Computer Vision

Deep Clustering for Unsupervised Learning of Visual Features

Project Objective

- To implement an end-to-end training of visual features on large scale dataset which requires little domain knowledge and no specific signal from the inputs.

- Deep Clustering is a clustering method that jointly learns the parameters of a neural network and the cluster assignments of the resulting features.

- Paper Link: Facebook Deepcluster

Overview

- f_θ the convnet mapping, where θ is the set of corresponding parameters.
- Given a training set X = $\{x_1, x_2, \ldots, x_N\}$ of N images, we want to find a parameter θ* such that the mapping $f_\theta{}^*$ produces good general-purpose features.
- Traditionally, each image x_n is associated with a label y_n in $\{0, 1\}^k$
- A parametrized classifier g_W predicts the correct labels on top of the features $f_\theta(x_n)$. The parameters W of the classifier and the parameter θ of the mapping are then jointly learned by optimizing the following problem:

$$\min_{\theta, W} \frac{1}{N} \sum_{n=1}^{N} \ell\left(g_W\left(f_\theta(x_n)\right), y_n\right),$$

where l is the multinomial logistic loss, also known as the negative log-softmax function. This cost function is minimized using mini-batch stochastic gradient descent and backpropagation to compute the gradient

Overview

- A multilayer perceptron classifier on top of the last convolutional layer of a random AlexNet achieves 12% in accuracy on ImageNet while the chance is at 0.1%(convolutional structure gives a strong prior on the input signal)
- Will exploit this weak signal to bootstrap the discriminative power of a convnet.
- Cluster the output of the convnet and use the subsequent cluster assignments as “pseudo-labels” to optimize the previous equation (iteratively learns the features and groups them).
- For clustering algorithm will use K-Means.

Overview

- K-Means jointly learns a d × k centroid matrix C and the cluster assignments yn of each image n by solving the following problem:

$$\min_{C\in\mathbb{R}^{d\times k}} \frac{1}{N}\sum_{n=1}^{N} \min_{y_n\in\{0,1\}^k} \|f_\theta(x_n) - Cy_n\|_2^2 \quad \text{such that} \quad y_n^\top 1_k = 1.$$

Solving this problem provides a set of optimal assignments $(y_n{}^*)_{n\leq N}$ and a centroid matrix C^* . These assignments are then used as pseudo-labels; we make no use of the centroid matrix.

- In summary we alternates between clustering the features to produce pseudo-labels using Eq. (2) and updating the parameters of the convnet by predicting these pseudo-labels using Eq. (1).

Proposed Method

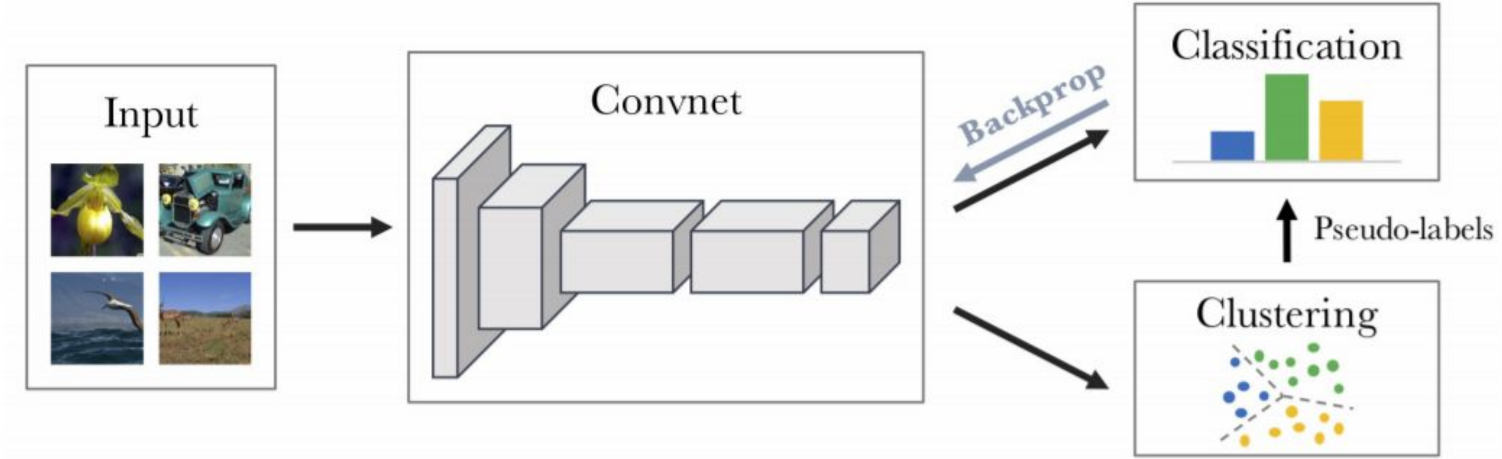
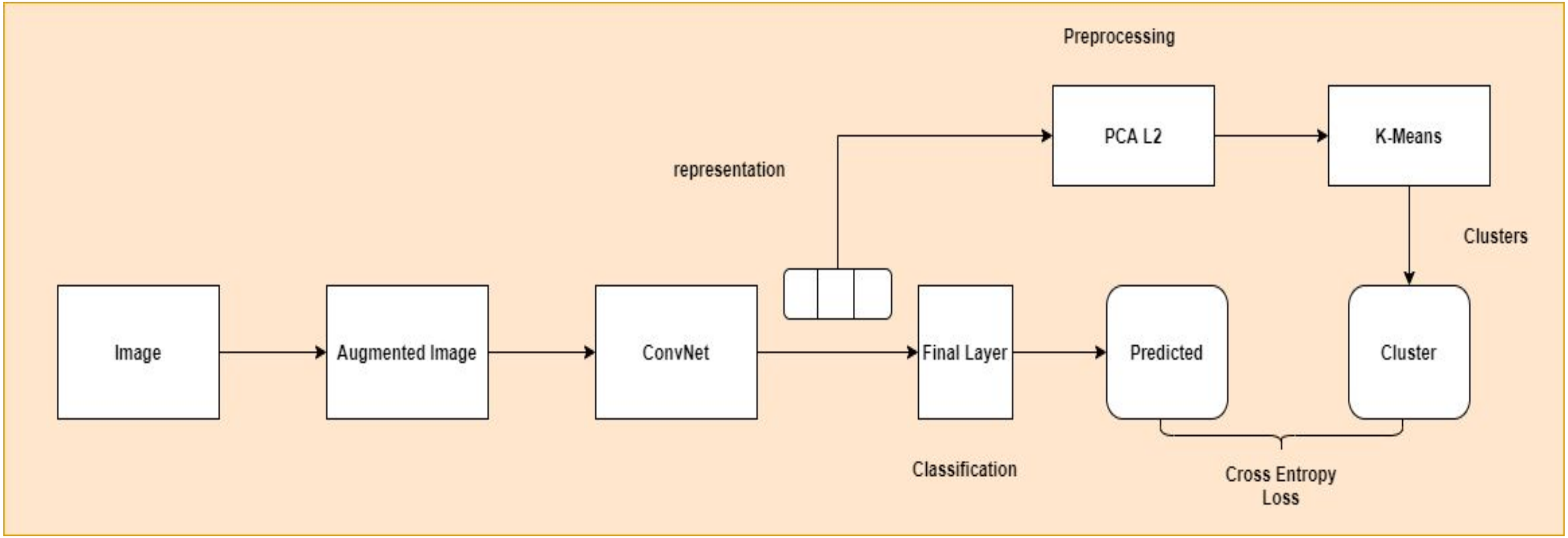


Fig. 1: Illustration of the proposed method: we iteratively cluster deep features and use the cluster assignments as pseudo-labels to learn the parameters of the convnet.

Pipeline Overview



Pipeline Details

1. Train Convolution Neural Net to generate features for Images
2. Cluster The features and generate pseudo labels for images with Kmeans
3. With CNN Do classification of the images
4. Calculate Loss using pseudo Labels generated from Kmeans and classification labels generated from CNN (Cross Entropy Loss)
5. Update the weights of Convolution network with this Cross Entropy loss

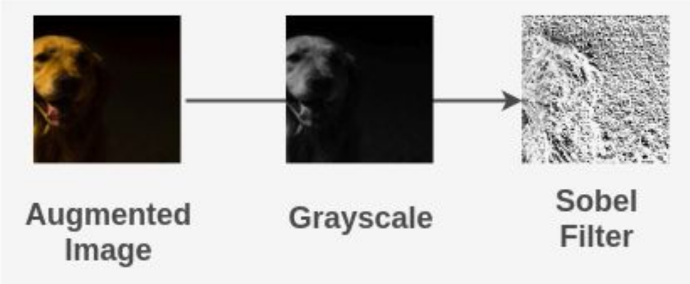
Implementation Details

1) **Images Transformations**

2. Data Augmentation

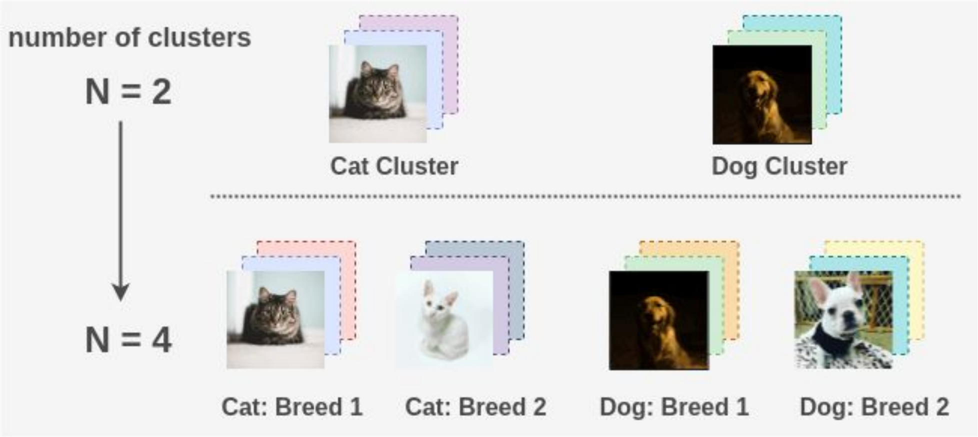
3. Filtering Image

- Sobel Filter is used to before feature generation from CNN to locate local features

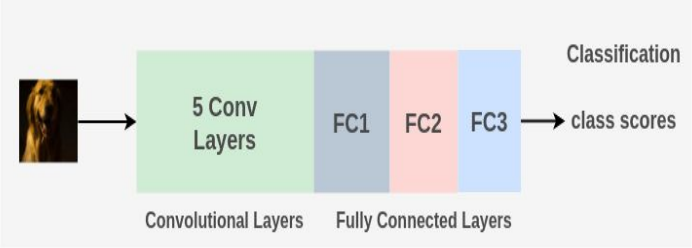


4) Clustering With Kmeans

Number Of Clusters = Number of classes in dataset



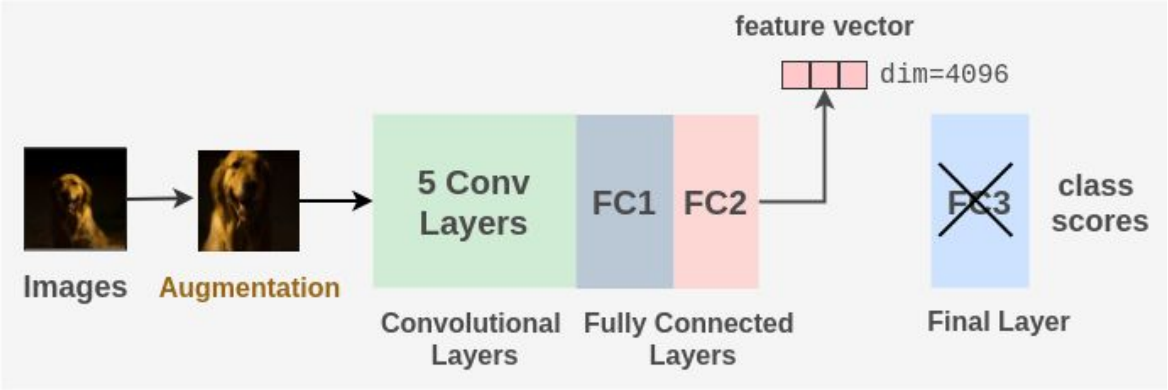
5) Model Architecture (Alexnet)



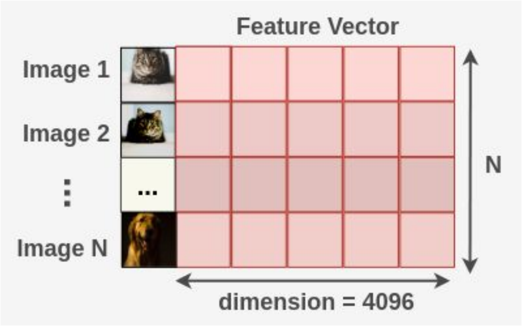

```
AlexNet(
  (features): Sequential(
    (0): Conv2d(2, 96, kernel_size=(11, 11), stride=(4, 4), padding=(2, 2))
    (1): BatchNorm2d(96, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU(inplace=True)
    (3): MaxPool2d(kernel_size=3, stride=2, padding=0, dilation=1, ceil_mode=False)
    (4): Conv2d(96, 256, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
    (5): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (6): ReLU(inplace=True)
    (7): MaxPool2d(kernel_size=3, stride=2, padding=0, dilation=1, ceil_mode=False)
    (8): Conv2d(256, 384, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (9): BatchNorm2d(384, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (10): ReLU(inplace=True)
    (11): Conv2d(384, 384, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (12): BatchNorm2d(384, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (13): ReLU(inplace=True)
    (14): Conv2d(384, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (15): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (16): ReLU(inplace=True)
    (17): MaxPool2d(kernel_size=3, stride=2, padding=0, dilation=1, ceil_mode=False)
  )
  (classifier): Sequential(
    (0): Dropout(p=0.5, inplace=False)
    (1): Linear(in_features=9216, out_features=4096, bias=True)
    (2): ReLU(inplace=True)
    (3): Dropout(p=0.5, inplace=False)
    (4): Linear(in_features=4096, out_features=4096, bias=True)
    (5): ReLU(inplace=True)
  )
  (top_layer): Linear(in_features=4096, out_features=2, bias=True)
  (sobel): Sequential(
    (0): Conv2d(3, 1, kernel_size=(1, 1), stride=(1, 1))
    (1): Conv2d(1, 2, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  )
)
```

6) Generating Pseudo Labels

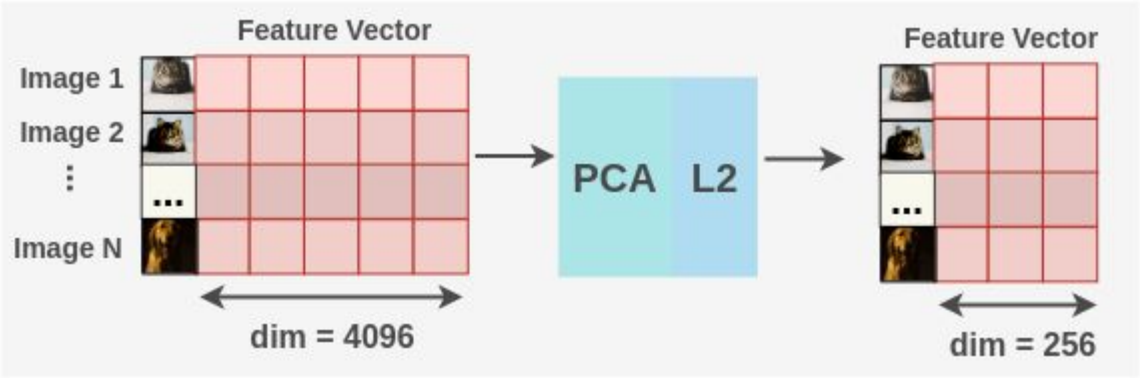
Use Convolution Layers to generate features of images from randomly initialized Alexnet



Feature Matrix :

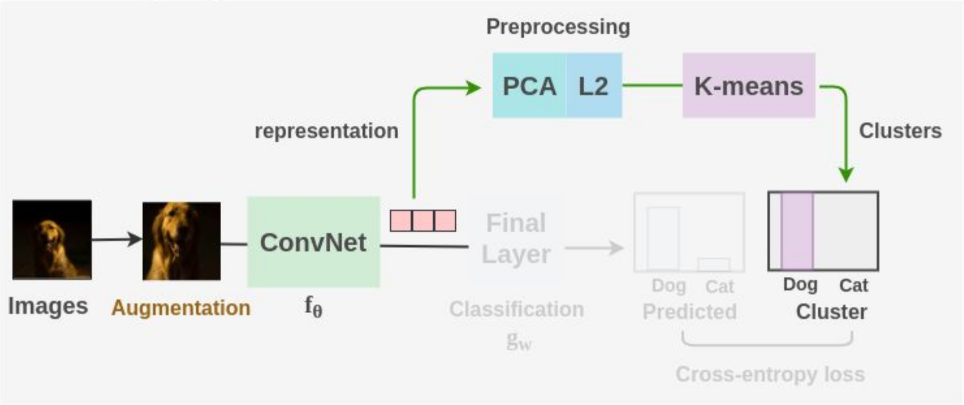


Reduce Dimension of features using PCA



Generate Pseudo Labels

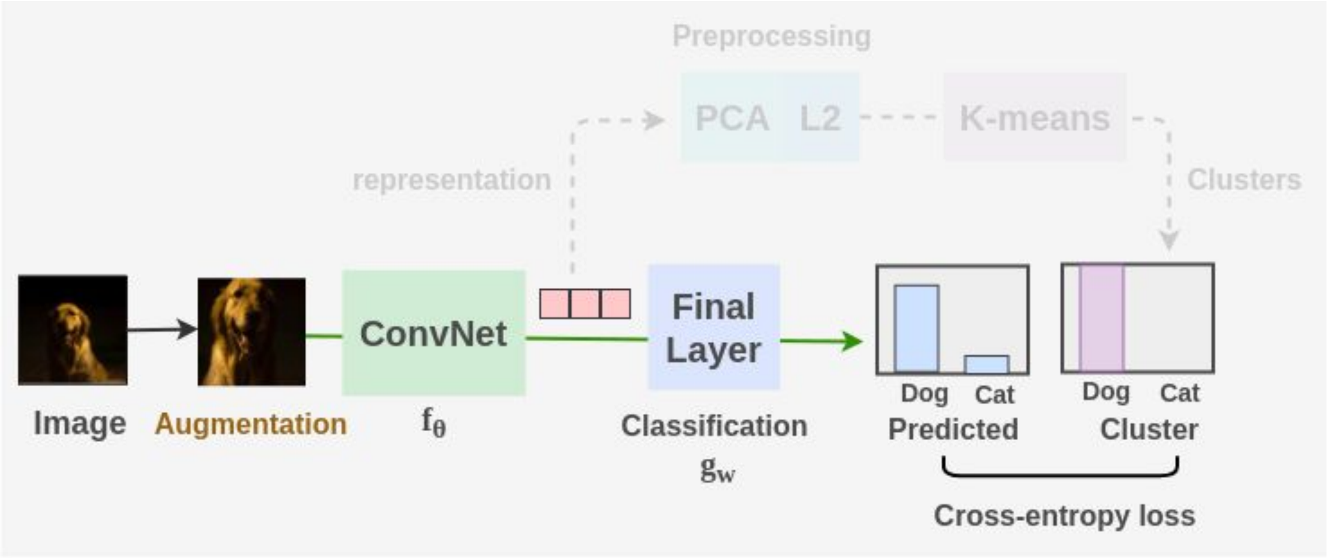
Clustering to generate labels



7) Training Convo Net

Once Pseudo Labels are generated from clusters , Train ConvNet same as regular supervised learning Model.

Use cross-entropy loss to compare model predictions to the ground truth cluster label.



8) Model Training and Clustering

For Each Epoch

1) First step is to generate Pseudo labels for whole dataset with clustering

2) Second step will have regular training of Convonet with cross entropy as a loss between predicted labels and pseudo labels

Dataset Details

- **CIFAR - 10**
    - Currently we considered 2 classes from CIFAR Dataset
    - Cats & Dogs
    - Image Resolution : 32 X 32

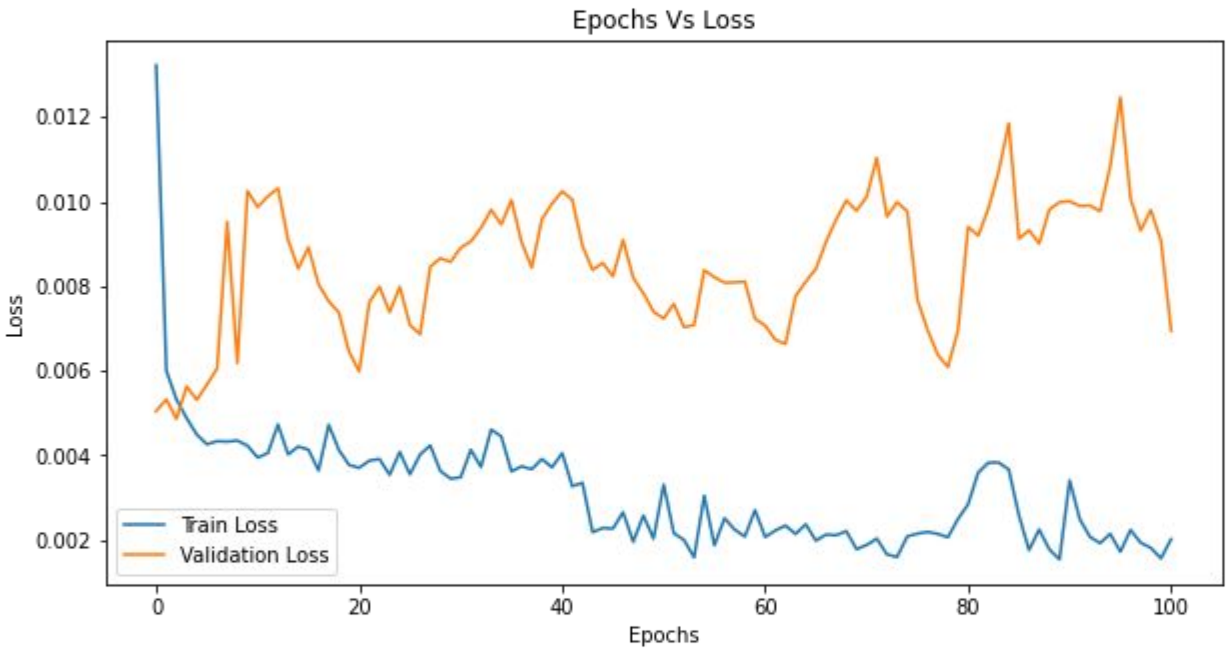
  Issues

  Less Resolution to get features

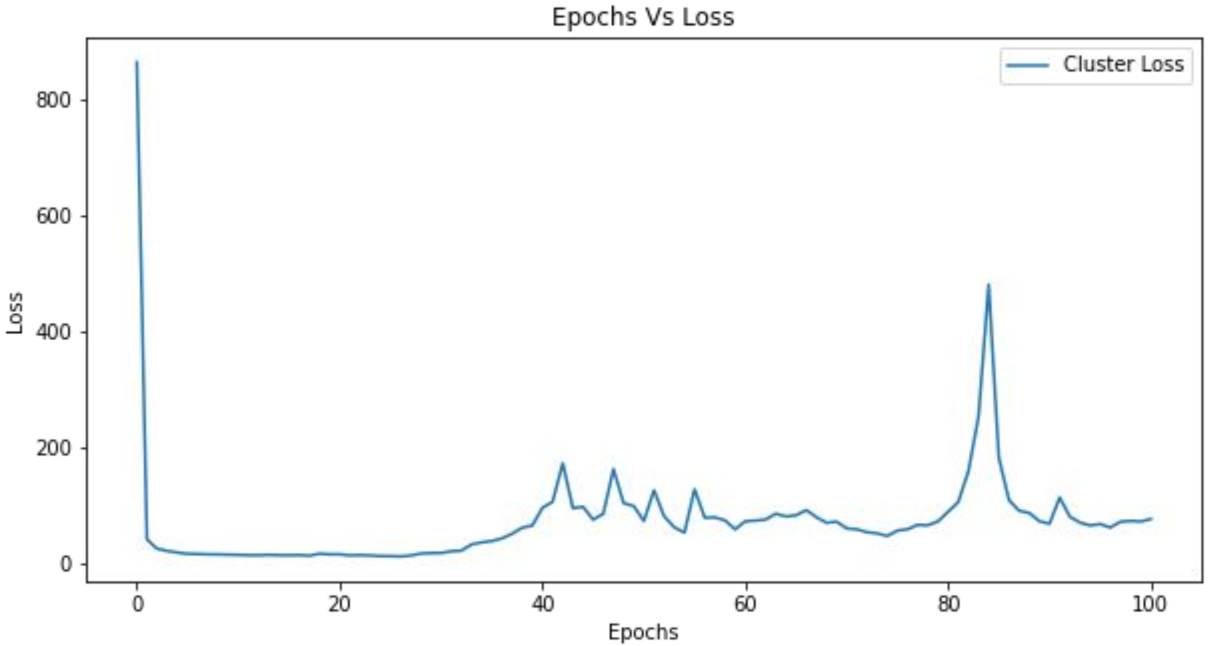
  Less amount of data to Train Large Architecture of the Alexnet

Results

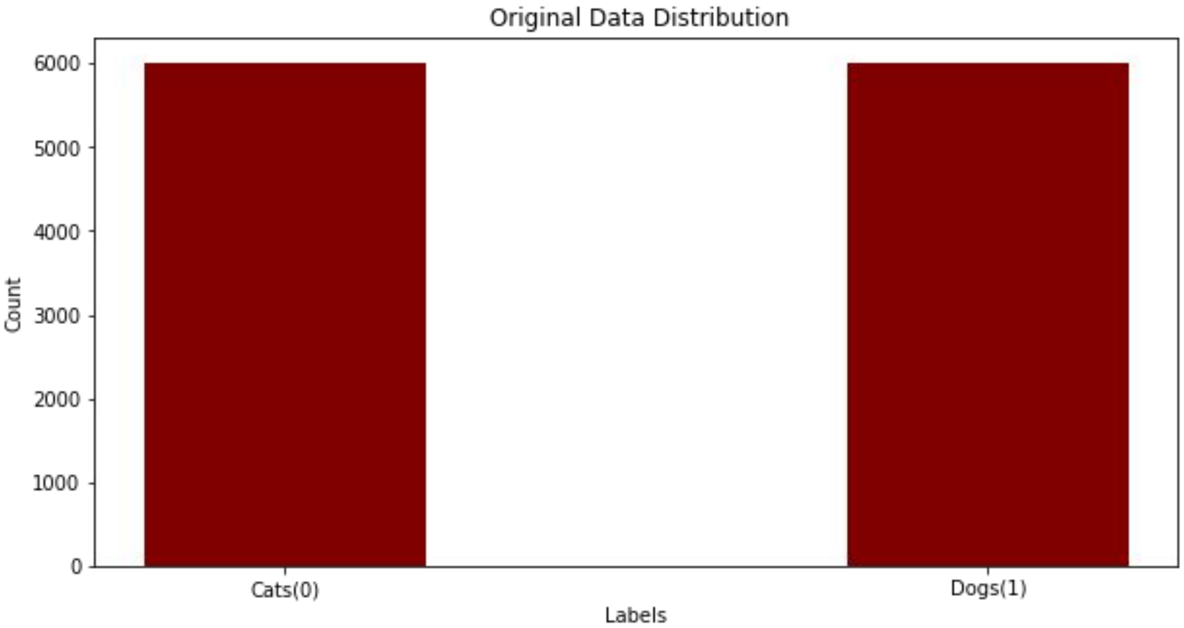
1. Training And Validation Loss (100 Epochs)



2. Clustering Loss (100 Epochs)

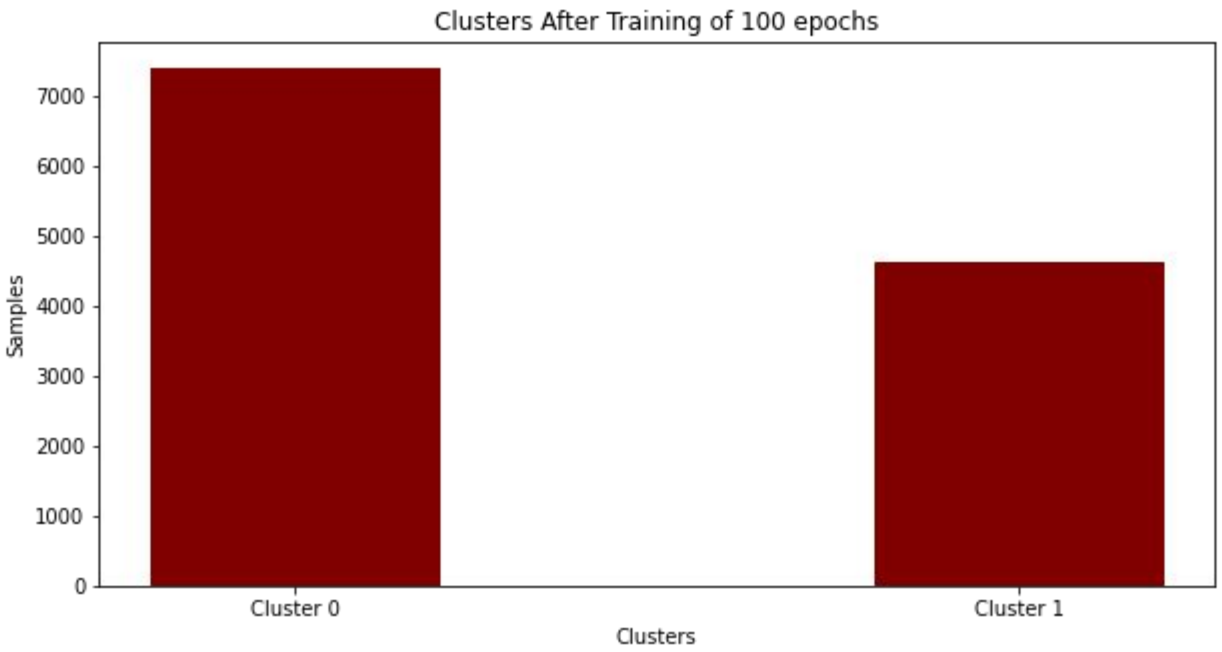


3. Cluster Analysis



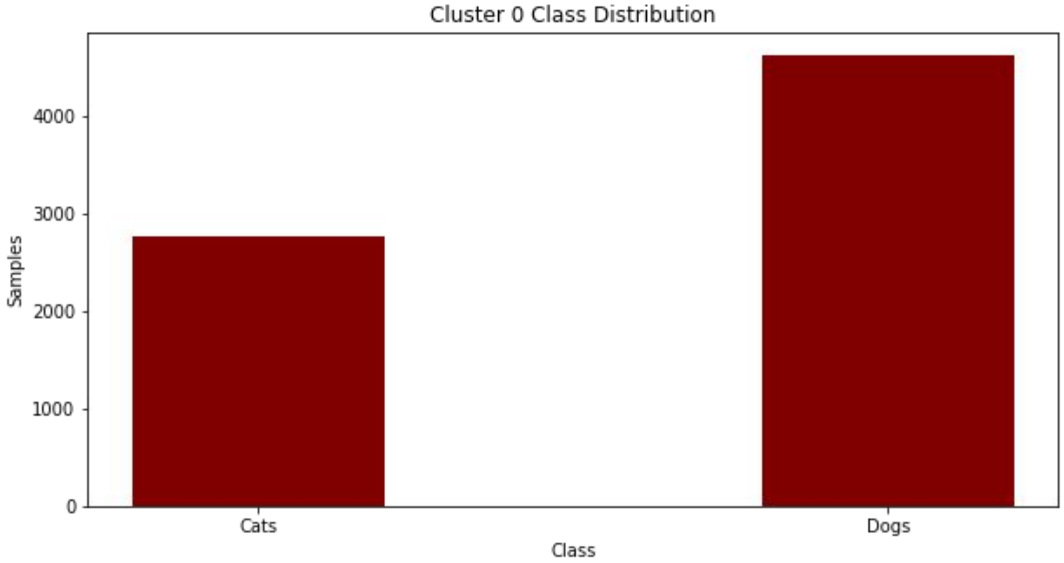
Cats - 5996
Dogs - 6006

Total Image = 12002



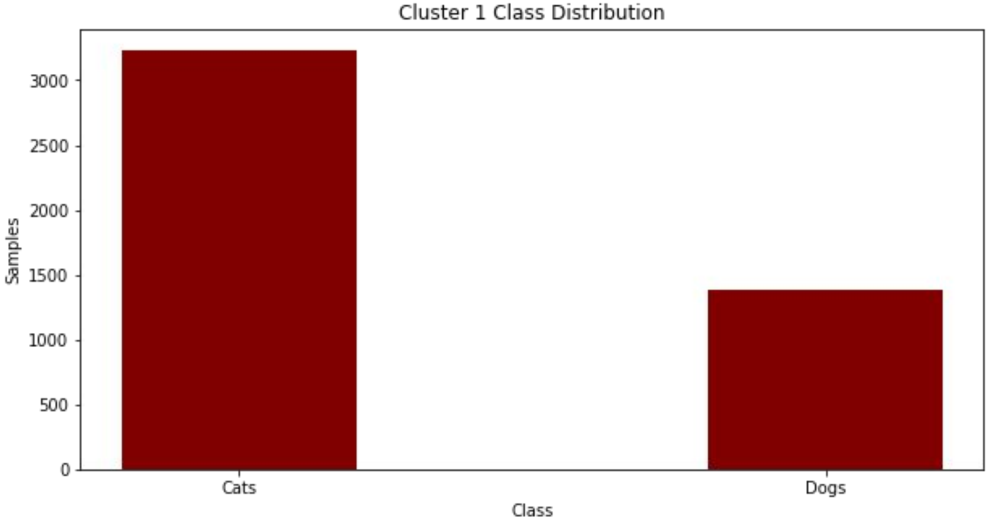
Cluster 0 - 7392
Cluster 1 - 4070

Total Samples = 12002



Dogs - 4627
Cats - 2765

Total Samples in Cluster 0
7392



Dogs - 1379
Cats - 3231

Total Samples in Cluster 1
4070

Classification

- After Training of 100 epoch of CNN with Kmeans , Model Learns the representation and features of the dataset and Generate Clusters
- Cluster Analysis Shows that 2 Cluster that are generated by the Model have one dominated class (Which is expected).
- So , we can safely assume Alexnet predicts the label of the Image same as the label assigned to the respective cluster dominated class

Cluster 0 :

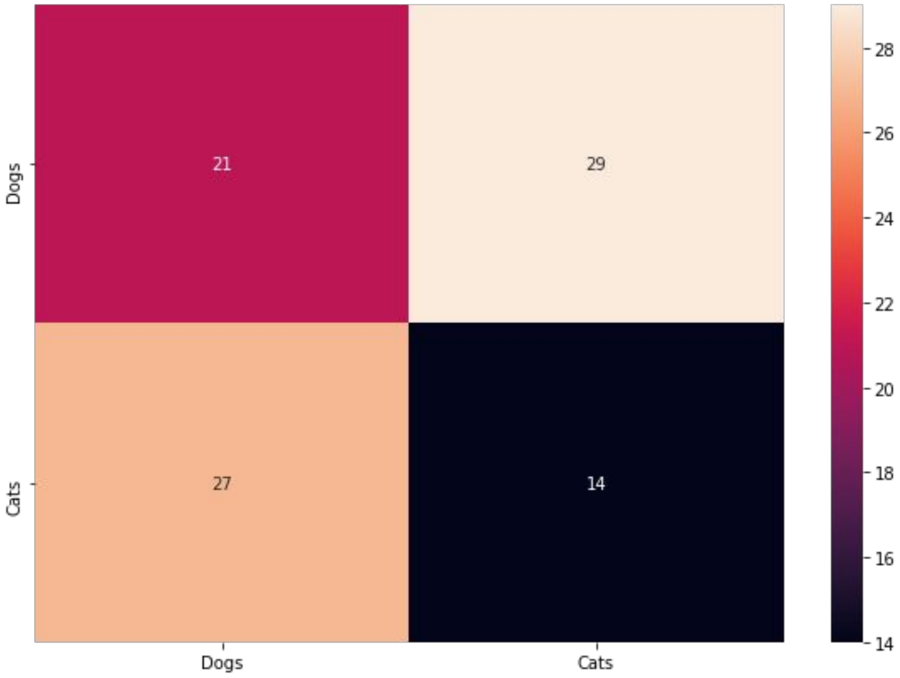
- Cluster 0 Distribution is Dominated By Dog Class
- So , We Considers Label - 0 Representing Dog class

Cluster 1 :

- Cluster 1 Distribution is Dominated By Cat Class
- So , we Considers Label - 1 Representing Cat class

Testing

Test Set Dogs - 50
Cats - 41

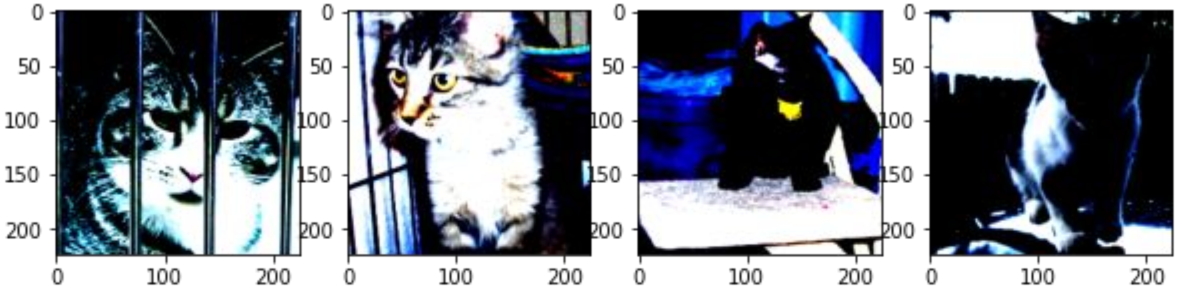


Model Accuracy

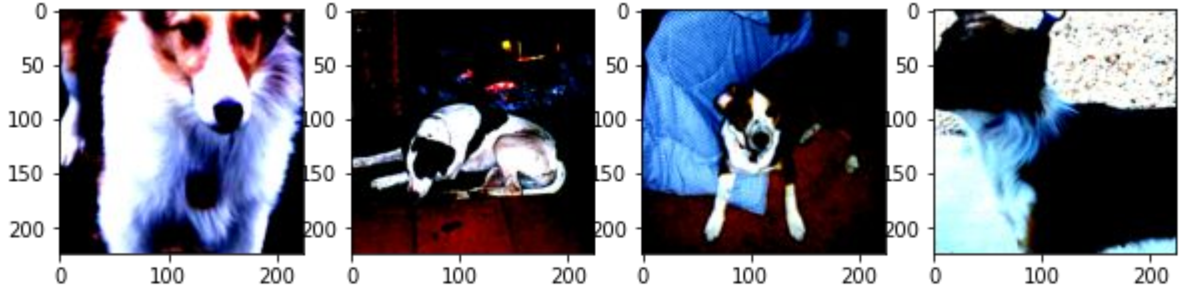
```
from sklearn.metrics import accuracy_score

print('Accuracy Score :' + str(accuracy_score(preds , orig_labels) ) )
```

```
Accuracy Score :0.38461538461538464
```

Correct Classification

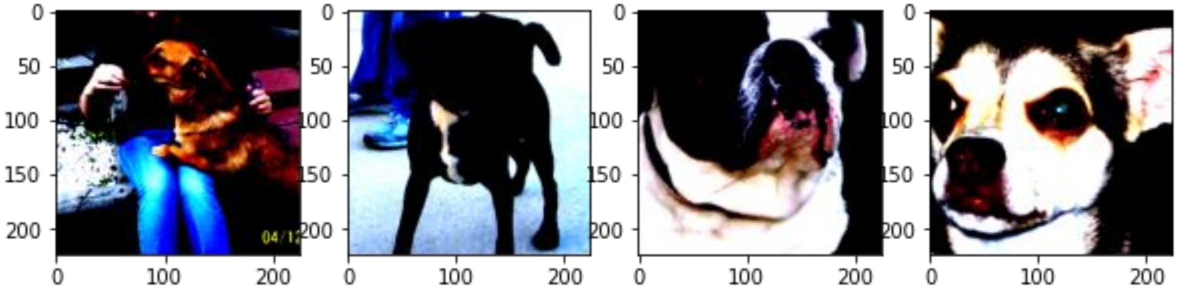


Correctly Predicted Cats

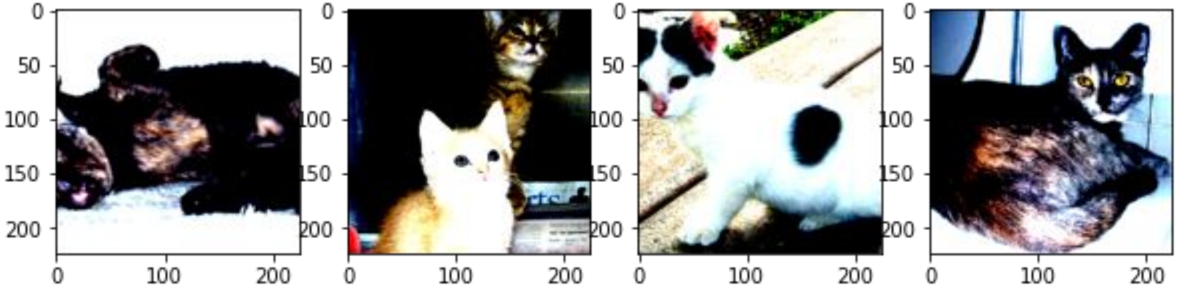


Correctly Predicted Dogs

InCorrect Classification



Predicted Cats



Predicted Dogs

References

1. [Deep Clustering for Unsupervised Learning of Visual Features]
2. [Unsupervised Learning of Visual Features by Contrasting Cluster Assignments]