Detecting Lesion Type Using Clustering and Dimensionality Reduction

Yu Xuan Yong, PID: A16078479 Kyle Huang, PID: A15747306 Emily Park, PID: A14340038 Caike Salles Campana, PID: A16555833[1,✉]

[1]University of California, San Diego, COGS 118B

Abstract - This report uses the DeepLesion data set and performs unsupervised learning, as well as present comparisons between different methods of dimensionality reduction.

1. Introduction

This project is based off of and motivated by the study: DeepLesion: automated mining of large-scale lesion annotations and universal lesion detection with deep learning by Ke Yan, Xiaosong Wang, Le Lu, Ronald M. Summers [4]. This study makes the use of image recognition techniques and unsupervised learning methods to detect lesions in the body and characterize their types. In this project, we want to be able to classify the lesion type based on data gathered from images of CT scans and see if we can derive the underlying structure of the data. We will test this using multiple clustering methods and dimensionality reduction techniques. We utilize the deconstructed coordinates of the image to conduct both the clustering and PCA. We compare the results of multiple clustering techniques such as k-means and Gaussian Mixture Models, and do dimensionality reduction using t-SNE, pca and autoencoders. In order to do this, we gather data from the Deeplesion dataset,which was collected from the PACS of a major medical institute. It contains 32,120 axial CT slices from 10,594 CT imaging studies of 4,427 patients[4]. By using clustering methods and dimensionality reduction we can understand the lesions and discover any relationships in the data. We also made use of scikit learn [3] to assist us in this.

2. Methodology

2.1 DeepLesion Dataset This dataset includes 32,120 axial CT slices from 10,594 CT scans from 4,427 different patients. Each row of the dataset contains information about a patient, the location of the lesion, and size of the lesion.However, since we only want samples that already have the classifier (coarselesiontype), we filtered the data set and utilized only 9816 samples. We also concluded that only a handful of features are useful and relevant to what we want to achieve, so we dropped certain features. According to the paper, lesion location and the size of the lesion are the two most important factors in lesion detection and identification, so the goal is to be able to try to identify the 8 different lesion types by using the location and size data from the DeepLesion dataset. The normalized lesion location columns were not standardized due to already being standardized previously.

Features we utilized in the DeepLesion data set include:

- Normalized Lesion Location: This shows where normalized coordinates of the location of the lesion in the form of a 3D vector. This refers to the relative body position of the center of the lesion.
- Measured Coordinates (in pixels): An 8D vector of the two RECIST diameters of the lesion
- Lesion Diameter (in pixels): A 2D vector that shows the lengths of the long and short axes of the lesion.
- Coarse Lesion Type: The type of the lesion, numbered from 1-8.
- Bounding Boxes (in pixel): A 4D vector of the coordinates of the bounding box that cordons off the lesion.

2.2 Unsupervised Learning

K-means Clustering From the dataset, the K-Means algorithm was run on the Normalized_lesion_location column. To prepare the data for the algorithm, the values in each row were split into three columns and the commas in between each float value was also deleted. The K-Means algorithm came from Homework 2 which includes functions for plotting, calculating square distances, determining the responsibility matrix, and recalculating mu values based on the cluster assignments. Since the K-Means algorithm from Homework 2 takes in a file string, the Normalized_lesion_location dataframe column was saved as a .txt file. The K-Means algorithm was run on the Normalized_lesion_location with setting the number of clusters to 2, 3, 4, and 7. Seeing that some clusters were overlapping, the dimensionality of the Normalized_lesion_location column was reduced using sklearn's PCA function from 3 to 2 dimensions.

Gaussian Mixture Models (GMM) We utilized Gaussian Mixture Models and applied it onto the normalized lesion location feature for clustering. It uses a linear combination of Gaussian distributions. The essential advantage of this method is that the cluster assignment is a distribution itself; hence, for each data point, we have a probability for it being on each cluster. In this project, we select the most probable cluster. Now there are two main ways of doing it. The

first is using the standard version, which uses a to describe the different clusters. The other one is a Bayesian approach, which “allows to infer an approximate posterior distribution over the parameters of a Gaussian mixture distribution.” [3] In our case, there were 7 different clusters from the definition given by the dataset, and as such, we have a more fluid assignment. In addition, we can also select how the covariance will be distributed; scikit-learn provides the following options:

- Full: each component has its own general covariance matrix
- Tied: all components share the same general covariance matrix.
- Diagonal: each component has its own diagonal covariance matrix
- Spherical: each component has its own single variance.

Bayesian Gaussian Mixture Models We also created Bayesian Gaussian Mixture Models for the normalized lesion location feature.The Bayesian approach “allows for us to infer an approximate posterior distribution over the parameters of a Gaussian mixture distribution.” We used a total of 4 different types of covariance matrices here as well, namely the spherical, tied, diagonal and full. The same thing was used to cluster the bounding box feature from the DeepLesion data set.

2.3 Dimensionality Reduction

Principal Component Analysis We did PCA multiple times using different vector columns in order to find the best results. We did PCA on the full dataset, measurement coordinates columns, bounding boxes, normalized lesion location, and lesion diameter. First to maximize the variance of the projected data, we calculate the sample mean. Using the mean we can center the values in the column by subtracting the mean from the data. We obtain the covariance matrix using the numpy covariance function on the transpose of the matrix containing the centered values. From there we use this covariance matrix to calculate the eigenvalues and eigenvectors. The eigenvalues and eigenvectors are sorted from largest to smallest value. We can get the amount of variance explained by each component by taking the eigenvalue matrix and dividing it by its sum. We choose the components to use in the feature vector based on how much they make up the explained variance. The data is then projected into the subspace by multiplying the columns containing the components that we want to use by the dataset. Finally, we visualize the components on a 2d or 1d space depending on how many components we chose to use, and run clustering methods on the projected data to figure out the cluster relationships.

t-SNE One way of making dimensionality reduction is using a technique called t-SNE, which stands for t-distributed stochastic neighbor embedding. This method can detect higher dimension structures in the dataset, as "most of the techniques are not capable of retaining both the local and the global structure of the data in a single map." [2] 1000 iterations were then compiled. The algorithm is as follows:

Algorithm 1: Simple version of t-Distributed Stochastic Neighbor Embedding.

Data: data set $X = \{x_1, x_2, ..., x_n\}$,
cost function parameters: perplexity $Perp$,
optimization parameters: number of iterations T, learning rate η, momentum $\alpha(t)$.
Result: low-dimensional data representation $\mathcal{Y}^{(T)} = \{y_1, y_2, ..., y_n\}$.
begin
  compute pairwise affinities $p_{j|i}$ with perplexity $Perp$ (using Equation 1)
  set $p_{ij} = \frac{p_{j|i} + p_{i|j}}{2n}$
  sample initial solution $\mathcal{Y}^{(0)} = \{y_1, y_2, ..., y_n\}$ from $\mathcal{N}(0, 10^{-4}I)$
  for t=1 **to** T **do**
    compute low-dimensional affinities q_{ij} (using Equation 4)
    compute gradient $\frac{\delta C}{\delta \mathcal{Y}}$ (using Equation 5)
    set $\mathcal{Y}^{(t)} = \mathcal{Y}^{(t-1)} + \eta \frac{\delta C}{\delta \mathcal{Y}} + \alpha(t)\left(\mathcal{Y}^{(t-1)} - \mathcal{Y}^{(t-2)}\right)$
  end
end

Fig. 1. Graph of number of actions taken by agent against the number of episodes.

Autoencoders We also utilized autoencoders for dimensionality reduction of features in the DeepLesion dataset. Autoencoders are artificial neural networks that learn data codings in an unsupervised manner, and we applied that to the data set in order to provide comparison of the visualization in component space as compared to other dimensionality reduction techniques. The keras [1] library was mainly used in the building of the autoencoder.

We first initialized a data frame of the normalized lesion location and the lesion diameters to use to build an autoencoder model. The data set was scaled using a MinMaxScaler. We then tuned the hyperparameters in order to achieve more accurate depictions of the reduced data. We assumed the input dimension *(input_dim)* to be of shape $(5, 0)$, which would be the number of features the data frame possesses. The encoding dimension *(encoding_dim)* was set to 2, which was the number of latent dimensions we want to reduce the data set to.

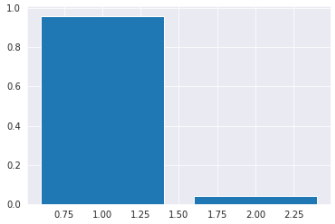
Using the keras library, we created three Dense encoding layers which serves to encode the samples from the data set. We used a bottleneck sort of approach and created layers of 10,5,2, in that order. These would subsequently represent the number of nodes each layer has. We used the **relu** as an activation function, also known as the rectified linear unit. We decided on this activation function for the input layers to allow nonzero input to be processed. We also made use of activity regularization for which we set the parameter to l1 loss. We believed using this would regularize the hidden layers of the autoencoder.

Similarly, we also created three Dense decoding layer which decodes the reduced dimensional output into a ’lossy’ reconstruction of the same data set. We decided on using rectified linear units for the hidden layer and a linear activation function for the output activation. Similarly, we also utilized activity regularization set to l1 loss for the bottleneck layer of the decoding layers.

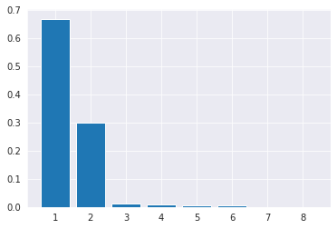
Finally, we create an autoencoder model with the inputs based on any input received with the shape $(5,)$ and outputs with the same shape. We also create an encoder model to visualize the reduced dimensional data. We then compile this model while setting using mean squared error as a loss function. Lastly, we fit the autoencoder model to the scaled data set, with the epoch set to 100, and the batch size set to 32. We proceed to then use the fitted autoencoder to predict the same data set, as well as use the encoder to predict how the data set looks like in 2 dimensions.

3. Results

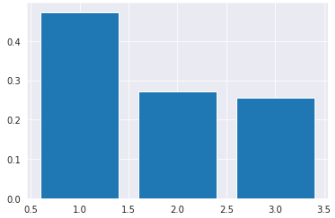
3.1 Dataset Analysis



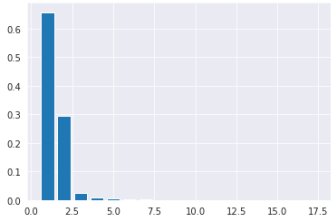
(a) Lesion Diameter



(b) Measurement Coordinates



(c) Normalized_Lesion_Location

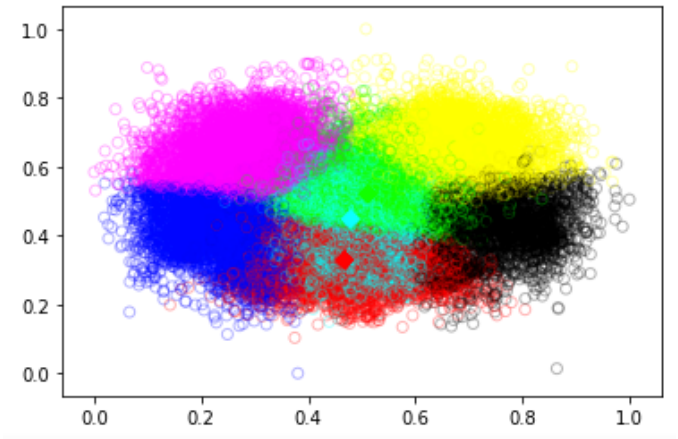


(d) Full Dataset

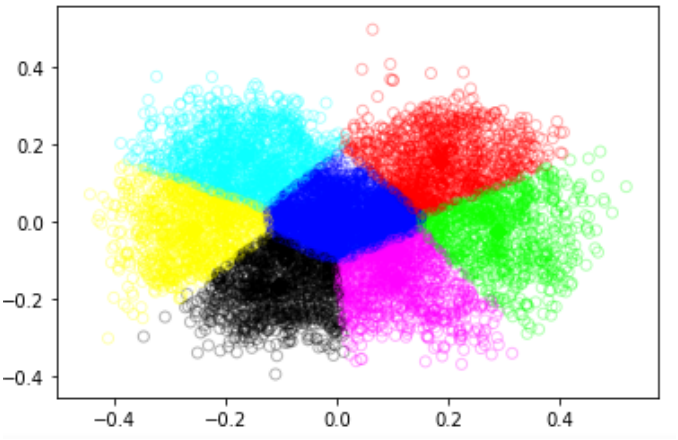
Fig. 2. Variance Graphs

3.2 k-Means

As shown from Figures 3a and 3b, the K-Means algorithm was able to cluster the Normalized_lesion_location with clear cluster outputs. The non-dimensionality reduced data did not perform as well as the data that was reduced from 3 to 2 dimensions. This demonstrates that K-Means does not work well on high dimensional data that has not been reduced in dimensions. However, there is significant improvement in the clusters when PCA is used on the data.



(a) k-Means on normalized_lesion_location



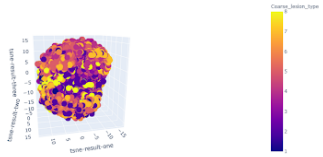
(b) k-Means on reduced normalized_lesion_location

Fig. 3. Graphs of k-Means

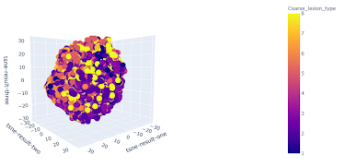
3.2 Dimensionality Reduction

Figures 4a, b and c show the results of running t-SNE. Since the data is visualized in 3D it would seem like the data is all clumped together. However, the data is clustered accordingly by the coarse lesion type. The t-SNE graph with a perplexity value of 5 gives a mean sigma of 7.42 and a KL divergence of 1.18. The t-SNE graph with a perplexity value of 10 gives a mean sigma of 8.45 and a KL divergence of 1.22. The t-SNE graph with a perplexity value of 50 gives a mean sigma of 10.95 and a KL divergence of 1.02.

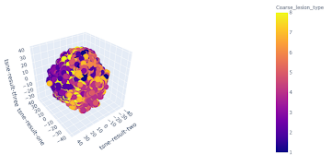
By doing PCA on our data we were able to reduce the dimensionality of the full dataset from 17 to 3. PCA



(a) t-SNE graph with perplexity value 5



(b) t-SNE graph with perplexity value 10



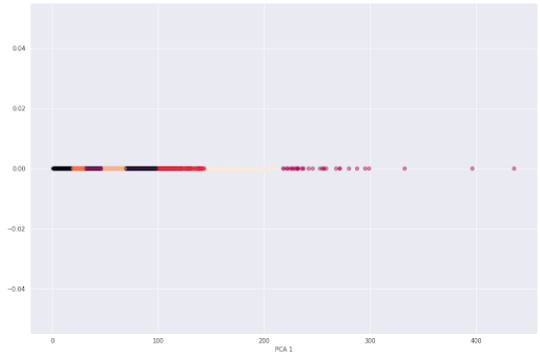
(c) t-SNE graph with perplexity value 50

Fig. 4. t-SNE graph with respective perplexity values

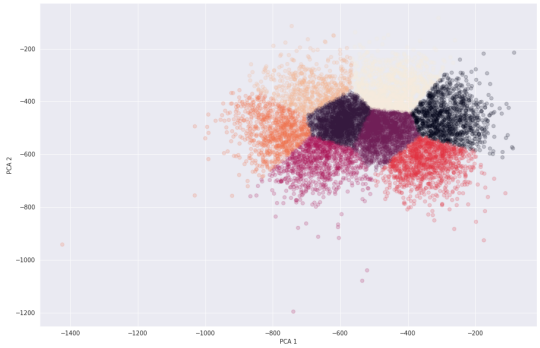
was done using multiple combinations of the 17 columns however after plotting the first three components, there were no visible clusters which would make it difficult to draw conclusions from running clustering methods. The projected data for measurement coordinates, and lesion diameter did not have more than 3 visible clusters, so it would be difficult to find 8 clusters using K-means on this data. One exception to this occurred however, with the projected data of normalized lesion location. The dimensions were not able to be reduced for this data as it uses all 3 components, but there are visible clusters when plotting the first and third components on a scatter plot. This is shown by Figures 5 a,b,c and d.

Utilizing autoencoders has yielded results similar to those of the original DeepLesion study. We see that the reconstructed data shown in Figure 7c is comparable and similar to the original clusters of data shown in Figure 7a, albeit with a little discrepancy, which is common when doing dimensionality reduction. The depiction of the loss function also shows that the loss is gradually minimized over the course of 500 epochs, as shown in Figure 6.

When we visualize the dataset in two latent dimensions, shown in Figure 7b, we can see that the clustering of the data still bears similarities to the original dataset in terms of its placement and distribution. This would go to show that the autoencoder has worked in explaining the dataset by using a minimal amount of components. This could also be partially due to the fact that we utilized the features that our EDA

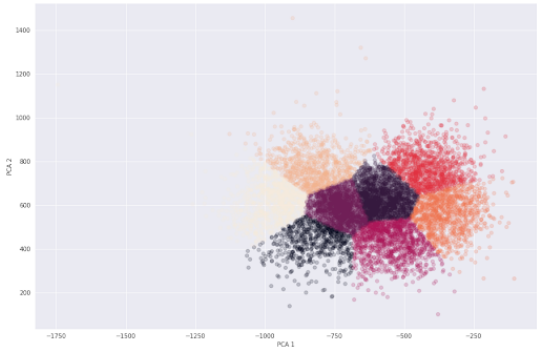


(a) Lesion Diameter



(b) Measurement Coordinates

(c) Normalized_Lesion_Location



(d) Full Dataset

Fig. 5. k-Means ran on reduced components through PCA

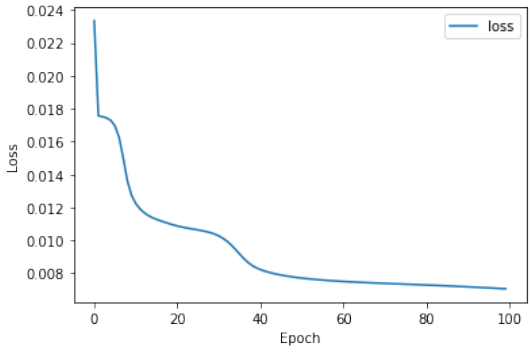
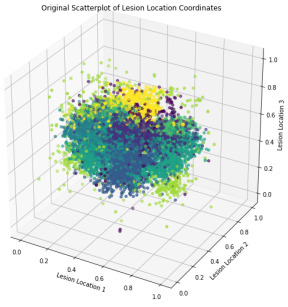
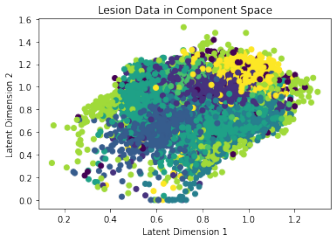


Fig. 6. Graph of the loss function vs the number of epochs

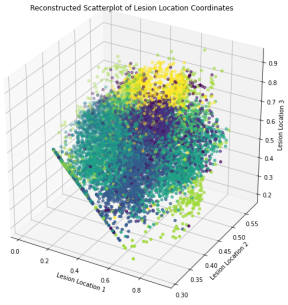
revealed has the most variance when accounting for the data set.



(a) Original Dataset with 5 dimensions expressed via normalized lesion location



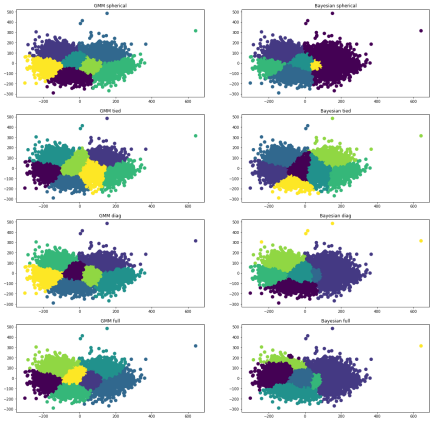
(b) Dataset expressed in 2 latent dimensions



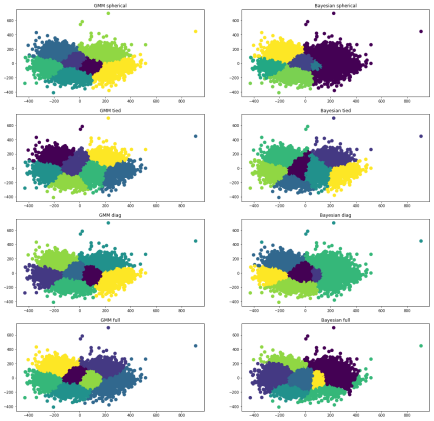
(c) Reconstructed Dataset expressed via normalized lesion location

Fig. 7. Dataset processed by autoencoders for dimensionality reduction

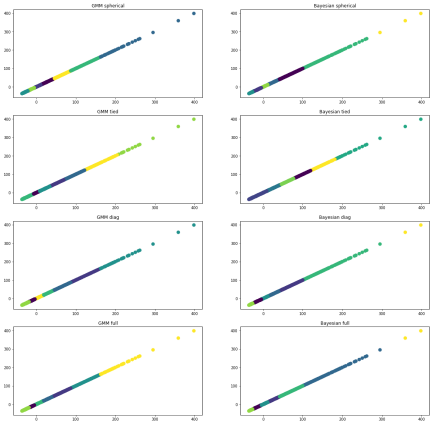
For the Gaussian Mixture Models, as shown in Figure , since our data appears to be superposed in a 2D plane, the different approaches to the Gaussian Mixture Model achieve different results and thus are not correctly clustered. However, despite its failure, the differences between each covariance and how that affects the standard method versus the bayesian are notable.



(a) Bounding Boxes



(b) Measurement Coordinates



(c) Lesion Diameter

Fig. 8. GMM and Bayesian GMM graphs

4. Discussion

Future projects as a continuation of this one could include using different clustering techniques like competitive learning and spectral clustering to perform clustering, as well as trying to reconstruct the entire lesion image given the data, since most of the data is in pixels. We could also use Support Vector Machines to do classification on the K-Means clustering.

Another possibility to more precisely predict the clusters is using convoluted neural networks on the result of the autoencoders.

5. Conclusion

To summarize, with the right starting cluster centroids, k-Means was useful in helping to cluster the dataset. Across all dimensionality reduction techniques, autoencoders seem to perform very well in preserving the clustering of the data, as well as reconstruction. GMM, Bayesian GMM, autoencoders seems to work well in predicting the placement of the different coarse lesion types, when referring back to the original DeepLesion study. We have also discovered through dimensionality techniques that the normalized lesion location and lesion diameter coordinates explain and account for most of the variance in the data set. One exception was the t-SNE graphs, which were not very useful in visualizing the different coarse lesion types. There was also the additional drawback of needing great computational power, since it ran for an extended period of time before results were generated. Another possible limitation would be the tuning of certain hyperparameters in both the k-Means and autoencoders, which might have causes a slight compromise on the results.

6. Links for Final Project

Youtube link for Final Project: https://youtu.be/IZx1X4Sg1Hs
Github link for Final Project: https://github.com/cscampana/COGS118B--FinalProject-Deeplesion

7. Contributions

Yu Yong (A16078479):

- Coding for Autoencoders
- Report Formatting
- Conclusions Section, Autoencoder Results and Methodology
- Video Presentation and Slides

Kyle Huang (A15747306):

- Dataset Analysis
- Coding for PCA
- Introduction Section, PCA Results and Methodology
- Video Presentation and Slides

Emily Park (A14340038):

- Coding for K-Means
- K-Means Results and Methodology
- Video Presentation and Slides
- Edit and Upload Video

Caike Salles Campana (A16555833):

- Initial exploratory data analysis
- t-SNE, coding and analysis
- Gaussian Mixture Models, coding and analysis
- Video presentation and Slides

References

[1] Francois Chollet et al. *Keras*. 2015. URL: https://github.com/fchollet/keras.

[2] Laurens van der Maaten and Geoffrey Hinton. Nov. 2008. URL: https://www.jmlr.org/papers/volume9/vandermaaten08a.

[3] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[4] Yan K;Wang X;Lu L;Summers RM; *DeepLesion: automated mining of large-scale lesion annotations and universal lesion detection with deep learning*. July 2018. URL: https://pubmed.ncbi.nlm.nih.gov/30035154/.