E1 213 : Pattern Recognition and Neural Networks
Assignment 3 Report

Bhartendu Kumar, Jeevithiesh Duggani, Nishanth Shetty, Rahul Raju Pogu

I. Deep Convolutional Generative Adversarial Network

Dataset - PneumoniaMNIST with labels $y_i \in \{0, 1\}$
We observed that the FID score decreases as training proceeds, this corresponds to increased realistic generation of images. GAN training is highly unstable and required several restarts.

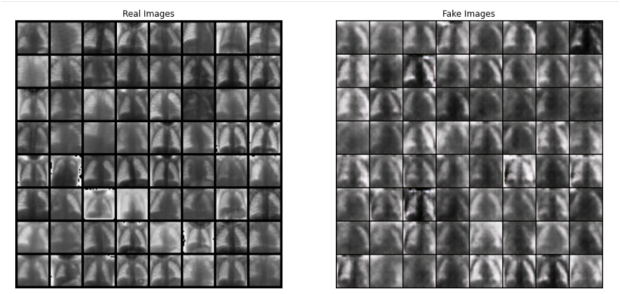

Fig. 1: Real and generated images with DCGAN

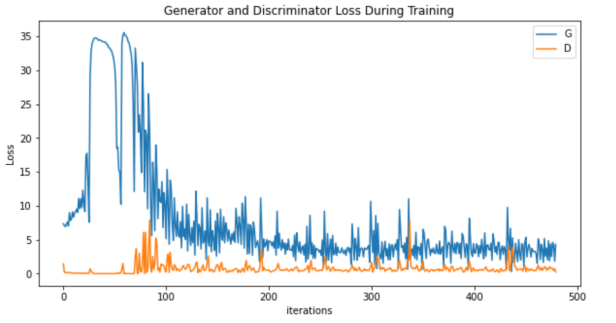

Fig. 2: Loss vs Epochs plot

Fig. 3: DCGAN FID Scores during training

II. Variational Autoencoders

Dataset - PneumoniaMNIST with labels $y_i \in \{0, 1\}$
We observed that the FID score decreases as training proceeds, this corresponds to increased realistic generation of images. The output is clearly not as good as GANs and is marked but a smoothing effect possibly due to the sampling layer in between encoder and decoder being a Gaussian.

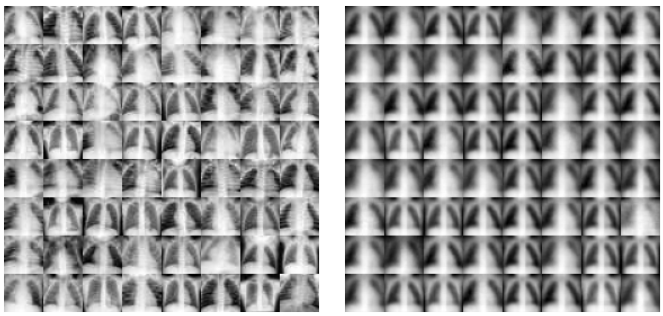

Fig. 4: Real and generated images with VAE

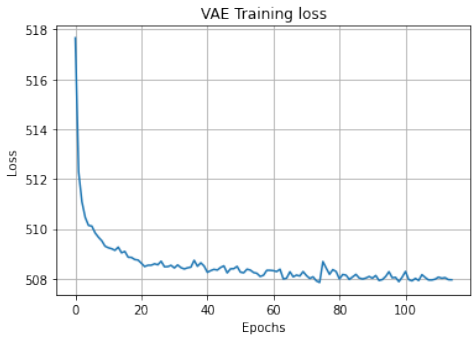

Fig. 5: Loss vs Epochs plot

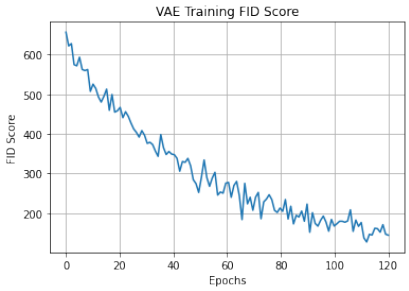

Fig. 6: VAE FID Scores during training

III. t-Stochastic Neighbour Embedding

Dataset - PneumoniaMNIST with labels $y_i \in \{0, 1\}$
The recommended way to apply t-SNE is usually in two steps. First we reduce high dimensional data to an intermediate dimension using t-SNE and then again use t-SNE to reduce dimension from this intermediate dimension to 2 or 3 dimension. In two dimensions, we see that the data is well separated.

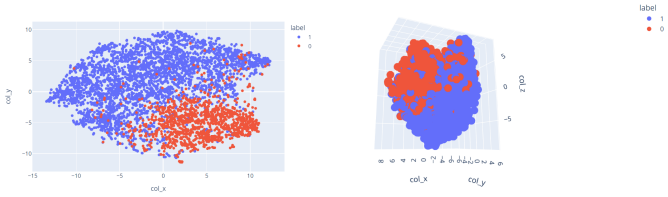

Fig. 7: t-SNE with $n = 2$, $n = 3$

IV. Principal Component Analysis

Dataset - PneumoniaMNIST with labels $y_i \in \{0, 1\}$
Principal component analysis applied for dimensionality reduction in the PneumoniaMNIST dataset. We see good separation of the two classes in the plots below.

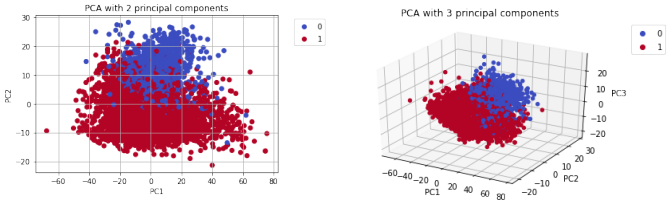

Fig. 8: PCA with $n = 2$, $n = 3$

V. Reconstruction of Images from Principal Components

The residual error is calculated as: $J = \sum_{i=M+1}^{D} \lambda_i$, where J is residue, M is the number of dimensions on which we project, D is total number of eigen vectors and λ_i is the i^{th} eigen value. These are plotted on a scale of 10^5.
We plot the eigen vectors of the data covariance matrix as images. Top 10 eigen vectors descending order is plotted.

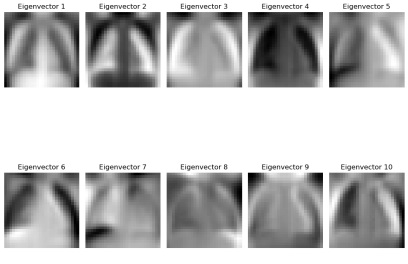

Fig. 9: Top 10 eigen vectors as images

The Eigen values (784) and the residual error are plotted in fig. 10.

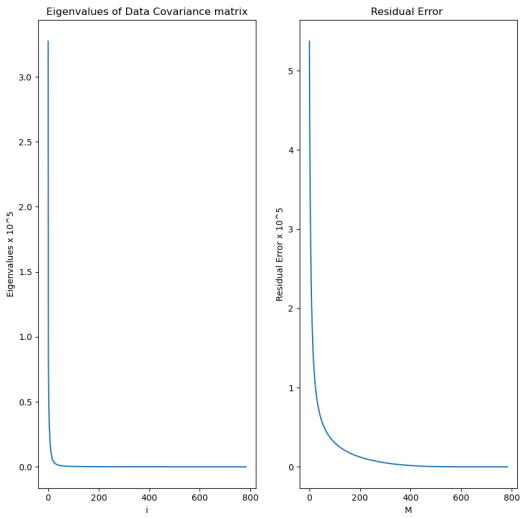

Fig. 10: Eigen values of Covariance Matrix and Residual

We plot the Reconstructed Images where k is the number of PCA dimensions the image is projected to.

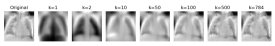

Fig. 11: PCA Reconstruction with different k

We see that at k=1 the reconstructed image is visually not like the original image but at k= 784 which is the total number of eigen vectors, its visually same. The MSE drops to 0 at 784 and is **non-increasing**.

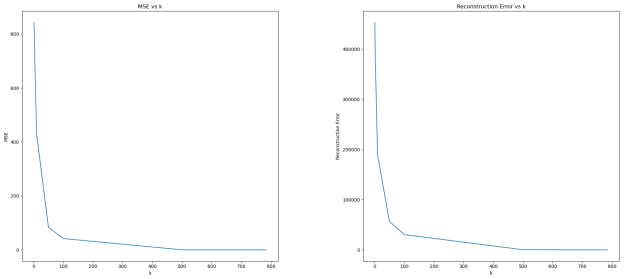

Fig. 12: MSE, Reconstruction Error vs k

VI. KNN with K-means clustering

Note: The Problem Statement in its original form might turn out to be unsolvable because we get **k** centroids from K-means clustering and we have to use those k centoids as training points in K nearest neighbour. **But it is well possible that some of these K-centoids are not ANY REAL DATA and thus we could not find the LABEL for that point and thus can't supervise KNN on those k centroid points as**

representation points.
Hack: We get the nearest data point to the found centroid and use that as **CENTROID**. This in some sense becomes a constrained Kmeans.

It is strongly recommended to Feature Normalize before applying the clustering or K-means until we do not have strong reasons to believe the behaviour of some features is different than others.

1) Difficulties and Solutions:

- **Time Complexity**: As each iteration in K-means clustering we need to process whole dataset, thus the learning is generally very slow.
  Solution Can reduce the number of features to make the algorithm fast and can **increase the accuracy**.
- **sensitive to Outliers** The solution of this could be to ideally pre-process the dataset for anomalies and outliers before running the algorithm.

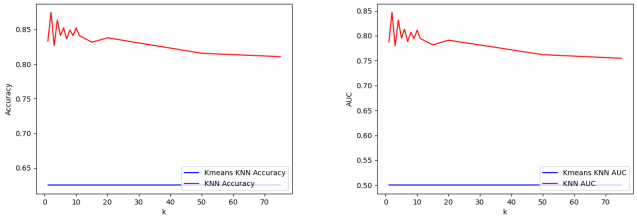

Fig. 13: Accuracy, AUC vs k

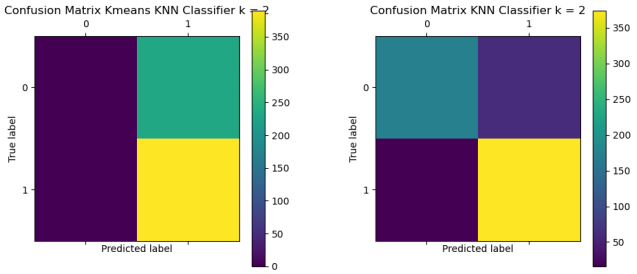

Fig. 14: Kmeans+KNN, KNN confusion Matrix

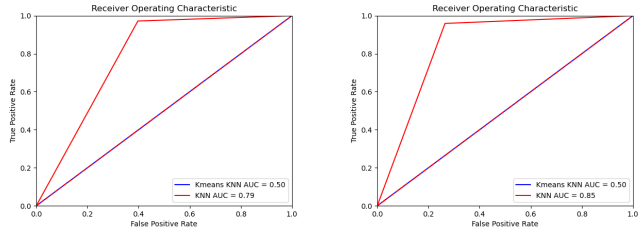

Fig. 15: Combined AUC curve both methods at k =1 and 2

Observations

This mechanism of finding **k-representative points** from centroids of k-clusters and then using **K Nearest Neighbour** is **VERY BAD** strategy.

In this as we applied to **binary classification** problem, the **k-clusters** formed and the **CLASS having MAJORITY Cluster** will always be the class our K Nearest Neighbour classifier will predict as **K is same of both, thus using KNN we are majority polling from all training points (in this case cluster centroids**, thus we have only **k - training points for KNN** and we are using **poll from K neighbours**. the the class with **majority clusters** in K means clustering will always be the **prediction output of our classifier**.

VII. Ensemble Learning

Dataset - PneumoniaMNIST with labels $y_i \in \{0, 1\}$
Using Base Classifiers, Decision Trees (DT) {number of estimators = 20} and Support Vector Classifiers (SVC) {number of estimators = 5}, We have run the AdaBoost Algorithm on the PneumoniaMNIST. We have observed a small Increase in accuracy while using AdaBoost when compared to just using the base estimator.

	DT	AdaBoost (DT)	SVC	AdaBoost (SVC)
Test Accuracy	0.793	0.820	0.846	0.857
F1 Score	0.850	0.868	0.888	0.891
AUC	0.812	0.837	0.880	0.894