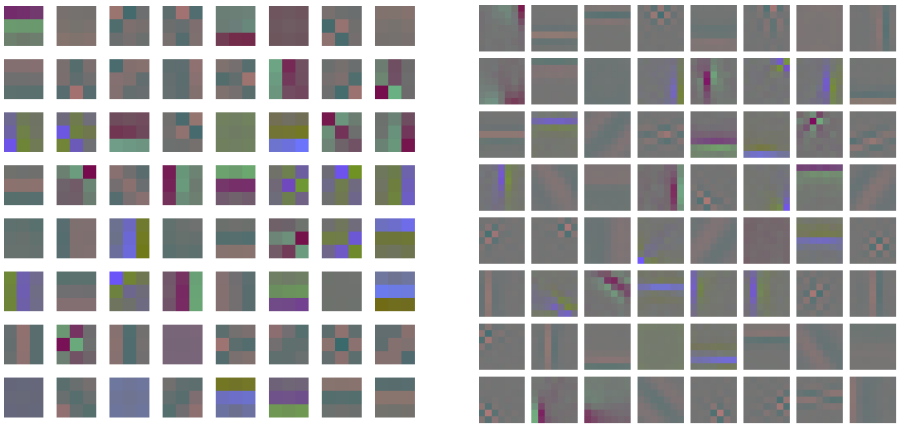

Figure 1: Left: Kmeans centroids of 3*3 patches. Right: Kmeans centroids of 7*7 patches.

are not so obvious to distinguish. But we can still see some edges and color blobs.

(ii) Visualizing a sample of augmentations

As we mentioned before, we included flipping, rotation, translation and scaling as augmentation methods. We randomly applied these augmentations in 36 images. The result is shown in figure 3, the left are original images and the right are transformed images.

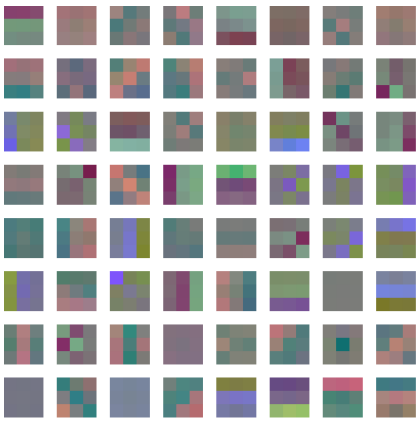

Figure 2: Visualization of first layer's filters.

2.5.2 t-SNE

We examined the learned features by clustering testing samples using the feature maps outputted from each feature extraction layer of a seven-convolution-layer model. If the clusters show more separated, it means that the features are more discriminative. In figure 4 (left) we use the raw image data as the feature vectors in t-SNE clustering. Result seems like there is no clear separation, but it does have different groups. For example, in the middle we have animals, on the bottom we have flights, and on the left we have steamships.

The discrimination becomes stronger as we use deeper features. Figure 4 (right) to figure 6 (left) are generated by t-SNE using outputs from the first convolution layer to the forth. The effects are really amazing even when we just use one single layer of convolution. The grouping at first seems based on the small features, like what we see in figure 4 (right) the images with similar color intensities stay together. As we go for deeper features, the feature vectors include more larger scale shape information, like the result in figure 6 (left).

Figure 3: Left: Original images. Right: Transformed images.

The most exciting part happens when we use the deepest features from the convolution layers just before the prediction softmax later. From figure 6 (right) to figure 7 (right), we can observe the amazing separation by t-SNE using deep features. In figure 6 (right), there are generally two groups, one is the group of transports and the other one is animals. Within each group there are sub-groups, but it is not such clear at this moment. The more clear grouping happens in figure 7 (left). It's obvious that in the group of transports, the vehicles, flights, and steamships are separated, and in the group of animals, the monkeys and horses also hold opposite positions. Figure 7 (right) displays our deepest features from the last convolution layer. The result is amazingly impressive. There are 4 groups separated for the transports on the right, and 6 groups for animals on the left. This result of the unsupervised clustering is consistent with our supervised information saying that 10 different labeled groups exist. This final result shows that we have successfully learned the features from both labeled and unlabeled data.

Figure 4: Left: Clustering based on raw images. Right: Clustering based on 1st CNN layer.

2.6 Results

Model	Baseline	+A	+A+Kmeans1	+A+Kmeans1&2
Accuracy	71.4%	80.9%	83.9%	84.4%

Table 1: Experiments Results

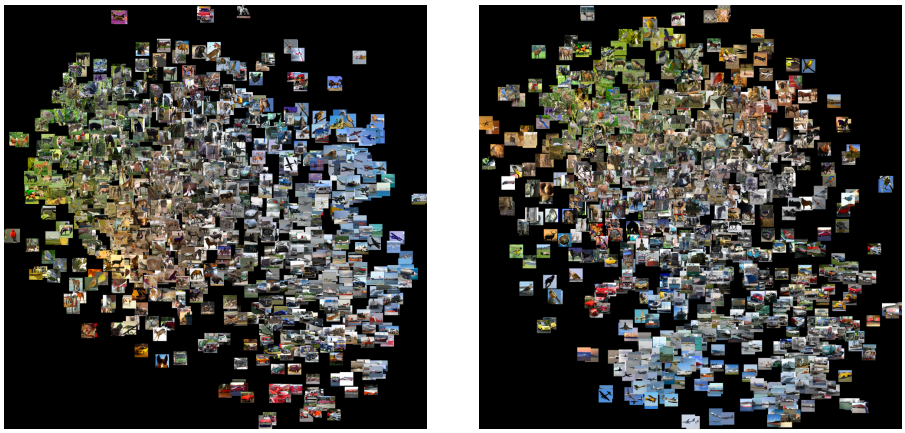

Figure 5: Left: Clustering based on 2nd CNN layer. Right: Clustering based on 3rd CNN layer.

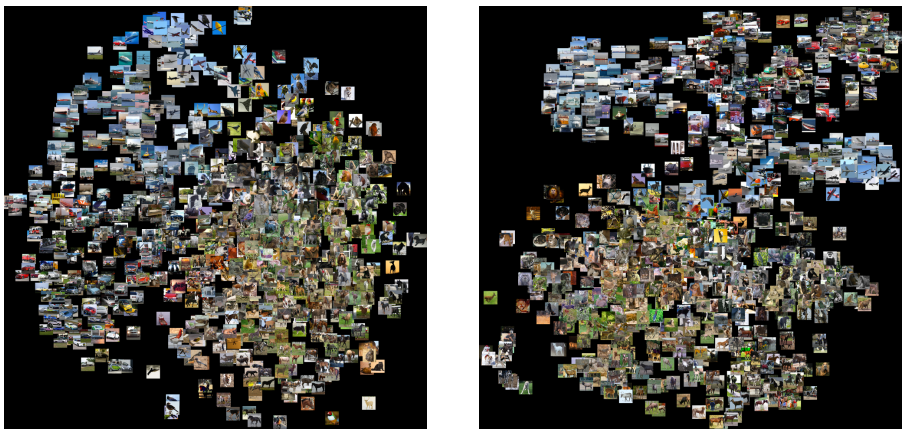

Figure 6: Left: Clustering based on 4th CNN layer. Right: Clustering based on 5th CNN layer.

2.7 Conclusion

The results display that data augmentation is generally very useful in image classification. Starting from baseline model, data augmentation gives 9% improvement in accuracy. In addition, when the labeled data is limited, using unlabeled data by unsupervised learning is also effective in improving the performance. In our work, we used KMeans to initialize the first layer's weight, which improves accuracy by 3%. By further initializing the second layer's weight, we improves accuracy by additional 0.5%. This shows that using KMeans to initialize layers' weights guides to a better local minimum.

References

[1] Yoshua Bengio. Practical recommendations for gradient-based training of deep architectures. In *Neural Networks: Tricks of the Trade*, pages 437–478. Springer, 2012.

[2] Adam Coates, Andrew Y Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In *International conference on artificial intelligence and statistics*, pages 215–223, 2011.

[3] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

Figure 7: Left: Clustering based on 6th CNN layer. Right: Clustering based on 7th CNN layer.

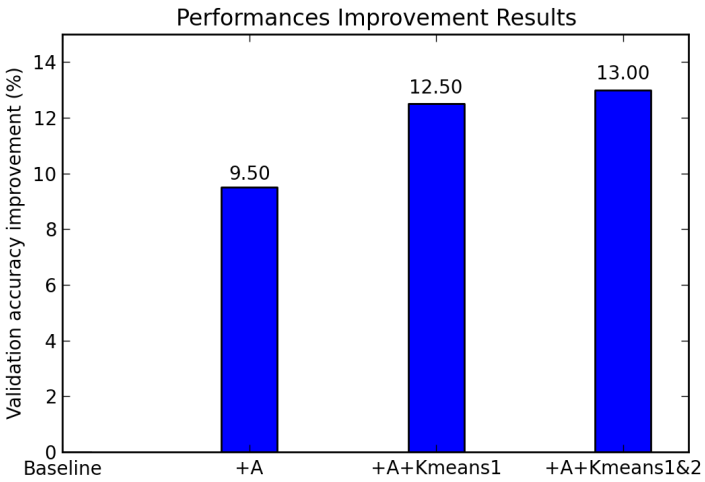

Figure 8: Validation accuracy improvement over basic CNN (absolute %). +A: plus augmentation, including flipping, rotation, translation and scaling; +Kmeans1: plus using Kmeans centroids as initialization of the first layer's weights; +Kmeans1&2: plus using Kmeans centroids as initialization of the first and the second layers' weights.

[4] Tom Le Paine, Pooya Khorrami, Wei Han, and Thomas S Huang. An analysis of unsupervised pre-training in light of recent advances. *arXiv preprint arXiv:1412.6597*, 2014.