Imperial College London

Department of Earth Science and Engineering

MSc in Applied Computational Science and Engineering

Independent Research Project
Final Report

Deep Learning Algorithms Applied to Surface Mapping of Mars

by

Niranjana Sundararajan

Email: niranjana.sundararajan21@imperial.ac.uk
GitHub username: acse-ns1321
Repository: https://github.com/ese-msc-2021/irp-acse-ns1321

Supervisors:

Dr Cédric M. John
Dr Philippa J. Mason

August 2022

Abstract

An accurate understanding of Martian Terrain and its features is crucial not only to further scientific research in the fields of planetary science but also essential to engineers responsible for planning rover landings and building models for landing site traversibility analysis. While current tools and research provide for supervised classification of landforms and features based on known/suspected terrain classifications,these often fail to acknowledge the possibility that Martian terrains may exhibit features that are significantly different from our present understanding of terrains. Using the Mars Reconnaissance Orbiter's HIRISE images that have a resolution of 25cm/pixel, this project focuses on creating the first publically available API package to query,filter,pre-process and download the images and extract hidden patterns and features in the images using three different deep learning based encoding algorithms.The images are then categorised using nine different clustering algorithms based on similarity of their derived features and tested based on multiple classification and clustering metrics.

Introduction

The Martian Terrain, much like earth, possesses a multitude of landscape features. The United States Geological Survey has divided the Martian land into thirty quadrangles, each defined by a specific latitude and longitude. Some characteristic features of the terrain include volcanoes, canyons, impact craters and dry lake beds. A brief study of the main features based on available information are summarized in Appendix A.

Presently, there are mainly two types of research objectives that are considered while building deep learning models involving Martian terrains. In the first kind, researchers have used automated image analysis and deep learning techniques for the detection of specific terrain features such as dunes fields[3], craters[12,19], volcanic cones[19] and aeolian ridges[20]. In the second, researches have used deep learning algorithms to build supervised algorithms that classify available images into defined classes. The Soil Property and Object Classification[22] software successfully classifies full resolution HIRISE images into 17 terrain classes using a specialized fully convolutional neural network.

A major shortcoming in these generalized classifiers using convolutional neural networks is that these models, that are supervised by nature, require images to be manually labelled before training. Therefore, considering the total data volume and the abundance of new images, even the recent, more accurate classifiers such as those deployed within the Planetary Data System's Imaging Atlas[29] will likely fail to account for newly captured features.

Objectives

This research has the following three objectives:

- The **first objective** in building this model is the ingestion of the HIRISE images and the associated metadata from both NASA's PDS as well as the secondary database from University of Arizona's HiWish Database . The built solution is the building of a python package, installed via pip, that allows for easy querying and downloads. The tool web scrapes the necessary file hierarchies from the websites as well as creates an updatable database that holds all the currently available information.
- The **second objective** is to supplement the package with different machine learning algorithms for encoding, dimension reduction, clustering and visualisation of the data. Each of these aspects can be implemented in a number of different ways depending on the objective of the researcher and the package must not only support these implementations but also allow for expansion of the package.

- The **third objective** is to use the built tool to conduct a number of machine learning experiments and analysis that allow for
  - Exploratory analysis of a randomly downloaded dataset by the user
  - Discovery of potentially new features and clustering results through fully unsupervised learning techniques

Code Metadata

The HIRISE database is populated with over 73,000 images and over 25 parameters that define each image. The Planetary Data System(PDS), maintained by NASA, requires the user to manually navigate to specific folders and download single image. The process being tedious and time-consuming, is a major barrier for both machine learning engineers and research scientists in the field. Therefore to consume this data, the first publically available tool, the "HIRISEimgs" package was built that allows for query, filtering, downloading and automatically updating to keep up with the growing database of images. The tool also integrates science theme from the University of Arizona's HiWish Database. The File Structure and the associated functions are shown in the figures below:

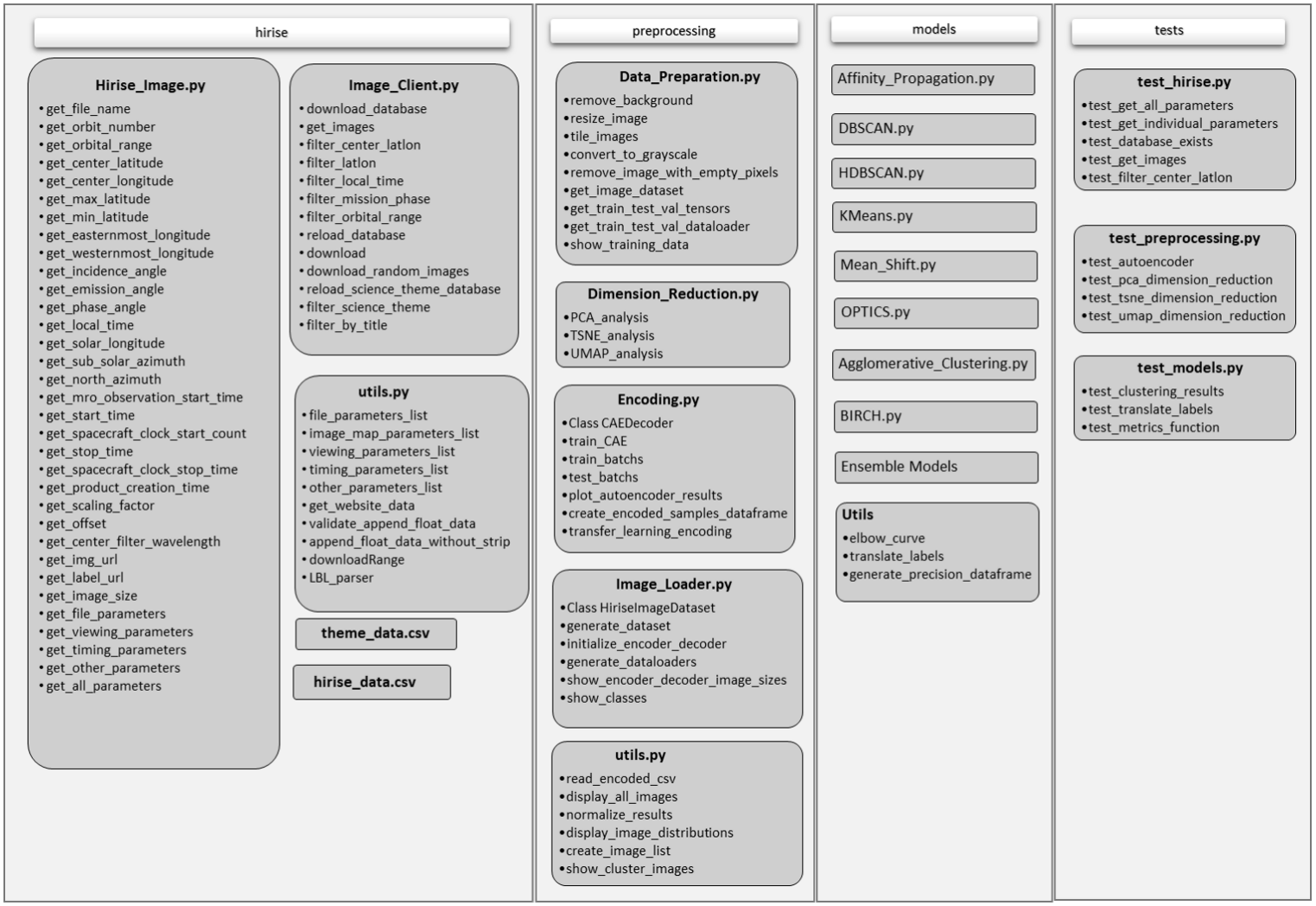


Figure 1: Package Structure and Functions

In the figure above, we see that hirise, preprocessing and modules are the three main sub-packages,

and tests(containing pytests) that rigorously test reach of these modules. These have 56, 29, 12 and 12 functions each, that can be further expanded depending on future requirements.

Coding Methodologies

A special consideration is given to reproducibility and scalability of the code. Continuous integration in git, versioning and automation through workflows were prioritised and implemented. The workflows test code for pep8 standards, automatically update documentation and automate the publishing of new versions to test-Pypi depending on the stage of development. This will allow researchers to expand the tool's use and functionalities to support their own research. Based on select considerations from the seminal agile manifesto[8] and the more recent revisions in[4] and[1], two main practices that were observed and implemented :

- **Continuous Delivery**: Frequent feedback and continuous improvement and integration of multiple versions of the software released during the development process. There were over 75 different versions available on the test Pypi website with a progressive improvement and bug fixes with each version.
- **Problem Breakdown**: The software was developed first by planning the overarching architecture of features and functions required, then individual classes, their associated functions and utility modules were planned and executed in stages. The code undergoes refactoring and optimisations when the logic is complete.

Methodology and Qualitative Observations

Curation of Dataset

HIRISE images have characteristics that pose many challenges when used in machine learning applications[24]. While many are explored in the literature studied, the ones specific to HIRISE Images include the fact that they are, firstly, **very large in size**(ranging from 1GB to 87GB). Secondly, the images are extremely **feature-rich** and contain minute details that need to be extracted and encoded correctly. Thirdly, not only are these images high resolution and feature-rich they are also grey-scaled, which creates higher **complexity in a single channel**. Tiling, a pre-processing method that is necessary, unintentionally creates noise in the dataset – both through generating areas that do not capture any features as well as creating tiles that are largely black-space due to the projected nature of the images. Thus, these challenges were to be kept in mind, not only during the machine learning stage of the project but also in the dataset curation stage.

The test dataset was generated considering aspects such as textures, objects and change detection using the best practices adapted from the research available for creating benchmark datasets, for aerial[14] and remote sensing[13] images. Considering the availability of images of a specific feature, the overall balance of features and with the objective of covering all features, the dataset had the following 14 image classes – dunes, glaciers, scalloped depressions, gullies, layers, cratered cones, concentric crater, pedestal crater, ridges, lava flow, dust devil, brain-terrain, chaos terrain and slope streaks.

A sample of the clusters from the curated test dataset are shown in the figure below:

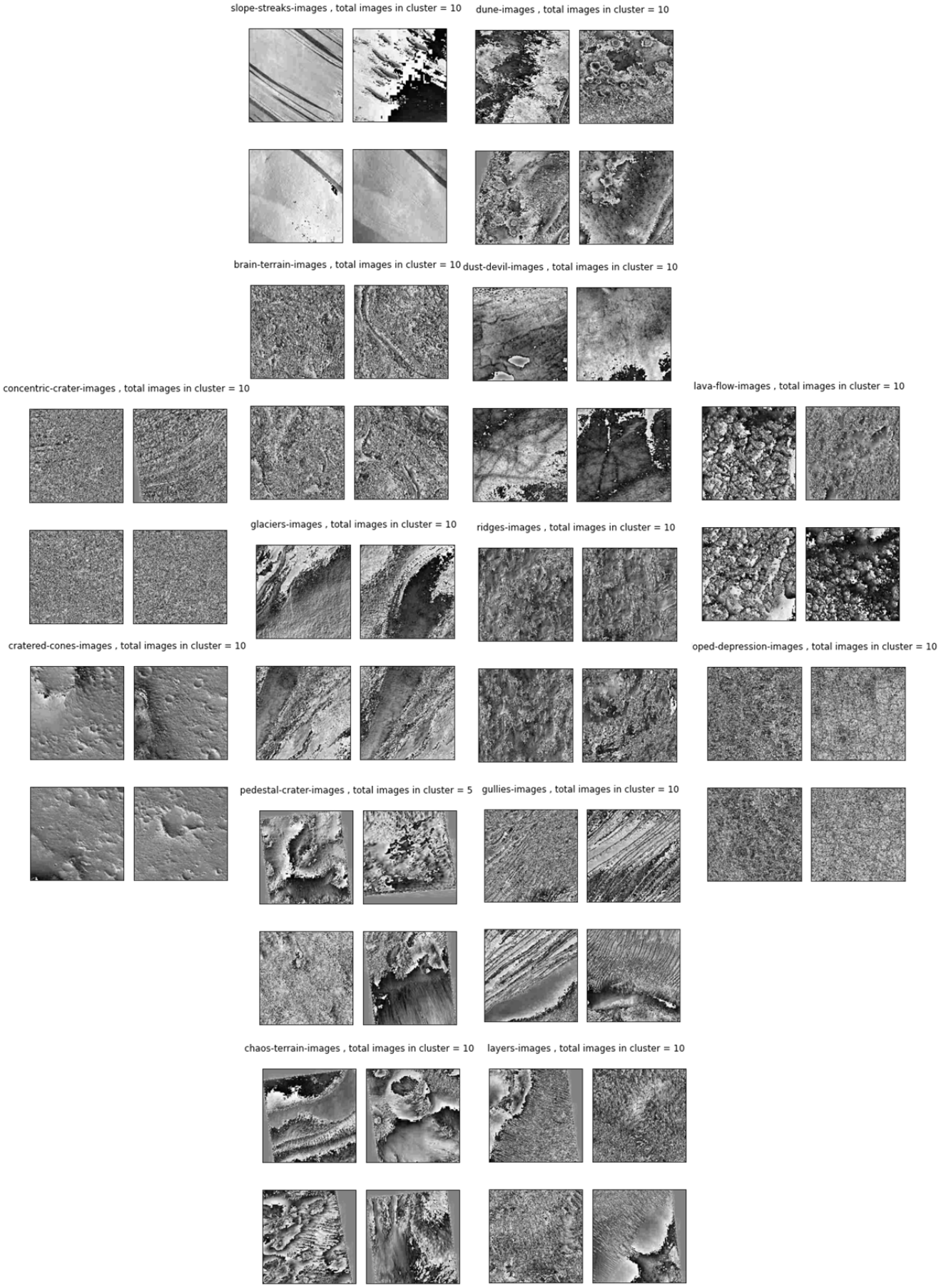


Figure 2: Curated Test Data Set with fourteen classes

Methodology:

The image below shows the different steps in the research:

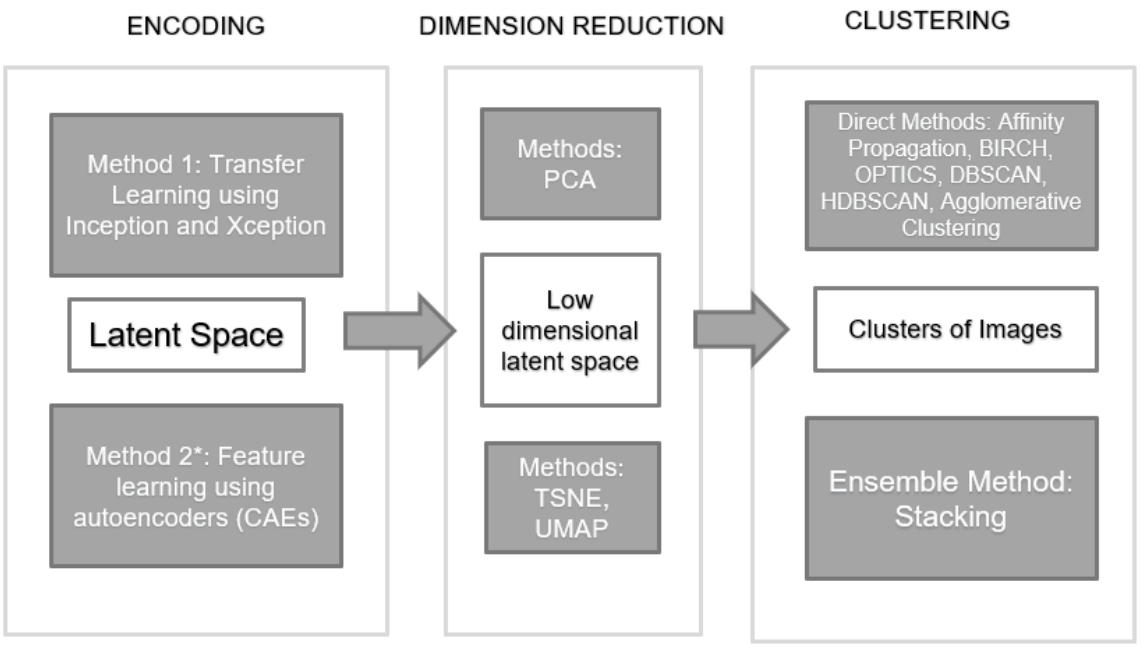


Figure 3: Methodology Schematic

STEP 1 : Encoding

The literature extensively explores encoding algorithms that allow for appropriate feature selection and accurate feature preservation in the cases of face detection[5]. Considering that the images range from 1GB to 87GB's, it is essential that encoding not only captures features correctly but also allows for dimension reduction during and after latent space generation. For this purpose **three models and two machine learning techniques** were considered to encode the data.
In the first method implemented is the **transfer learning technique**. This technique has been used in both research environments[23] and in production[26]. It is especially effective in the three scenarios – First, when the dataset is exceptionally large such that researchers without access to powerful supercomputers will be unlikely to use all the available data for training models. In this case, using previously trained, state-of-the-art models allows practitioners to exploit the capabilities of these models for a new dataset while making the necessary modifications to suit individual cases. Secondly, practitioners maybe limited by the difficultly and cost, of annotating datasets. Transfer learning may be used in an unsupervised manner such that the challenge of analysing domain specific data is not restricted to domain specialists. Finally, the practitioner may have a very limited dataset of samples available for training. Using models optimised on a limited dataset can severely limit performance. Transfer learning's generalisation capabilities have been widely used in these cases and are called fuzzy transfer learning techniques[9].
Of the conditions discussed above, the HIRISE dataset satisfies the first two cases. The two pre-trained models used for this section were **InceptionV3** and **Xception**, along with the custom designed convolutional auto-encoder.

The schematic for the models are shown in the figure below. For a more detailed model plot, view Appendix B and C. The schematic and outputs after encoding for the two models are given below:

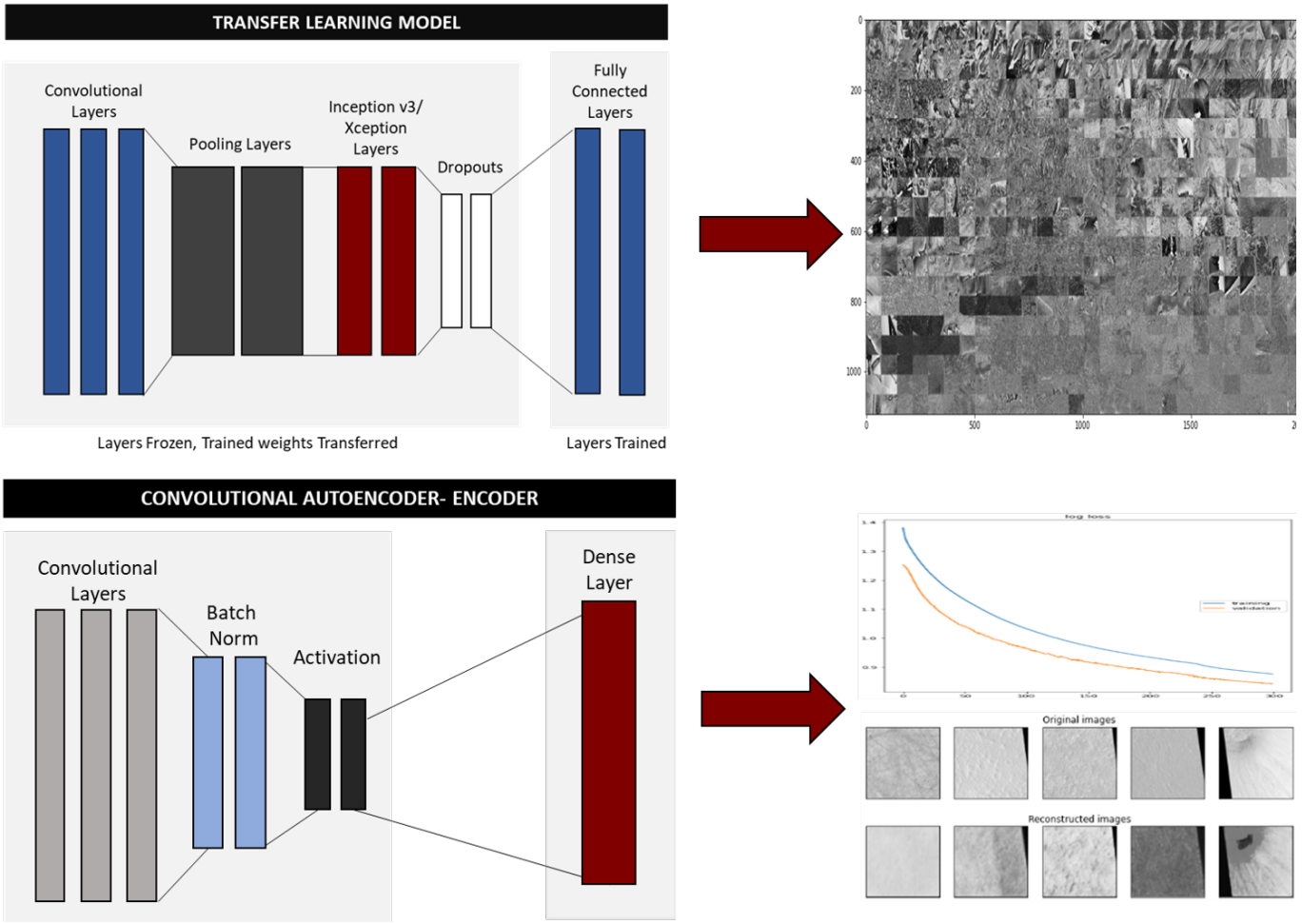


Figure 4: Encoding model Schematics - Top (a) Transfer Learning Model and Image Distribution Results after Inception and Bottom (b) Convolutional Auto-Encoder and Image Reconstruction Results

The general characteristics and the reasons for selection are discussed below:

The Inception Model : The Inception model, as described in the research[25] with 23.62 million parameters, was developed with the aim of improving the classification prowess of existing deep networks by using filter-level sparsity as an architectural improvement, and to do so considering hardware capabilities of existing systems. The network was trained on a dataset of 1000 classes, originally a part of the ImageNet network. The network is 27 layers deep, when accounting for pooling and includes convolutional, average pooling , a fully connected layer, SoftMax and activation layers and a dropout layer with 70% dropout. This lightweight network was an ideal first choice for the HIRISE dataset.

The Xception Model : Xception[7], with 22.85 million parameters, modifies Inception to improve the computationally costly convolutions present in Inception both spatially and across the depth of the model. This model, unlike Inception also does not introduce any non-linearity which leads to an improvement in validation accuracy. A slightly faster model compared to inception, that significantly outperforms Inception on larger datasets with many classes, Xception showed good promise for the HIRISE dataset and therefore was the second choice. Other common models considered included VGG which although more complex than Inception and Xception , when used on the ImageNet dataset

showed both lower accuracy and higher computational costs due to large number of weight parameters. Resnet, although a highly complex advanced architecture is trained to 'unlearn' features the network considers not useful, which in the case of our geospatial images can lead to loss of relevant features.

The second method uses a **convolutional auto-encoder**(CAE) which extracts image features and encodes the image using a combination of down-sampling and convolutions. Auto-encoders are used to both reconstruct[27] and reduce dimensions of high-resolution images[6]. Further, the convolutional component of an auto-encoder is very useful to maintain spatial features of the images. A major challenge is the difficulty in tuning the auto-encoder to be able to generate images with adequate details, while using the available computational resources. This is especially difficult for the HIRISE dataset, since the land-forms are highly complex and features maybe similar and/or overlapping. The reconstructions observed were visually verifiable as shown in figure

STEP 2 : Dimension Reduction

Once the high dimensional latent space is generated, it is useful to further reduce the dimensions of the stored latent space and reorient the data such that the clusters can be best represented visually. The algorithms to achieve dimension reduction rely on either projection or manifold learning. The three most commonly used are Principal Component Analysis(PCA), t-distributed stochastic neighbour embedding(t-SNE) and Uniform Manifold Approximation and Projection for Dimension Reduction(UMAP) and the impact of these have been rigoursly analysed in the literature.[17]. The UMAP has demonstrably shown, in previous research conducted, structure preservation on Mars images[2]. These were the three-dimension reduction algorithms that were chosen and implemented. Their characteristics are as follows:

Principal Component Analysis: PCA is the simplest, fastest and most computationally efficient technique to reduce the dimensionality of the data by plotting the data along the components that capture the most variance in the data. The non-linear nature of PCA fails to capture the nuances image data[15] in various applications[21]. For the HIRISE dataset, 30% of the variance is captured by the first 10 components for the test dataset and 60 components capture around 83% which demonstrates the complexity of the data. As the size of the dataset increases, the variance decreases. Thus, while PCA is a good initial preprocessing step and is useful for detailed analysis of specific clusters, it is not the best choice.

T-Distributed Stochastic Neighbour Embedding: T-SNE uses the idea of dimension reduction and, unlike PCA , applies it effectively to high dimensional datasets. This method[28] plots each high dimensional data point into a two- or three-dimensional space such that similar clusters are closer together on a probabilistic scale. T-SNE when applied to HIRISE data shows good separation and when on its own is sufficient to observe distinguishable clusters. However, for lager datasets it was observed both in the literature[11] and on HIRISE datasets to produce better results after using PCA for the preprocessing step .

Uniform Manifold Approximation and Projection(UMAP) Like the t-SNE, the UMAP is a non-linear dimensionality reduction method but the difference is in the scalability of the mathematical method[16]. For larger datasets with high dimensional data, UMAP produces better and more distinct separations and visualizations[18]. On the HIRISE dataset, especially for non-test cases that require a large number of images to be analysed(tested on 700 tiles of the dataset), UMAP performs significantly better than both t-SNE and PCA.

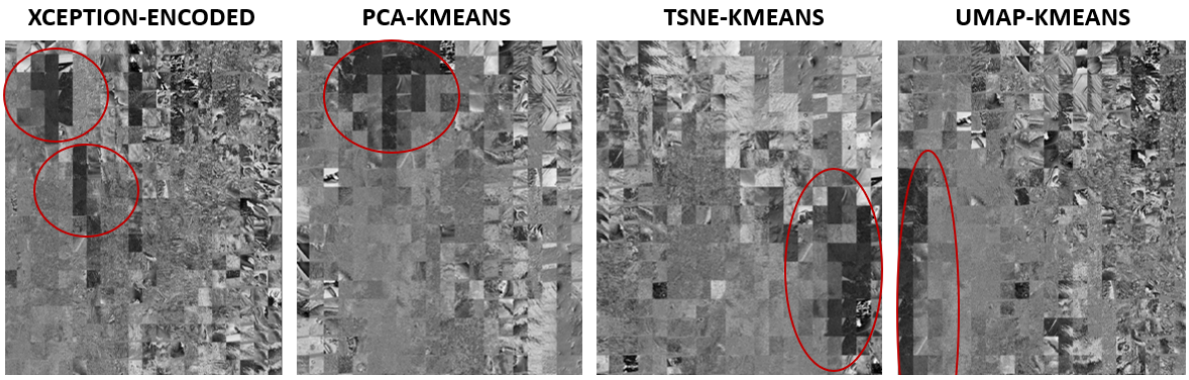


Figure 5: Change in Representation of Encoded Images after PCA, UMAP and t-SNE

Distinct patterns of clustering are observed when each of the dimension reduction techniques are applied on the encoded Xception image data. In Figure 5, as highlighted areas show how the darker set of images transforms from the two distinct groups in the Xception encoded dataset to differently placed and arranged clusters in different dimension reduction techniques.

STEP 3 : Clustering

Two types of clustering algorithms were considered depending on the objective of the research, namely **clustering for exploratory data analysis** and **clustering for the discovery of new features**. These were chosen based on a study described in "A Comprehensive Survey of Clustering Algorithms" [31]. (Refer Appendix G, H, I for further details)

Exploratory Analysis : Group 1 Models

For exploratory analysis, the program specifies fourteen major clusters as reviewed previously.The three models implemented(hereafter referred to as Group 1 Models) that fall under this category include :

1. **K- Means**: Each encoded sample is categorized into a cluster based on the randomly chosen centroid that is nearest to it. K Means minimises variances within clusters using Euclidean distances. K Means, due to its simplicity, scalability and guaranteed convergence is often a reliable first choice as the classification algorithm for our purpose.
   K Means requires the user to specify the number of clusters. In the case of exploratory analysis, this is useful. A drawback observed is the requirement to set the seed manually, to ensure that the results are reproducible because K means is highly dependent on initial values. Further it is difficult to cluster images of varying sizes and densities and to detect outliers, since all images are assigned to one of the specified clusters.

2. **Agglomerative Clustering(AC)**: This is a form of hierarchical clustering using the "bottom-up" approach where each image first forms its own cluster and through a greedy approach connect with similar images in pairs until distinct clusters are formed. Like K-Means the number of clusters are predetermined and the images merge up the hierarchy to form the required number of clusters. While this algorithm performs quite well in terms of precision for the HIRISE dataset(always above 60%), it is exceedingly slow in its execution and is therefore unsuitable for medium to large datasets.

3. **BIRCH(Balanced Iterative Reducing and Clustering using Hierarchies) Analysis** BIRCH, like AG is derived from the family of hierarchical clustering algorithms. BIRCH is especially useful to cluster large datasets – the likes of which will not fit in memory and will likely cause issues in the two aforementioned clustering algorithms. BIRCH, therefore not only optimises for precision but also manages both time and memory constraints effectively and is therefore an appropriate first choice in many cases.

Discovery Analysis: Group 2 Models

When the purpose of clustering is to discover new features, it necessary that the chosen algorithms do not require the user to specify the number of clusters. These algorithms are characteristically density-based clustering algorithms that group together points that are closely packed together in a highly dense area. The algorithms chosen that satisfy this condition(hereafter referred to as Group 2 Models) are explained as follows :

1. **Density-Based Spatial Clustering of Applications with Noise (DBSCAN)**: DBSCAN uses the described density-based method to create clusters such that all samples within the cluster are "density-connected and "density-reachable" from one other. It also accounts for noise in the dataset and is therefore robust to outliers. This is especially useful for the HIRISE dataset, since preprocessing images creates tiles that may not contain distinguishable features. These points in the DBSCAN algorithm will then be classified as the noise in the dataset. DBSCAN has two tuning parameters – epsilon, which is a distance measure and second, the minimum number of points that one would expect to have in a cluster. A grid search method is be used to estimate the most appropriate values for the HIRISE dataset.

2. **HDBSCAN(Hierarchical DBSCAN)**: DBSCAN with hierarchical tendencies, this method allows for the consideration of clusters with varying densities. This eliminates the need to specify the epsilon parameter, with min samples as the only other tuning parameter. We see from the results(Table 2), for the HIRISE dataset, the clustering is more effective with HDBSCAN than DBSCAN, thus implying that the clusters in HIRISE datasets are of varying densities.

3. **Affinity Propagation(AP)**: This method relies on a message passing mechanism where the each of the samples communicate to the others their 'attractiveness' to the other, such that each sample after multiple epochs chooses an exemplar whose cluster it wishes to belong to. Consequently, all samples with the same exemplar are chosen to be part of the same cluster. For the purpose of discovery, the AP produced particularly interesting results. The tuning resulted in a wide range of clusters from 19- 44 and when observed each of the results were visually sensible. Specialists in the domain could, therefore, analyse potential new features not recognised by the literature into their own clusters, thus potentially discovering new features using AP.

4. **Ordering Points to Identify the Clustering Structure (OPTICS)** Similar to HDBSCAN, OPTICS is a density-based clustering algorithm that further refines the ability of the algorithm to generate accurate clusters with varying densities, represented through dendrograms. For the HIRISE dataset it is observed that OPTICS is in certain cases generates more distinct clusters compared to DBSCAN and HDBSCAN while handling noisy data and is therefore a useful option to have in order to tackle these specific clustering problems.

5. **Mean Shift**: The mean shift algorithm, similar to the K-Means clusters images based on their

proximity to a predetermined centroid. However, it does not require the user to specify the number of clusters. Mean shift works on the concept of kernel density estimation such that it shifts the sample points progressively towards the centroids, re-calibrating the centroids based on the density of the forming cluster. HIRISE dataset performed poorly with the mean shift, however, there maybe applications or specific selections of images where mean shift is the preferred option. This area of application remains relatively less explored in this research.

Ensemble Model

Stacking, a type of ensemble modelling defined used base models(first predictions used) and meta models(contributing models). This method substantially improves model performances while retaining the performance benefits of individual models. For the HIRISE dataset it was observed that stacking DBSCAN with BIRCH, KMeans and affinity propagation, with KMeans as the base model allows the user to both extract noise as well as achieve better clustering performance between 25-30% in certain combinations.

Quantitative Analysis and Results

Thirty-six models combinations, based on the selections of techniques at each stage were generated and tested on the curated test dataset.
Results from the analysis at each stage can be visualised as follows :

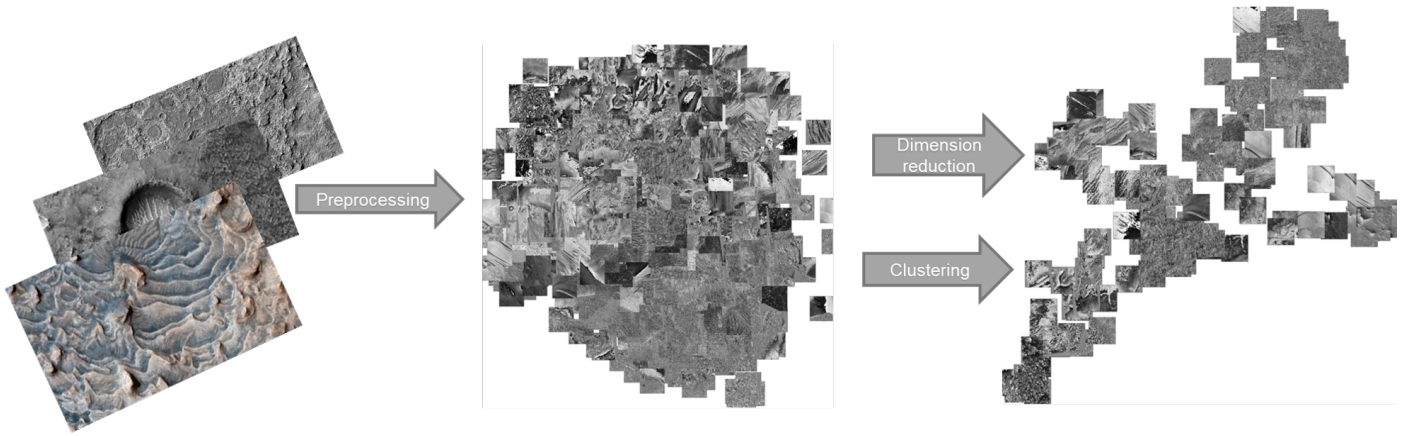


Figure 6: Results at Each Stage of the Process

Preprocessing Results

The three main results from the preprocessing were the resizing of the images, the tiling of the images and the removal of black spaces from the tiled image. Each of these operations can both be performed and tuned using according to the user's research needs in the HIRISE package. For the test case, the resizing function was tuned such that each pixel of the image represents 250cm of ground area. This parameter was tuned in conjunction to the tile size which was chosen to be 256x256 pixels. Together these parameters lead to an image tile that represented 640x640 m of Mars's surface. This size was tested to be the most appropriate size to adequately represent most of the selected features. Thus, the chosen size reduction factor was 1:10. Further, in order to account for projected images and to avoid aberrations from affecting clustering, tiles that contain greater than 10%(tune-able) black space were removed. The image size was reduced from 42.9GB to 19.2MB, which is approximately a reduction factor of 1300:1.

Encoding Results

Each of the two transfer learning models and the auto-encoder generated encoded results with different number of feature variables depending on the features selected by the model. Inception selected a total of 6912 feature variables, Xception encoded 131071 features variables while the convolutional auto-encoder, tuned to have 2000 latent dimensions, stored 2000 feature variables.(Appendix D)
The convolutional auto-encoder was tuned based not only the latent dimensions, but also the learning rate and the batch size.(Appendix D for grid search results)
The transfer learning models were used such that all layers except the last layer were non-trainable. The output layer was then saved and used as the encoded latent space. The file size was further reduced to 6.75MB for inception and a mere 33.4KB for the convolutional auto-encoder. This is a further 64.84 % - 100 % reduction in image size.
The auto-encoder, notoriously hard to train, after multiple architectural changes and hyper-parameter tuning, produced results that were not able to correctly identify all the features for further clustering. Therefore, although the package allows for the implementation of this encoding method, further results will not discuss results using the convolutional auto-encoder.

Dimension Reduction Results

The dimension reduction techniques further reduced the complexity of the data by either capturing the axes with most variance in data or by projecting high dimensional data to a lower dimension. This results in a reorganised latent space(Appendix E) of embedded that can be used for further analysis.

Clustering Results

Individual characteristics of each of the 36 algorithm are described below. The main observations are summarised at the end of the section.
The Group 1 models are analysed depending on best and worst predicted classes, and group 2 based on the number of clusters discovered as shown in the table below:

Table 1: Clustering Results : Group 1 Models for Exploratory Analysis

Sr no	Base Model	Dimension Reduction	Clustering Algorithm	Best predicted	Poorly Predicted
1	Inception	PCA	K-Means	cratered-cones, dunes, dust-devils, lava-flows, ridges	layers, pedestal craters
2			Agglomerative Clustering	cratered-cones, dunes, dust-devils, lava-flows, slope-streaks	concentric crater, pedestal crate ridges
3			BIRCH	cratered-cones, dunes, lava-flows, slope-streaks	pedestal crater, ridges, scallope depression
4		T-SNE	K-Means	brain-terrain, concentric-crater, cratered-cones, dune, dust-devil, gullies, lava-flow, ridges	layers, pedestal craters
5			Agglomerative Clustering	cratered -cones, dust-devil, lava-flow, ridges	-
6			BIRCH	cratered -cones, dust-devil, lava-flow, ridges	
7		UMAP	K-Means	dust-devil	concentric-crater, cratered-cone dune, glaciers, gullies, lava-flow, layers, pedestal-crater, ridges, scalloped-depression, slope-stre
8			Agglomerative Clustering	cratered-cones	brain-terrain, dune, dust-devil, glaciers, lava-flow, layers, ridge scalloped-depression, slope-stre
9			BIRCH	gullies	chaos-terrain, concentric-crater, cratered-cones, dust-devil, lava-flow, ridges, scalloped-depressi
1	Xception	PCA	K-Means	-	brain-terrain, chaos-terrain, concentric-crater, cratered-cone dune, dust-devil, glaciers, gullies, lava-flow, layers, pedest crater, ridges, scalloped-depress slope-streaks
2			Agglomerative Clustering	cratered-cones	brain-terrain, concentric-crater, dune, dust-devil, gullies, lava-flo layers, pedestal-crater, ridges, scalloped-depression, slope-stre
3			BIRCH	-	brain-terrain, cratered-cones, du dust-devil, glaciers, gullies, pede crater, ridges, scalloped-depress slope-streaks
4		T-SNE	K-Means	concentric-crater, cratered-cones, glaciers, lava-flow, ridges, scalloped-depression	chaos-terrain
5			Agglomerative Clustering	concentric-crater, cratered-cones, glaciers, lava-flow, ridges	pedestal crater
6			BIRCH	lava-flow	brain-terrain, concentric-crater, cratered-cones, dune, dust-devi gullies, layers, pedestal-crater, ridges, scalloped-depression, slo streaks
7		UMAP	K-Means	-	cratered-cones, dune, dust-devi gullies, lava-flow, layers, ridges
8			Agglomerative Clustering	-	brain-terrain, chaos-terrain, concentric-crater, cratered-cone dune, glaciers, gullies, lava-flow,

Table 2: Clustering Results : Group 2 Models for Discovery

Sr no	Base Model	Dimension Reduction	Clustering Algorithm	Predicted Clusters	Hyperparameters Tuned
1	Inception	PCA	Affinity Propagation	44	Damping: tuned from ranges of 0.5-0.9. Tuned to 0.
2			HDBSCAN	8	Minimum samples = 6. From a range of 5 - 15
3			DBSCAN	26	Eps = 0.3, minimum samples = 5 . From a range of 0 – 0.95 and 5 to 15 respectively
5			OPTICS	9	Eps = 0.5, minimum samples = 5 . From a range of 0 0.9 and 5 to 15 respectively
5		T-SNE	Affinity Propagation	24	Damping: tuned from ranges of 0.5-0.9. Tuned to 0.
6			HDBSCAN	7	Minimum samples = 11. From a range of 5 - 15
7			DBSCAN	3	Eps = 0.9, minimum samples = 5 . From a range of 0 – 0.95 and 5 to 15 respectively
8			OPTICS	16	Eps = 0.9, minimum samples = 8 . From a range of 0 0.9 and 5 to 15 respectively
9		UMAP	Affinity Propagation	19	Damping: tuned from ranges of 0.5-0.9. Tuned to 0.
10			HDBSCAN	10	Minimum samples = 6. From a range of 5 - 15
11			DBSCAN	15	Eps = 0.25, minimum samples = 11 . From a range o 0.01 – 0.95 and 5 to 15 respectively
12			OPTICS	22	Eps = 0.9, minimum samples = 8 . From a range of 0 0.9 and 5 to 15 respectively
1	Xception	PCA	Affinity Propagation	45	Damping: tuned from ranges of 0.5-0.9. Tuned to 0.
2			HDBSCAN	5	Minimum samples = 6. From a range of 5 - 15
3			DBSCAN	-	Eps = 0.3, minimum samples = 5 . From a range of 0 – 0.95 and 5 to 15 respectively
4			OPTICS	12	Eps = 0.5, minimum samples = 5 . From a range of 0 0.9 and 5 to 15 respectively
5		T-SNE	Affinity Propagation	24	Damping: tuned from ranges of 0.5-0.9. Tuned to 0.
6			HDBSCAN	11	Minimum samples = 10. From a range of 5 - 15
7			DBSCAN	15	Eps = 0.7, minimum samples = 3 . From a range of 0 – 0.95 and 5 to 15 respectively
8			OPTICS	12	Eps = 0.6, minimum samples = 9 . From a range of 0 0.9 and 5 to 15 respectively
9		UMAP	Affinity Propagation	19	Damping: tuned from ranges of 0.5-0.9. Tuned to 0.
10			HDBSCAN	14	Minimum samples = 6. From a range of 5 - 15
11			DBSCAN	15	Eps = 0.37, minimum samples = 15 . From a range o 0.01 – 0.95 and 5 to 15 respectively
12			OPTICS	23	Eps = 0.5, minimum samples = 5 . From a range of 0 0.9 and 5 to 15 respectively

Metrics for Quantitative analysis

Depending on whether the dataset has corresponding true labels, we can use different metrics to assess the quality of the clustering results. For test datasets, datasets with known true labels, the v-score, homogeneity and completeness can be used(classification metrics). If the dataset has no true labels to be assessed against, rand, adjusted rand, mutual, adjusted, normalized and adjusted mutual information score(clustering metrics).

A. Classification Metrics: Data Exploration

1. **Precision** In a multi-class classification problem, precision is the number of samples correctly identified by the classifier divided by the sum of correctly and incorrectly identified samples. Fig (b) shows how well each of the group 1 models perform at the designated classes, stacked against each other. This gives a clear idea of the algorithms that do poorly against the other(K-Means Xception PCA, Agglomerative Clustering Xception PCA, Agglomerative Clustering Xception UMAP, BIRCH Xception PCA). The further analyses will further help refine the choice between the other model combinations.

2. **Confusion Matrix** A visual representation table of the accuracy of the classification results such that the predicted and actual labels from the two dimensions of the table with each class being a ticker on the axes. These labels divide the table into categories that allows the user to understand the nature and characteristics of the results produced.

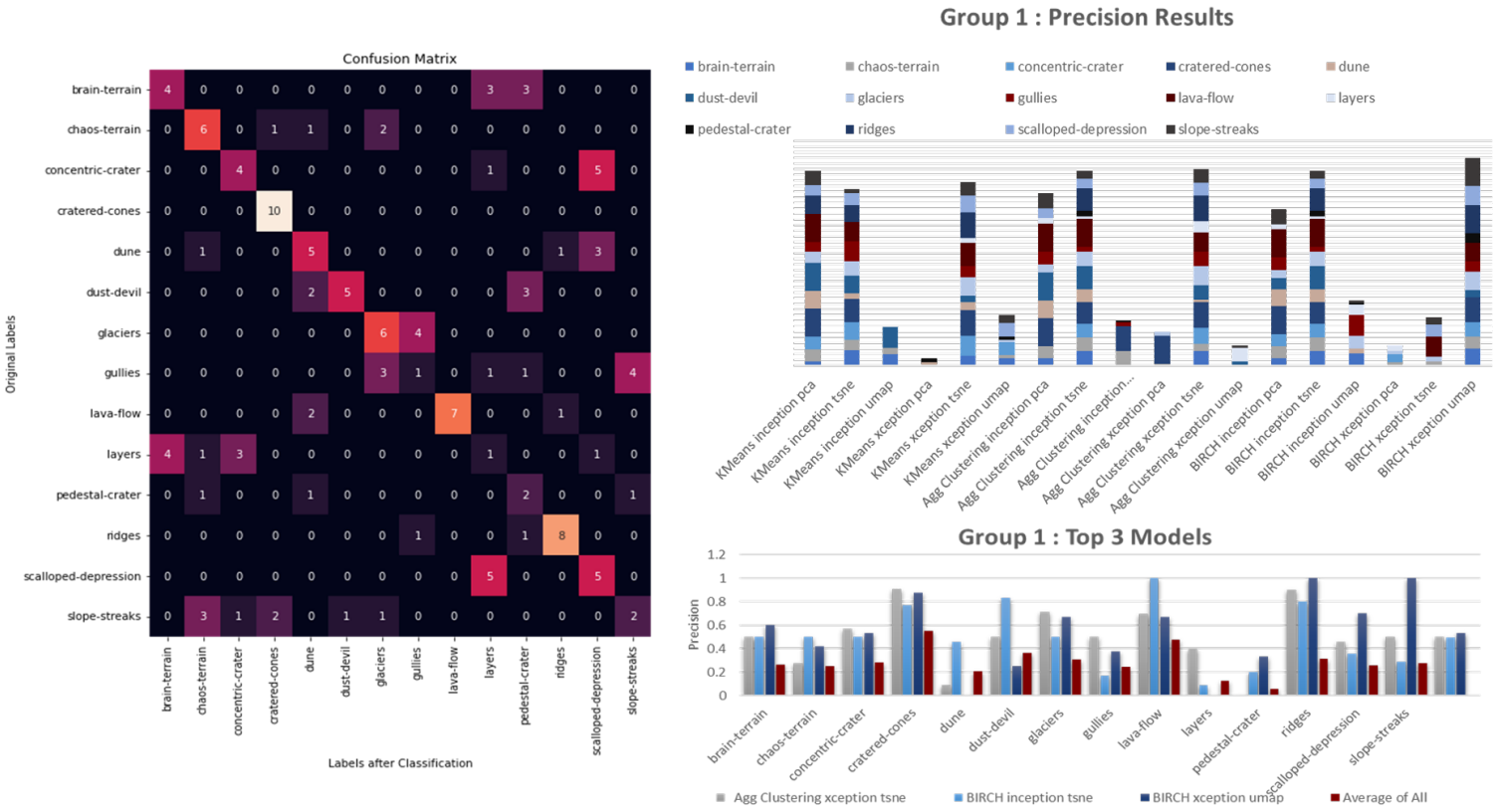


Figure 7: Clockwise from the left - (a) Confusion Matrix, (b) Precision Results, (c) Top 3 Group 1 Compared on classification metrics

Figure (a) shows the confusion matrix for the Agglomerative Clustering model, pre-processed with a t-SNE using inception encoded samples.
Figure (c) describes the top 3 models that produce the best results based on clustering metrics.

B. Clustering Metrics: Data Exploration and Feature Discovery

Clustering metrics can be used for both data exploration and discovery. The metrics used to explain the results are as follows:

1. **RAND, Adjusted RAND** The Rand Index measures the similarity between two cluster samples and the adjusted rand index adjusts the rand measure with a probability feature, such that the measure is adjusted for chance. The measure ranges from 0 -1 where 1 is perfect similarity.

$$ARI = (RI - Expected\_RI)/(max(RI) - Expected\_RI)$$

2. **Adjusted and Normalized Mutual Information Score** These measures are useful when the ground truth is not known. This is especially useful when a labelled test dataset is unavailable. The adjusted MI score is one that is adjusted for chance and the normalized score is scaled such that 0 represents no mutual information while 1 represents perfect correlation between both the clusters.
Mathematically,

MI :

$$MI(U,V) = \sum_{i=1}^{|U|}\sum_{j=1}^{|V|}\frac{|U_i \cap V_j|}{N}\log\frac{N|U_i \cap V_j|}{|U_i||V_j|}$$

Where $|Ui|$, $|Vj|$ are the samples in cluster U_i and V_j.

Adjusted MI:

$$AMI(U,V) = [MI(U,V) - E(MI(U,V))]/[avg(H(U),H(V)) - E(MI(U,V))]$$

3. **Homogeneity and Completeness Score** Homogeneity is the property that results in a cluster having encoded samples of a single class , while completeness is how many of the same class are put together in a cluster.
Mathematically,

$$H(C/K) = -\sum\frac{n_{ck}}{N}log(\frac{n_{ck}}{n_k})$$

Where,
n_{ck} = No. of samples of specific class in cluster
N = Total Samples n_k = Number of samples in a cluster
Homogeneity:

$$h = 1 - \frac{H(Y_{true}|Y_{predicted}}{H_{true}}$$

Completeness :

$$c = 1 - \frac{H(Y_{predicted}|Y_{true}}{H(Y_{predicted}}$$

4. **V Score** V measure is a derived index that is defined to be the harmonic mean between homogeneity and completeness. Perfect clustering will result in a v-score of 1, which means the cluster is both completely homogeneous and complete.

$$v = \frac{(1+beta)*homogeneity*completeness}{(beta*homogeneity + completeness)}$$

Figure 8 (a) and (b) below show the top 3 models for exploration and discovery respectively based on clustering metrics and are explained in the summarization.

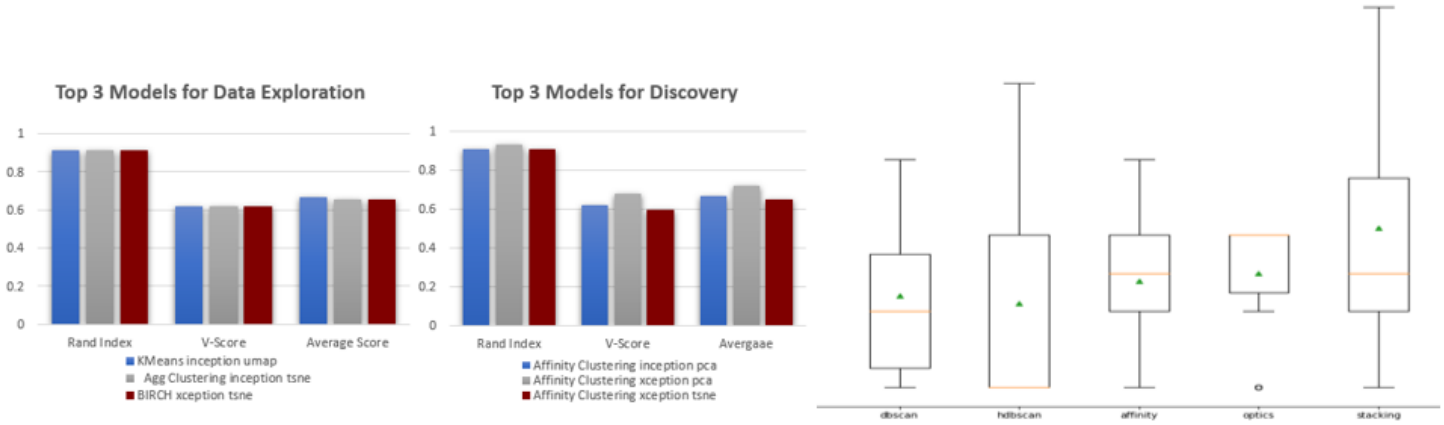


Figure 8: From left (a)Clustering metrics on Group 1 Models, (b) Clustering Metrics on Group 2 Models, (c) Ensemble Stacking of DBSCAN, HDBSCAN, Affinity Propagation and Optics

The main observations(Appendix F for details) for the test dataset are as follows:

- The Inception encoding method works better than Xception for most clustering algorithms and dimension reduction combinations. Xception combined with PCA or UMAP dimension reduction methods perform particularly poorly(Table 1, 2).

- For group 1 models(figure 7(c), figure 8(a), the best clustering models that predict labelled data in an unsupervised manner are agglomerative clustering(with t-SNE on Xception and Inception samples) and BIRCH(with UMAP or t-SNE on Xception samples). The precision values are low, around 50-53%.This is expected since the encoded images are categorized arbitrarily based on the literature. The algorithm is sensitive to minute differences, which is confirmed visually by K-Means categorising the layers on the edges of craters in the layers class rather than craters.

- Based on the defined classes, certain classes(concentric craters, ridges, dune, lava flows, slope-streaks) are well predicted, over 70% precision scores by almost all the algorithms(fig 7 - (c)). The most difficult to predict classes were layers and pedestal craters. Layers are notoriously hard to predict as a distinct class because they tend to be present on other land forms such as layers on cratered cones or layers in chaos terrains. This, however is an issue with the premise that one-to-one mapping of dataset images with classes is the correct approach. The pedestal craters in the dataset were of varying sizes and hence the scaling method chosen was limited in identifying the same feature that was displayed on different scales. This is a limitation of the data curation approach. The package however allows the user to scale features different and this feature should be used for future research.

- For group 2 models, the best clustering models are Affinity Clustering(with PCA on both inception and Xception samples) as well as K-Means(with UMAP on Inception samples). These have exceptionally good RAND indexes of 0.91 - 0.93, with v-scores of 62- 70%. Clustering metrics, as expected, prove to be better indicators for unsupervised clustering compared to precision values.

- The Ensemble model(stacking) used, fig 8 (c) shows how stacking multiple models(DBSCAN, HDBSCAN, AP and Optics with DBSCAN as the base model) increases the v-score range by 0.3. This is especially useful for larger datasets (tested on 700 images) with greater noise and greater complexities.

Conclusions and Future Scope

The project successfully covers the objectives set out on the plan.
The first result achieved is the version 1.0.0 of the publicly available software tool allows researchers to filter, download, pre-process and analyse HIRISE images as required. The package further allows researchers to apply thirty-six different models combinations, with over 100 supporting functions that can be used to better understand the downloaded HIRISE dataset.These cover objectives one and two outlined previously.
The second aspect of the project uses the 36 model combinations, split into Group 1 (K-Means, Agglomerative Clustering, BIRCH) and Group 2(DBSCAN, HDBSCAN, OPTICS, Affinity Propagation, Means Shift), and finally the ensemble model to analyse the curated image test dataset for the purpose of data exploration and feature discovery. This covers the third objective.

There are multiple avenues to be explored further to both expand the tool and in this research, these include:

- On the software end, the tool can be expanded to support further preprocessing and clustering functions especially methods such as semi-supervised generative-adversarial networks and graph-based clustering methods such as spectral clustering, SNN-cliq and Seurat.
- On the research end, the analysis on the HIRISE dataset should be expanded to test rigorously the for impact of scaling, impact of resizing and impact of dataset size on the clustering performance. The tool in its present version(1.0.0) supports all these above mentioned areas of research, with computational(availability of GPUs) and data storage cost(for larger datasets) being the only factors to be kept in mind.

Therefore the project allows for many avenues of further research in the application of deep learning algorithms to understand and discover complexities in the mapping of Martian terrains using the continuously expanding HIRISE dataset.

References

1. Pekka Abrahamsson, Outi Salo, Jussi Ronkainen, and Juhani Warsta. Agile software development methods: Review and analysis. *CoRR*, abs/1709.08439, 2017. URL `http://arxiv.org/abs/1709.08439`.

2. Mebarka Allaoui, Mohammed Lamine Kherfi, and Abdelhak Cheriet. Considerably improving clustering algorithms using umap dimensionality reduction technique: A comparative study. *Image and Signal Processing*, 12119:317 – 325, 2020.

3. Lourenço Bandeira, Jorge S. Marques, José Saraiva, and Pedro Pina. Automated detection of martian dune fields. *IEEE Geoscience and Remote Sensing Letters*, 8(4):626–630, 2011. doi: 10.1109/LGRS.2010.2098390.

4. Kent L. Beck, Michael A. Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andy Hunt, Ron Jeffries, Jon Kern, Brian Marick, Robert C. Martin, Stephen J. Mellor, Ken Schwaber, Jeff Sutherland, and Dave A. Thomas. Manifesto for agile software development. In *Manifesto for Agile Software Development*, 2013.

5. Dong Chen, Xudong Cao, Fang Wen, and Jian Sun. Blessing of dimensionality: High-dimensional feature and its efficient compression for face verification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2013.

6. Zhengxue Cheng, Heming Sun, Masaru Takeuchi, and Jiro Katto. Deep convolutional autoencoder-based lossy image compression. In *2018 Picture Coding Symposium (PCS)*, pages 253–257, 2018. doi: 10.1109/PCS.2018.8456308.

7. François Chollet. Xception: Deep learning with depthwise separable convolutions, 2016. URL https://arxiv.org/abs/1610.02357.

8. Martin Fowler, Jim Highsmith, et al. The agile manifesto. *Software development*, 9(8):28–35, 2001.

9. Valeriy Gavrishchaka, Zhenyi Yang, Rebecca Miao, and Olga Senyukova. Advantages of hybrid deep learning frameworks in applications with limited data. *International Journal of Machine Learning and Computing*, 8(6):549–558, 2018.

10. Shancheng Jiang, Fan Wu, K.L. Yung, Yingqiao Yang, W.H. Ip, Ming Gao, and James Abbott Foster. A robust end-to-end deep learning framework for detecting martian landforms with arbitrary orientations. *Knowledge-Based Systems*, 234:107562, 2021. ISSN 0950-7051. doi: https://doi.org/10.1016/j.knosys.2021.107562. URL https://www.sciencedirect.com/science/article/pii/S0950705121008248.

11. Dmitry Kobak and Philipp Berens. The art of using t-sne for single-cell transcriptomics. *Nature communications*, 10(1):1–14, 2019.

12. Christopher Lee. Automated crater detection on mars using deep learning. *Planetary and Space Science*, 170:16–28, jun 2019. doi: 10.1016/j.pss.2019.03.008. URL https://doi.org/10.1016%2Fj.pss.2019.03.008.

13. Yang Long, Gui-Song Xia, Shengyang Li, Wen Yang, Michael Ying Yang, Xiao Xiang Zhu, Liangpei Zhang, and Deren Li. Dirs: On creating benchmark datasets for remote sensing image interpretation. *arXiv preprint arXiv:2006.12485*, 2020.

14. Yang Long, Gui-Song Xia, Shengyang Li, Wen Yang, Michael Ying Yang, Xiao Xiang Zhu, Liangpei Zhang, and Deren Li. On creating benchmark dataset for aerial image interpretation: Reviews, guidances, and million-aid. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 14:4205–4230, 2021. doi: 10.1109/JSTARS.2021.3070368.

15. E.C. Malthouse. Limitations of nonlinear pca as performed with generic neural networks. *IEEE Transactions on Neural Networks*, 9(1):165–173, 1998. doi: 10.1109/72.655038.

16. Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018.

17. Moses Njue and Billy Franklin. Dimensionality reduction on mnist dataset using pca, t-sne and umap. *arXiv*, 2020.

18. Krishan Pal and Mayank Sharma. Performance evaluation of non-linear techniques umap and t-sne for data in higher dimensional topological space. In *2020 Fourth International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)*, volume 1, pages 1106–1110, 2020. doi: 10.1109/I-SMAC49090.2020.9243502.

19. Leon F. Palafox, Alexander M. Alvarez, and Christopher W. Hamilton. Automated detection of impact craters and volcanic rootless cones in mars satellite imagery using convolutional neural networks and support vector machines. In *AUTOMATED DETECTION OF IMPACT CRATERS AND VOLCANIC ROOTLESS CONES IN MARS SATELLITE IMAGERY USING CONVOLUTIONAL NEURAL NETWORKS AND SUPPORT VECTOR MACHINES*, 2015.

20. Leon F Palafox, Christopher W Hamilton, Stephen P Scheidt, and Alexander M Alvarez. Automated detection of geological landforms on mars using convolutional neural networks. *Computers & geosciences*, 101:48–56, 2017.

21. Pavel Potapov. On the loss of information in pca of spectrum-images. *Ultramicroscopy*, 182: 191–194, 2017.

22. Brandon Rothrock, Ryan Kennedy, Christopher T. Cunningham, Jeremie Papon, Matthew Heverly, and Masahiro Ono. Spoc: Deep learning-based terrain classification for mars rover missions. In *SPOC: Deep Learning-based Terrain Classification for Mars Rover Missions*, 2016.

23. Manali Shaha and Meenakshi Pawar. Transfer learning for image classification. In *2018 Second International Conference on Electronics, Communication and Aerospace Technology (ICECA)*, pages 656–660, 2018. doi: 10.1109/ICECA.2018.8474802.

24. Michael Steinbach, Levent Ertöz, and Vipin Kumar. The challenges of clustering high dimensional data. In *New Directions in Statistical Physics*, pages 273–309. Springer Berlin Heidelberg, 2004. doi: 10.1007/978-3-662-08968-2_16. URL https://doi.org/10.1007%2F978-3-662-08968-2_16.

25. Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. *CoRR*, abs/1409.4842, 2014. URL http://arxiv.org/abs/1409.4842.

26. Hasan Tercan, Alexandro Guajardo, and Tobias Meisen. Industrial transfer learning: Boosting machine learning in production. In *2019 IEEE 17th International Conference on Industrial Informatics (INDIN)*, volume 1, pages 274–279, 2019. doi: 10.1109/INDIN41052.2019.8972099.

27. Ayush Tewari, Michael Zollhofer, Hyeongwoo Kim, Pablo Garrido, Florian Bernard, Patrick Perez, and Christian Theobalt. Mofa: Model-based deep convolutional face autoencoder for unsupervised monocular reconstruction. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV) Workshops*, Oct 2017.

28. Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

29. Kiri L. Wagstaff, You Lu, Alice Stanboli, Kevin Grimes, Thamme Gowda, and Jordan Padams. Deep mars: Cnn classification of mars imagery for the pds imaging atlas. In *AAAI*, 2018.

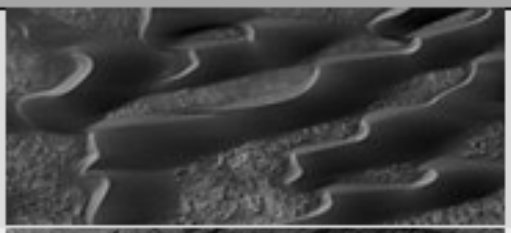
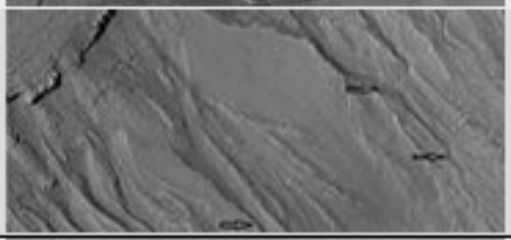
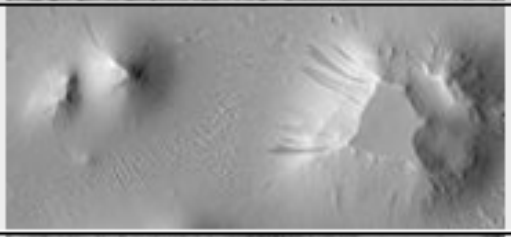
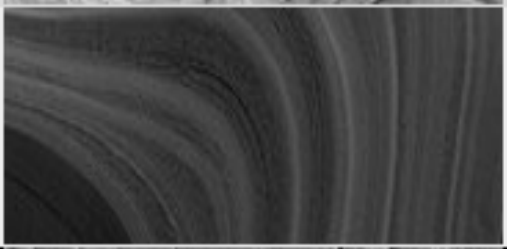
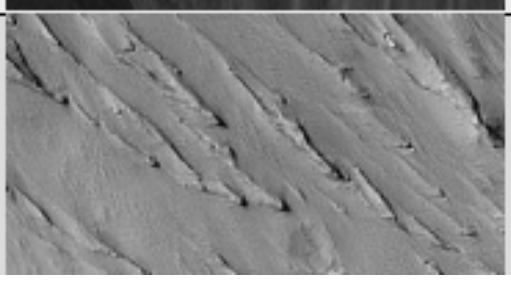
30. Yasi Wang, Hongxun Yao, and Sicheng Zhao. Auto-encoder based dimensionality reduction. *Neurocomputing*, 184:232–242, 2016. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2015.08.104. URL https://www.sciencedirect.com/science/article/pii/S0925231215017671. RoLoD: Robust Local Descriptors for Computer Vision 2014.

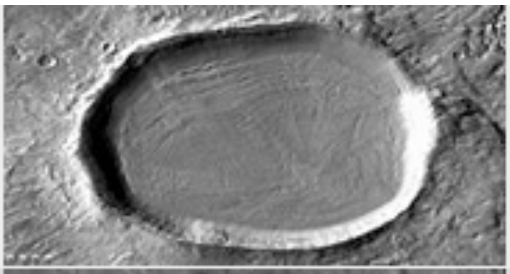
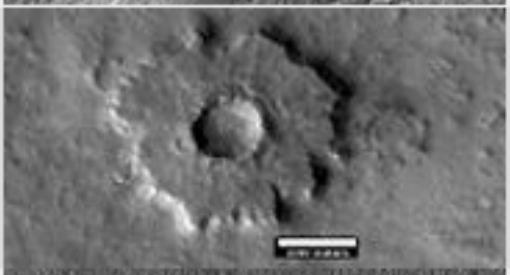
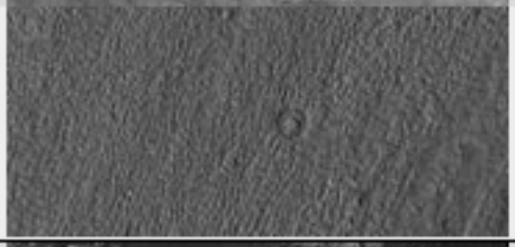
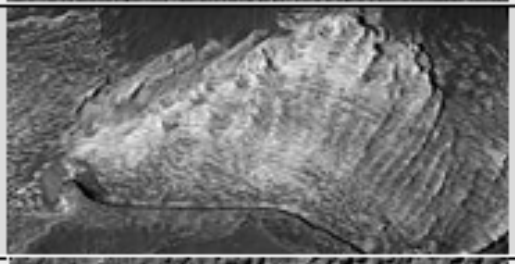
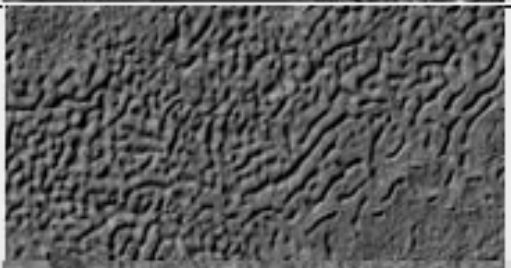
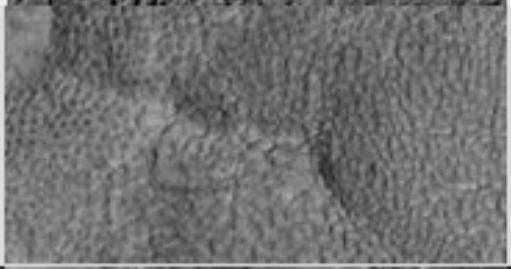
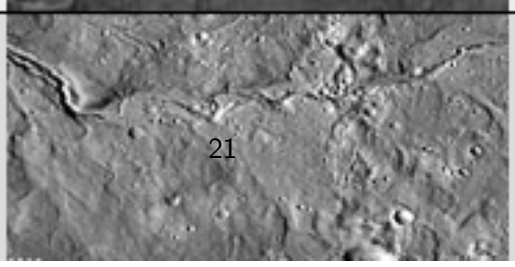
31. Dongkuan Xu and Yingjie Tian. A comprehensive survey of clustering algorithms. *Annals of Data Science*, 2(2):165–193, 2015.

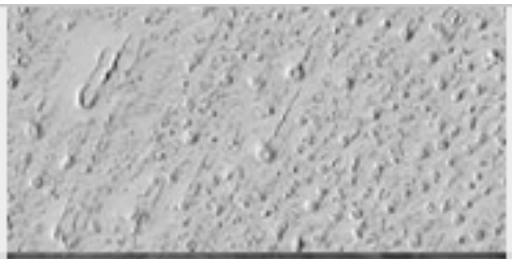
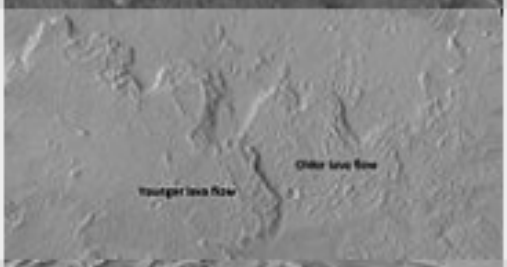
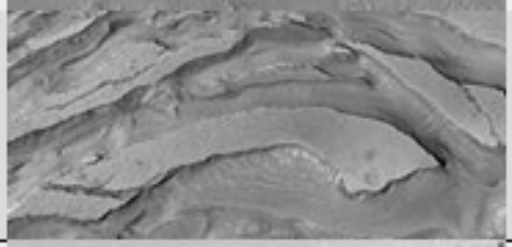
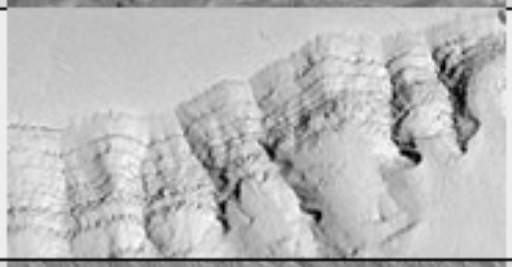
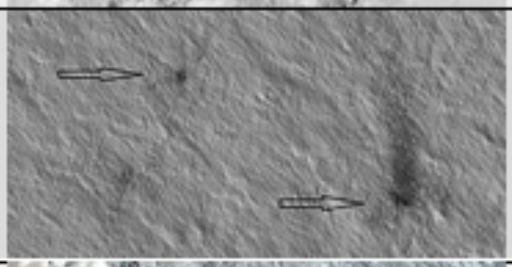
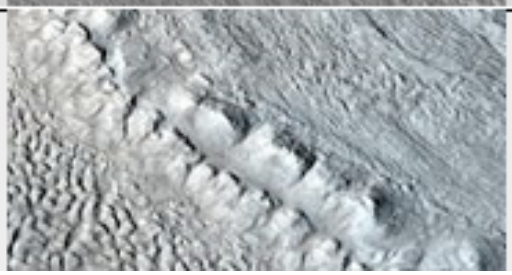
32. Fan Zhang, Bo Du, Liangpei Zhang, and Lefei Zhang. Hierarchical feature learning with dropout k-means for hyperspectral image classification. *Neurocomputing*, 187:75–82, 2016. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2015.07.132. URL https://www.sciencedirect.com/science/article/pii/S0925231215018743. Recent Developments on Deep Big Vision.

Appendices

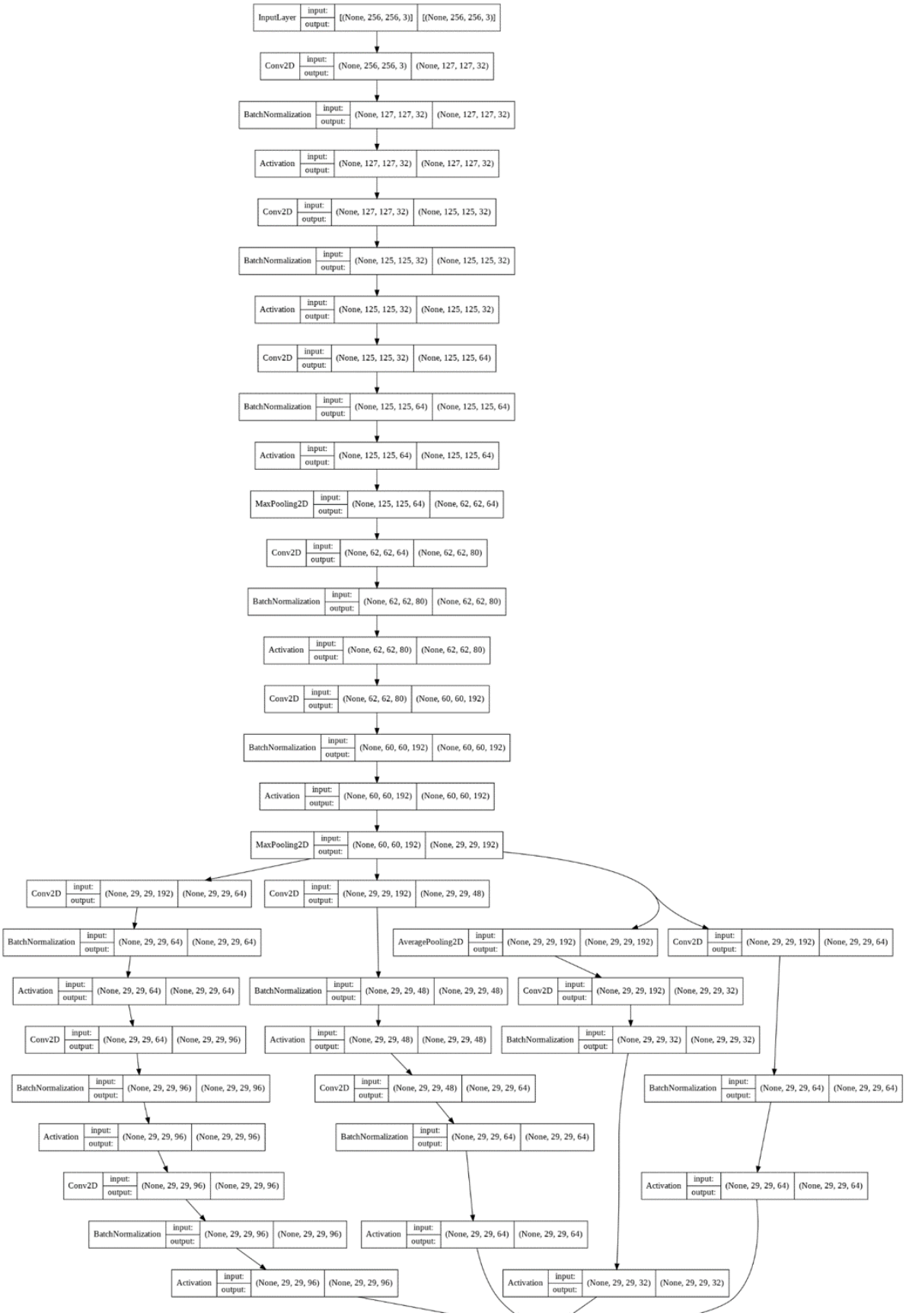
Appendix A

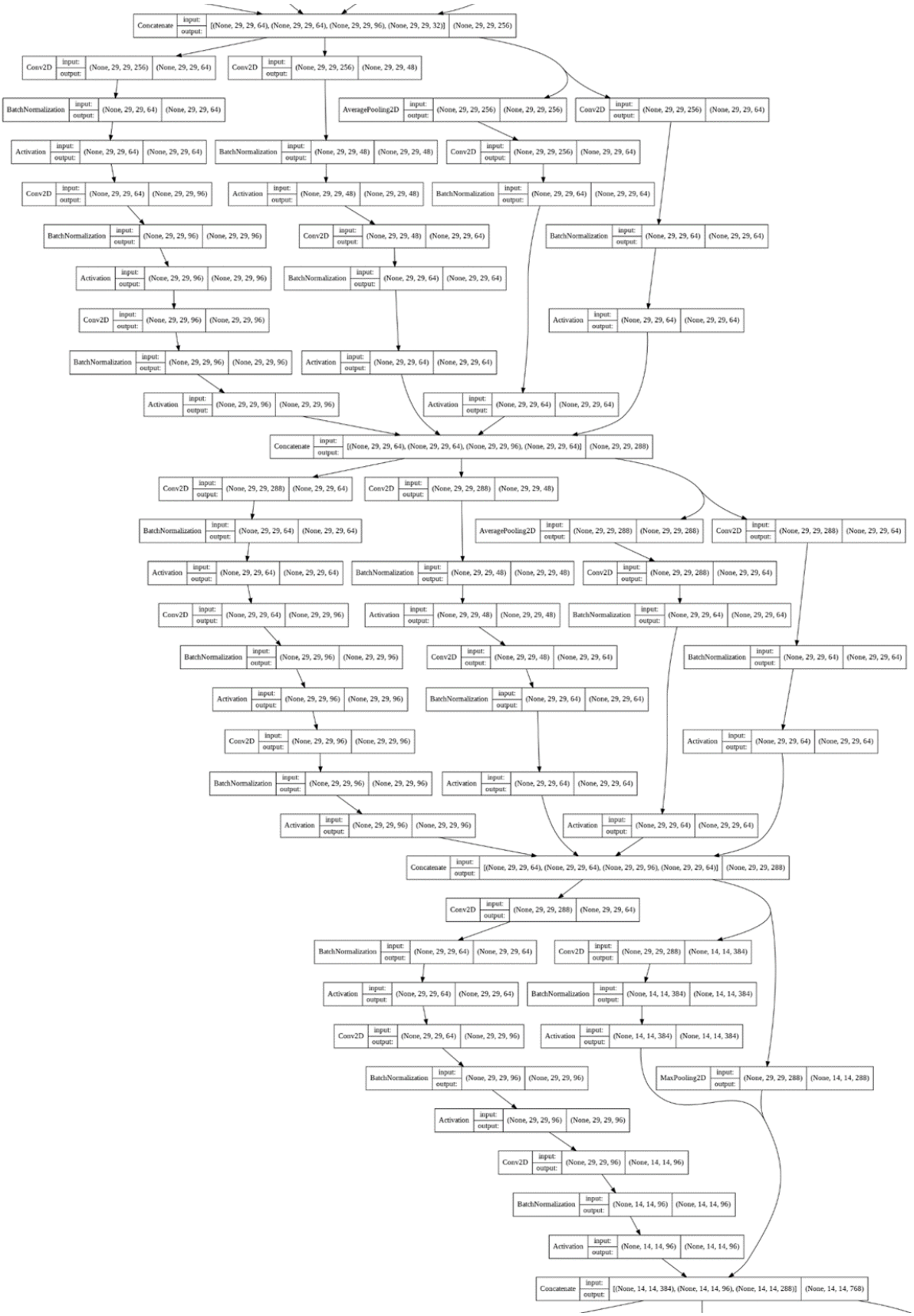
	Landform Name	HIRISE Image (from HiWish)	Location and Features
1	Sand Dunes		Many locations mostly in the northern polar caps. Including Eridania quadrangle, Mare Tyrrhenum quadrangle
	Dust Devil Tracks		Multiple areas – constantly changing including in the Noachis quadrangle
	Martian Gullies		Multiple Locations- along craters. Including in Phaethontis, Thaumasia, Diacria, Mare Acidalium, and Eridania quadrangle
2	Recurrent Slope Streaks		Multiple locations – along craters and gullies
3	Layers In the Sandy Terrain		Margaritifer Sinus quadrangle, Aeolis quadrangle, Memnonia quadrangle
	Layers In the Polar Ice-caps		The Northern Ice Cap
4	Linear Ridges (Yardangs)		Restricted regions mainly in the Amazonis quadrangle of the Medusae Fossae Formation

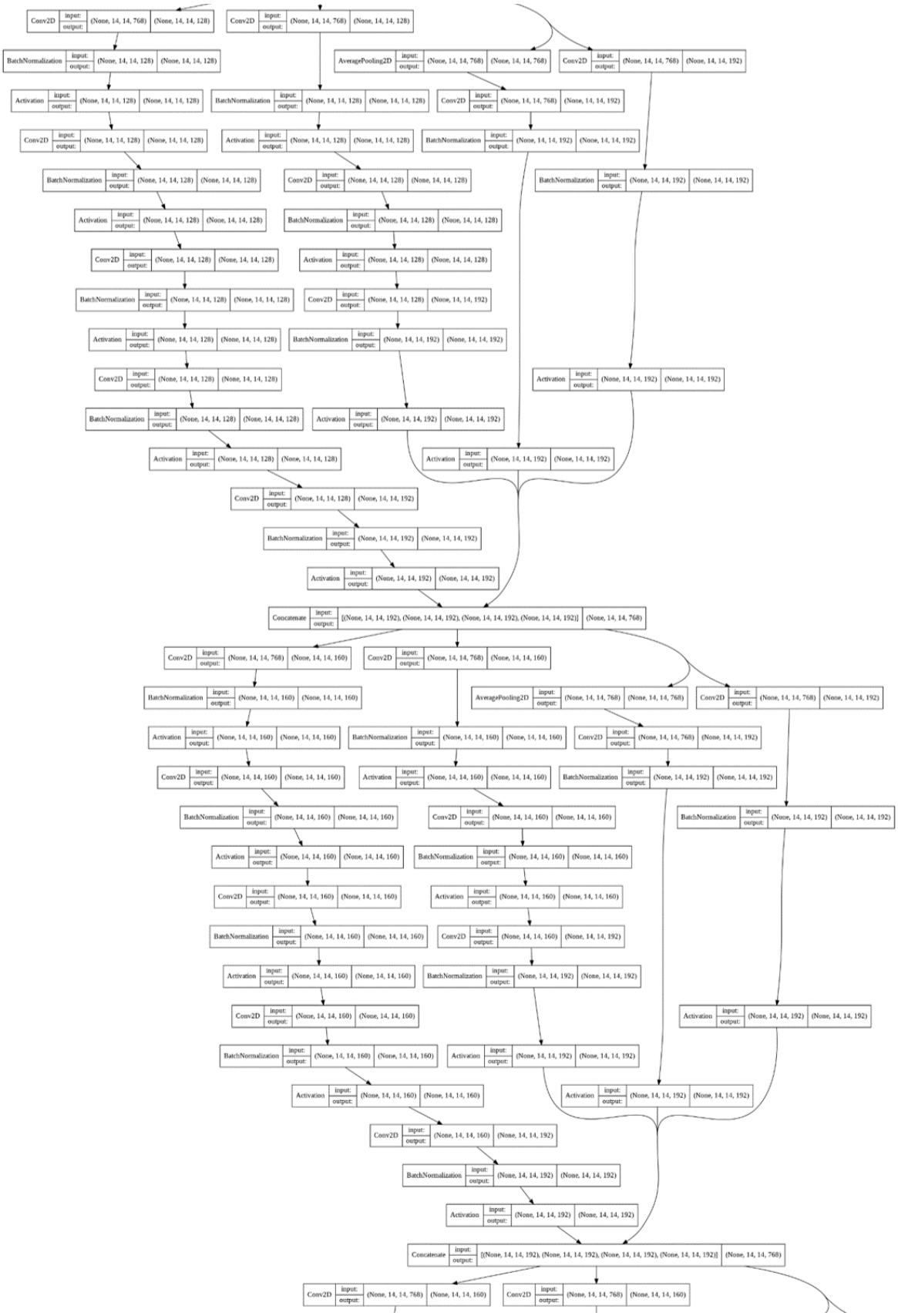
5	Craters : Concentric Craters		Phaethontis quadrangle, Casius quadrangle
	Craters: Pedestal Crater		Casius quadrangle, Amazonis quadrangle
	Craters : Ring Mould Craters		Mostly Ismenius Lacus quadrangle
6	Chaos Terrain		Multiple locations. (Left is Eos Chaos, others include Ister Chaos, Aureum Chaos, Arsinoes Chaos, Iani Chaos, Margaritifer Chaos etc)
7	Mantle : Brain Terrain		In the "Upper Plains Unit" in the Northern hemisphere the Ismenius Lacus quadrangle
	Mantle : Polygonal Patterns		Some regions including in the Casius quadrangle, Ismenius Lacus quadrangle
8	Scalloped Depressions		Multiple Locations mainly between 45° and 60° north and south
9	Dry Water Channels		Multiple loactions – mostly in craters and lake beds – including in Memnonia quadrangle, Cebrenia quadrangle

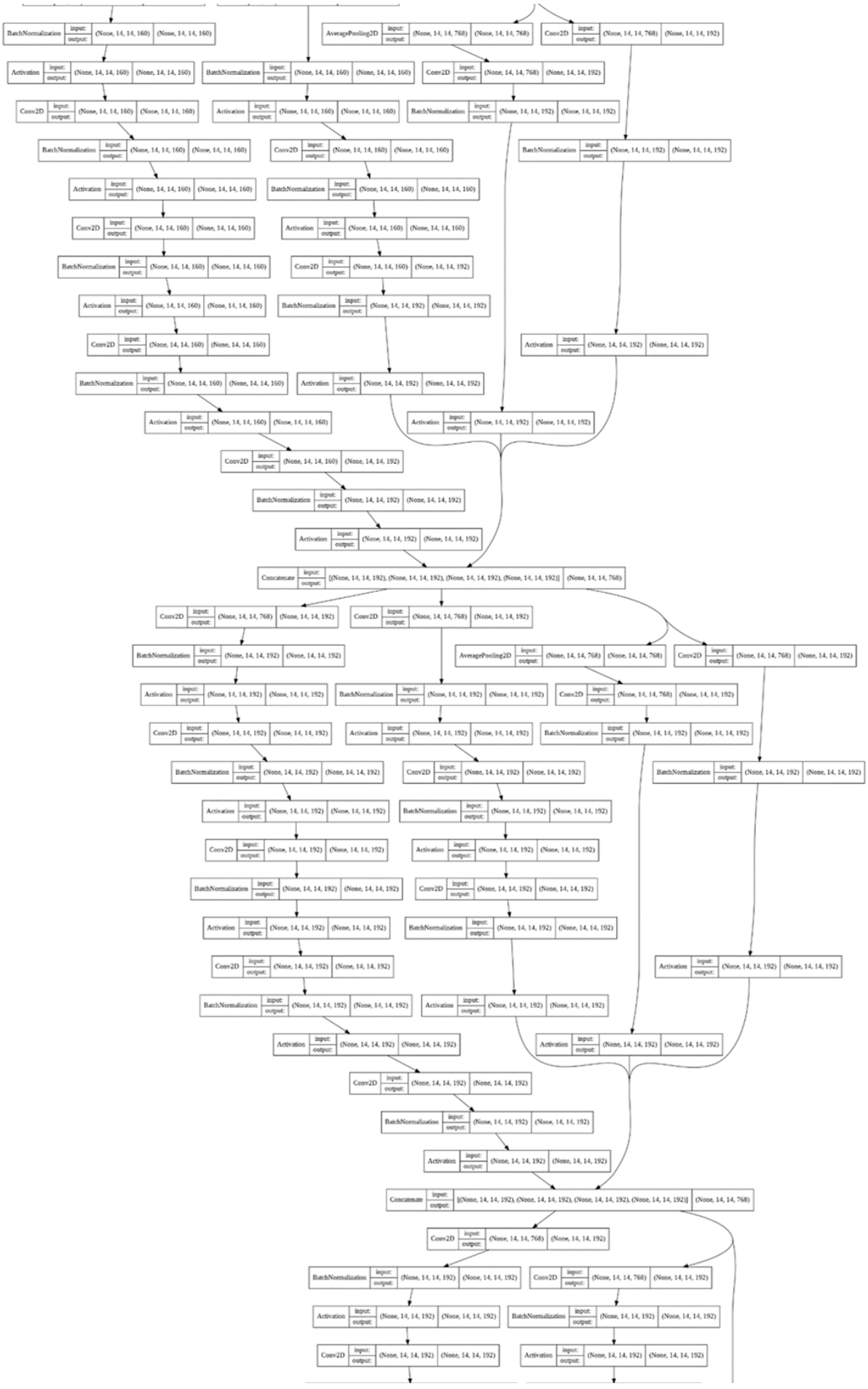
10	Volcanic Cones		Equatorial Regions
	Mud Volcanoes		Mostly the Northern Hemisphere
	Laval flows		Multiples Locations – including Tharsis quadrangle, Phoenicis Lacus quadrangle and Amazonis quadrangle
	Volcanos below Ice		Multiples Locations – mainly in the Ismenius Lacus quadrangle
11	Valleys : Noctis Labyrinthus		Phoenicis Lacus quadrangle
12	Spiders and Plumes: Dark channels formed after defrosting		Appear in winter across multiple locations
13	Glaciers		Restricted areas mainly in the Ismenius Lacus quadrangle

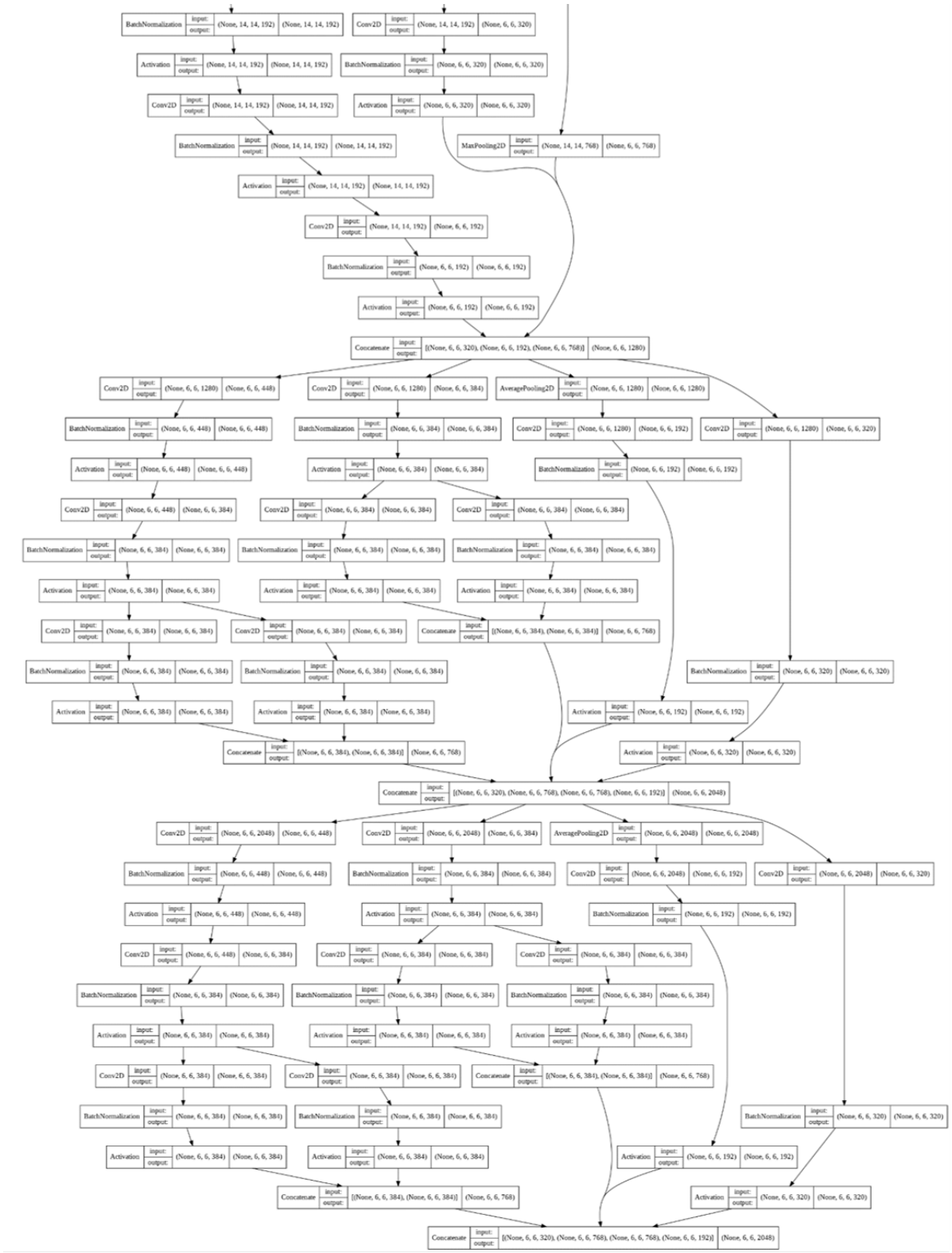
Appendix B


InputLayer | input: | [(None, 256, 256, 3)] | output: | [(None, 256, 256, 3)]
Conv2D | input: | (None, 256, 256, 3) | output: | (None, 127, 127, 32)
BatchNormalization | input: | (None, 127, 127, 32) | output: | (None, 127, 127, 32)
Activation | input: | (None, 127, 127, 32) | output: | (None, 127, 127, 32)
Conv2D | input: | (None, 127, 127, 32) | output: | (None, 125, 125, 32)
BatchNormalization | input: | (None, 125, 125, 32) | output: | (None, 125, 125, 32)
Activation | input: | (None, 125, 125, 32) | output: | (None, 125, 125, 32)
Conv2D | input: | (None, 125, 125, 32) | output: | (None, 125, 125, 64)
BatchNormalization | input: | (None, 125, 125, 64) | output: | (None, 125, 125, 64)
Activation | input: | (None, 125, 125, 64) | output: | (None, 125, 125, 64)
MaxPooling2D | input: | (None, 125, 125, 64) | output: | (None, 62, 62, 64)
Conv2D | input: | (None, 62, 62, 64) | output: | (None, 62, 62, 80)
BatchNormalization | input: | (None, 62, 62, 80) | output: | (None, 62, 62, 80)
Activation | input: | (None, 62, 62, 80) | output: | (None, 62, 62, 80)
Conv2D | input: | (None, 62, 62, 80) | output: | (None, 60, 60, 192)
BatchNormalization | input: | (None, 60, 60, 192) | output: | (None, 60, 60, 192)
Activation | input: | (None, 60, 60, 192) | output: | (None, 60, 60, 192)
MaxPooling2D | input: | (None, 60, 60, 192) | output: | (None, 29, 29, 192)
Conv2D | input: | (None, 29, 29, 192) | output: | (None, 29, 29, 64)
Conv2D | input: | (None, 29, 29, 192) | output: | (None, 29, 29, 48)
AveragePooling2D | input: | (None, 29, 29, 192) | output: | (None, 29, 29, 192)
Conv2D | input: | (None, 29, 29, 192) | output: | (None, 29, 29, 64)
BatchNormalization | input: | (None, 29, 29, 64) | output: | (None, 29, 29, 64)
Activation | input: | (None, 29, 29, 64) | output: | (None, 29, 29, 64)
BatchNormalization | input: | (None, 29, 29, 48) | output: | (None, 29, 29, 48)
Conv2D | input: | (None, 29, 29, 192) | output: | (None, 29, 29, 32)
Conv2D | input: | (None, 29, 29, 64) | output: | (None, 29, 29, 96)
Activation | input: | (None, 29, 29, 48) | output: | (None, 29, 29, 48)
BatchNormalization | input: | (None, 29, 29, 32) | output: | (None, 29, 29, 32)
BatchNormalization | input: | (None, 29, 29, 96) | output: | (None, 29, 29, 96)
Conv2D | input: | (None, 29, 29, 48) | output: | (None, 29, 29, 64)
BatchNormalization | input: | (None, 29, 29, 64) | output: | (None, 29, 29, 64)
Activation | input: | (None, 29, 29, 96) | output: | (None, 29, 29, 96)
BatchNormalization | input: | (None, 29, 29, 64) | output: | (None, 29, 29, 64)
Activation | input: | (None, 29, 29, 64) | output: | (None, 29, 29, 64)
Conv2D | input: | (None, 29, 29, 96) | output: | (None, 29, 29, 96)
BatchNormalization | input: | (None, 29, 29, 96) | output: | (None, 29, 29, 96)
Activation | input: | (None, 29, 29, 64) | output: | (None, 29, 29, 64)
Activation | input: | (None, 29, 29, 96) | output: | (None, 29, 29, 96)
Activation | input: | (None, 29, 29, 32) | output: | (None, 29, 29, 32)

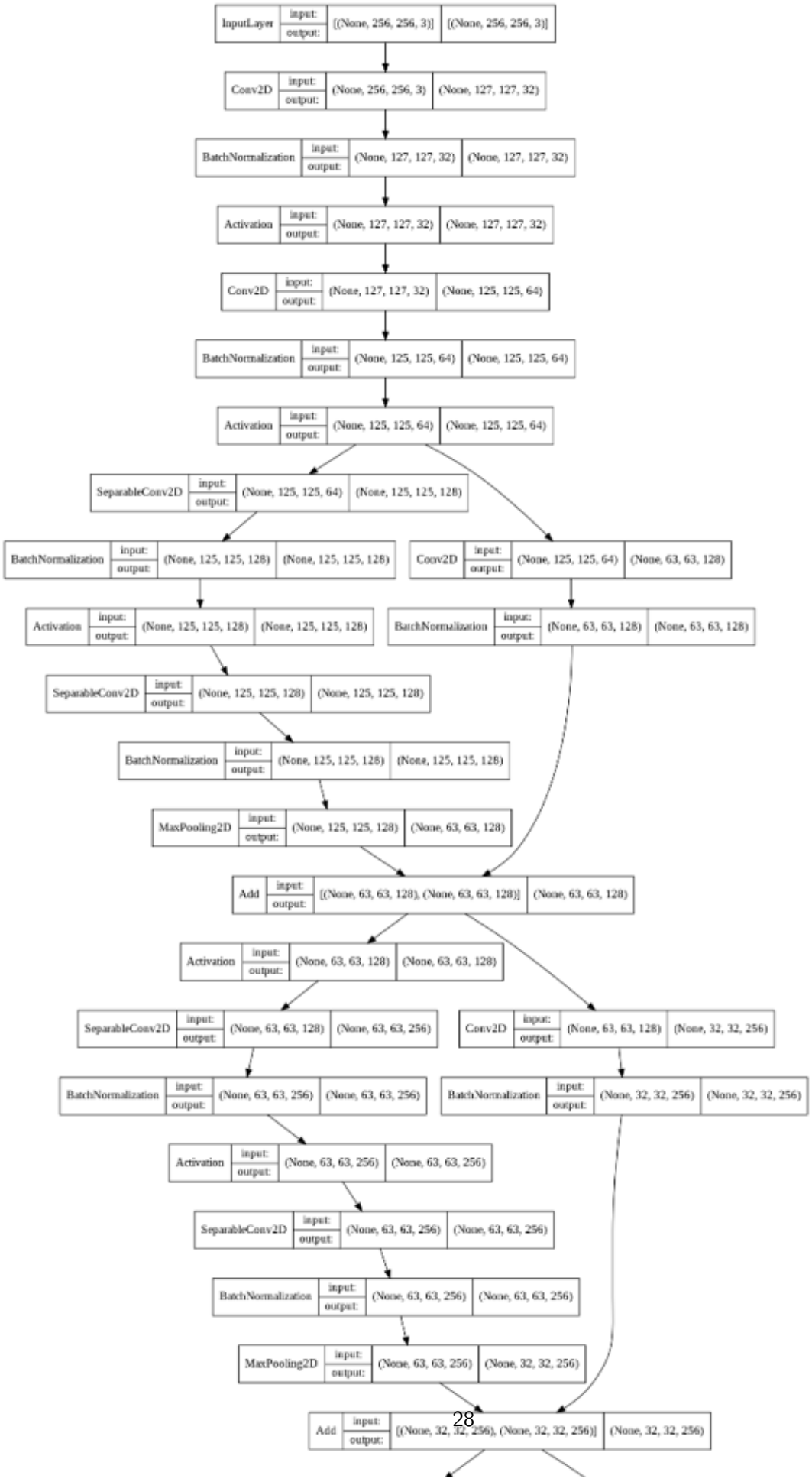

Concatenate input: [(None, 29, 29, 64), (None, 29, 29, 64), (None, 29, 29, 96), (None, 29, 29, 32)] output: (None, 29, 29, 256)
Conv2D input: (None, 29, 29, 256) output: (None, 29, 29, 64)
Conv2D input: (None, 29, 29, 256) output: (None, 29, 29, 48)
BatchNormalization input: (None, 29, 29, 64) output: (None, 29, 29, 64)
AveragePooling2D input: (None, 29, 29, 256) output: (None, 29, 29, 256)
Conv2D input: (None, 29, 29, 256) output: (None, 29, 29, 64)
Activation input: (None, 29, 29, 64) output: (None, 29, 29, 64)
BatchNormalization input: (None, 29, 29, 48) output: (None, 29, 29, 48)
Conv2D input: (None, 29, 29, 256) output: (None, 29, 29, 64)
Conv2D input: (None, 29, 29, 64) output: (None, 29, 29, 96)
Activation input: (None, 29, 29, 48) output: (None, 29, 29, 48)
BatchNormalization input: (None, 29, 29, 64) output: (None, 29, 29, 64)
BatchNormalization input: (None, 29, 29, 96) output: (None, 29, 29, 96)
Conv2D input: (None, 29, 29, 48) output: (None, 29, 29, 64)
BatchNormalization input: (None, 29, 29, 64) output: (None, 29, 29, 64)
Activation input: (None, 29, 29, 96) output: (None, 29, 29, 96)
BatchNormalization input: (None, 29, 29, 64) output: (None, 29, 29, 64)
Conv2D input: (None, 29, 29, 96) output: (None, 29, 29, 96)
Activation input: (None, 29, 29, 64) output: (None, 29, 29, 64)
BatchNormalization input: (None, 29, 29, 96) output: (None, 29, 29, 96)
Activation input: (None, 29, 29, 64) output: (None, 29, 29, 64)
Activation input: (None, 29, 29, 96) output: (None, 29, 29, 96)
Activation input: (None, 29, 29, 64) output: (None, 29, 29, 64)
Concatenate input: [(None, 29, 29, 64), (None, 29, 29, 64), (None, 29, 29, 96), (None, 29, 29, 64)] output: (None, 29, 29, 288)
Conv2D input: (None, 29, 29, 288) output: (None, 29, 29, 64)
Conv2D input: (None, 29, 29, 288) output: (None, 29, 29, 48)
BatchNormalization input: (None, 29, 29, 64) output: (None, 29, 29, 64)
AveragePooling2D input: (None, 29, 29, 288) output: (None, 29, 29, 288)
Conv2D input: (None, 29, 29, 288) output: (None, 29, 29, 64)
Activation input: (None, 29, 29, 64) output: (None, 29, 29, 64)
BatchNormalization input: (None, 29, 29, 48) output: (None, 29, 29, 48)
Conv2D input: (None, 29, 29, 288) output: (None, 29, 29, 64)
Conv2D input: (None, 29, 29, 64) output: (None, 29, 29, 96)
Activation input: (None, 29, 29, 48) output: (None, 29, 29, 48)
BatchNormalization input: (None, 29, 29, 64) output: (None, 29, 29, 64)
BatchNormalization input: (None, 29, 29, 96) output: (None, 29, 29, 96)
Conv2D input: (None, 29, 29, 48) output: (None, 29, 29, 64)
BatchNormalization input: (None, 29, 29, 64) output: (None, 29, 29, 64)
Activation input: (None, 29, 29, 96) output: (None, 29, 29, 96)
BatchNormalization input: (None, 29, 29, 64) output: (None, 29, 29, 64)
Activation input: (None, 29, 29, 64) output: (None, 29, 29, 64)
Conv2D input: (None, 29, 29, 96) output: (None, 29, 29, 96)
BatchNormalization input: (None, 29, 29, 96) output: (None, 29, 29, 96)
Activation input: (None, 29, 29, 64) output: (None, 29, 29, 64)
Activation input: (None, 29, 29, 96) output: (None, 29, 29, 96)
Activation input: (None, 29, 29, 64) output: (None, 29, 29, 64)
Concatenate input: [(None, 29, 29, 64), (None, 29, 29, 64), (None, 29, 29, 96), (None, 29, 29, 64)] output: (None, 29, 29, 288)
Conv2D input: (None, 29, 29, 288) output: (None, 29, 29, 64)
BatchNormalization input: (None, 29, 29, 64) output: (None, 29, 29, 64)
Conv2D input: (None, 29, 29, 288) output: (None, 14, 14, 384)
Activation input: (None, 29, 29, 64) output: (None, 29, 29, 64)
BatchNormalization input: (None, 14, 14, 384) output: (None, 14, 14, 384)
Conv2D input: (None, 29, 29, 64) output: (None, 29, 29, 96)
Activation input: (None, 14, 14, 384) output: (None, 14, 14, 384)
BatchNormalization input: (None, 29, 29, 96) output: (None, 29, 29, 96)
MaxPooling2D input: (None, 29, 29, 288) output: (None, 14, 14, 288)
Activation input: (None, 29, 29, 96) output: (None, 29, 29, 96)
Conv2D input: (None, 29, 29, 96) output: (None, 14, 14, 96)
BatchNormalization input: (None, 14, 14, 96) output: (None, 14, 14, 96)
Activation input: (None, 14, 14, 96) output: (None, 14, 14, 96)
Concatenate input: [(None, 14, 14, 384), (None, 14, 14, 96), (None, 14, 14, 288)] output: (None, 14, 14, 768)

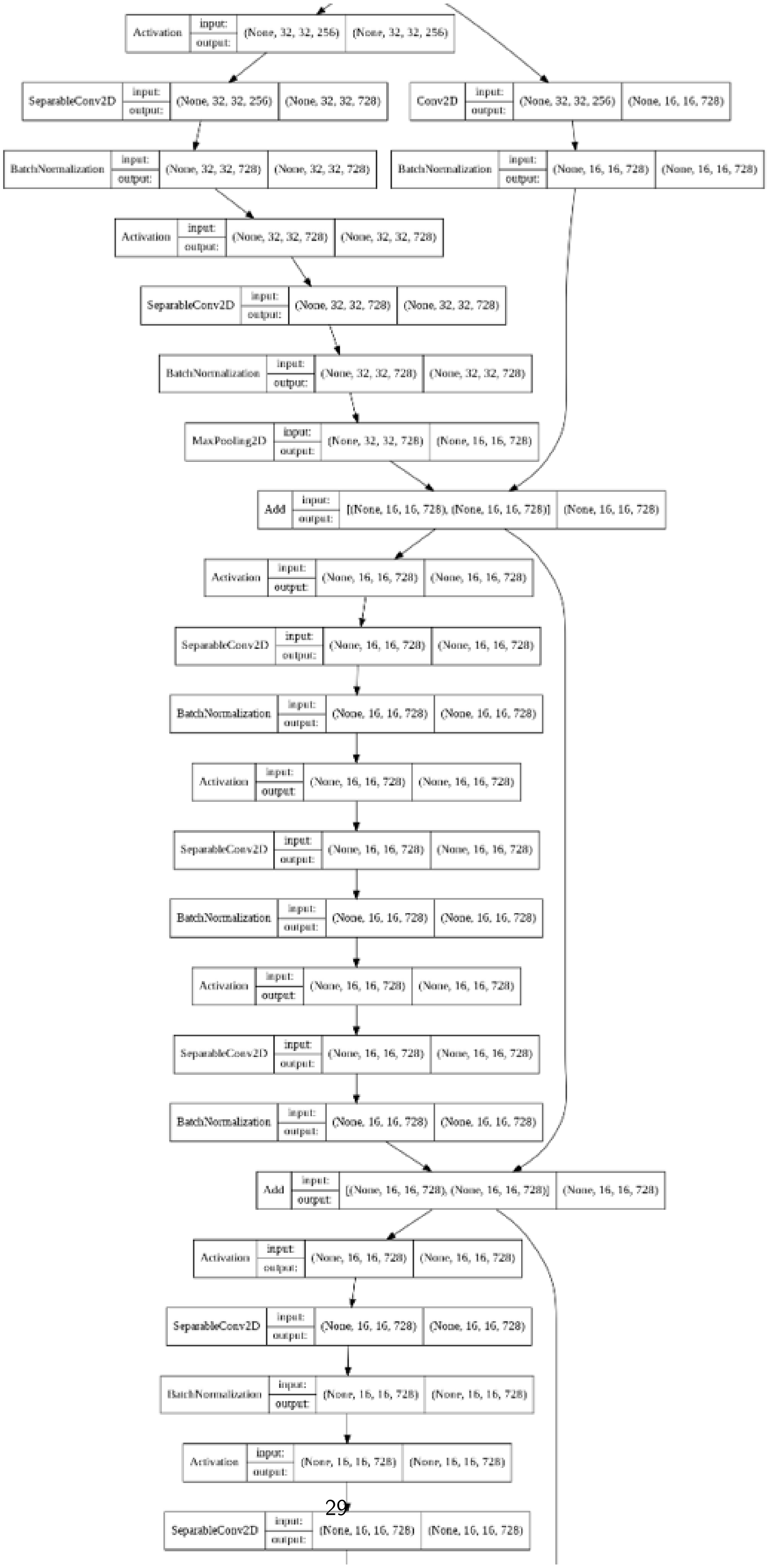

Conv2D input: (None, 14, 14, 768) output: (None, 14, 14, 128)
Conv2D input: (None, 14, 14, 768) output: (None, 14, 14, 128)
AveragePooling2D input: (None, 14, 14, 768) output: (None, 14, 14, 768)
Conv2D input: (None, 14, 14, 768) output: (None, 14, 14, 192)
BatchNormalization input: (None, 14, 14, 128) output: (None, 14, 14, 128)
Activation input: (None, 14, 14, 128) output: (None, 14, 14, 128)
BatchNormalization input: (None, 14, 14, 128) output: (None, 14, 14, 128)
Conv2D input: (None, 14, 14, 768) output: (None, 14, 14, 192)
Conv2D input: (None, 14, 14, 128) output: (None, 14, 14, 128)
Activation input: (None, 14, 14, 128) output: (None, 14, 14, 128)
BatchNormalization input: (None, 14, 14, 192) output: (None, 14, 14, 192)
BatchNormalization input: (None, 14, 14, 128) output: (None, 14, 14, 128)
Conv2D input: (None, 14, 14, 128) output: (None, 14, 14, 128)
BatchNormalization input: (None, 14, 14, 192) output: (None, 14, 14, 192)
Activation input: (None, 14, 14, 128) output: (None, 14, 14, 128)
BatchNormalization input: (None, 14, 14, 128) output: (None, 14, 14, 128)
Conv2D input: (None, 14, 14, 128) output: (None, 14, 14, 128)
Activation input: (None, 14, 14, 128) output: (None, 14, 14, 128)
BatchNormalization input: (None, 14, 14, 128) output: (None, 14, 14, 128)
Conv2D input: (None, 14, 14, 128) output: (None, 14, 14, 192)
Activation input: (None, 14, 14, 128) output: (None, 14, 14, 128)
BatchNormalization input: (None, 14, 14, 192) output: (None, 14, 14, 192)
Conv2D input: (None, 14, 14, 128) output: (None, 14, 14, 128)
Activation input: (None, 14, 14, 192) output: (None, 14, 14, 192)
BatchNormalization input: (None, 14, 14, 128) output: (None, 14, 14, 128)
Activation input: (None, 14, 14, 192) output: (None, 14, 14, 192)
Activation input: (None, 14, 14, 128) output: (None, 14, 14, 128)
Activation input: (None, 14, 14, 192) output: (None, 14, 14, 192)
Conv2D input: (None, 14, 14, 128) output: (None, 14, 14, 192)
BatchNormalization input: (None, 14, 14, 192) output: (None, 14, 14, 192)
Activation input: (None, 14, 14, 192) output: (None, 14, 14, 192)
Concatenate input: [(None, 14, 14, 192), (None, 14, 14, 192), (None, 14, 14, 192), (None, 14, 14, 192)] output: (None, 14, 14, 768)
Conv2D input: (None, 14, 14, 768) output: (None, 14, 14, 160)
Conv2D input: (None, 14, 14, 768) output: (None, 14, 14, 160)
BatchNormalization input: (None, 14, 14, 160) output: (None, 14, 14, 160)
AveragePooling2D input: (None, 14, 14, 768) output: (None, 14, 14, 768)
Conv2D input: (None, 14, 14, 768) output: (None, 14, 14, 192)
Activation input: (None, 14, 14, 160) output: (None, 14, 14, 160)
BatchNormalization input: (None, 14, 14, 160) output: (None, 14, 14, 160)
Conv2D input: (None, 14, 14, 768) output: (None, 14, 14, 192)
Conv2D input: (None, 14, 14, 160) output: (None, 14, 14, 160)
Activation input: (None, 14, 14, 160) output: (None, 14, 14, 160)
BatchNormalization input: (None, 14, 14, 192) output: (None, 14, 14, 192)
BatchNormalization input: (None, 14, 14, 160) output: (None, 14, 14, 160)
Conv2D input: (None, 14, 14, 160) output: (None, 14, 14, 160)
BatchNormalization input: (None, 14, 14, 192) output: (None, 14, 14, 192)
Activation input: (None, 14, 14, 160) output: (None, 14, 14, 160)
BatchNormalization input: (None, 14, 14, 160) output: (None, 14, 14, 160)
Conv2D input: (None, 14, 14, 160) output: (None, 14, 14, 160)
Activation input: (None, 14, 14, 160) output: (None, 14, 14, 160)
Conv2D input: (None, 14, 14, 160) output: (None, 14, 14, 192)
BatchNormalization input: (None, 14, 14, 160) output: (None, 14, 14, 160)
Activation input: (None, 14, 14, 160) output: (None, 14, 14, 160)
BatchNormalization input: (None, 14, 14, 192) output: (None, 14, 14, 192)
Activation input: (None, 14, 14, 192) output: (None, 14, 14, 192)
BatchNormalization input: (None, 14, 14, 160) output: (None, 14, 14, 160)
Activation input: (None, 14, 14, 192) output: (None, 14, 14, 192)
Activation input: (None, 14, 14, 160) output: (None, 14, 14, 160)
Activation input: (None, 14, 14, 192) output: (None, 14, 14, 192)
Conv2D input: (None, 14, 14, 160) output: (None, 14, 14, 192)
BatchNormalization input: (None, 14, 14, 192) output: (None, 14, 14, 192)
Activation input: (None, 14, 14, 192) output: (None, 14, 14, 192)
Concatenate input: [(None, 14, 14, 192), (None, 14, 14, 192), (None, 14, 14, 192), (None, 14, 14, 192)] output: (None, 14, 14, 768)
Conv2D input: (None, 14, 14, 768) output: (None, 14, 14, 160)
Conv2D input: (None, 14, 14, 768) output: (None, 14, 14, 160)

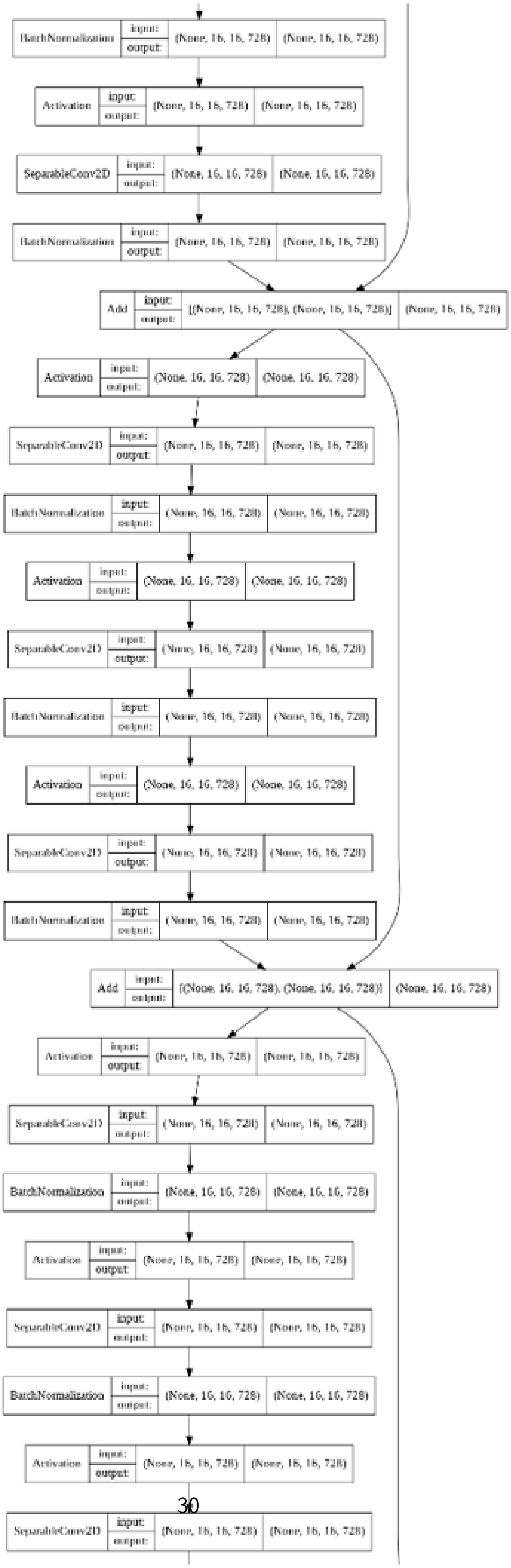

BatchNormalization input: (None, 14, 14, 160) output: (None, 14, 14, 160)
AveragePooling2D input: (None, 14, 14, 768) output: (None, 14, 14, 768)
Conv2D input: (None, 14, 14, 768) output: (None, 14, 14, 192)
Activation input: (None, 14, 14, 160) output: (None, 14, 14, 160)
BatchNormalization input: (None, 14, 14, 160) output: (None, 14, 14, 160)
Conv2D input: (None, 14, 14, 768) output: (None, 14, 14, 192)
Conv2D input: (None, 14, 14, 160) output: (None, 14, 14, 160)
Activation input: (None, 14, 14, 160) output: (None, 14, 14, 160)
BatchNormalization input: (None, 14, 14, 192) output: (None, 14, 14, 192)
BatchNormalization input: (None, 14, 14, 160) output: (None, 14, 14, 160)
Conv2D input: (None, 14, 14, 160) output: (None, 14, 14, 160)
BatchNormalization input: (None, 14, 14, 192) output: (None, 14, 14, 192)
Activation input: (None, 14, 14, 160) output: (None, 14, 14, 160)
BatchNormalization input: (None, 14, 14, 160) output: (None, 14, 14, 160)
Conv2D input: (None, 14, 14, 160) output: (None, 14, 14, 160)
Activation input: (None, 14, 14, 160) output: (None, 14, 14, 160)
BatchNormalization input: (None, 14, 14, 160) output: (None, 14, 14, 160)
Conv2D input: (None, 14, 14, 160) output: (None, 14, 14, 192)
Activation input: (None, 14, 14, 160) output: (None, 14, 14, 160)
BatchNormalization input: (None, 14, 14, 192) output: (None, 14, 14, 192)
Conv2D input: (None, 14, 14, 160) output: (None, 14, 14, 160)
Activation input: (None, 14, 14, 192) output: (None, 14, 14, 192)
BatchNormalization input: (None, 14, 14, 160) output: (None, 14, 14, 160)
Activation input: (None, 14, 14, 192) output: (None, 14, 14, 192)
Activation input: (None, 14, 14, 160) output: (None, 14, 14, 160)
Activation input: (None, 14, 14, 192) output: (None, 14, 14, 192)
Conv2D input: (None, 14, 14, 160) output: (None, 14, 14, 192)
BatchNormalization input: (None, 14, 14, 192) output: (None, 14, 14, 192)
Activation input: (None, 14, 14, 192) output: (None, 14, 14, 192)
Concatenate input: [(None, 14, 14, 192), (None, 14, 14, 192), (None, 14, 14, 192), (None, 14, 14, 192)] output: (None, 14, 14, 768)
Conv2D input: (None, 14, 14, 768) output: (None, 14, 14, 192)
Conv2D input: (None, 14, 14, 768) output: (None, 14, 14, 192)
BatchNormalization input: (None, 14, 14, 192) output: (None, 14, 14, 192)
AveragePooling2D input: (None, 14, 14, 768) output: (None, 14, 14, 768)
Conv2D input: (None, 14, 14, 768) output: (None, 14, 14, 192)
Activation input: (None, 14, 14, 192) output: (None, 14, 14, 192)
BatchNormalization input: (None, 14, 14, 192) output: (None, 14, 14, 192)
Conv2D input: (None, 14, 14, 768) output: (None, 14, 14, 192)
Conv2D input: (None, 14, 14, 192) output: (None, 14, 14, 192)
Activation input: (None, 14, 14, 192) output: (None, 14, 14, 192)
BatchNormalization input: (None, 14, 14, 192) output: (None, 14, 14, 192)
BatchNormalization input: (None, 14, 14, 192) output: (None, 14, 14, 192)
Conv2D input: (None, 14, 14, 192) output: (None, 14, 14, 192)
Activation input: (None, 14, 14, 192) output: (None, 14, 14, 192)
BatchNormalization input: (None, 14, 14, 192) output: (None, 14, 14, 192)
Activation input: (None, 14, 14, 192) output: (None, 14, 14, 192)
Conv2D input: (None, 14, 14, 192) output: (None, 14, 14, 192)
Activation input: (None, 14, 14, 192) output: (None, 14, 14, 192)
BatchNormalization input: (None, 14, 14, 192) output: (None, 14, 14, 192)
Conv2D input: (None, 14, 14, 192) output: (None, 14, 14, 192)
Activation input: (None, 14, 14, 192) output: (None, 14, 14, 192)
BatchNormalization input: (None, 14, 14, 192) output: (None, 14, 14, 192)
Activation input: (None, 14, 14, 192) output: (None, 14, 14, 192)
Conv2D input: (None, 14, 14, 192) output: (None, 14, 14, 192)
BatchNormalization input: (None, 14, 14, 192) output: (None, 14, 14, 192)
Activation input: (None, 14, 14, 192) output: (None, 14, 14, 192)
Activation input: (None, 14, 14, 192) output: (None, 14, 14, 192)
Conv2D input: (None, 14, 14, 192) output: (None, 14, 14, 192)
BatchNormalization input: (None, 14, 14, 192) output: (None, 14, 14, 192)
Activation input: (None, 14, 14, 192) output: (None, 14, 14, 192)
Concatenate input: [(None, 14, 14, 192), (None, 14, 14, 192), (None, 14, 14, 192), (None, 14, 14, 192)] output: (None, 14, 14, 768)
Conv2D input: (None, 14, 14, 768) output: (None, 14, 14, 192)
BatchNormalization input: (None, 14, 14, 192) output: (None, 14, 14, 192)
Conv2D input: (None, 14, 14, 768) output: (None, 14, 14, 192)
Activation input: (None, 14, 14, 192) output: (None, 14, 14, 192)
BatchNormalization input: (None, 14, 14, 192) output: (None, 14, 14, 192)
Conv2D input: (None, 14, 14, 192) output: (None, 14, 14, 192)
Activation input: (None, 14, 14, 192) output: (None, 14, 14, 192)

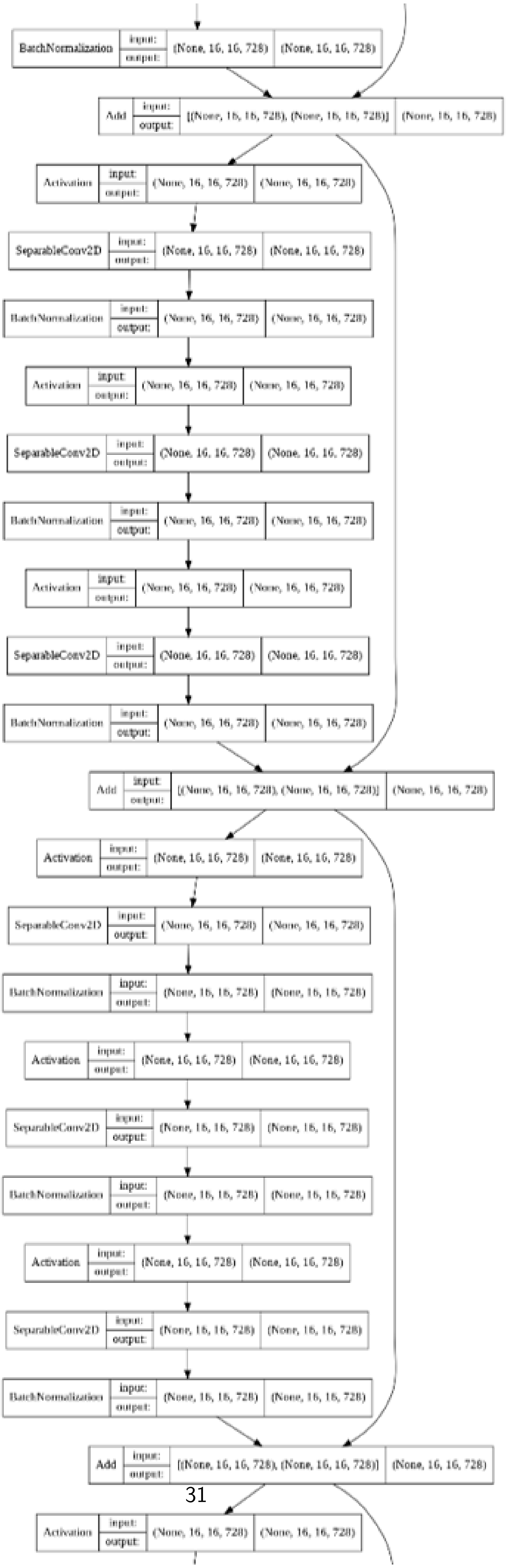

BatchNormalization input: (None, 14, 14, 192) output: (None, 14, 14, 192)
Conv2D input: (None, 14, 14, 192) output: (None, 6, 6, 320)
Activation input: (None, 14, 14, 192) output: (None, 14, 14, 192)
BatchNormalization input: (None, 6, 6, 320) output: (None, 6, 6, 320)
Conv2D input: (None, 14, 14, 192) output: (None, 14, 14, 192)
Activation input: (None, 6, 6, 320) output: (None, 6, 6, 320)
BatchNormalization input: (None, 14, 14, 192) output: (None, 14, 14, 192)
MaxPooling2D input: (None, 14, 14, 768) output: (None, 6, 6, 768)
Activation input: (None, 14, 14, 192) output: (None, 14, 14, 192)
Conv2D input: (None, 14, 14, 192) output: (None, 6, 6, 192)
BatchNormalization input: (None, 6, 6, 192) output: (None, 6, 6, 192)
Activation input: (None, 6, 6, 192) output: (None, 6, 6, 192)
Concatenate input: [(None, 6, 6, 320), (None, 6, 6, 192), (None, 6, 6, 768)] output: (None, 6, 6, 1280)
Conv2D input: (None, 6, 6, 1280) output: (None, 6, 6, 448)
Conv2D input: (None, 6, 6, 1280) output: (None, 6, 6, 384)
AveragePooling2D input: (None, 6, 6, 1280) output: (None, 6, 6, 1280)
BatchNormalization input: (None, 6, 6, 448) output: (None, 6, 6, 448)
BatchNormalization input: (None, 6, 6, 384) output: (None, 6, 6, 384)
Conv2D input: (None, 6, 6, 1280) output: (None, 6, 6, 192)
Conv2D input: (None, 6, 6, 1280) output: (None, 6, 6, 320)
Activation input: (None, 6, 6, 448) output: (None, 6, 6, 448)
Activation input: (None, 6, 6, 384) output: (None, 6, 6, 384)
BatchNormalization input: (None, 6, 6, 192) output: (None, 6, 6, 192)
Conv2D input: (None, 6, 6, 448) output: (None, 6, 6, 384)
Conv2D input: (None, 6, 6, 384) output: (None, 6, 6, 384)
Conv2D input: (None, 6, 6, 384) output: (None, 6, 6, 384)
BatchNormalization input: (None, 6, 6, 384) output: (None, 6, 6, 384)
BatchNormalization input: (None, 6, 6, 384) output: (None, 6, 6, 384)
BatchNormalization input: (None, 6, 6, 384) output: (None, 6, 6, 384)
Activation input: (None, 6, 6, 384) output: (None, 6, 6, 384)
Activation input: (None, 6, 6, 384) output: (None, 6, 6, 384)
Activation input: (None, 6, 6, 384) output: (None, 6, 6, 384)
Conv2D input: (None, 6, 6, 384) output: (None, 6, 6, 384)
Conv2D input: (None, 6, 6, 384) output: (None, 6, 6, 384)
Concatenate input: [(None, 6, 6, 384), (None, 6, 6, 384)] output: (None, 6, 6, 768)
BatchNormalization input: (None, 6, 6, 384) output: (None, 6, 6, 384)
BatchNormalization input: (None, 6, 6, 384) output: (None, 6, 6, 384)
BatchNormalization input: (None, 6, 6, 320) output: (None, 6, 6, 320)
Activation input: (None, 6, 6, 384) output: (None, 6, 6, 384)
Activation input: (None, 6, 6, 384) output: (None, 6, 6, 384)
Activation input: (None, 6, 6, 192) output: (None, 6, 6, 192)
Concatenate input: [(None, 6, 6, 384), (None, 6, 6, 384)] output: (None, 6, 6, 768)
Activation input: (None, 6, 6, 320) output: (None, 6, 6, 320)
Concatenate input: [(None, 6, 6, 320), (None, 6, 6, 768), (None, 6, 6, 768), (None, 6, 6, 192)] output: (None, 6, 6, 2048)
Conv2D input: (None, 6, 6, 2048) output: (None, 6, 6, 448)
Conv2D input: (None, 6, 6, 2048) output: (None, 6, 6, 384)
AveragePooling2D input: (None, 6, 6, 2048) output: (None, 6, 6, 2048)
BatchNormalization input: (None, 6, 6, 448) output: (None, 6, 6, 448)
BatchNormalization input: (None, 6, 6, 384) output: (None, 6, 6, 384)
Conv2D input: (None, 6, 6, 2048) output: (None, 6, 6, 192)
Conv2D input: (None, 6, 6, 2048) output: (None, 6, 6, 320)
Activation input: (None, 6, 6, 448) output: (None, 6, 6, 448)
Activation input: (None, 6, 6, 384) output: (None, 6, 6, 384)
BatchNormalization input: (None, 6, 6, 192) output: (None, 6, 6, 192)
Conv2D input: (None, 6, 6, 448) output: (None, 6, 6, 384)
Conv2D input: (None, 6, 6, 384) output: (None, 6, 6, 384)
Conv2D input: (None, 6, 6, 384) output: (None, 6, 6, 384)
BatchNormalization input: (None, 6, 6, 384) output: (None, 6, 6, 384)
BatchNormalization input: (None, 6, 6, 384) output: (None, 6, 6, 384)
BatchNormalization input: (None, 6, 6, 384) output: (None, 6, 6, 384)
Activation input: (None, 6, 6, 384) output: (None, 6, 6, 384)
Activation input: (None, 6, 6, 384) output: (None, 6, 6, 384)
Activation input: (None, 6, 6, 384) output: (None, 6, 6, 384)
Conv2D input: (None, 6, 6, 384) output: (None, 6, 6, 384)
Conv2D input: (None, 6, 6, 384) output: (None, 6, 6, 384)
Concatenate input: [(None, 6, 6, 384), (None, 6, 6, 384)] output: (None, 6, 6, 768)
BatchNormalization input: (None, 6, 6, 384) output: (None, 6, 6, 384)
BatchNormalization input: (None, 6, 6, 384) output: (None, 6, 6, 384)
BatchNormalization input: (None, 6, 6, 320) output: (None, 6, 6, 320)
Activation input: (None, 6, 6, 384) output: (None, 6, 6, 384)
Activation input: (None, 6, 6, 384) output: (None, 6, 6, 384)
Activation input: (None, 6, 6, 192) output: (None, 6, 6, 192)
Concatenate input: [(None, 6, 6, 384), (None, 6, 6, 384)] output: (None, 6, 6, 768)
Activation input: (None, 6, 6, 320) output: (None, 6, 6, 320)
Concatenate input: [(None, 6, 6, 320), (None, 6, 6, 768), (None, 6, 6, 768), (None, 6, 6, 192)] output: (None, 6, 6, 2048)

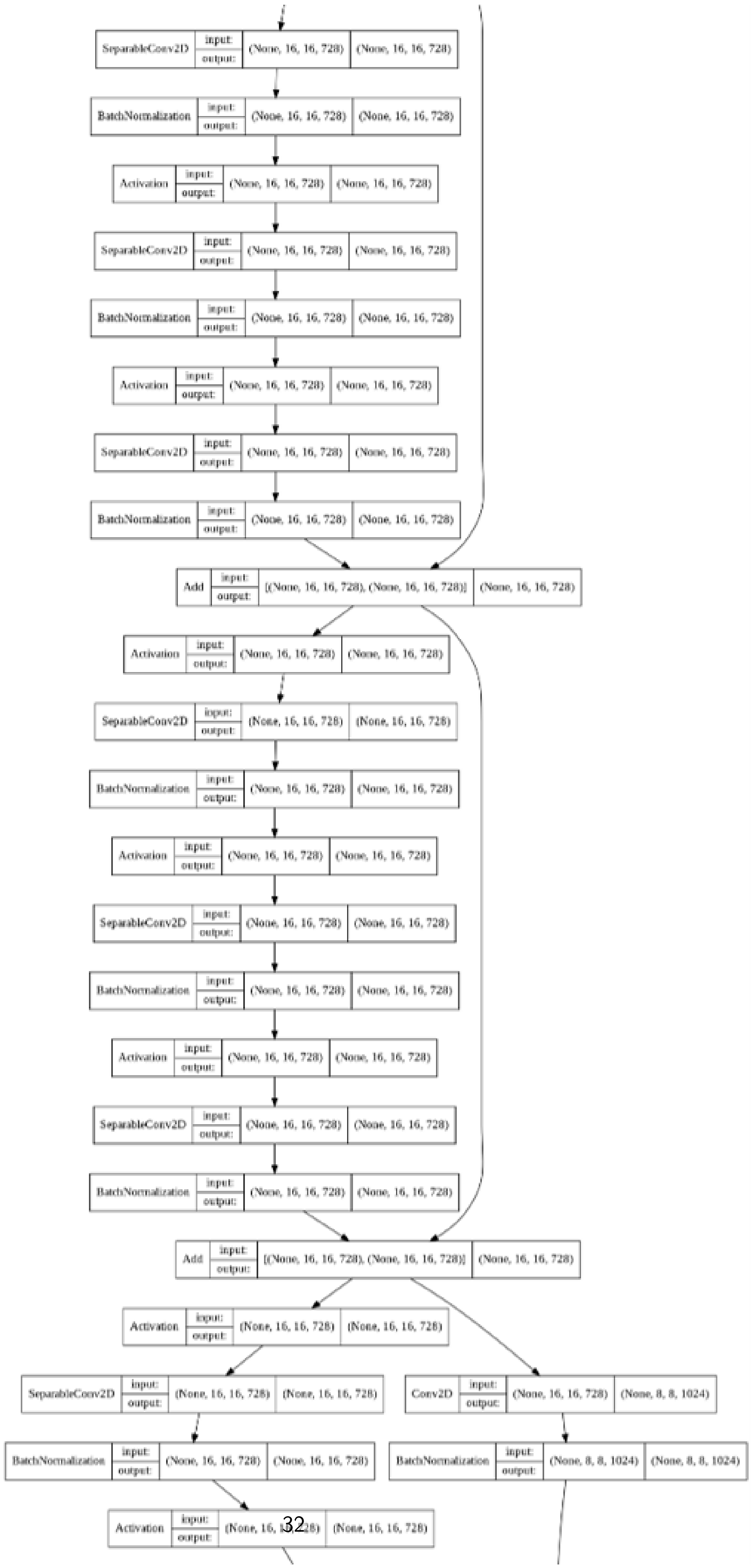
Appendix C


InputLayer | input: | [(None, 256, 256, 3)] | [(None, 256, 256, 3)]
Conv2D | input: | (None, 256, 256, 3) | (None, 127, 127, 32)
BatchNormalization | input: | (None, 127, 127, 32) | (None, 127, 127, 32)
Activation | input: | (None, 127, 127, 32) | (None, 127, 127, 32)
Conv2D | input: | (None, 127, 127, 32) | (None, 125, 125, 64)
BatchNormalization | input: | (None, 125, 125, 64) | (None, 125, 125, 64)
Activation | input: | (None, 125, 125, 64) | (None, 125, 125, 64)
SeparableConv2D | input: | (None, 125, 125, 64) | (None, 125, 125, 128)
BatchNormalization | input: | (None, 125, 125, 128) | (None, 125, 125, 128)
Conv2D | input: | (None, 125, 125, 64) | (None, 63, 63, 128)
Activation | input: | (None, 125, 125, 128) | (None, 125, 125, 128)
BatchNormalization | input: | (None, 63, 63, 128) | (None, 63, 63, 128)
SeparableConv2D | input: | (None, 125, 125, 128) | (None, 125, 125, 128)
BatchNormalization | input: | (None, 125, 125, 128) | (None, 125, 125, 128)
MaxPooling2D | input: | (None, 125, 125, 128) | (None, 63, 63, 128)
Add | input: | [(None, 63, 63, 128), (None, 63, 63, 128)] | (None, 63, 63, 128)
Activation | input: | (None, 63, 63, 128) | (None, 63, 63, 128)
SeparableConv2D | input: | (None, 63, 63, 128) | (None, 63, 63, 256)
Conv2D | input: | (None, 63, 63, 128) | (None, 32, 32, 256)
BatchNormalization | input: | (None, 63, 63, 256) | (None, 63, 63, 256)
BatchNormalization | input: | (None, 32, 32, 256) | (None, 32, 32, 256)
Activation | input: | (None, 63, 63, 256) | (None, 63, 63, 256)
SeparableConv2D | input: | (None, 63, 63, 256) | (None, 63, 63, 256)
BatchNormalization | input: | (None, 63, 63, 256) | (None, 63, 63, 256)
MaxPooling2D | input: | (None, 63, 63, 256) | (None, 32, 32, 256)
Add | input: | [(None, 32, 32, 256), (None, 32, 32, 256)] | (None, 32, 32, 256)


Activation | input: | output: | (None, 32, 32, 256) | (None, 32, 32, 256)
SeparableConv2D | input: | output: | (None, 32, 32, 256) | (None, 32, 32, 728)
Conv2D | input: | output: | (None, 32, 32, 256) | (None, 16, 16, 728)
BatchNormalization | input: | output: | (None, 32, 32, 728) | (None, 32, 32, 728)
BatchNormalization | input: | output: | (None, 16, 16, 728) | (None, 16, 16, 728)
Activation | input: | output: | (None, 32, 32, 728) | (None, 32, 32, 728)
SeparableConv2D | input: | output: | (None, 32, 32, 728) | (None, 32, 32, 728)
BatchNormalization | input: | output: | (None, 32, 32, 728) | (None, 32, 32, 728)
MaxPooling2D | input: | output: | (None, 32, 32, 728) | (None, 16, 16, 728)
Add | input: | output: | [(None, 16, 16, 728), (None, 16, 16, 728)] | (None, 16, 16, 728)
Activation | input: | output: | (None, 16, 16, 728) | (None, 16, 16, 728)
SeparableConv2D | input: | output: | (None, 16, 16, 728) | (None, 16, 16, 728)
BatchNormalization | input: | output: | (None, 16, 16, 728) | (None, 16, 16, 728)
Activation | input: | output: | (None, 16, 16, 728) | (None, 16, 16, 728)
SeparableConv2D | input: | output: | (None, 16, 16, 728) | (None, 16, 16, 728)
BatchNormalization | input: | output: | (None, 16, 16, 728) | (None, 16, 16, 728)
Activation | input: | output: | (None, 16, 16, 728) | (None, 16, 16, 728)
SeparableConv2D | input: | output: | (None, 16, 16, 728) | (None, 16, 16, 728)
BatchNormalization | input: | output: | (None, 16, 16, 728) | (None, 16, 16, 728)
Add | input: | output: | [(None, 16, 16, 728), (None, 16, 16, 728)] | (None, 16, 16, 728)
Activation | input: | output: | (None, 16, 16, 728) | (None, 16, 16, 728)
SeparableConv2D | input: | output: | (None, 16, 16, 728) | (None, 16, 16, 728)
BatchNormalization | input: | output: | (None, 16, 16, 728) | (None, 16, 16, 728)
Activation | input: | output: | (None, 16, 16, 728) | (None, 16, 16, 728)
SeparableConv2D | input: | output: | (None, 16, 16, 728) | (None, 16, 16, 728)


BatchNormalization
input:
output:
(None, 16, 16, 728)
Activation
SeparableConv2D
Add
[(None, 16, 16, 728), (None, 16, 16, 728)]


BatchNormalization input: output: (None, 16, 16, 728) (None, 16, 16, 728)
Add input: output: [(None, 16, 16, 728), (None, 16, 16, 728)] (None, 16, 16, 728)
Activation input: output: (None, 16, 16, 728) (None, 16, 16, 728)
SeparableConv2D input: output: (None, 16, 16, 728) (None, 16, 16, 728)
BatchNormalization input: output: (None, 16, 16, 728) (None, 16, 16, 728)
Activation input: output: (None, 16, 16, 728) (None, 16, 16, 728)
SeparableConv2D input: output: (None, 16, 16, 728) (None, 16, 16, 728)
BatchNormalization input: output: (None, 16, 16, 728) (None, 16, 16, 728)
Activation input: output: (None, 16, 16, 728) (None, 16, 16, 728)
SeparableConv2D input: output: (None, 16, 16, 728) (None, 16, 16, 728)
BatchNormalization input: output: (None, 16, 16, 728) (None, 16, 16, 728)
Add input: output: [(None, 16, 16, 728), (None, 16, 16, 728)] (None, 16, 16, 728)
Activation input: output: (None, 16, 16, 728) (None, 16, 16, 728)
SeparableConv2D input: output: (None, 16, 16, 728) (None, 16, 16, 728)
BatchNormalization input: output: (None, 16, 16, 728) (None, 16, 16, 728)
Activation input: output: (None, 16, 16, 728) (None, 16, 16, 728)
SeparableConv2D input: output: (None, 16, 16, 728) (None, 16, 16, 728)
BatchNormalization input: output: (None, 16, 16, 728) (None, 16, 16, 728)
Activation input: output: (None, 16, 16, 728) (None, 16, 16, 728)
SeparableConv2D input: output: (None, 16, 16, 728) (None, 16, 16, 728)
BatchNormalization input: output: (None, 16, 16, 728) (None, 16, 16, 728)
Add input: output: [(None, 16, 16, 728), (None, 16, 16, 728)] (None, 16, 16, 728)
Activation input: output: (None, 16, 16, 728) (None, 16, 16, 728)


SeparableConv2D input: output: (None, 16, 16, 728) (None, 16, 16, 728)
BatchNormalization input: output: (None, 16, 16, 728) (None, 16, 16, 728)
Activation input: output: (None, 16, 16, 728) (None, 16, 16, 728)
SeparableConv2D input: output: (None, 16, 16, 728) (None, 16, 16, 728)
BatchNormalization input: output: (None, 16, 16, 728) (None, 16, 16, 728)
Activation input: output: (None, 16, 16, 728) (None, 16, 16, 728)
SeparableConv2D input: output: (None, 16, 16, 728) (None, 16, 16, 728)
BatchNormalization input: output: (None, 16, 16, 728) (None, 16, 16, 728)
Add input: output: [(None, 16, 16, 728), (None, 16, 16, 728)] (None, 16, 16, 728)
Activation input: output: (None, 16, 16, 728) (None, 16, 16, 728)
SeparableConv2D input: output: (None, 16, 16, 728) (None, 16, 16, 728)
BatchNormalization input: output: (None, 16, 16, 728) (None, 16, 16, 728)
Activation input: output: (None, 16, 16, 728) (None, 16, 16, 728)
SeparableConv2D input: output: (None, 16, 16, 728) (None, 16, 16, 728)
BatchNormalization input: output: (None, 16, 16, 728) (None, 16, 16, 728)
Activation input: output: (None, 16, 16, 728) (None, 16, 16, 728)
SeparableConv2D input: output: (None, 16, 16, 728) (None, 16, 16, 728)
BatchNormalization input: output: (None, 16, 16, 728) (None, 16, 16, 728)
Add input: output: [(None, 16, 16, 728), (None, 16, 16, 728)] (None, 16, 16, 728)
Activation input: output: (None, 16, 16, 728) (None, 16, 16, 728)
SeparableConv2D input: output: (None, 16, 16, 728) (None, 16, 16, 728)
Conv2D input: output: (None, 16, 16, 728) (None, 8, 8, 1024)
BatchNormalization input: output: (None, 16, 16, 728) (None, 16, 16, 728)
BatchNormalization input: output: (None, 8, 8, 1024) (None, 8, 8, 1024)
Activation input: output: (None, 16, 16, 728) (None, 16, 16, 728)

Appendix D

PART 1 : BY TRAINING LOSS

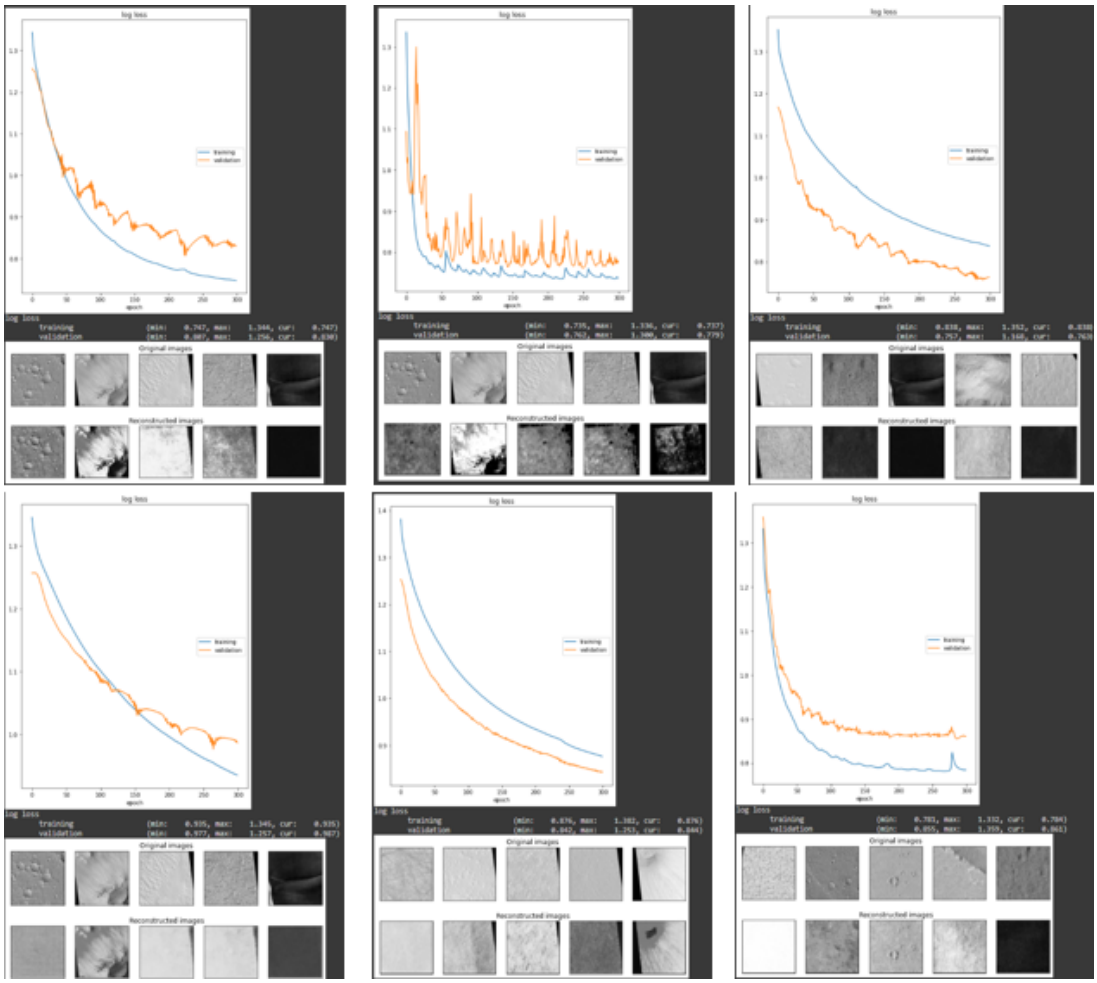
Model No.	Learning Rate	Batch Size	Latent Dimensions	Training Loss	Validation Loss	Diff
513	0.009	64	100	0.624226809	1.010261655	#VALUE!
503	0.009	32	100	0.624235451	1.005691528	#VALUE!
483	0.009	8	100	0.624298811	1.014780164	#VALUE!
533	0.009	256	100	0.625053287	1.006362557	0.00018
433	0.0008	16	100	0.626246214	1.0176512	0.000595
453	0.0008	64	100	0.62646234	1.010237932	0.001127
423	0.0008	8	100	0.6264925	1.004957557	0.001364
493	0.009	16	100	0.626830697	1.005199075	0.001622
443	0.0008	32	100	0.626869917	1.012652993	0.002101
473	0.0008	256	100	0.626915753	1.01342833	0.002127
403	0.0007	128	100	0.628757596	1.01237905	0.002792
393	0.0007	64	100	0.629181504	1.015418649	0.003076
523	0.009	128	100	0.629454255	1.005549788	0.00318
373	0.0007	16	100	0.629458368	1.013601542	0.003184
363	0.0007	8	100	0.630118847	1.01427114	0.00426
383	0.0007	32	100	0.630409241	1.019189477	0.005433
303	0.0006	8	100	0.632940173	1.032733202	0.006463
333	0.0006	64	100	0.633112729	1.027141809	0.007426
323	0.0006	32	100	0.633231878	1.026400805	0.007693
343	0.0006	128	100	0.634303868	1.025708079	0.010406
313	0.0006	16	100	0.635282397	1.022289157	0.01068

PART 2 : BY VALIDATION LOSS

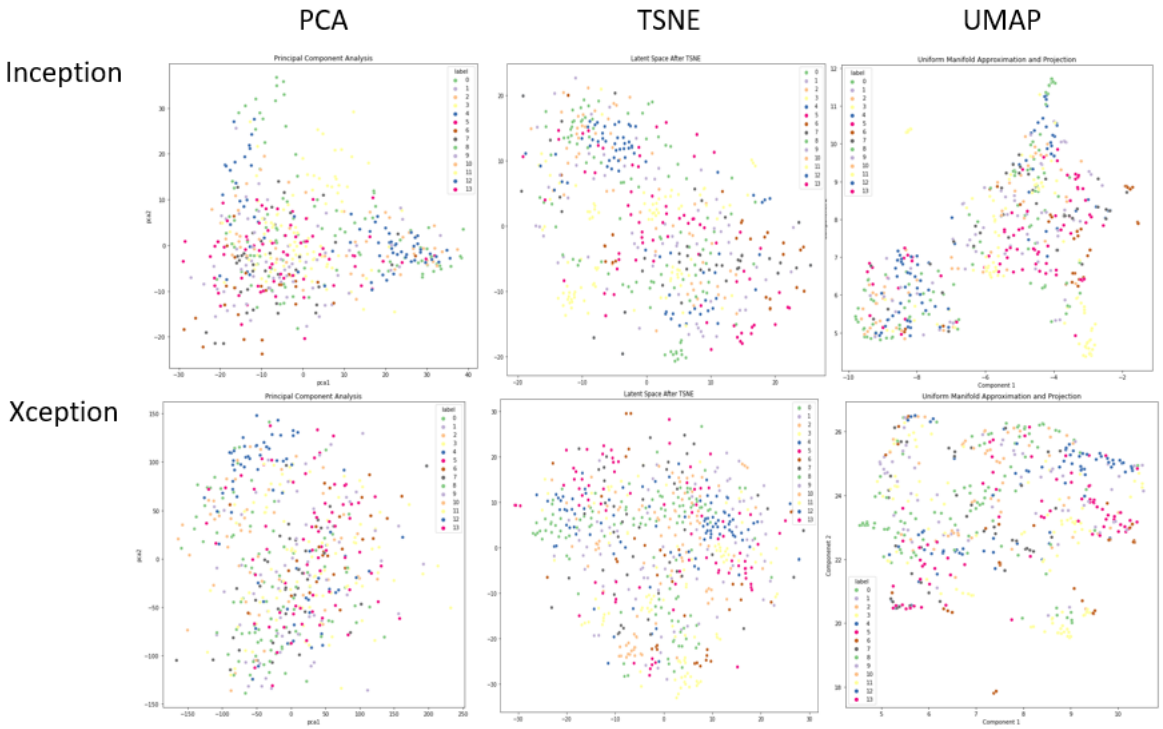
Model	Learning Rate	Batch Size	Latent Dimensions	Training Loss	Validation Loss	Diff
514	0.009	64	120	0.694744587	0.644101799	0.052088
504	0.009	32	120	0.692000508	0.647637308	0.052099
434	0.0008	16	120	0.697401762	0.650553286	0.052322
394	0.0007	64	120	0.697587967	0.650907636	0.052535
374	0.0007	16	120	0.698337317	0.651141524	0.0526
494	0.009	16	120	0.692774296	0.651308179	0.052989
364	0.0007	8	120	0.699170411	0.651769459	0.053522
464	0.0008	128	120	0.695126176	0.652526259	0.053575
484	0.009	8	120	0.693000317	0.652648449	0.053644
424	0.0008	8	120	0.696368337	0.654276371	0.053858
404	0.0007	128	120	0.697655678	0.654978156	0.054094
474	0.0008	256	120	0.693969607	0.655763686	0.054269
314	0.0006	16	120	0.703797996	0.65670222	0.054308
384	0.0007	32	120	0.697814584	0.65839988	0.054586
324	0.0006	32	120	0.703073025	0.658413708	0.054691
354	0.0006	256	120	0.704109907	0.659182847	0.054843
344	0.0006	128	120	0.702579618	0.659824073	0.054954
244	0.0005	8	120	0.714835405	0.659880996	0.054954
304	0.0006	8	120	0.702921391	0.66057688	0.055249
524	0.009	128	120	0.691885114	0.660686314	0.05588

PART 3 : BY DIFFERENCE

Model	Learning Rate	Batch Size	Latent Dimensions	Training Loss	Validation Loss	Diff
8	0.0001	8	256	0.909699678	0.909879506	0.163738
48	0.0001	128	256	0.910177052	0.910771966	0.188955
15	0.0001	16	160	0.933409929	0.932282925	0.197604
25	0.0001	32	160	0.933524728	0.932160556	0.198131
5	0.0001	8	160	0.933740079	0.93211782	0.198709
38	0.0001	64	256	0.909556687	0.91165781	0.20276
58	0.0001	256	256	0.909588099	0.911714733	0.218366
45	0.0001	128	160	0.933622837	0.930831313	0.220414
55	0.0001	256	160	0.933921218	0.930845618	0.222323
18	0.0001	16	256	0.909586966	0.91276741	0.222833
35	0.0001	64	160	0.934344232	0.931159794	0.22505
414	0.0007	256	120	0.699897885	0.695637703	0.226027
28	0.0001	32	256	0.909602284	0.915035546	0.226114
265	0.0005	32	160	0.809641838	0.816104531	0.226978
358	0.0006	256	256	0.753964067	0.746537864	0.227712
268	0.0005	32	256	0.763494611	0.771187603	0.228898
78	0.0002	16	256	0.818252802	0.828659058	0.229761
278	0.0005	64	256	0.746107221	0.756786883	0.231168
465	0.0008	128	160	0.792895973	0.804323673	0.231271
298	0.0005	256	256	0.750139713	0.762186885	0.231598
448	0.0008	32	256	0.728885889	0.74096185	0.231724



Appendix E


PCA
TSNE
UMAP
Inception
Xception
Principal Component Analysis
Latent Space After TSNE
Uniform Manifold Approximation and Projection

Appendix F

Group 1 : Top 3 Model Results

	brain-terrain	chaos-terrain	concentric-crater	cratered-cones	dune	dust-devil	glaciers	gullies	lava-flow	layers	pedestal-crater	ridges	scalloped-depression	slope-streaks	Average
Agg Clustering xception tsne	0.50	0.27	0.57	0.91	0.09	0.50	0.71	0.50	0.69	0.40	0.00	0.90	0.45	0.50	0.50
BIRCH xception umap	0.60	0.42	0.53	0.88	0.00	0.25	0.67	0.38	0.67	0.00	0.33	1.00	0.70	1.00	0.53
Agg Clustering inception tsne	0.50	0.50	0.50	0.77	0.45	0.83	0.50	0.17	1.00	0.09	0.20	0.80	0.36	0.29	0.50

Group 2 : Top 3 Model Results

	Rand Index	Adjusted RI	Mutual Information	Normalized MI	Adjusted MI	Balanced Accuracy	Homogeneity	Completeness	V-Score	Average
Affinity Clustering inception pca	0.91	0.35	1.57	0.62	0.49	0.19	0.64	0.60	0.62	0.66
Affinity Clustering xception pca	0.93	0.43	1.83	0.68	0.54	0.02	0.66	0.70	0.68	0.72
KMeans inception umap	0.92	0.35	1.61	0.62	0.48	0.18	0.62	0.61	0.62	0.67

Appendix G

Models	brain-terrain	chaos-terrain	concentric-crater	cratered-cones	dune	dust-devil	glaciers	gullies	lava-flow	layers	pedestal-crater	ridges	scalloped-depression	slope-streaks	Average
KMeans inception PCA	0.15	0.43	0.45	1.00	0.63	1.00	0.40	0.33	1.00	0.00	0.00	0.67	0.38	0.50	0.50
KMeans inception t-SNE	0.56	0.36	0.63	0.83	0.20	0.63	0.50	0.71	0.69	0.00	0.00	0.60	0.44	0.13	0.45
KMeans inception UMAP	0.40	0.22	0.00	0.00	0.00	0.75	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.10
KMeans Xception PCA	0.00	0.00	0.00	0.00	0.13	0.00	0.00	0.00	0.00	0.00	0.13	0.00	0.00	0.00	0.02
KMeans Xception t-SNE	0.33	0.00	0.73	0.90	0.30	0.22	0.67	0.40	0.82	0.18	0.00	0.90	0.60	0.50	0.47
KMeans Xception UMAP	0.25	0.11	0.46	0.00	0.00	0.00	0.10	0.00	0.00	0.00	0.10	0.00	0.50	0.29	0.13
Agg Clustering inception PCA	0.25	0.44	0.00	1.00	0.63	1.00	0.29	0.44	1.00	0.20	0.00	0.00	0.36	0.55	0.44
Agg Clustering inception t-SNE	0.50	0.50	0.50	0.77	0.45	0.83	0.50	0.17	1.00	0.09	0.20	0.80	0.36	0.29	0.50
Agg Clustering inception UMAP	0.00	0.50	0.00	0.91	0.00	0.00	0.00	0.14	0.00	0.00	0.05	0.00	0.00	0.00	0.11
Agg Clustering Xception PCA	0.00	0.06	0.00	1.00	0.00	0.00	0.15	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.09
Agg Clustering Xception t-SNE	0.50	0.27	0.57	0.91	0.09	0.50	0.71	0.50	0.69	0.40	0.00	0.90	0.45	0.50	0.50
Agg Clustering Xception UMAP	0.00	0.00	0.00	0.00	0.00	0.13	0.00	0.00	0.00	0.50	0.00	0.00	0.00	0.08	0.05
BIRCH inception PCA	0.25	0.44	0.41	1.00	0.63	0.40	0.29	0.44	1.00	0.17	0.00	0.00	0.00	0.55	0.40
BIRCH inception t-SNE	0.50	0.50	0.50	0.77	0.45	0.83	0.50	0.17	1.00	0.09	0.20	0.80	0.36	0.29	0.50
BIRCH inception UMAP	0.43	0.00	0.00	0.00	0.18	0.00	0.44	0.75	0.00	0.38	0.05	0.00	0.00	0.09	0.17

Appendix H

Model	Rand Index	Adjusted RI	Mutual Information	Normalized MI	Adjusted MI	Balanced Accuracy	Homogeneity	Completeness	V-Score	Average Score
KMeans inception PCA	0.91	0.34	1.57	0.61	0.47	0.04	0.62	0.60	0.61	0.64
KMeans inception t-SNE	0.91	0.35	1.60	0.62	0.48	0.08	0.62	0.61	0.62	0.65
KMeans inception UMAP	0.92	0.35	1.61	0.62	0.48	0.18	0.62	0.61	0.62	0.67
KMeans Xception PCA	0.91	0.35	1.60	0.62	0.49	0.09	0.64	0.61	0.62	0.66
KMeans Xception t-SNE	0.91	0.32	1.54	0.59	0.44	0.03	0.59	0.59	0.59	0.62
KMeans Xception UMAP	0.92	0.34	1.54	0.59	0.43	0.06	0.59	0.59	0.59	0.63
Agg Clustering inception PCA	0.91	0.34	1.53	0.60	0.46	0.06	0.62	0.58	0.60	0.63
Agg Clustering inception t-SNE	0.92	0.33	1.62	0.62	0.48	0.08	0.62	0.61	0.62	0.65
Agg Clustering inception UMAP	0.91	0.30	1.53	0.59	0.44	0.02	0.60	0.58	0.59	0.62
Agg Clustering Xception PCA	0.90	0.32	1.57	0.61	0.48	0.06	0.63	0.60	0.61	0.64
Agg Clustering Xception t-SNE	0.91	0.35	1.57	0.61	0.47	0.03	0.62	0.60	0.61	0.64
Agg Clustering Xception UMAP	0.91	0.31	1.53	0.58	0.43	0.07	0.59	0.58	0.58	0.62
BIRCH inception PCA	0.90	0.32	1.57	0.61	0.48	0.06	0.63	0.60	0.61	0.64
BIRCH inception t-SNE	0.91	0.34	1.53	0.60	0.45	0.05	0.61	0.58	0.60	0.63
BIRCH inception UMAP	0.91	0.29	1.48	0.57	0.41	0.09	0.58	0.56	0.57	0.61
BIRCH Xception PCA	0.91	0.34	1.53	0.60	0.46	0.06	0.62	0.58	0.60	0.63
BIRCH Xception t-SNE	0.92	0.33	1.62	0.62	0.48	0.08	0.62	0.61	0.62	0.65
BIRCH Xception UMAP	0.91	0.31	1.56	0.60	0.46	0.00	0.61	0.59	0.60	0.63

Appendix I

Models	Rand Index	Adjusted RI	Mutual Information	Normalized MI	Adjusted MI	Balanced Accuracy	Homogeneity	Completeness	V-Score	Averag
DBSCAN inception PCA	0.066	0.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000	0.118
DBSCAN inception t-SNE	0.066	0.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000	0.118
DBSCAN inception UMAP	0.767	0.109	1.065	0.466	0.311	0.071	0.547	0.405	0.466	0.468
DBSCAN Xception PCA	0.066	0.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000	0.118
DBSCAN Xception t-SNE	0.066	0.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000	0.118
DBSCAN Xception UMAP	0.690	0.080	0.979	0.455	0.310	0.093	0.583	0.372	0.455	0.446
HDBSCAN inception PCA	0.636	0.075	0.559	0.299	0.232	0.014	0.505	0.213	0.299	0.315
HDBSCAN inception t-SNE	0.700	0.084	0.775	0.378	0.279	0.000	0.527	0.295	0.378	0.379
HDBSCAN inception UMAP	0.838	0.204	1.044	0.465	0.369	0.029	0.560	0.397	0.465	0.485
HDBSCAN Xception PCA	0.555	0.056	0.370	0.215	0.167	0.000	0.451	0.141	0.215	0.241
HDBSCAN Xception t-SNE	0.638	0.105	0.668	0.359	0.297	0.071	0.613	0.254	0.359	0.374
HDBSCAN Xception UMAP	0.838	0.204	1.086	0.476	0.367	0.021	0.561	0.413	0.476	0.494
OPTICS inception PCA	0.521	0.080	0.768	0.424	0.346	0.079	0.774	0.292	0.424	0.412
OPTICS inception t-SNE	0.663	0.109	0.950	0.468	0.366	0.007	0.665	0.361	0.468	0.451
OPTICS inception UMAP	0.791	0.146	1.092	0.485	0.363	0.057	0.583	0.415	0.485	0.491
OPTICS Xception PCA	0.375	0.020	0.433	0.256	0.157	0.036	0.576	0.165	0.256	0.253
OPTICS Xception t-SNE	0.661	0.112	1.008	0.492	0.382	0.000	0.686	0.384	0.492	0.469
OPTICS Xception UMAP	0.814	0.171	1.126	0.487	0.354	0.079	0.565	0.428	0.487	0.501
Mean Shift inception PCA	0.066	0.000	0.000	0.000	0.000	0.071	1.000	0.000	0.000	0.126
Mean Shift inception t-SNE	0.066	0.000	0.000	0.000	0.000	0.071	1.000	0.000	0.000	0.126
Mean Shift inception UMAP	0.523	0.065	0.365	0.221	0.195	0.043	0.538	0.139	0.221	0.257
Mean Shift Xception PCA	0.130	0.001	0.080	0.057	0.009	0.071	0.434	0.030	0.057	0.097
Mean Shift Xception t-SNE	0.066	0.000	0.000	0.000	0.000	0.071	1.000	0.000	0.000	0.126
Mean Shift Xception UMAP	0.693	0.129	0.612	0.329	0.288	0.050	0.559	0.233	0.329	0.358
Affinity Clustering inception PCA	0.905	0.352	1.568	0.619	0.490	0.186	0.643	0.597	0.619	0.664
Affinity Clustering inception t-SNE	0.887	0.299	1.347	0.561	0.456	0.079	0.620	0.513	0.561	0.591
Affinity Clustering inception UMAP	0.883	0.335	1.369	0.586	0.498	0.229	0.669	0.521	0.586	0.631
Affinity Clustering Xception PCA	0.932	0.434	1.831	0.679	0.538	0.021	0.663	0.697	0.679	0.719
Affinity Clustering Xception t-SNE	0.906	0.351	1.488	0.596	0.477	0.207	0.630	0.566	0.596	0.646
Affinity Clustering Xception UMAP	0.896	0.312	1.357	0.554	0.434	0.293	0.597	0.516	0.554	0.612