# Classification and clustering analysis of skin lesion images

## 1. Introduction

The 2019 ISIC Skin Lesion Images Classification dataset was chosen for this study. It is essentially an image classification problem, with the data consisting of 25331 labeled images from 8 different diagnostic categories, namely Melanoma, Melanocytic nevus, Basal cell carcinoma, Actinic keratosis, Benign keratosis, Dermatofibroma, Vascular lesion, and Squamous cell carcinoma. The total data size is approximately 9.2 GB. The images depict skin lesions, which are any areas of skin that differ from the surrounding skin in color, shape, size, and texture.


**Figure 1:** *Sample Image of labeled skin lesion data*

The ground truth histogram shown below demonstrates the data imbalance in which a few categories dominate over others. It is always advisable to test several algorithms on such data to see which one performs best. In terms of image classification, convolutional neural nets are said to perform better than other machine learning techniques because when we convert an image to a feature vector, we lose a lot of spatial information in the form of interaction between pixel intensities. CNNs, on the other hand, take this information into account when recognizing edges and thus outperform other methods.

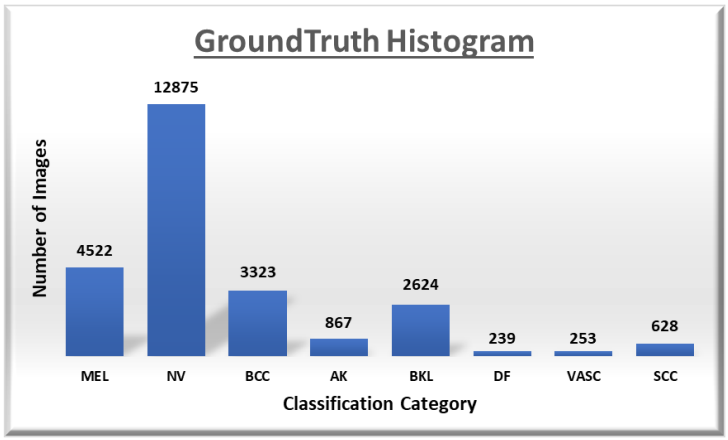



**Figure 2:** *Ground truth histogram of classification categories*

Nonetheless, before implementing CNN, we tried and tested other machine learning techniques, and the performance and results of these methods are discussed in the following sections.

## 2. Image Transformation

We had to transform our image data into a vector form with features before using any machine learning techniques so that the features could be fed into various algorithms for training and testing. The process we used to extract features from an image is shown in the figure below.


**Figure 3:** *Image transformation procedure*

The first step was to install and load the EBImage library in R, which contains all of the basic functions for image processing and analysis. Once the library was loaded, we fixed the pixel size. In this study, we used two independent approaches. In the first approach, we fixed the pixel size to be 64 x 64 x 3, resulting in a feature vector of size 12288 x 25331, whereas, in the second approach, the image size was fixed to be 34 x 48 x 3. The pixel size represents its length, and width and 3 represents the R, G, and B values. The image is then read from the dataset, resized to our specified dimensions, and saved in vector form. This process is looped to read images one at a time, resize them to the desired dimensions, and save the resized data in the final feature vector. Once all of the images have been read, the feature vector is saved locally in the system and used for all subsequent analyses.

## 3. Exploratory data analysis and dimension reduction

The data analysis presented in the current section and the next one deal with images of size 36x48. The original raw data downloaded from Kaggle consisted of 25,331 colored images of varying sizes. However, we observed that an aspect ratio of 4:3 was maintained in all images. Hence, all images were resized to a size that respects the aspect ratio. Of the 25,331 images, 75% of the data i.e. 18998 images were randomly selected to be used as the training set while the rest of the 6333 images were used as test set images. Each image is represented as a 36x48x3 vector of pixel intensities. From Table1, it is clear that there is a heavy **imbalance in the data** and that it is bound to affect classification accuracy.

Table 1: Number of images from each class label in the training set.

| Class | DF | VASC | SCC | AK | BKL | BCC | MEL | NV |
|---|---|---|---|---|---|---|---|---|
| Proportion | 0.0095 | 0.0097 | 0.0246 | 0.0350 | 0.1023 | 0.1310 | 0.1798 | 0.5080 |

A histogram of average red (R), blue (B), and green (G) pixel intensities corresponding to images of each class are shown in Figure 4. We observe that the intensity histograms corresponding to blue and green are all left-skewed with maximum intensities being close to 0.6. On the other hand, the red intensity histogram appears to be bimodal with the maximum intensity being close to 1. If we view image data to be sampled from the corresponding histograms in Figure 4, then the distributions from which different class images are being sampled look very similar.

Since each image corresponds to a feature vector of size 36x48x3 = 5184, the next logical step is to perform dimension reduction. Here, we apply Principal Component Analysis, a linear dimension reduction approach. Scree plots in Figure 5, show that 15 principal components explain 90% of the variation in the data while 90 principal

components are required to capture 96% of the variation in the data. Keeping the computational costs in mind, we believe that retaining 15 principal components is enough. Furthermore, retaining 15 principal components is backed by Kaiser's rule as well i.e. there are exactly 15 eigenvalues greater than or equal to 1.

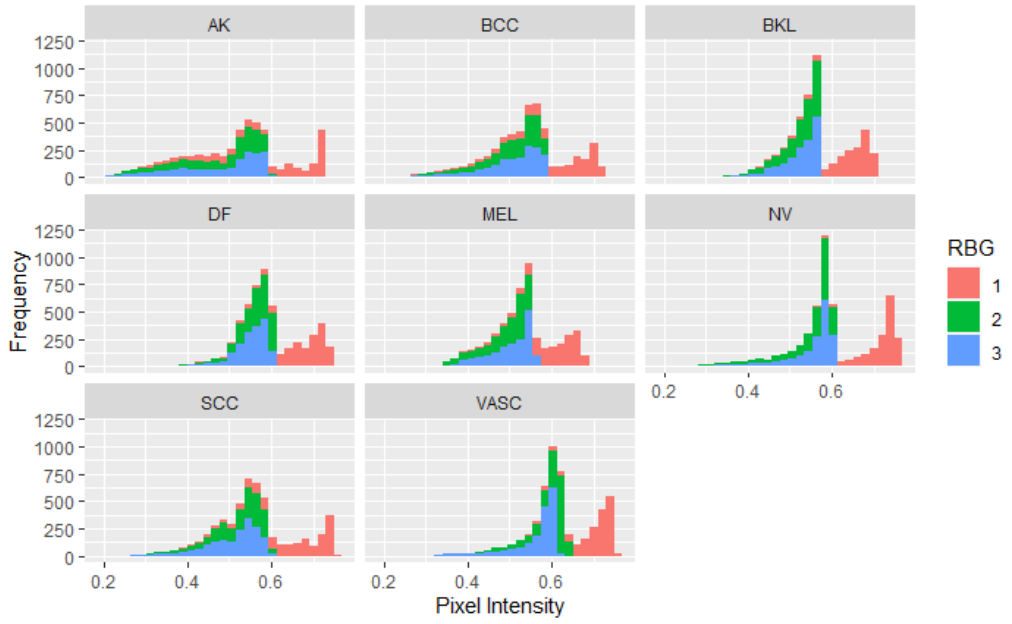
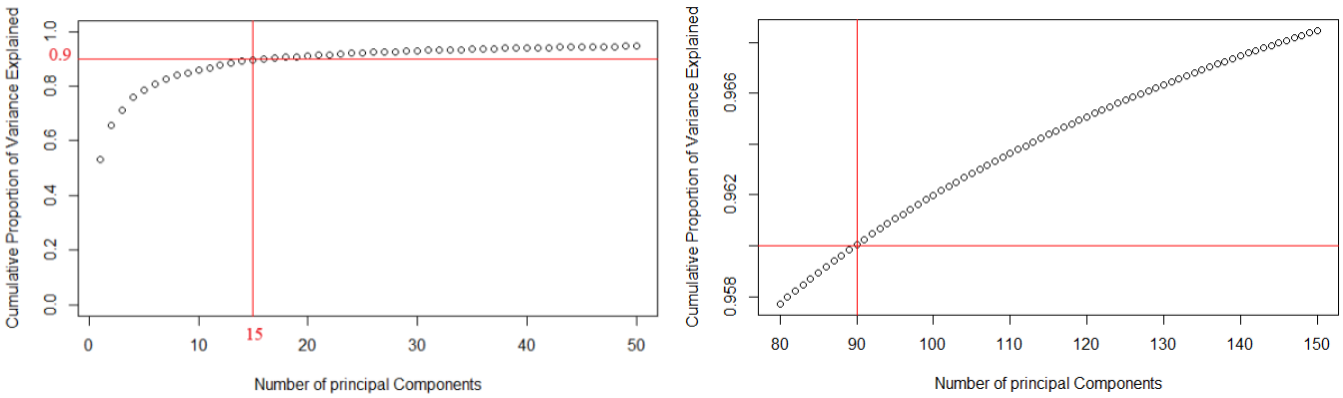



Figure 4: Histogram of RBG pixel intensities categorized by type of skin lesion.

Using these 15 principal components, we obtain a reduced feature vector of length five corresponding to each image. Different classification techniques are applied to the reduced feature set as it is to identify potential problems with the data at hand so that modifications can accordingly be made. Table 2 summarizes the results of the classification methods applied to the reduced feature set, their accuracy, and a brief comment on classification performance. Note that the data was not scaled prior to dimension reduction since all features are pixel intensities measured on the same scale.

Figure 5: Scree plots to identify the required number of PCs.




Based on our observations mentioned in Table 2, the following are the problems that must be addressed to obtain models with improved classification performance:

1. Data imbalance: This issue is dealt with by using a technique called **Synthetic Minority Over-Sampling Technique** (SMOTE). SMOTE is an over-sampling approach that over-samples the minority class by creating synthetic examples.

2. Use of information gain: Instead of bagging and boosting traditional decision trees that split based on information gain, we can focus on bagging and boosting C5.0 trees that use Gain Ratio as splitting criteria.

Table 2: Classifiers applied to the unaltered reduced feature set. The accuracy given is on the test data set.

| **Classifier** | **Accuracy** | **Comments on classifier performance** |
|---|---|---|
| Single decision tree with information gain as the splitting criteria | 51.5% | Due to data imbalance and the use of information gain as the splitting criteria, we observe that the trained decision tree classifies any new point into one of the three most prevalent classes - NV, MEL, and BCC |
| C5.0 which uses Gain ratio as the splitting criteria. | 43.6% | Although lower in accuracy than the first classifier, we observe that C5.0 fits a better model than the first one in the sense that the trained tree classifies new data points into 5 different classes. This improvement is due to the use of gain ratio. However, data imbalance is still an issue. |
| Bagged trees with information gain as splitting criteria (500 trees) | 36% | Poor performance of the bagged classifier is because the weak learners are not independent due to strong predictors like PC1. More prevalent classes like NV are classified the best while less prevalent classes are almost always misclassified. Like in the previous cases, we have the issues of data imbalance, and the use of information gain in splitting. |
| Random Forest involving 500 trees and 3 predictors per tree | 39.5% | The performance of RF classifier is slightly better than the bagged classifier because RF decorrelates the tree topologies. However, the performance is still not high enough and very similar to the bagged trees due to the issues of data imbalance and the use of information gain. |
| Boosting with multiclass cross-entropy loss and 500 trees | 29.7% | Although it is expected that boosting would perform better than random forests, the opposite has been observed implying the possibility of potential outliers in the data that have to be dealt with first. Like in earlier cases, data imbalance and use of information gain are still issues. |
| SVM with linear kernel and cost 10 | 51% | Linear kernel was chosen not only for its low computation cost but also because it performed better than SVMs with the radial kernel. All test set points are classified as either NV or MEL, the two most prevalent classes. The only issue here is the data imbalance. |

## 4. Dealing with data imbalance

Upon applying SMOTE to the current training data, we observe that DF (the least prevalent class) is oversampled while NV (the most prevalent class) is downsampled. The total number of samples we have after applying SMOTE technique is 37,460. The proportion of images of each class after applying SMOTE technique is given in Table 3.

Table 3: Proportion of images in different classes after applying SMOTE over-sampling technique

| Class | AK | BCC | BKL | DF | MEL | NV | SCC | VASC |
|---|---|---|---|---|---|---|---|---|
| Prop. (original) | 0.0350 | 0.1310 | 0.1023 | 0.0095 | 0.1792 | 0.5080 | 0.0246 | 0.0097 |
| Prop. (SMOTE) | 0.0178 | 0.0664 | 0.0519 | 0.4977 | 0.0912 | 0.2576 | 0.0125 | 0.0049 |

Although the issues of data imbalance and use of information gain have been addressed, we haven't yet dealt with outliers. Therefore, boosted C5.0 trees are likely to perform poorly in comparison to bagged C5.0 trees if we continue to use multi-class cross-entropy. An attempt was made to look for R packages that implement boosting

using the multi-class Huber loss, however, we weren't able to find one. Even if we bag C5.0 trees using synthetic data, the problem of having strong predictors like PC1 is not lost.

Therefore, the best way ahead is to consider an RF classifier made using C5.0 trees. Or one can use an SVM as well since the only issue identified in Table 2 regarding SVM was data imbalance which is solved by SMOTE technique. A summary of the performance of classifiers applied to SMOTE synthetic data is given in Table 4.

Table 4: Classifiers applied to data generated by using the SMOTE over-sampling technique. Here we replaced decision trees with C5.0.

| **Classifier** | **Accuracy** | **Comments on classifier performance** |
|---|---|---|
| Boosted C5.0 with 5 boosting iterations | 39% | Boosting performance with as many as 500 trees was 29% when we used a regular decision tree. However, boosted C5.0, with just 5 boosting iterations produced an accuracy of 39%. |
| Bagged C5.0 with 10 trees | 43% | Bagging performance improved from 36% to 43%. Also, note that we used 500 trees earlier (Table 2) while we attained higher accuracy with just 10 trees. Bagging with 100 C5.0 trees further increases the accuracy to 46.8%. |

## 5. Drawbacks of current data analysis

In the data analysis that we did so far, we did not attempt to deal with outliers in the data which are clearly present based on the boosting results we obtained, and also the PC score plot obtained. (PC score plot is not included in the report due to lack of space.) Even if all the issues identified are somehow rectified, there are some inherent drawbacks of this approach that cannot be rectified.

1. Representing images as a vector leads to a loss of spatial information that is present in the 3D array representation of an image.
2. PCA is restricted to looking at only linear combinations of our features. Therefore, it wouldn't be as powerful as a non-linear dimension reduction approach. (Eg. Extracting features using a neural network)

## 6. Results for 64x64x3 Feature Vector

As mentioned earlier in the report we followed two independent approaches to test different machine learning algorithms. This section briefly discuss the results obtained by using a feature vector of size 64x64x3.

**SVM:**

```
Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  radial
       cost:  300

Number of Support Vectors:  636

 ( 128 113 277 76 17 12 6 7 )


Number of Classes:  8

Levels:
 AK BCC BKL DF MEL NV SCC VASC
```

```
                Reference
Prediction  AK BCC BKL  DF MEL  NV SCC VASC
      AK     0   3   1   0   1   2   0    0
      BCC    1  15   7   0   2   5   0    1
      BKL    0   9   5   0   5  12   0    0
      DF     0   1   1   0   0   0   0    0
      MEL    0  10   2   0  15  21   0    0
      NV     0  11   2   0   6 104   0    0
      SCC    0   5   0   1   0   2   0    0
      VASC   0   0   0   0   0   0   0    0

Overall Statistics

                  Accuracy : 0.556
                    95% CI : (0.4921, 0.6186)
    No Information Rate : 0.584
    P-Value [Acc > NIR] : 0.8321
```

- *Training Accuracy: 92.7%*
- *Runtime: 1.33 mins*
- *Testing Accuracy: 55.6%*

**Random Forest:**

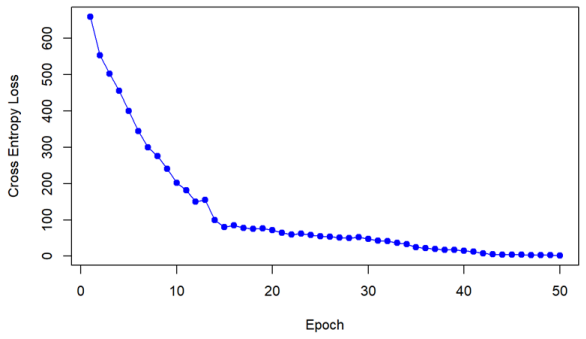
```
                Length Class  Mode
call               3   -none- call
type               1   -none- character
predicted        750   factor numeric
err.rate        4500   -none- numeric
confusion         72   -none- numeric
votes           6000   matrix numeric
oob.times        750   -none- numeric
classes            8   -none- character
importance     12288   -none- numeric
importanceSD       0   -none- NULL
localImportance    0   -none- NULL
proximity          0   -none- NULL
ntree              1   -none- numeric
mtry               1   -none- numeric
forest            14   -none- list
y                750   factor numeric
test               0   -none- NULL
inbag              0   -none- NULL
terms              3   terms  call
```

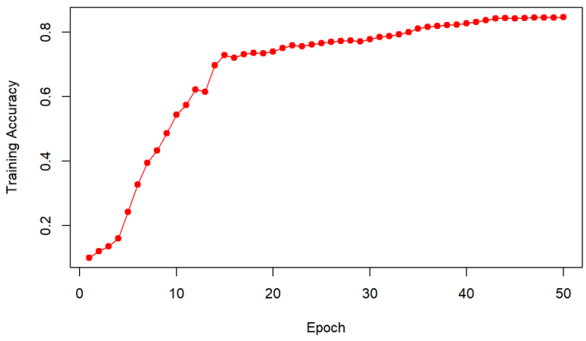
```
          Reference
Prediction  AK BCC BKL  DF MEL  NV SCC VASC
      AK     0   4   0   0   0   3   0    0
      BCC    0  18   0   0   1  12   0    0
      BKL    0   7   1   0   3  20   0    0
      DF     0   1   0   0   0   1   0    0
      MEL    0   7   0   0   8  33   0    0
      NV     0   4   0   0   3 116   0    0
      SCC    0   7   0   0   0   1   0    0
      VASC   0   0   0   0   0   0   0    0

Overall Statistics

               Accuracy : 0.572
                 95% CI : (0.5081, 0.6342)
    No Information Rate : 0.744
    P-Value [Acc > NIR] : 1
```

- *Training Accuracy: 100%*
- *Runtime: 7.9 mins*
- *Testing Accuracy: 57.2%*

## 7. Convolutional Neural Network (CNN)

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 conv2d_11 (Conv2D)           (None, 64, 64, 32)        2432
 conv2d_10 (Conv2D)           (None, 64, 64, 32)        9248
 max_pooling2d_9 (MaxPooling2D)  (None, 32, 32, 32)     0
 conv2d_9 (Conv2D)            (None, 32, 32, 64)        18496
 max_pooling2d_8 (MaxPooling2D)  (None, 16, 16, 64)     0
 conv2d_8 (Conv2D)            (None, 16, 16, 128)       73856
 max_pooling2d_7 (MaxPooling2D)  (None, 8, 8, 128)      0
 conv2d_7 (Conv2D)            (None, 8, 8, 256)         295168
 max_pooling2d_6 (MaxPooling2D)  (None, 4, 4, 256)      0
 flatten_3 (Flatten)          (None, 4096)              0
 dense_3 (Dense)              (None, 64)                262208
 Output (Dense)               (None, 8)                 520
=================================================================
Total params: 661,928
Trainable params: 661,928
Non-trainable params: 0
_________________________________________________________________
```




- *Training Accuracy: 85%*
- *Runtime: 93.6 mins*
- *Testing Accuracy: 79.6%*

## 8. References

*[1] Tschandl P., Rosendahl C. & Kittler H. The HAM10000 dataset, a large collection of multi-source dermatoscopic images of common pigmented skin lesions. Sci. Data 5, 180161 doi.10.1038/sdata.2018.161 (2018)*

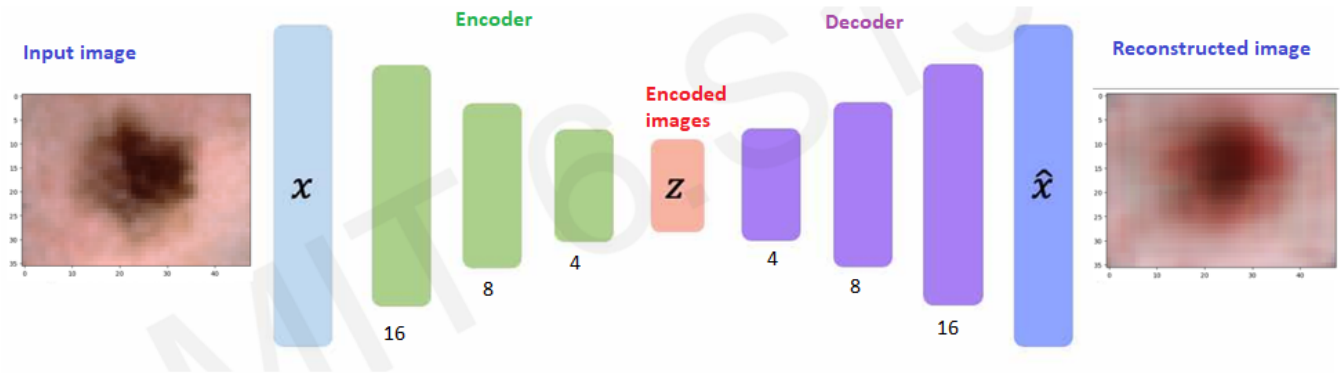
*[2] Noel C. F. Codella, David Gutman, M. Emre Celebi, Brian Helba, Michael A. Marchetti, Stephen W. Dusza, Aadi Kalloo, Konstantinos Liopyris, Nabin Mishra, Harald Kittler, Allan Halpern: "Skin Lesion Analysis Toward Melanoma Detection: A Challenge at the 2017 International Symposium on Biomedical Imaging (ISBI), Hosted by the International Skin Imaging Collaboration (ISIC)", 2017; arXiv:1710.05006.*

*[3] Marc Combalia, Noel C. F. Codella, Veronica Rotemberg, Brian Helba, Veronica Vilaplana, Ofer Reiter, Allan C. Halpern, Susana Puig, Josep Malvehy: "BCN20000: Dermoscopic Lesions in the Wild", 2019; arXiv:1908.02288.*

## Contributions (First names mentioned alphabetically)

1. Aditya: Resized the images to 64*64*3 and extracted the features of images. Implemented support vector machine (SVM), boosting, random forest (RF), and logistic regression. Implemented Convolutional Neural Network (CNN) for image classification.
2. Padma: Attempted extracting features corresponding to images. The R implementation, and report writing correspond to sections 3, 4, and 5. (Implementation of standard classifiers, SMOTE oversampling)
3. Sudhir: Data Exploration, Image transformation process in R, feature extraction and implementing R inbuilt machine learning algorithm support vector machine, logistic regression.

*Figure 5: Convolutional Autoencoder*




### 4.1. Encoder Architecture

Pictorially, the three green layers in Figure 5 represent the three layers of our encoder. Each green layer represents one convolutional layer followed by a max-pooling layer. The three convolutional layers were padded such that the output of the convolutional layer has the same dimension as that of the input. The number of filters used in each of the three layers is 16, 8, and 4 respectively. The convolution kernels were matrices of size 3x3 while the max-pooling layers used filters of size 2x2 (i.e. downsampling by a factor of 2).

### 4.2 Decoder Architecture

The purple layers in Figure 5 denote the layers associated with the decoder. Each layer constitutes a convolutional layer followed by an upsampling procedure. Since we want to reconstruct the original image, the decoder architecture should be an "inverse" of the encoder architecture. As is evident from Figure 5, the number of filters used in each convolutional layer is inverted. Convolutional layers are padded like in the Encoder. A convolutional kernel of size 3x3 is again used followed by upsampling the rows and columns by a factor of 2. Upsampling is performed by repeating each row and each column twice.

Note that the choice of kernel dimension and up/downsampling factor is based on the parameter values that are used most often.

## 5. Autoencoder Training

Of the 25,331 images, we used 18,998 images to train the autoencoder for 10 epochs using mini-batch stochastic gradient descent with batch size 512. In particular, we used the Adam optimizer to minimize binary cross-entropy loss between input and reconstructed images. The loss function value was monitored on the validation set consisting of 6333 images. The loss converged to 0.58 approximately after 10 epochs. The total number of parameters involved in the autoencoder is 3947.

In the sections that follow, K-means and Gaussian mixture model techniques were implemented on the 3x4x4 dimensional encoded images instead of the entire images. A point to note is that these reduced features are **correlated.** Unlike PCA, autoencoders, do not output uncorrelated features.

## 6. K-Means Algorithm

Kmeans algorithm is an iterative algorithm that tries to partition the dataset into Kpre-defined distinct non-overlapping subgroups (clusters) where each data point belongs to only one group. It tries to make the intra-cluster data points as similar as possible while also keeping the clusters as different (far) as possible. It assigns data points to a cluster such that the sum of the squared distance between the data points and the cluster's centroid (arithmetic mean of all the data points that belong to that cluster) is at the minimum. The less variation we have within clusters, the more homogeneous (similar) the data points are within the same cluster.

The Adjusted Rand Index (ARI) results for different random initializations are shown in the table below.

| Random State | ARI |
|---|---|
| 0 | 0.0326 |
| 1 | 0.0325 |
| 2 | 0.0332 |
| 3 | 0.0331 |
| 4 | 0.0334 |

It is evident form the table that the ARI value does not change significantly with different random initializations. Further, the ARI values for all of the random initializations are close to 0 indicating that the predicted labels do not match well with the true labels..

The average ARI values for different number of clusters (K) are shown below

| Number of Clusters | ARI |
|---|---|
| 4 | 0.058 |
| 5 | 0.057 |
| 6 | 0.052 |
| 7 | 0.027 |
| 8 | 0.033 |
| 9 | 0.027 |

Again, the ARI value does not change significantly with the number of cluster. However, ARI is maximum for K=4, this is probably due to the imbalance in the dataset. With lesser number of clusters, the data belonging to classes having fewer number of images are combined with the dominant classes, improving the overall ARI.

## 7. Gaussian Mixture Model (GMM)

Gaussian mixture models are probabilistic models that are used to represent normally distributed subpopulations within a larger population. In general, mixture models do not require knowing which subpopulation a data point belongs to, allowing the model to learn the subpopulations on its own. Because subpopulation assignment is unknown, this is a form of unsupervised learning. GMM is an expectation maximisation algorithm that is similar to the K-means algorithm. The gaussian mixture has a more flexible decision boundary than the K-means algorithm. Furthermore, GMM employs a probabilistic algorithm, which means that we know the degree of certainty behind the classification of a specific data point to a specific cluster.

The encoded data is passed through GMM algorithm and following ARI values are obtained for various cluster values.

| Number of Clusters | ARI |
|---|---|
| 4 | 0.0861 |
| 5 | 0.1199 |
| 6 | 0.0797 |
| 7 | 0.0634 |
| 8 | 0.0541 |
| 9 | 0.0497 |

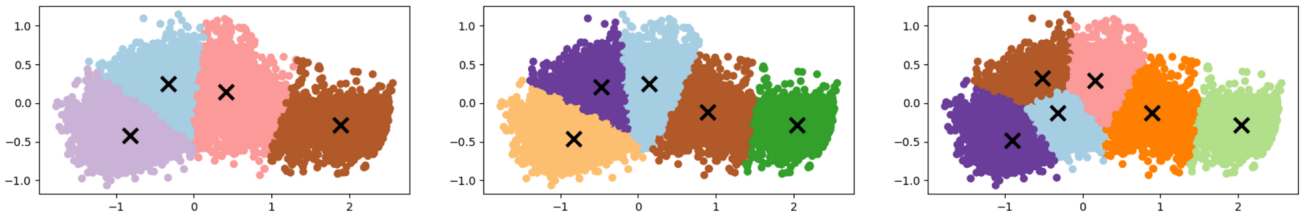
From the above ARI values it is evident that when compared to the k-means algorithm, the GMM performance is undeniably better. The ARI varies with cluster size, reaching a maximum at cluster size 5. Furthermore, for any cluster size, the ARI value still remains close to zero, indicating that the predicted labels do not match well with the True labels.

## 8. Principal Component Analysis (PCA)

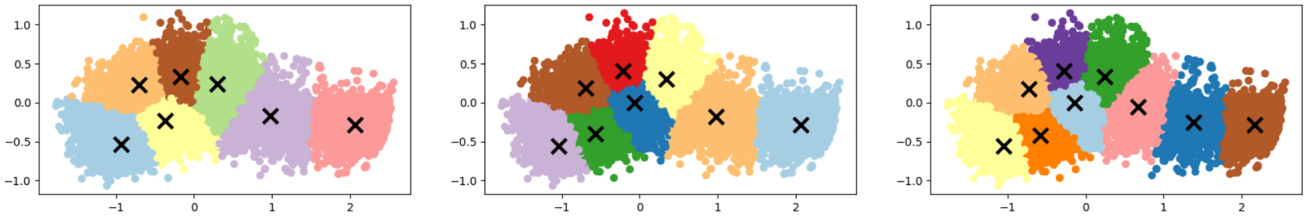
PCA was applied to 3*4*4 encoded data outputted by autoencoders to reduce the data to 2D.

K-means algorithm implemented for different number of clusters on the 2D data yielded the results as shown in the figure below.

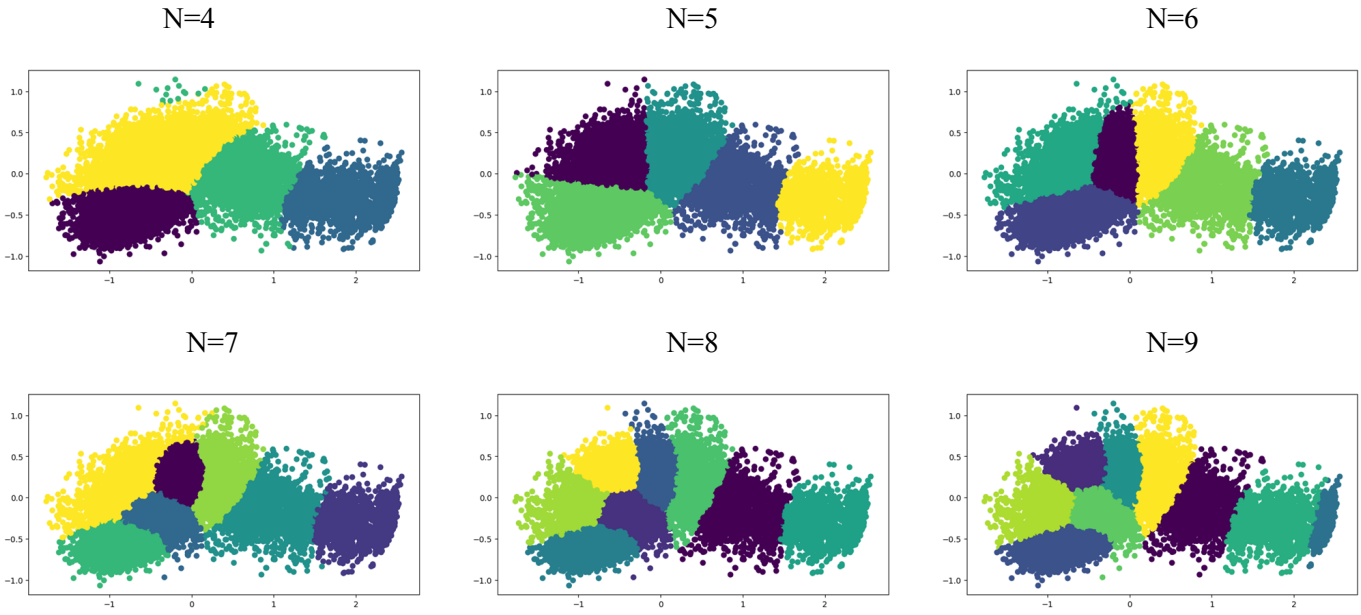
N=4 N=5 N=6



N=7 N=8 N=9



GMM algorithm implemented for different number of clusters on the 2D data yielded the results as shown in the figure below.




## 9. Scope for further improvement

First, the most obvious improvement in clustering performances can be obtained by using images of better resolution rather than the current resolution of 36x48x3. The architecture of the autoencoder used is very basic. Fine-tuning the autoencoder by increasing the number of convolutional layers, increasing the number of filters used per layer, changing the filter size, and up/downsampling factors can improve the autoencoder's performance further.

Another direction of improvement is to apply a log transform to the pixel data before applying the GMM algorithm. Since the pixel intensities are all right-skewed (Figure 4). This will further improve the performance of GMM as a log transform will make the distribution more symmetric, as assumed in the GMM model.

## 10 . References

*[1] Tschandl P., Rosendahl C. & Kittler H. The HAM10000 dataset, a large collection of multi-source dermatoscopic images of common pigmented skin lesions. Sci. Data 5, 180161 doi.10.1038/sdata.2018.161 (2018)*

*[2] Noel C. F. Codella, David Gutman, M. Emre Celebi, Brian Helba, Michael A. Marchetti, Stephen W. Dusza, Aadi Kalloo, Konstantinos Liopyris, Nabin Mishra, Harald Kittler, Allan Halpern: "Skin Lesion Analysis Toward Melanoma Detection: A Challenge at the 2017 International Symposium on Biomedical Imaging (ISBI), Hosted by the International Skin Imaging Collaboration (ISIC)", 2017; arXiv:1710.05006.*

*[3] Marc Combalia, Noel C. F. Codella, Veronica Rotemberg, Brian Helba, Veronica Vilaplana, Ofer Reiter, Allan C. Halpern, Susana Puig, Josep Malvehy: "BCN20000: Dermoscopic Lesions in the Wild", 2019; arXiv:1908.02288.*

*[4] Blogpost given by the official keras website - click here*

## Contributions (First names mentioned alphabetically)

1. Aditya: Applied K-means algorithm to the encoded data from the autoencoder to identify the clusters and evaluated it's performance for different random initialization and cluster sizes. Applied Principal Companent Analysis(PCA), to the encoded data to convert it to 2D and applied K-means to visually evaluate the clusters.
2. Padma: The python/R implementation and report writing correspond to sections 1 (Section 1 done together with Sudhir), 3, 4, 5, and 9. (EDA, Implementation of autoencoder)
3. Sudhir: Unsupervised clustering was performed using the Gaussian Mixture Model algorithm on encoded data. The algorithm's performance was evaluated using the ARI index, and the variation in ARI with cluster size was investigated. PCA was used to reduce the data dimension to two, and the GMM was repeated and plotted for better visualisation and understanding.