Image Classification using Deep Neural Networks and TDA

Sidhanth Krishna (ID: 5733479)

ABSTRACT

For the success of any data-science experiment one of the most promising entities of the pipeline is in the quality of features that's being used by the machine learning model. Even with a linear model and a set of robust features the accuracy of prediction can be on par with state-of-the art neural networks. A major domain of the same application manifests itself in image classification data and in this experiment I intend to focus on the problem by building on two core concepts - Topological Data Analysis and Deep Learning. TDA deals with the underlying shape of the data and tools like Persistent Homology help in understanding the global and prominent structures present in the image. Deep Learning on the other hand concentrates on finding non-linear patterns from the data and thus helps in supervised learning tasks. These methods can be robustly combined using ensemble models(GBDT) to achieve state-of-the-art results. With a 75-25 train-test split on the dataset the ensemble model gave a classification accuracy of ~94%. (Please do read the note towards the end)

PROBLEM STATEMENT

The dataset for this experiment is the notMNISTdataset taken from Kaggle. The notMNIST dataset is created using some publicly available fonts and extracted glyphs(fig. below) from them to make a dataset similar to MNIST. There are 10 classes, with letters A-J taken from different fonts. Judging by the examples, one would expect this to be a harder task than MNIST. This seems to be the case -- logistic regression on top of stacked auto-encoder with fine-tuning gets about 89% accuracy whereas the same approach gives 98% on MNIST. Dataset consists of a small hand-cleaned part, about 19k instances, and a large uncleaned dataset, 500k instances. Two parts have approximately 0.5% and 6.5% label error rate. For this project I've stuck to using the smaller version of the dataset for computational reasons.

DEEP LEARNING APPROACH - VARIATIONAL AUTOENCODERS

INTRODUCTION:

In a nutshell, a Variational Autoencoder(VAE) is an autoencoder whose encodings distribution is regularised during the training in order to ensure that its latent space has good properties allowing us to generate some new data and also compress the original data. VAE is a generalized neural network framework for the task of dimensionality reduction and it consists of two parts - Encoder and Decoder.

Encoder aids in producing a new feature representation of the original data whereas the Decoder is responsible for reversing the same process. A dimensionality reduction can then be viewed as a data compression strategy where the Encoder compresses the original data to the encoded space, also called the latent space, whereas the Decoder decompresses them. The objective of the dimensionality reduction technique is to find the best encoder-decoder pair that keeps the maximum information when encoding and has the minimum reconstruction error when decoding. Unlike a generic autoencoder, a VAE also has the regularization component where instead of encoding input data as a single vector, we encode it as a distribution over latent space. This step is done so as to avoid the autoencoder from taking advantage of the overfitting possibilities.

OBJECTIVE FUNCTION:

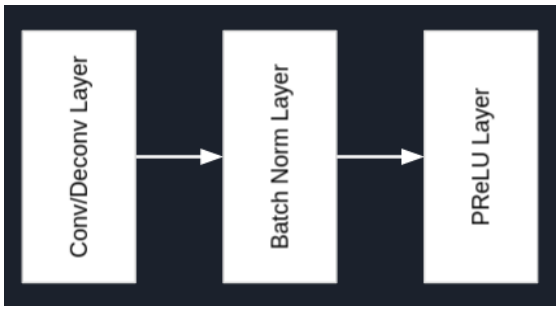
VAEs are trained using what's known as the ELBO loss which captures the essence as mentioned above. The loss function constitutes two terms - KL Divergence loss and Reconstruction Log loss. The KL Divergence loss is responsible for regularizing the latent space distribution by computing a metric between itself and a Normal distribution. The Reconstruction Log loss transforms to a Mean-Squared Error between the original-data and the reconstructed output. For the project, the input is a 28*28 grayscale image and the latent vector is of size 32. More about this shall be mentioned in the network model section below. Thus the ELBO loss function can be written as show below:

$$\widetilde{\mathcal{L}}^{B}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}^{(i)}) = -D_{KL}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}^{(i)})||p_{\boldsymbol{\theta}}(\mathbf{z})) + \frac{1}{L}\sum_{l=1}^{L}(\log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}|\mathbf{z}^{(i,l)}))$$

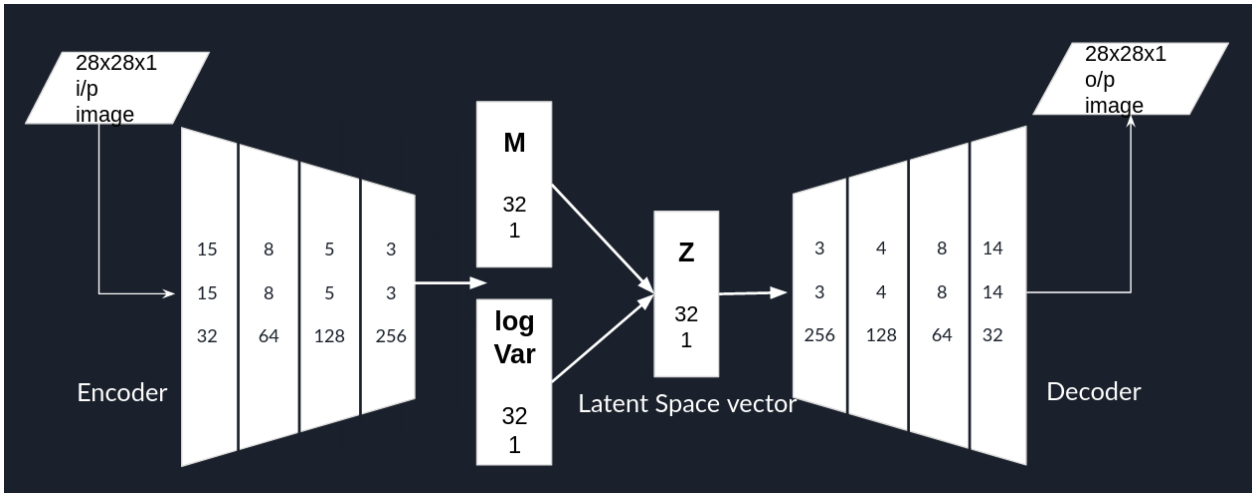
NETWORK MODEL:

The network architecture of the Encoder and the Decoder is a series of convolution and deconvolution units respectively with kernel size of 4, stride values of 2 and varying padding to match the reconstructed output image size with the output. Each of these units consists of either a convolution or a deconvolution layer followed by a batch-normalization layer and a parametric ReLU activation function. To get latent space distribution from the convolution layers, the output is first flattened and connected to a couple of linear layers each for the mean and log-variance of the distribution. The latent-space vector is sampled from these distribution parameters using the reparameterization trick.

Convolution Unit:



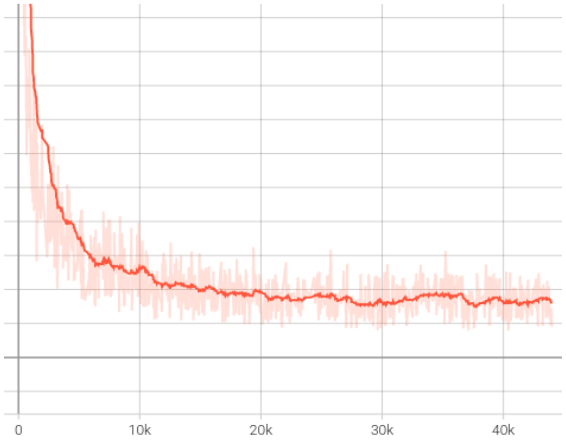
Network Architecture:



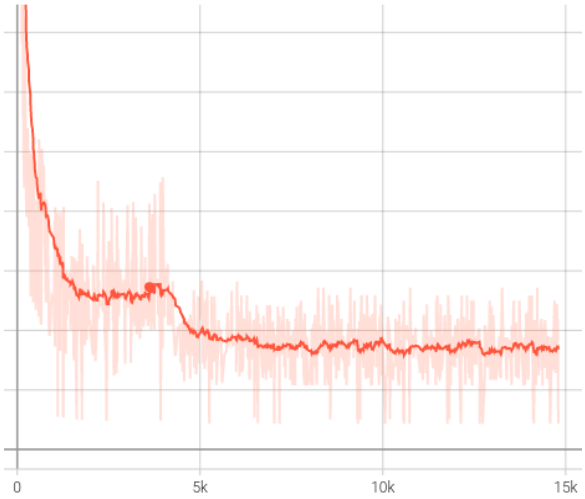
TRAINING PROCEDURE:

- The training-set consists of 14k grayscale images and the validation-set consists of 4.5k grayscale images.
- Initial Learning Rate of 1e-4 was used along with a learning rate scheduler - ReduceLROnPlateau that decides to lower the learning rate by monitoring the validation loss.
- The network was L2-regularized with weight decay parameter equal to 2 and the optimizer used was Adam
- The network trained for 200 epochs and was saved every 10 epochs if the current validation loss decreased when compared to the previous validation loss. The tensorboard logs are as shown below:

  Train Loss



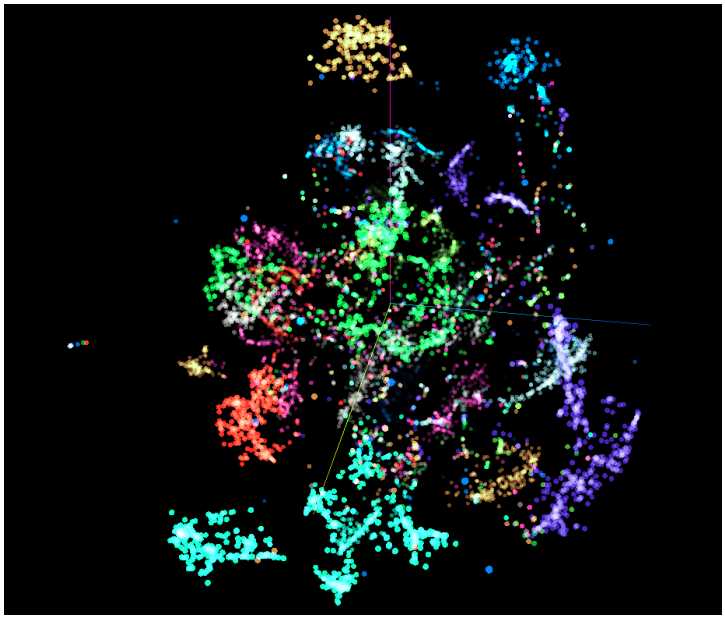
  Test Loss



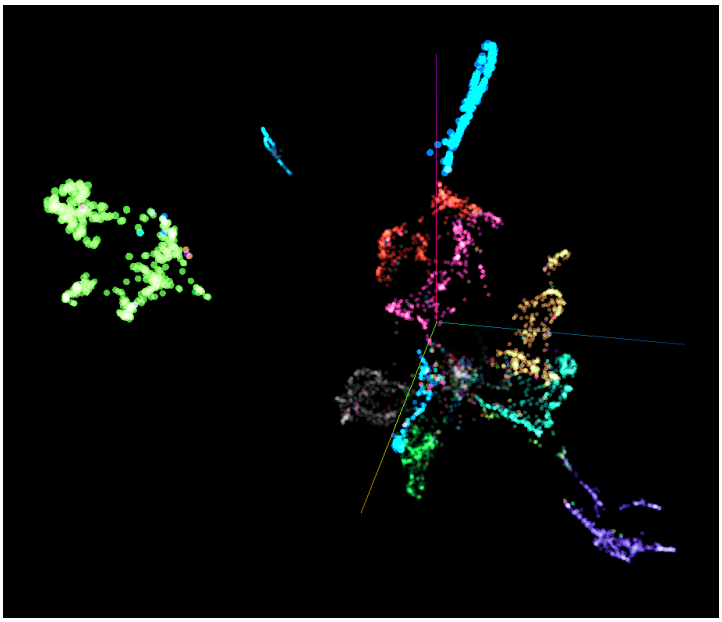
LOGISTIC REGRESSION USING VAE FEATURES:

Based on the trained VAE, a new dataset consisting of latent vectors and the corresponding data label can be generated. The latent representation seems to make the original data linearly separable in the embedded space(fig. below) and can be visualized using tensorboard and T-SNE and UMAP plots. A linear classifier such as logistic regression is chosen to fit the embedding space data and the classification report is shown below.

T-SNE Plot (Colors represent different labels):



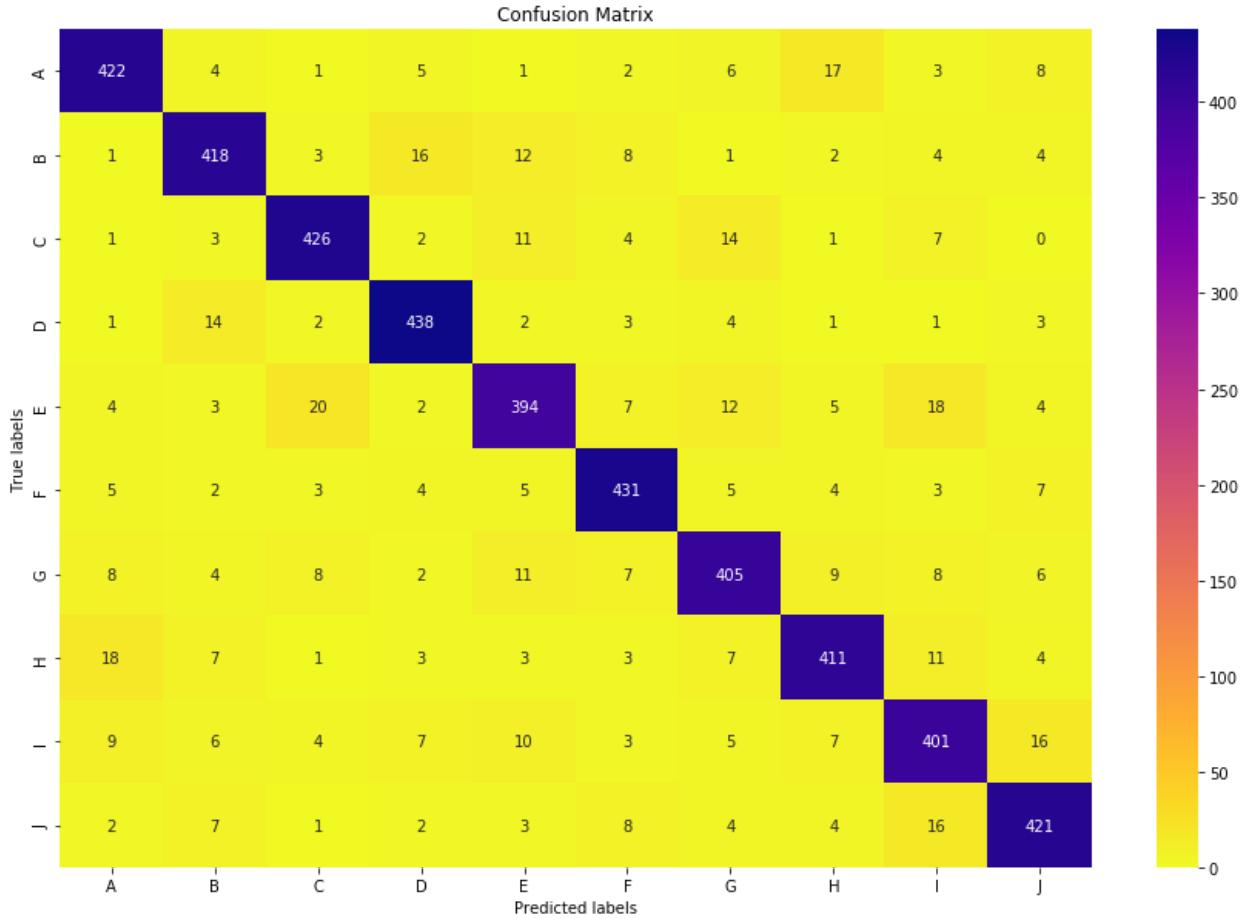
UMAP Plot:



Classification Report:

```
Classifier Accuracy:  0.8892445582586428
Classification Report:
              precision    recall  f1-score   support

           A       0.90      0.90      0.90       469
           B       0.89      0.89      0.89       469
           C       0.91      0.91      0.91       469
           D       0.91      0.93      0.92       469
           E       0.87      0.84      0.86       469
           F       0.91      0.92      0.91       469
           G       0.87      0.87      0.87       468
           H       0.89      0.88      0.88       468
           I       0.85      0.86      0.85       468
           J       0.89      0.90      0.89       468

    accuracy                           0.89      4686
   macro avg       0.89      0.89      0.89      4686
weighted avg       0.89      0.89      0.89      4686
```

Confusion Matrix:



TOPOLOGICAL DATA ANALYSIS

INTRODUCTION:

From the Confusion Matrix above, it's visible that some of the misclassifications caused are between the following letters:

1. A with H
2. B with D and E
3. C with E and G
4. I with J

All these letters structurally are similar, in the sense that if we were to attach the top ends of H to form a loop it'd look like an A. Although they are structurally similar, they are topologically very different. Topological Data Analysis(TDA) helps us in understanding different shapes present in the data. They are powerful tools to both visualize and aid machine learning algorithms in achieving a certain task. For example, let's look at one of the topological characteristics (in this case the number of loops present in the structure) of the class labels:

1. A has 1 loop
2. B has 2 loops
3. C has 0 loops
4. D has 1 loop
5. E has 0 loops
6. F has 0 loops
7. G has 0 loops
8. H has 0 loops
9. I has 0 loops
10. J has 0 loops

From the topological characteristics, if we were to find a way to calculate the number of loops, then we can predict the letter B with lesser misclassifications as it's the only label with 2 loops. The purpose of this experiment is to understand how we can use such topological features that can aid in classification. The following topics will be discussed briefly with examples to comprehend and compute relevant topological features:

1. Filtration
2. Simplicial Complex and Persistent Homology
3. Feature Representation
4. TDA Feature Generation Pipeline

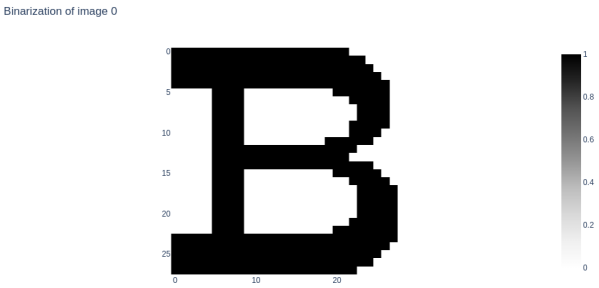
FILTRATION:

In mathematics, a filtration F is an indexed family of subobjects of a given algebraic structure S, with the index i running over some totally ordered index set I, subject to the condition that if $i \leq j$ in I, then $S[i] \subseteq S[j]$

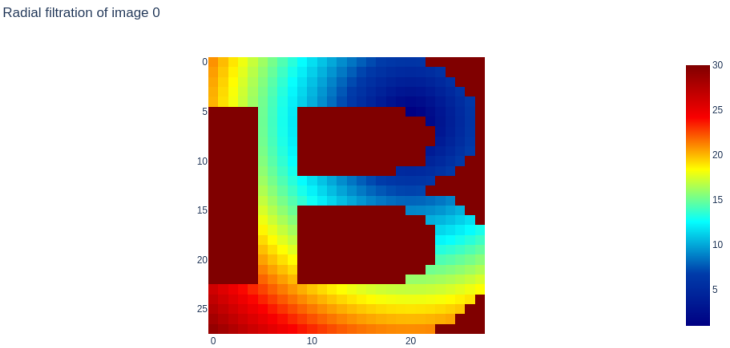
Filtration is an important step in any TDA pipeline as it's the first step to create something called Complex. Complex is a way of forming a topological space from distances in a set of points. Distances between points are defined by something called the Metric and points closer than a specified threshold are connected by an edge in a complex. Construction of a complex is necessary to understand the global structure of the data.

One such example of a filtration for images is the Radial filtration. It assigns each pixel of a binary image(grayscale) a value that's computed from a reference pixel, called the "center", and of a "radius". If the binary pixel is active(non-zero) and lies within the ball defined by this reference center and radius, then the assigned value equals its distance from the reference pixel. For an inactive pixel, the assigned value equals the maximum distance between any pixel of the image and the reference center pixel, plus one. Lets visualize the Radial filtration:

Binarized Image:



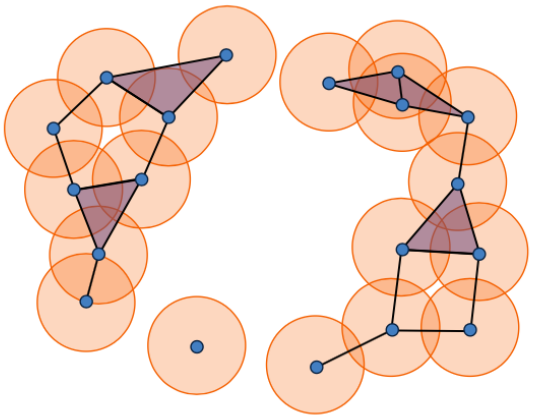
Radial Filtration (center is the pixel at upper right corner):



SIMPLICIAL COMPLEX AND PERSISTENT HOMOLOGY:

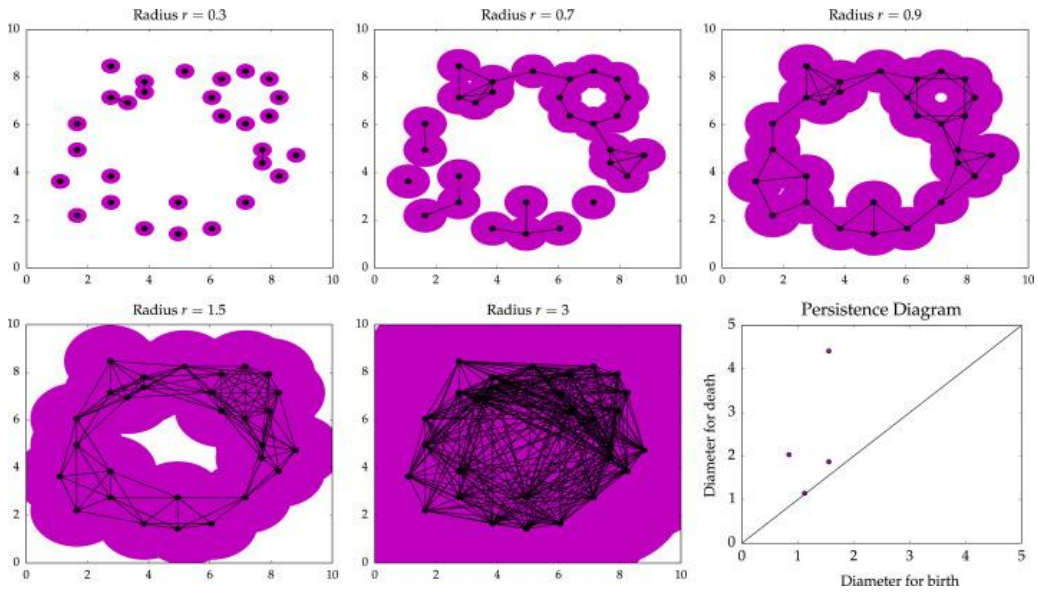
In mathematics, a simplicial complex is a set composed of points, line segments,triangles, and their n-dimensional counterparts (see illustration). A simplex is defined as the fundamental entity in a simplicial complex. Eg. 0-simplex is a point, 1-simplex is an edge, 2-simplex is a triangle and so on. They are formed as a result of edge formation, as described above, between points that lie within a certain ball of radius determined by the distance threshold (fig. below).

Simplicial Complex:



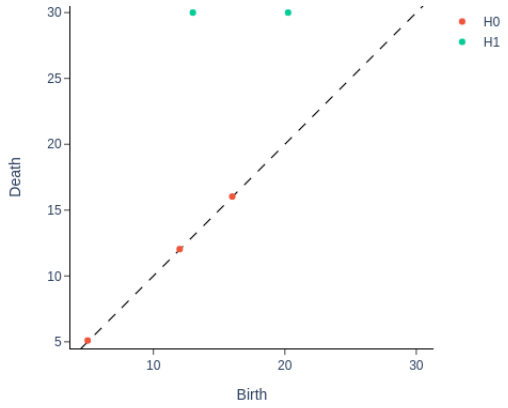
Homology is a rigorous mathematical method for defining and categorizing holes in a manifold formed by the simplicial complex. Persistent Homology can be considered as a historical ledger of different homology formations when the distance threshold (mentioned above) varies from 0 to its maximum value.

Persistent Homology:



The relevance of Persistent Homology to the current exercise is that it helps to compute different topological features like loops. As seen above, the letter B has two loops and the Persistence Diagram should reflect this. We can compute the Persistent Homology by forming a cubical simplicial complex from the filtration that we computed before and then plotting the historical events of hole formations using Homology (fig. below)

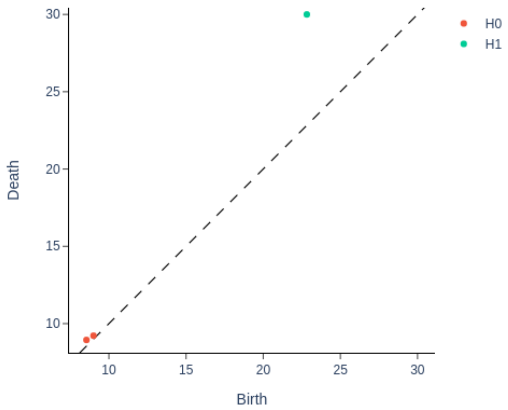
Persistence Diagram for B



H0 - Edges (Red Dots)

H1 - Loops (Green Dots)

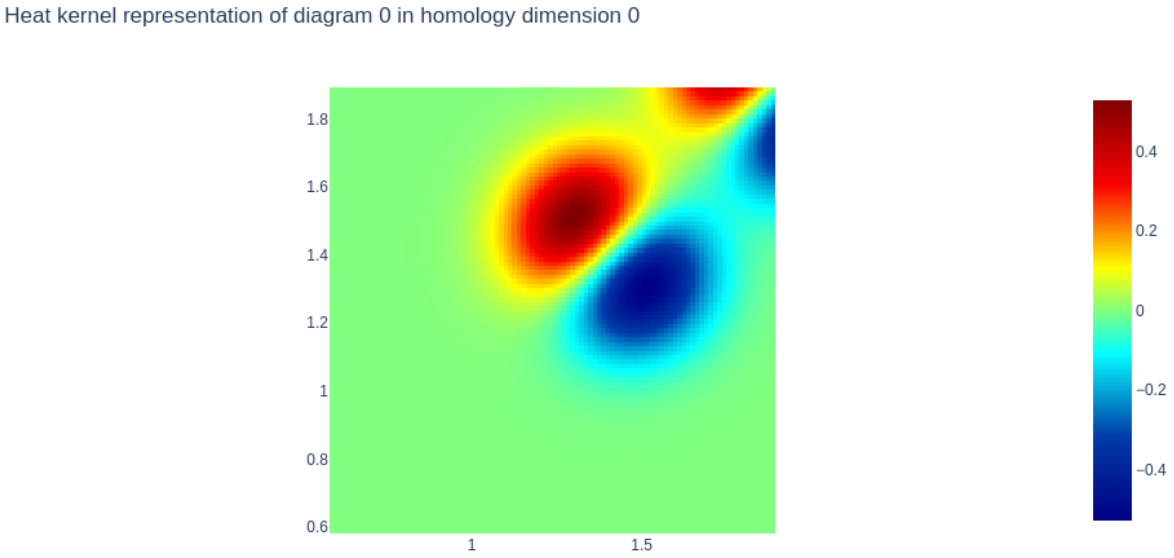
As shown in the Persistence Diagram, the filtration for image B has two loops and three edges. Hence different topological information can be captured for different images, Persistence Diagram for D is shown below and as expected it captures the single D-loop in H1 (fig. below)

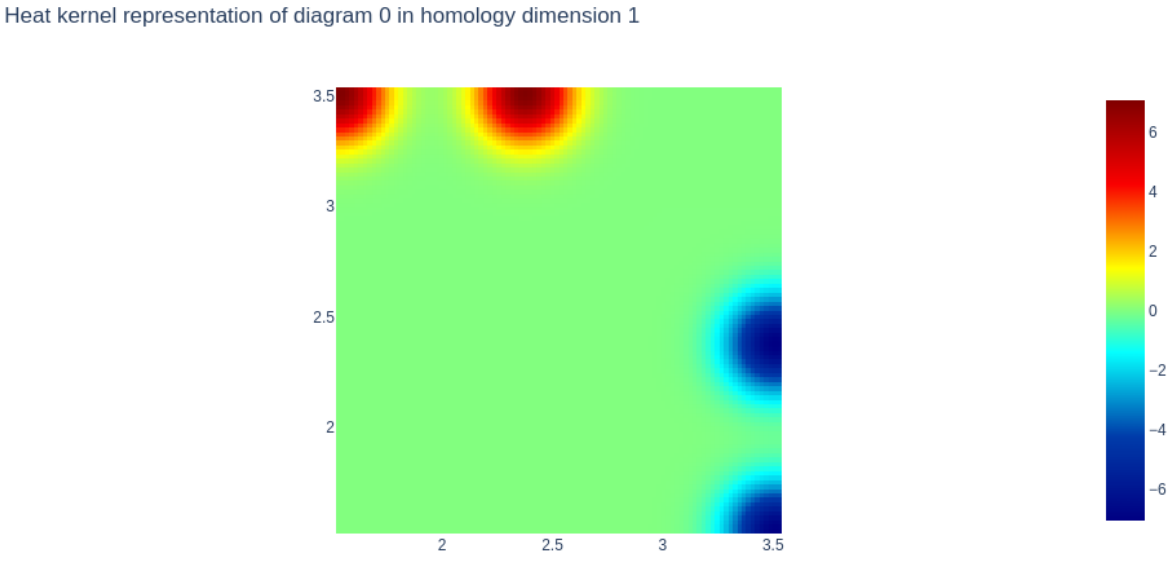


FEATURE REPRESENTATION:

Persistent Homology needs to be translated to a meaningful metric for it to be considered as a feature. One of the ways this can be achieved is by convolving a gaussian kernel on the persistence diagram, a procedure achieved via the heat kernel. For each of the Homology dimensions, the Heat Kernels amplitude is computed to get the vectorized feature.

Heat Kernel Maps for H0 and H1 of B



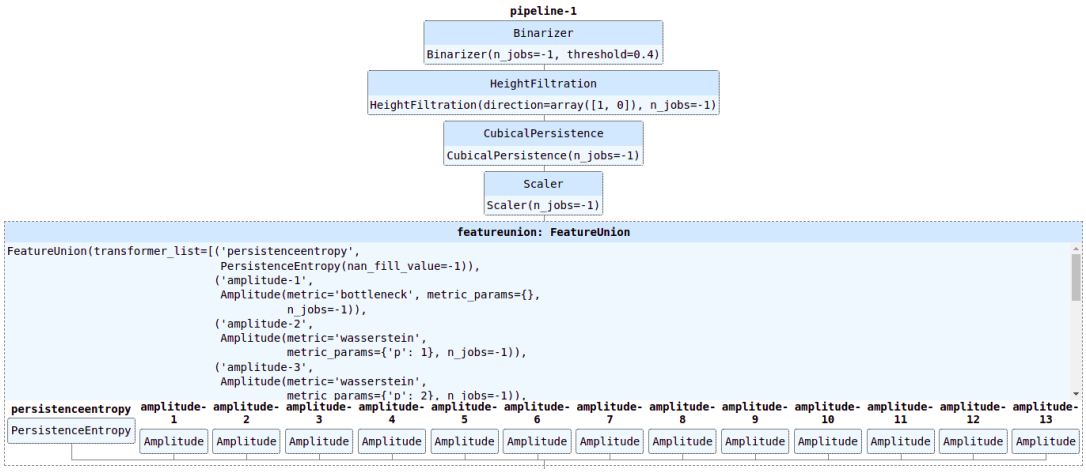


TDA FEATURE GENERATION PIPELINE

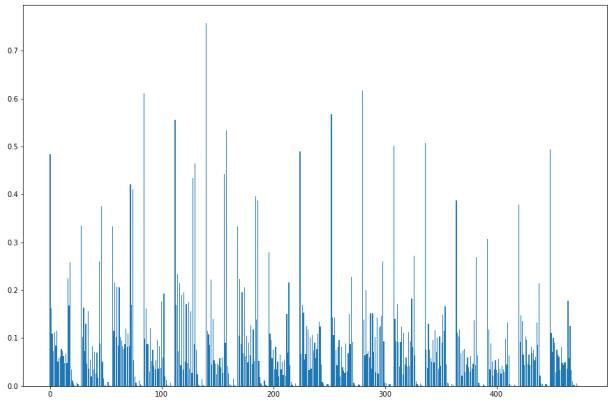
Giotta-tda python library is a high performance TDA library that helps us compute Persistent Homology features and it's compatible with the Sklearn Pipeline method. There's a whole bunch of feature vectorization that can be done and for this exercise the following features were computed resulting in 476 TDA based features:

- Amplitudes using the following metrics:
    - Bottleneck distances
    - Wasserstein distances
    - Persistence Landscapes
    - Betti vectors
    - Heat-Kernels
- Persistence Entropy

Pipeline configuration:



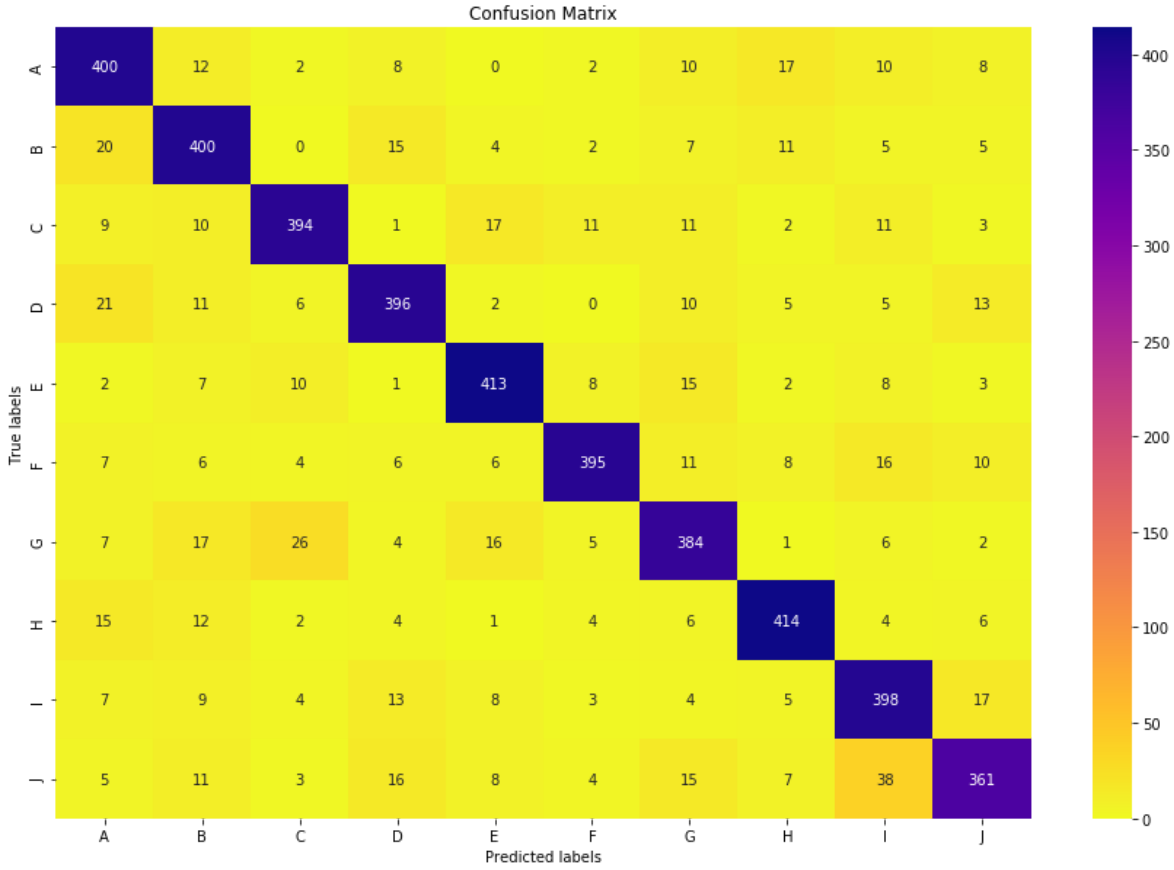
Before combining these features with the VAE features, we can inspect the predictive power of TDA features by modeling it using a logistic regression model and checking the feature importance across all classes (fig below.)



LOGISTIC REGRESSION USING TDA FEATURES:

From the above plot, we can see that there's a good cohort of features that have predictive power. A logistic regression model was trained based on these TDA features to check if they can provide any potential boost and the confusion-matrix and classification report are as follows:

```
Classifier Accuracy:  0.8440034144259496
Classification Report:
              precision    recall  f1-score   support

           A       0.81      0.85      0.83       469
           B       0.81      0.85      0.83       469
           C       0.87      0.84      0.86       469
           D       0.85      0.84      0.85       469
           E       0.87      0.88      0.87       469
           F       0.91      0.84      0.87       469
           G       0.81      0.82      0.82       468
           H       0.88      0.88      0.88       468
           I       0.79      0.85      0.82       468
           J       0.84      0.77      0.81       468

    accuracy                           0.84      4686
   macro avg       0.85      0.84      0.84      4686
weighted avg       0.85      0.84      0.84      4686
```



ENSEMBLE MODELS

The summary of exercises done so far are as follows:

- Modelled a Variational Autoencoder to get the latent-space representation of the notMNIST dataset.
- Modelled a Logistic Regression Classifier using the VAE features (also called latent space features) and produced a classification accuracy of ~88.9%
- Implemented a pipeline to generate TDA based Persistent Homology features using Giotta-tda library.
- Modelled a Logistic Regression Classifier using the TDA features to check for predictive power of the TDA features and produced a classification accuracy of ~84.4%

Not all features generated by the TDA pipeline seemed useful and there was never an effort to check how they'd interact when combined with the VAE features.

The next step in the project is to fetch a good cohort of these features for the best predictive performance and there are multiple ways to approach this problem. In this project I have decided to automate this step by using a tree-based ensemble model called Gradient Boosted Decision Tree (GBDT).

The GBDT model inherently checks for the best features to create multiple simple decision trees based on varying subsets of data. The GBDT model then combines these simple trees based on their respective cross-validation accuracy to form its prediction strategy which results in a strong classifier with low variance and low bias.

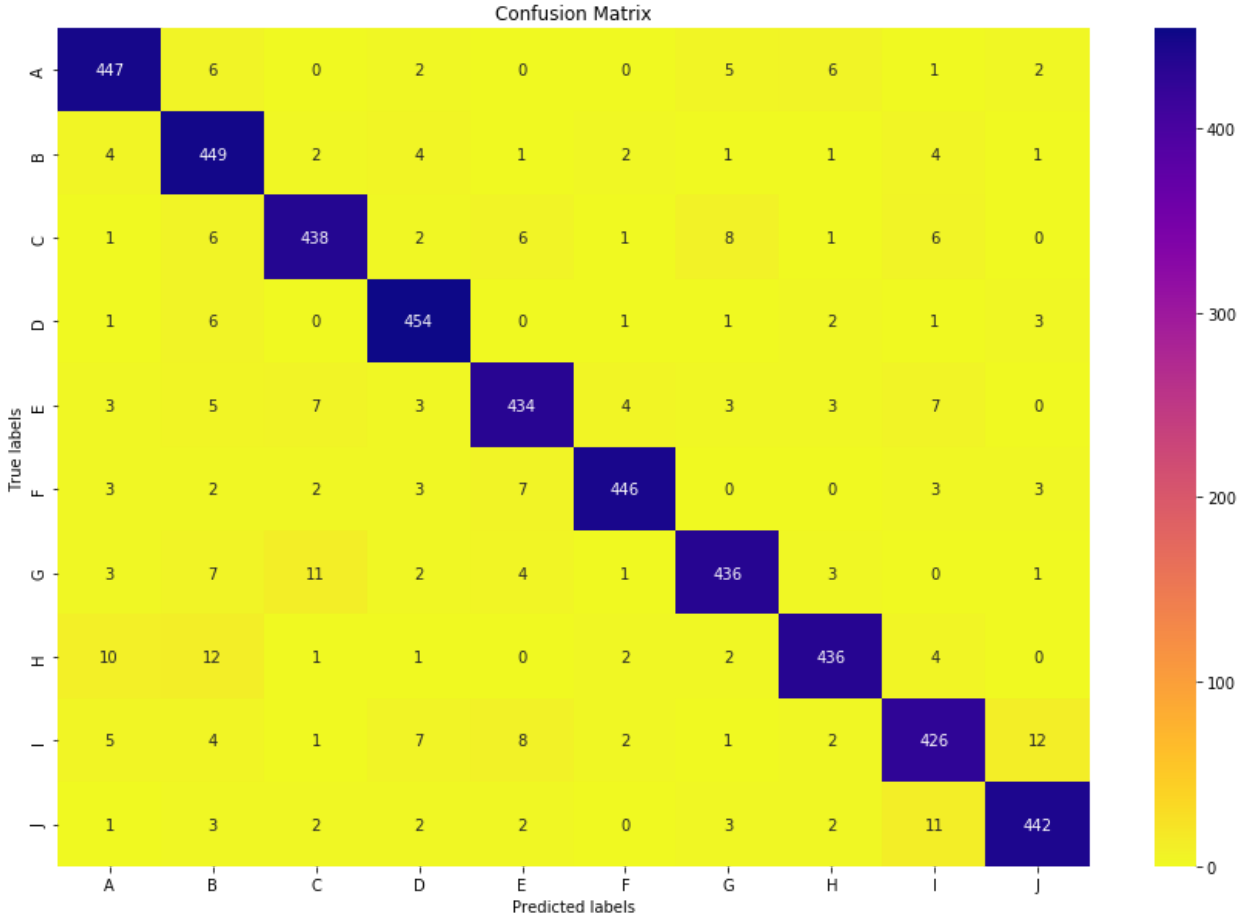
As discussed before, the features from TDA methods and VAE were concatenated and the resulting dataset shape turned out to have 508 features and 14k rows.

GBDT is a resource intensive algorithm and a new high performance library called LightGBM (By Microsoft) was used to implement it. The final model that was trained had the following hyper-parameters:

```
                        LGBMClassifier
LGBMClassifier(boosting='gbdt', learning_rate=0.2, max_depth=30,
               metric='auc_mu', n_estimators=1000, num_iterations=500,
               objective='multiclass', random_state=1234)
```

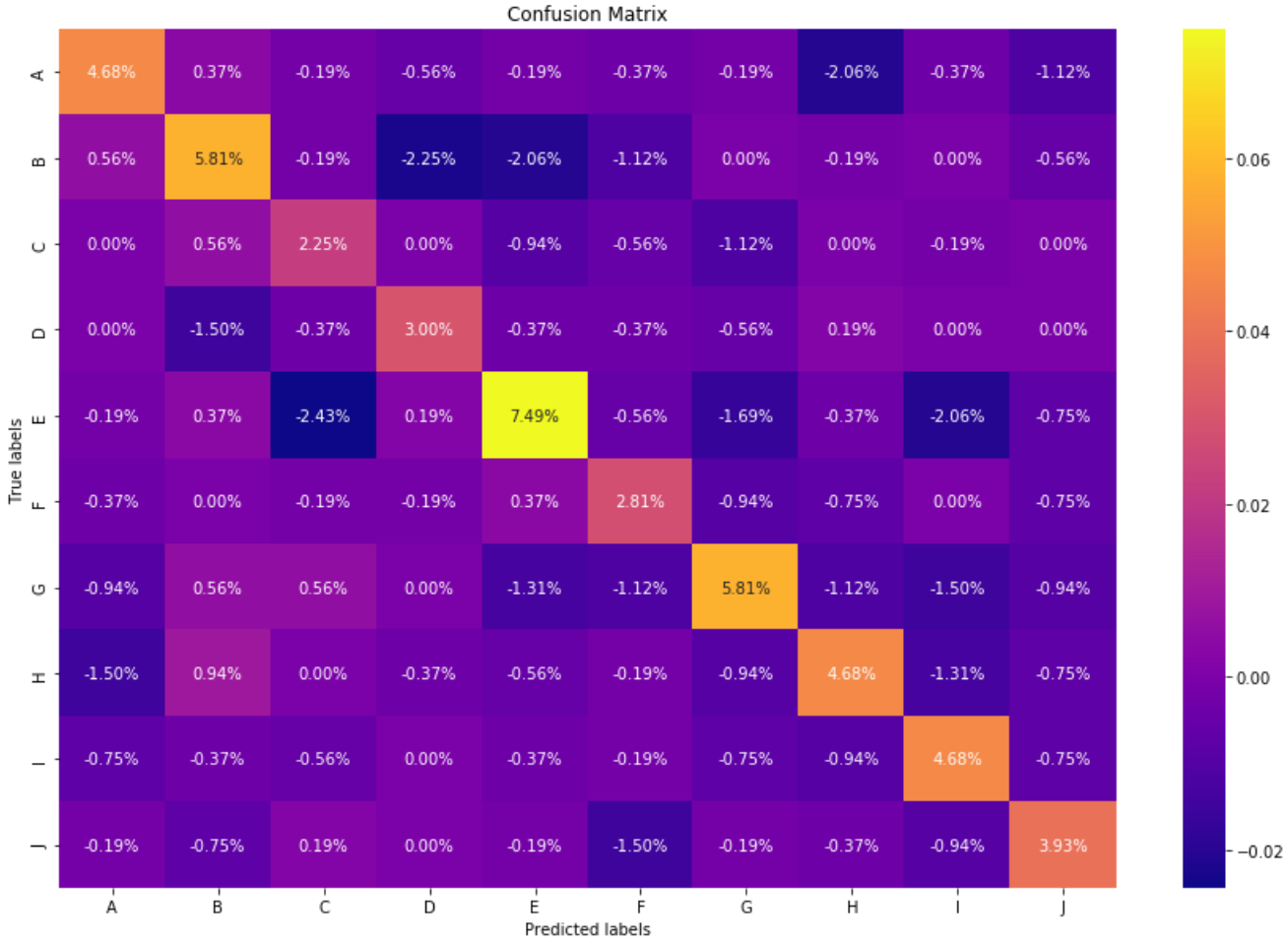
The Classification Report and Confusion Matrix of GBDT are as shown below:

```
Classifier Accuracy:  0.9406743491250533
Classification Report:
              precision    recall  f1-score   support

           A       0.94      0.95      0.94       469
           B       0.90      0.96      0.93       469
           C       0.94      0.93      0.94       469
           D       0.95      0.97      0.96       469
           E       0.94      0.93      0.93       469
           F       0.97      0.95      0.96       469
           G       0.95      0.93      0.94       468
           H       0.96      0.93      0.94       468
           I       0.92      0.91      0.92       468
           J       0.95      0.94      0.95       468

    accuracy                           0.94      4686
   macro avg       0.94      0.94      0.94      4686
weighted avg       0.94      0.94      0.94      4686
```



CONFUSION MATRIX ANALYSIS

As seen from the confusion matrix of the GBDT classifier, misclassifications have reduced drastically for almost all of the class labels. We can visualize the percentage improvement by checking the trace of difference of confusion matrices between the two methods. The resultant percentage heatmap is as follows:



Almost every class had 3% and above improvement with the top-4 being:

1. E (Became robust to misclassification with class C and I)
2. B (Became robust to misclassification with class D and E)
3. G (Became robust to misclassification with class E, F, H and I)
4. A (Became robust to misclassification with class H and J)

CONCLUSION

Topological Data Analysis is a powerful tool in identifying global structures of the data. The features generated from TDA are highly robust to noise. When combined with Deep-Learning approaches, a powerful class of feature engineering algorithms evolve. The best part about the whole exercise is that simple and linear models can perform like state-of-the-art models with more interpretability and explainability just by providing the right set of features.

NOTE

- The initial approach (mentioned in the progress report) used a different dataset. The previous dataset was noisy and there was a substantial amount of data-leakage associated with it. I was getting close to 99% accuracy on the test-set but the model failed when it was checked against the leaderboard dataset. I wanted to concentrate more on the approach rather than solving problems with the dataset and hence took up the notMNIST challenge.
- The previous data was a time-series dataset and thus the final approach was modified to fit in for the current image dataset.

REFERENCES

1. TDA for Time-Series
2. Persistent Homology of Complex Networks for Dynamic State Detection
3. Topological Autoencoders
4. Harnessing the power of TDA for Change Point Detection in Time-Series
5. LightGBM
6. Giotta-TDA