Machine Learning Engineer Nanodegree

Capstone Project

Valérian Wrobel April 5th, 2017

I. Definition

Project Overview

To foster the development of photovoltaic energy, methods that identify surfaces suitable for installations are crucial. When wanting to evaluate the potential of a given roof, one as to take several factors into account: the local climate, the shadows casted by surounding buildings or landscape and the characteristics of the roof such as the size, the shape and the orientation. Hence it is essential to get some data that allows the 3D representation of buildings. LIDAR data is very powerfull for that matter but is costly to aquire and is not available in all regions. To the contrary, satellite and aerial images as well as building footprints are widely available. In this project, we are interested in applying machine learning methods in order to determine the shape of a roof from the latter data.

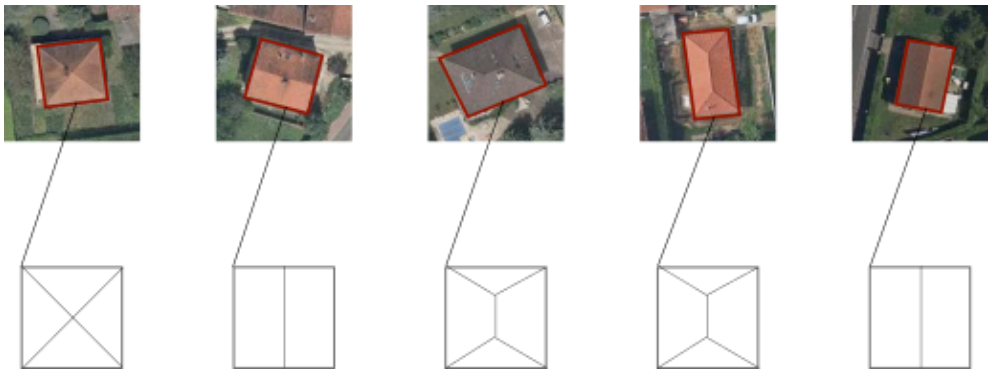


figure 1: the intended labeling of roofs

Currently, computer vision is one of the most prominent application of machine learning and encompasses various tasks from digit classification to face recognition. In recent years, various Deep Neural Networks architectures have been studied for supervised learning as well as for unsupervised learning applied to images. Since raw images are high dimensional datapoints with strong redundancy, performant feature (or representation) learning that allows dimensionality reduction is a key aspect for any computer vision problem. Since the data we are interested in do not offer any labeling of the roof shapes, we will focus on a deep neural network architecture fitted for unsupervised learning problems.

Generative adversarial networks (GAN) have been introduced by *Ian Goodfellow et al.* in 2014 and have since been used in a wide variety of unsupervised machine learning applications to computer vision. For example, in this article published in 2016, *Alec Radford et al.* describe a method for feature learning based on Deep convolutional generative adversarial networks. These methods do not aim specifically at the clustering of datapoints during feature learning. In this article entitled Unsupervised Deep Embedding for Clustering Analysis published in 2016, *Junyuan Xie et al.* present a model architecture - according to their own words - "that simultaneously learns feature representations and cluster assignments using deep neural networks". We will apply the Deep Embedding for Clustering (DEC) model with the aim of determining a roof segmentation by shapes.

Problem Statement

The metropole of Lyon in France promotes open data sharing. In that context, aerial images of the city are made available from a web map service as well as the footprints of the buildings available on a web feature service. Combining both data sources, we can get roof images for the buildings of a given area. For these unlabelled roof images, finding a segmentation requires some data preprocessing and a performant model for feature learning and clustering.

figure 2: the data from which we get roof images

We do not have some *a priori* knowledge of the expected segmentation of the roofs: we expect the model we chose to reveal different clusters (between 5 and 10) that each corresponds to a given roof shape - for example hip, pyramid, gable or flat. Since we are more interested in the shape of the roofs than the material they are made of, the preprocessing will aim at lowering differences between a hip roof made of slates or made of tiles. In order to build a solution to this sgmentation problem, we will proceed with the following steps:

1. Prepare the dataset:
    - Select the footprints of buildings we are interested in,
    - Prepare the roof images crossing the aerial images and the footprints,
    - Remove potential outliers from images,
    - Process the images so as to highlight the shape of the roofs,
    - Format the images to fit the input shape of the model.
2. Train the model:
    - Test the DEC model on a well-kwown and labelled dataset - MNIST,
    - Adapt the DEC model to our roofs dataset,
    - Measure the model performance for different parameters (e.g. the number of clusters),
    - Select and finetune the best model to obtain a *good* clustering of roofs.
3. Analyse the results:
    - Visualise the clusters of images defined by the model,
    - Measure the final performance of clustering with some metrics,
    - Interpret the results visually and decide on the usefulness of the model for roofs segmentation.

Metrics

Measuring performance of an unsupervised learning model on the roofs dataset can be tricky since there is no ground truth available. We can measure how well the model creates data segmentation independantly of any labelling with the silhouette coefficient. Nevertheless, our final goal is to obtain clusters that correspond to roof shapes: there is no way to check that a model can help with that task without labels. Therefore we will draw a 200 points sample in the dataset and label it manually after a basic clustering (described in the next part). We will then measure the accuracy of clustering relatively to these labels.

Silhouette score

The silhouette value measures how an object is similar to the cluster it is assigned to compared to other clusters. Let us consider a dataset segmented in k clusters. We call C the set of clusters. We define the dissimiliarity of a datapoint i to any cluster c, $d_c(i)$, as the average of the distance from i to all points in c. We will consider the euclidian distance for the dissimilarity calculation. The silhouette score of a datapoint i assigned to the cluster c is:

$$s(i) = \begin{cases} 1 - a(i)/b(i), & \text{if } a(i) < b(i) \\ 0, & \text{if } a(i) = b(i) \\ b(i)/a(i) - 1, & \text{if } a(i) > b(i) \end{cases}$$

where:

- $a(i) = d_c(i)$, is the dissimilarity of i to its cluster,
- $b(i) = \min_{c' \in C, c' \neq c} d_{c'}(i)$ is the lowest average dissimilarity of i to any other cluster.

From the above definition, we have $-1 \leq s(i) \leq 1$. If the silhouette value is close to 1, it means that $a(i) \ll b(i)$ and so that the clustering is clear for the datapoint i. $s(i)$ close to 0 means that the datapoint is at the border of two clusters. $s(i)$ close to -1 means that the clustering is inapropriate for this point.

Hence averaging the silhouette score over all datapoints for a given clustering model and a given number of clusters is a way to measure the quality of the clustering without needing ground truth labels. It can help to determine an optimal number of clusters for a given model.

Accuracy

For a labelled sample of the dataset, we define the accuracy score as:

$$ACC = \max_m \frac{\sum_{i=1}^{n} \mathbf{1}\{l_i = m(c_i)\}}{n}$$

where:

- l_i is the ground-truth label,
- c_i is the cluster assignment produced by the algorithm,
- m ranges over all possible surjections between clusters and labels.

We consider surjections so as to take into account the possible merging of clusters to better fit labelling. For example, if there are 5 possible labels and that we have a 8 clusters model, we evaluate the accuracy over all possible mergingfs of 8 clusters to 5 labels.

We will use the silhouette score as an indicator to choose the number of clusters for each model: the DEC and the benchmark model. However, we will use the accuracy measured for the sample set to select the best parameters for each model and to compare the performance of both models.

II. Analysis

Data Exploration

So as to create the dataset of roof images, we first focus on selecting only appropriate footprints. For this study, we aim at selecting only individual suburban houses and small buildings with rectangular shapes (more complex shapes would be harder to handle with rectangular images).

The full footprints dataset for the metropole of Lyon has 331.334 records.

From these, we remove the footprints in a 6km diameter disk centered on the city center so as to remove high residential buildings. We also remove buildings flagged as light constructions, keeping 200.930 records.

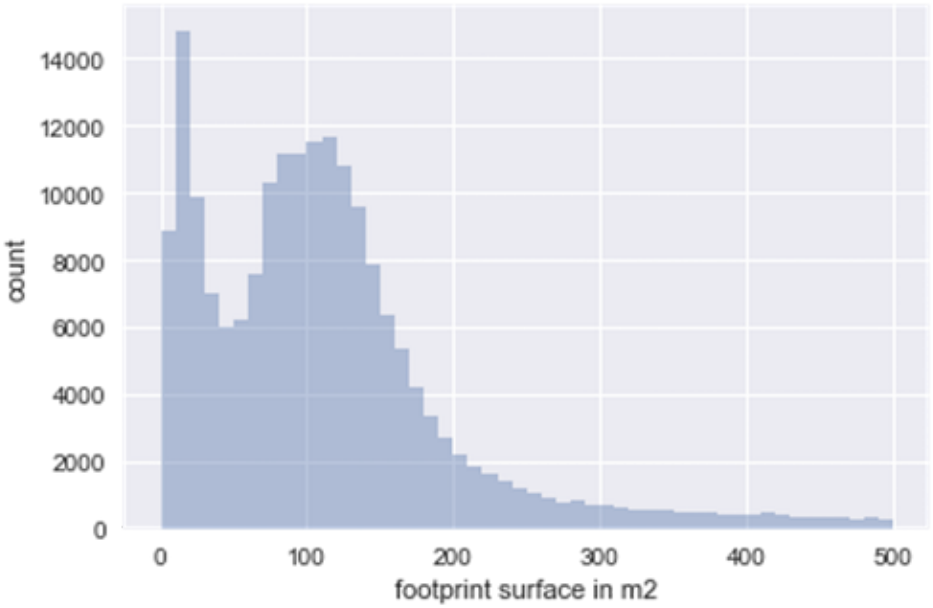


figure 3: histogram of footprints by surface in m2

So as to remove large buildings (like schools or churches) and small constructions (like pools), we only select the footprints that have a surface between 100m2 and 200m2, keeping 73.435 records.

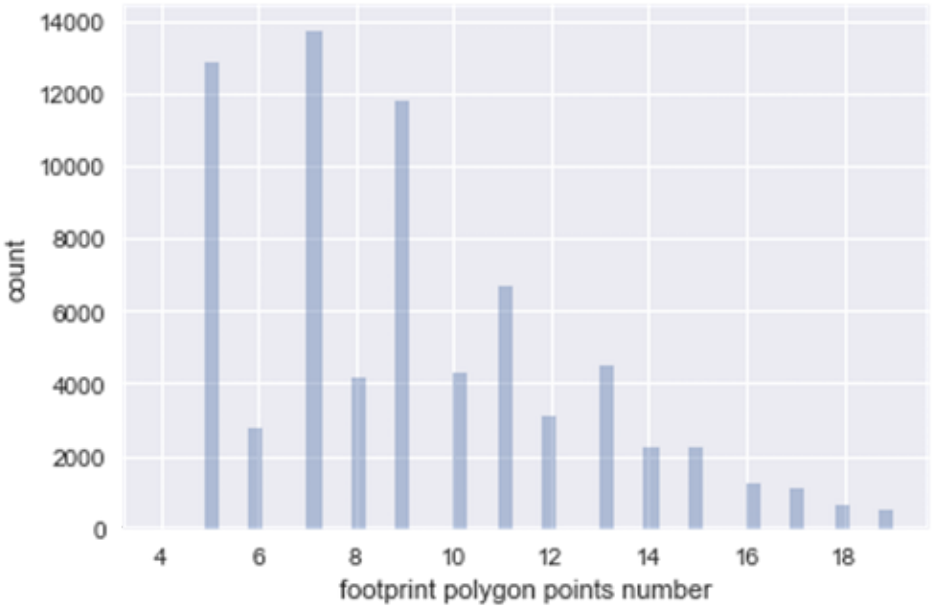


figure 4: histogram of footprints by number of points

Finally with select only polygons defined by 5 point (rectangles since two of the points are superposed). The final footprint dataset has 12.870 records.

Crossing the footprints with the aerial images and using an affine transform, we are able to get a roof image stretched to a square (64x64) for each footprint.

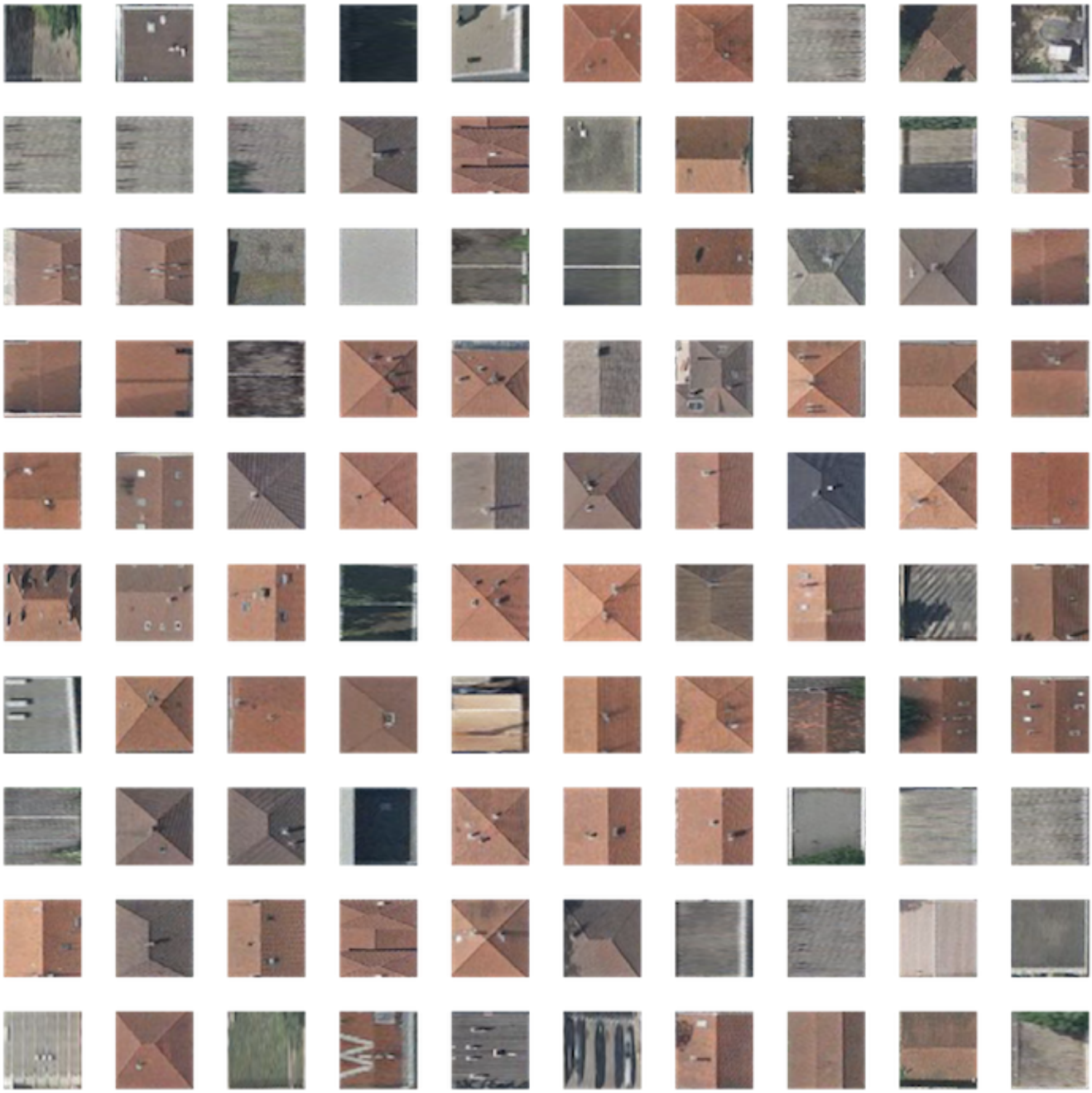


figure 5: a sample of 100 roof images

Then we analyse the set of images computing the mean Hue, Saturation and Value for each image.

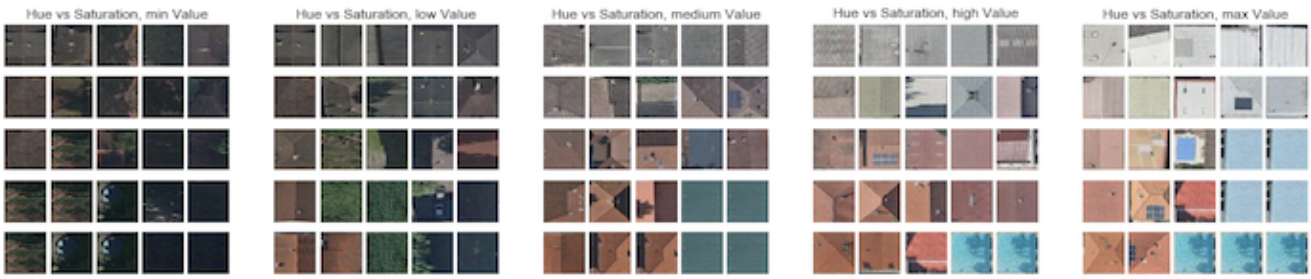


figure 6: the closest roof image for each point of a grid in the HSV color space

We decide to remove some images based on their HSV values because they do not correspond to roofs or are of bad quality:

- blue images - some correspond to pools since a pool is considered as a building in the footprints dataset,
- green images - some footprints are not well defined and we can get trees or grass instead of roofs,

- too dark or too light images because they often correspond to very shady roofs or very bright roofs for which it is hard to determine the shape.

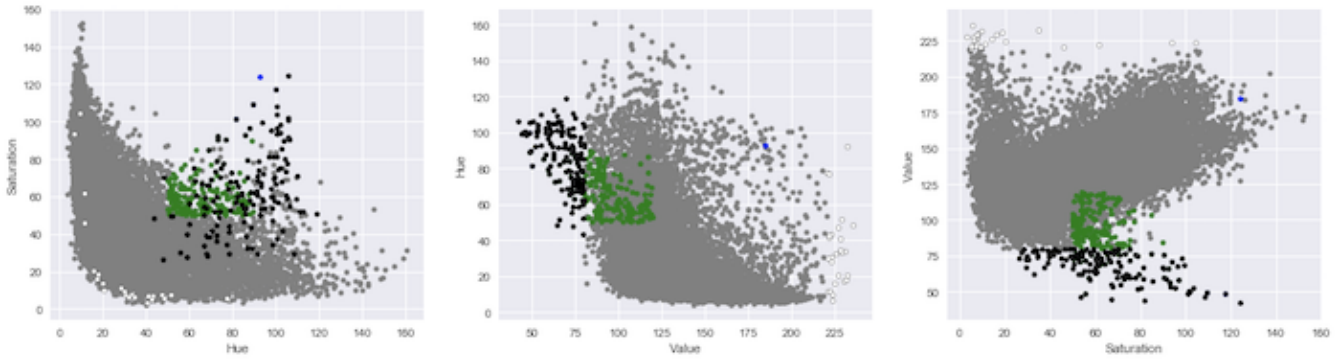


figure 7: the removed images highlighted on HSV plots of the dataset

The final dataset has 13.408 images of size (64,64) with 3 channels (RGB).

Exploratory Visualization

One key aspect of the roofs dataset is the symetry of images. For the following visualization, we use a processing on the images that will be described thoroughly in the next part. In procecessed dataset, images are on a gray scale and oriented such that the darker parts of the images are at the top and to the right.

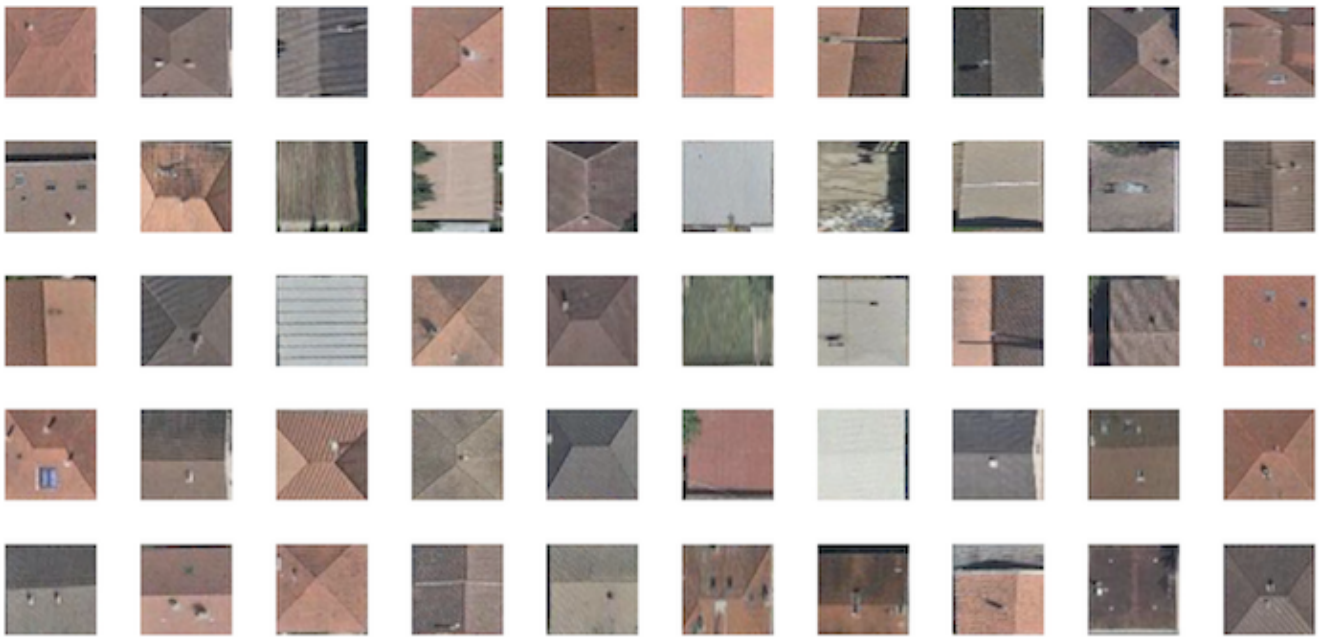


figure 8: a sample of the final roof images before processing

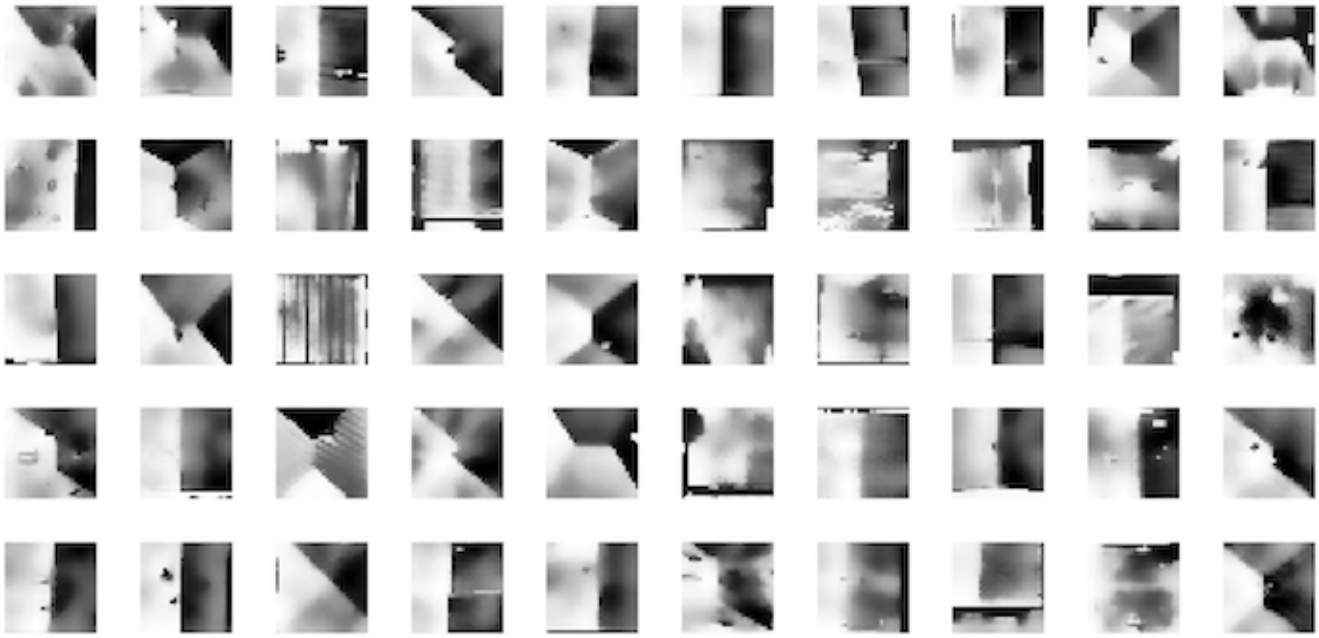


figure 9: the same sample of the final roof images after processing

For each processed image, we measure $Hsym$ and $Vsym$ coefficients defined as follows:

$$Vsym = 1 - \frac{\|L - R\|}{\|M\|}, Hsym = 1 - \frac{\|T - B\|}{\|M\|}$$

where:

- L is the left half of the grayscale image,
- R is the vertically flipped right half of the grayscale image,
- T is the top half of the grayscale image,
- B is the horizontally flipped bottom half of the grayscale image,
- M is a half image where each coefficient has the maximum value of 255,
- the norm is the L2 norm.

$Vsym = 1$ for images where the left and the right halves are symetric along the vertical axis and $Vsym = 0$ for images where the left and the right halves are extremely opposed (one half is black while the other is white).

$Hsym = 1$ for images where the top and the bottom halves are symetric along the horizontal axis and $Hsym = 0$ for images where the top and the bottom halves are extremely opposed (one half is black while the other is white).

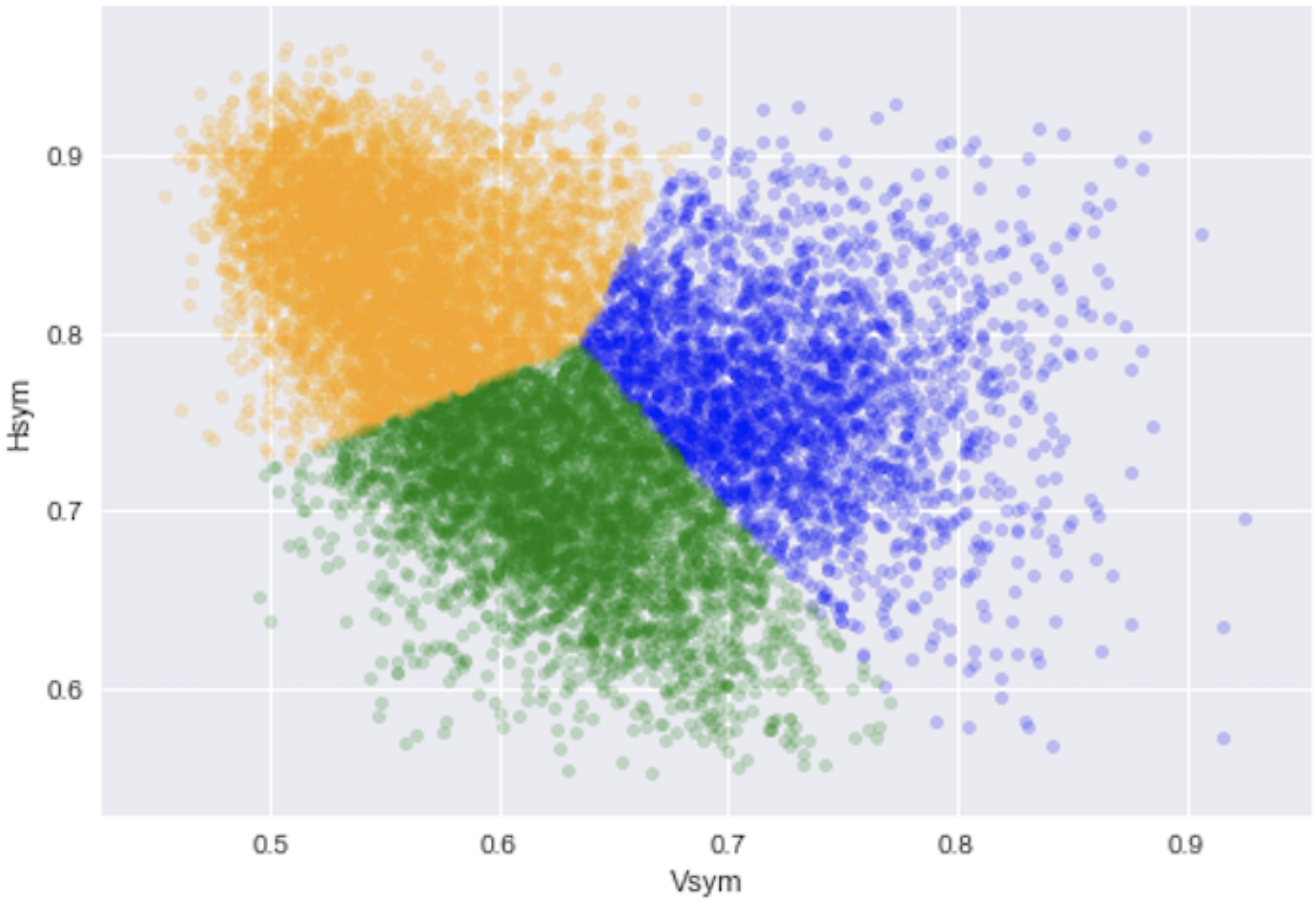
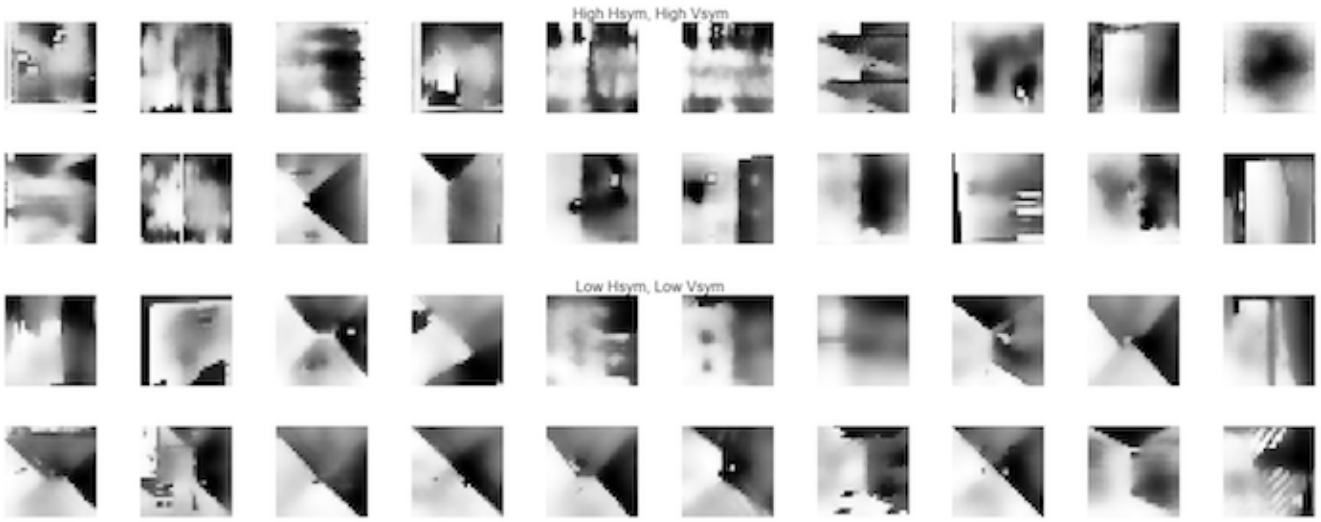


figure 10: a scatter plot of the dataset according to $Vsym$ and $Hsym$ coefficients

On the above plot, the dataset is segmented in 3 clusters with k-means algorithm:

- roofs for which the vertical and horizontal symetry are low - hip and pyramid roofs for example,
- roofs for which the vertical symetry is high while the horizontal symetry is low - gable roofs for example,
- roofs for which the vertical and horizontal symetry are high - flat roofs for example.

It is obvious on the scatter plot above that there is no clear separation of the clusters and that the assignments are not very clear around the borders.



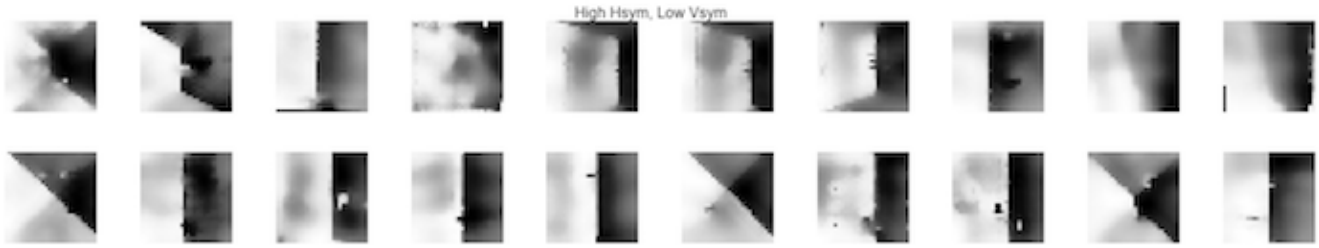


figure 11: a sample of roof images for each cluster

This study of the symetry of the dataset images allows some segmentation of the dataset with k-mean clustering. With the DEC model, we expect to find a finer clustering that will also leverage features that represent the symetry of images.

Labelling of a sample dataset

We use the above segmentation to initialize the labelling of a sample of 200 points randomly chosen in the dataset.

We consider five possible labels for a roof:

- flat,
- gable,
- hip or pyramid,
- complex,
- not a roof.

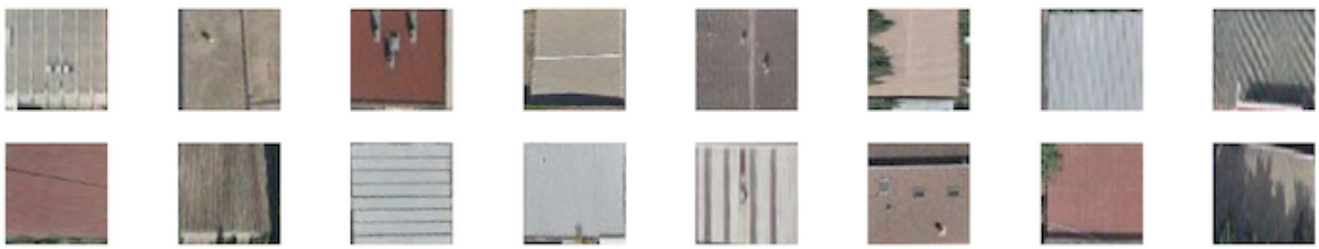
We initialize the labelling, the following way:

- low Hsym and Vsym: hip or pyramid,
- low Hsym and high Vsym: gable,
- high Hsym and Vsym: flat.

We manually adjust the labels with a visual control of the images. The result of the label counts for each label is the following:

shape	count
flat	27
gable	92
hip or pyramid	61
complex	16
not a roof	4
TOTAL	200

flat



gable

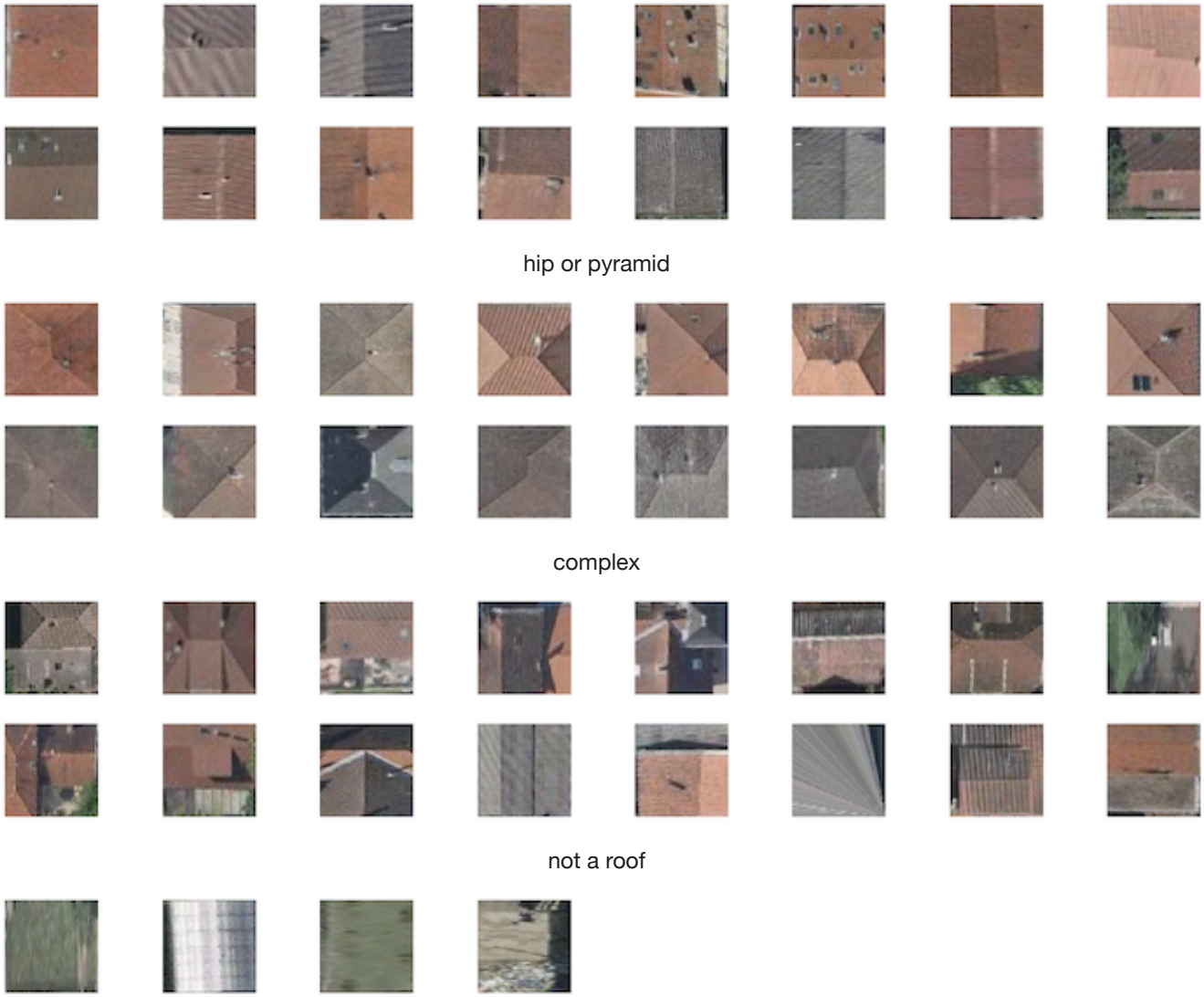


figure 12: a sample of roof images for each label

The dataset appears unbalanced between the chosen labels. Eventhough 200 is not a big sample, we will use this sample to calculate the accuracy of the DEC model and the benchmark model.

Algorithms and Techniques

The DEC model is thoroughly discussed in the *Junyuan Xie et al.* article, I will expose the main points here.

The DEC model aims at simultaneously learning a the centers of k clusters in a feature space Z as well as the mapping with parameter θ that goes from the input space X to the feature space Z.

DEC training has two phases:

1. parameter initialization:
    - we train a deep autoencoder in order to find a feature space with reduced dimension for the dataset - the input dataset of dimension 784 is reduced to dimension 10 with a bootleneck coding layer with 10 nodes in the middle of the autoencoder,
    - we initialize the clusters centers with k-means applied on the dataset in the feature space,
2. clustering by optimizing the parameters, for each iteration:
    - we compute the soft assignment of the point z_i in the feature space to the cluster j with center μ_j:

$$q_{ij} = \frac{(1 + \|z_i - \mu_j\|^2/\alpha)^{-\frac{\alpha+1}{2}}}{\sum_{j'}(1 + \|z_i - \mu_{j'}\|^2/\alpha)^{-\frac{\alpha+1}{2}}}$$

- we define an auxialliary distribution that strengthen predictions, put more emphasis on data points assigned with high confidence and normalize loss contribution of each centroid:

$$p_{ij} = \frac{q_{ij}^2 / f_j}{\sum_{j'} q_{ij'}^2 / f_{j'}}$$

where $f_j = \sum_i q_{ij}$ are soft cluster frequencies,

- we minimize the KL divergence loss L between q and p with stochastic gradient descent with momentum:

$$L = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}}$$

The gradients $\frac{\partial L}{\partial z_i}$ and $\frac{\partial L}{\partial \mu_j}$ are computed to update the parameters. The parameters of the Deep Neural Network are updated with backpropagation using $\frac{\partial L}{\partial z_i}$.

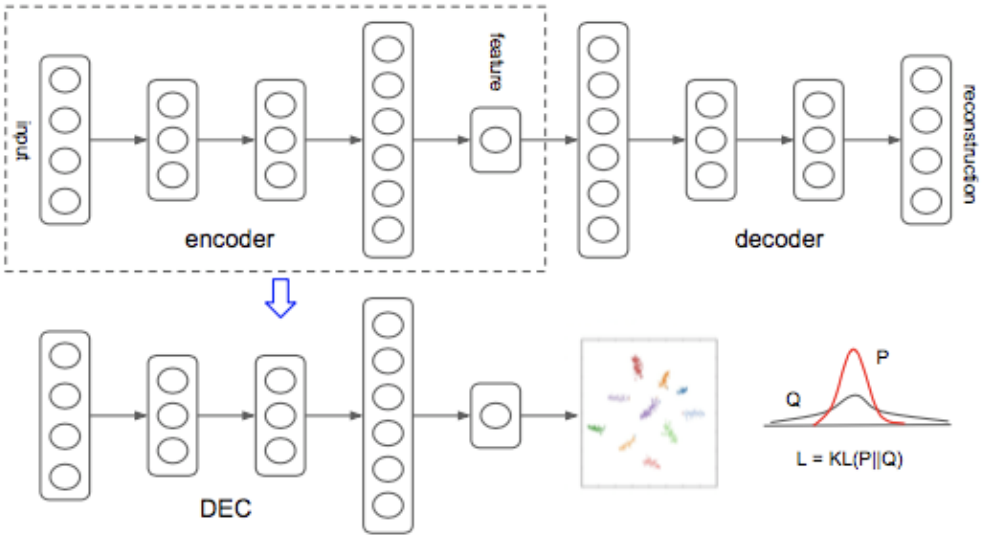


figure 13: the structure of the DEC model - source Junyuan Xie et al.

Benchmark

t-distributed stochastic neighbor embedding (t-SNE) is a very popular machine learning algorithm for dimensionality reduction of data and clustering developed by Geoffrey Hinton and Laurens van der Maaten. For the MNIST dataset for exemple, this algorithm allows an efficient segmentation (as can be seen in the middle of this webpage).

We will compare the clustering obtained with the DEC model to the following benchmark model:

- flattening of the processed image to an array of vectors - $(28 \times 28) \rightarrow (784)$,
- Principal Component Analysis (PCA) for dimensionality reduction - $(784) \rightarrow (25)$,
- t-SNE for dimensionality reduction to $(25) \rightarrow (3)$,
- k-means clustering followed by merging of clusters that gives the best accuracy $(3) \rightarrow (label)$.

We have used the default scikit-klearn perplexity of 30 for the t-SNE algorithm.

For reference the authors of the article claim an accuracy of 84.30% for clustering the DEC data. We have reached a 82% accuracy when testing a DEC implementation on MNIST data and X% with this benchmark model. For this reason, we think that it is relevant to compare both models for an image dataset with the same shape (28x28x1) but different characteristics.

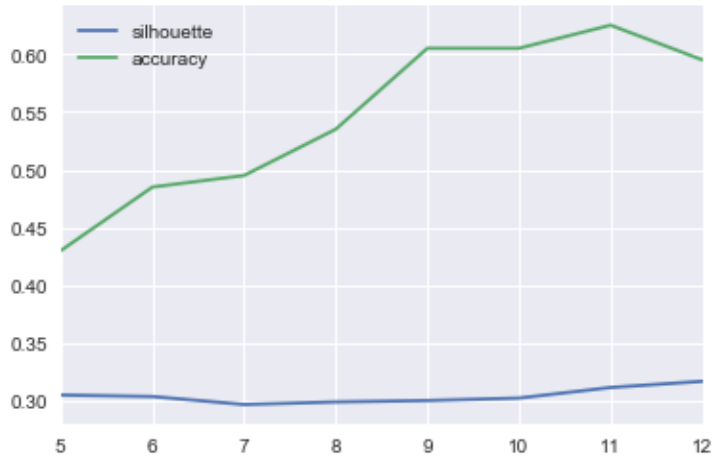


figure 14: accuracy of the benchmark model on MNIST dataset for different clusters number

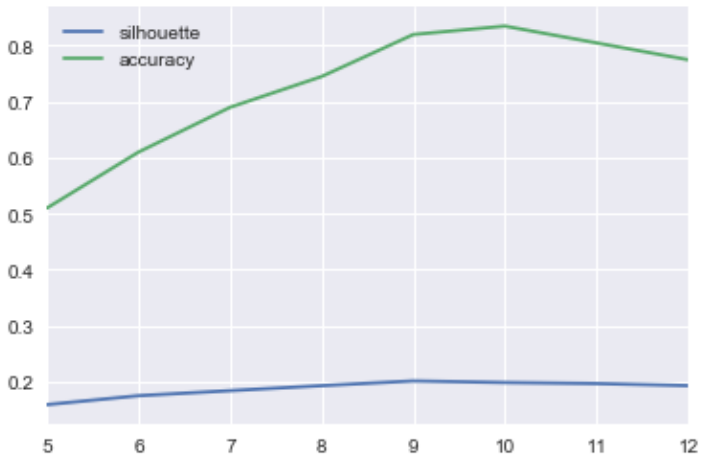


figure 15: accuracy of the DEC model on MNIST dataset for different clusters number

III. Methodology

Data Preprocessing

The original (64x64x3) images are not well suited for our aim of clustering the data by roof shape using the DEC model.

figure 16: a sample of the original images

First we remove some noise and keep the edges sharps using a bilateral filter. We then strengthen the edges by substracting a Gaussian blurred version of the resulting image. The aim of this process is to emphasize the elements that represent the shape (the edges) while reducing the ones that represent the material (the noise...).

figure 17: a sample of the sharpened images

Then we apply histogram normalization over all the images so as to have a strong contrast in all images.

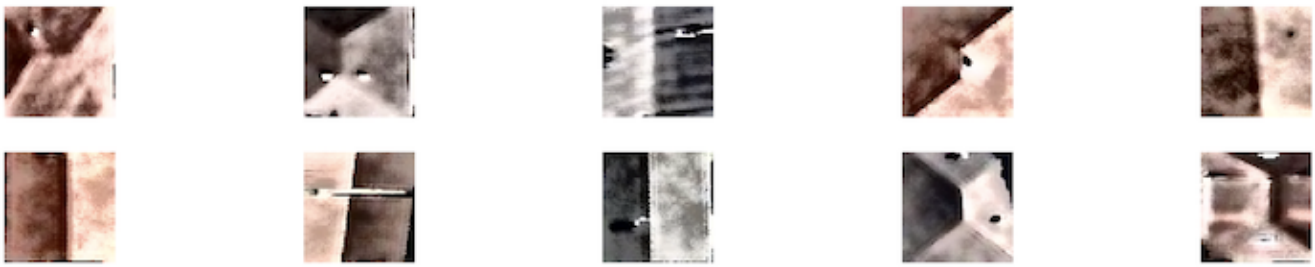


figure 18: a sample of the images after histogram normalization

After that, converting the images to grayscale and reducing the size allows a strong reduction of the dimensionnality without loosing much information regarding the shape of the roof. From (64, 64, 3) to (28, 28, 1) the dimension goes from 12.288 to 784.

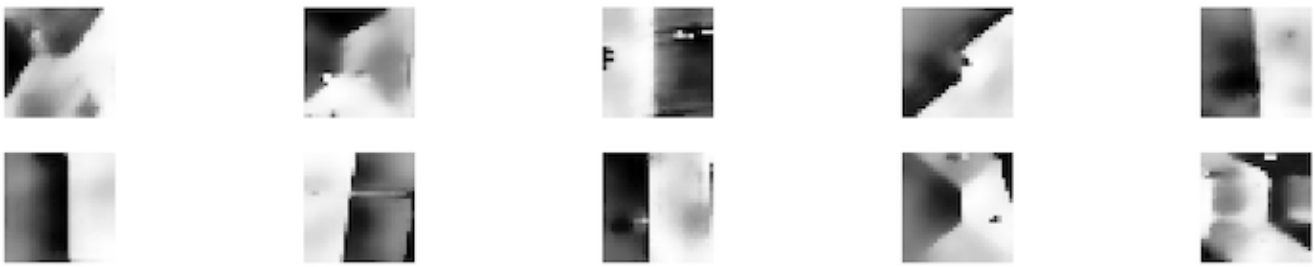


figure 19: a sample of the images after grayscale and size reduction

Finally, we rotate and flip the images so that the darker half of the image is to the right, and the darker half on the remaining axis is on the top.

figure 20: a sample of the images after orientation

The resulting images are of reasonnable dimensionnality and keep enough information for a human to recognize the shape of the roof. Furthermore, two roofs with different colors and orientations would have very close processed images if they have similar shapes.

Besides, before using the dataset as an input for the DEC model, we normalize all images so that for any data point x of the dataset, we have $\frac{1}{d}\|x\|_2^2$ close to 1 and we also have to flatten each image so that each input point is a 1-D array.

Implementation

The authors of the DEC article have based their publication on a Caffe implementation of their model. They have also shared a MXNET implementation that is easier to set up.

I have first tested the MXNET implementation and have switched to a Keras implementation (I find this framework easier to use). I have based my code on this Github repository. At first the results were not satisfactory (60% accuracy for the clustering of the MNIST dataset). I have discussed with the author of the code and we agree that the training of the autoencoder did not seem to be as performant as with the MXNET model. An other Github user has forked the initial projet and have improved the performance by adding layer-wise pre-training for the encoder and removing the dropout for the second phase. With this improved model slightly adapted, I have been able to get a clustering accuracy above to 84% on MNIST.

Then I have adapted the model to cluster the roof dataset with the following steps:

1. I have trained the autoencoder based on the dataset with the parameter values used in the paper for the autoencoder layers node numbers, the batch size, the adjustment of the learning rate,
2. Apply the KL divergence minimization phase for a range of number of clusters between 3 and 12,
3. Evaluate the metrics for each number of clusters.

I have noticed that for a given set of parameters, the results could be quite different depending on the random initialization.

At first, I intended to use the silhouette score to determine the best number of clusters but it appeared to be irrelevant for the task of clustering roofs.

Refinement

Since the silhouette scores was an insufficient metric to evaluate the performance of the model, I have labelled 200 datapoints to measure the accuracy and evaluate the model with different parameters as well as 100 additional datapoints to measure the accuracy on *unspoiled* data at the end of the improvement process to avoid the overfitting of the parameters to the first 200 datapoints.

With this labelled sample, I have been able to measure the clustering accuracy with merging as defined above for each tested set of parameters.

I have proceeded the following way to refine my model:

1. Initial model: the model with the parameters of the paper
    - Train the autoencoder for 5000 iterations during layerwise training and 10000 iterations during finetuning,
    - Evaluate the clustering accuracy for a range of number of clusters between 3 and 12 with a maximum number of iteration of 3000.

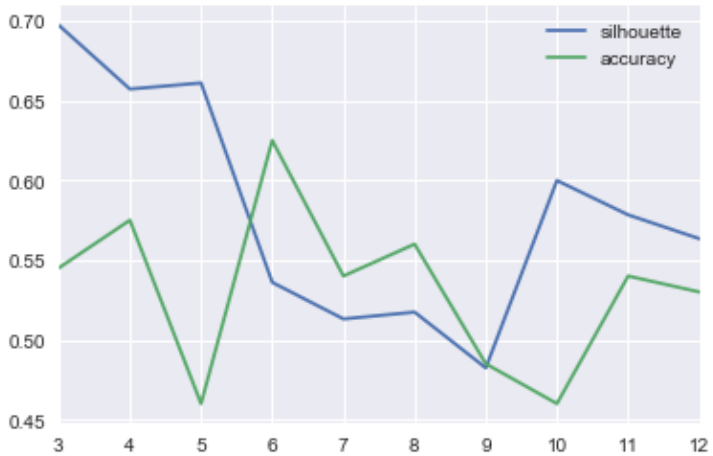


figure 21: the accuracy and silhouette score of the initial model for different number of clusters

We notice that the silhouette score does not follow the accuracy which is the metric that we are interested in. Furthermore, the accuracy with merging ranges between 46% and 62% for the different numbers of clusters with a maximum for 6 clusters.

2. Test of different parameters: I have tested the model for 6, 9 and 12 clusters on a grid of parameters. I have chosen to test different sizes of the bottleneck of the autoencoder [8, 10, 12] and values for the α parameter of the Student's t-distribution kernel in [0.1, 1, 10].

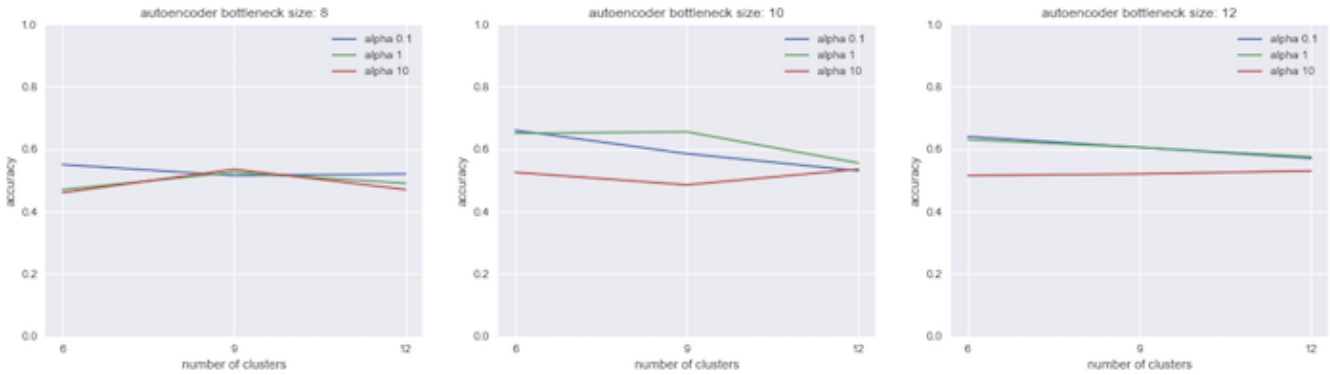


figure 22: the accuracy for different parameters

The best accuracy is obtained for 6 clusters with a bottleneck size of 10 and α equals 1 (the paper parameters). It is nevertheless quite difficult to determine a relationship between the accuracy and the parameters from the graph.

3. Final model: In the end my final model has the same paraeters as my initial model. I have trained it with more iterations and tested again the same range of numbers of clusters.
    - Train the autoencoder for 50000 iterations during layerwise training and 100000 iterations during finetuning,
    - Evaluate the clustering accuracy with a maximum number of iteration of 50000 or stop when the change of assignments is less than 1% between two iterations.

figure 23: the accuracy and silhouette score of the final model for different number of clusters

Eventhough the model is trained for longer, the accuracy is not better than for the initial model. The maximum of accuracy is reached for 12 clusters (62%) and then for 5 (61.5%). However, since these results can vary depending on the initial conditions, I have chosen to keep this final model with 6 clusters as it is the number of cluster for which I have obtained an accuracy of 62.5% (1% represents only 2 points in our sample dataset).

The final model does not seem to perform better than the initial model but it is quite hard to measure since we aim at measuring the performance of an unsupervided learning model with labelled data that the model has not access to!

IV. Results

Model Evaluation and Validation

Finally, we cluster the dataset with the final model for 6 clusters and stop when the label assignment changes by less than 0.5 % between two iterations. The accuracy on the 200 points labelled dataset is 63% and 60% on the 100 points labelled dataset.

Our choice for this model parameters and the number of clusters is based on the accuracy calculation on a 200 points labelled dataset. It appears that their is not clear overfitting during the refinement process since the clustering accuracy of the model for a different sample remains around 60%.

As can be seen on the accuracy plots above, the clustering accuracy (and therefore the clustering) can change for the same parameters and same number of clusters but different runs of the model: depending on the initialization, K-means algorithm

can provide different cluster centers at the first step. Eventhough, the resulting accuracy can vary by several percents between different runs, this problem is common to many unsupervised learning algorithm since there is no ground truth available during learning.

To get an feel of this, if have run the final model 5 times (stopping when label assigment change between two iterations was less than 2%) and got the following accuracies for 6 clusters: 50%, 50.5%, 51%, 61.5%, 53.5%. It is therefore necessary to run the model several times to get a satisfactory accuracy on the sample dataset.

Then, we are interested in knowing how the model behaves when it has fewer datapoints to cluster: could we find a usefull segmentation of roofs with less data? Hence, we have used a dataset of 2000 points including the 200 labelled datapoint and used the final model with different numbers of clusters. As can be seen on the graph bellow the accuracy drops to a level around 45% for all numbers of clusters: the number of datapoints seems to be important to train a DEC model for roof segmentation.

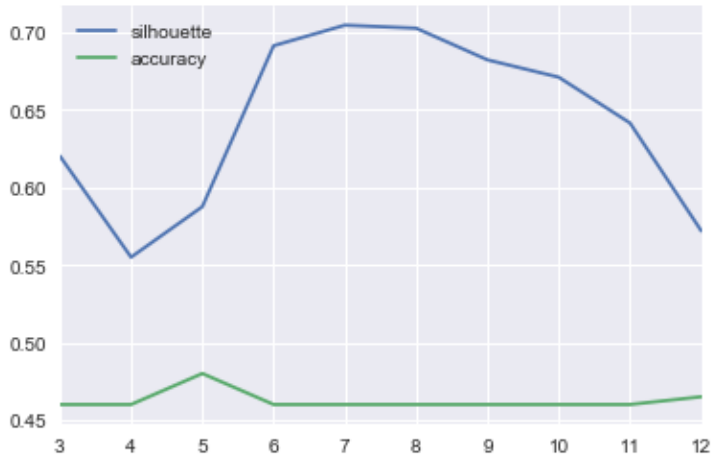


figure 24: the accuracy and silhouette score of the final model trained on 2000 datapoints for different number of clusters

From the discussion above, it appears that **the model does not reach a very high classification accuracy on the roofs dataset and that it may have to be run several times before providing satisfying clustering for the task of roofs segmentation.**

Justification

I appears that contrary to the MNIST dataset, the benchmark model outperform the DEC model for clustering the roofs dataset. Indeed, with 3 clusters merged to two, the t-SNE algorithm is able to reach an accuracy of 71% on the 200 points sample (and it is much faster).

figure 24: accuracy of the benchmark model on roofs dataset for different clusters number

To visualize how the two models cluster the data, let us remind the labels we have used:

index	shape
0	flat
1	gable
2	hip or pyramid
3	complex
4	not a roof

Finding the best correpondance between clusters and labels, we are able to build confusion matrices.

Confusion matrix for the DEC model on the 200 roofs sample labelled dataset (rows correspond to ground truth labels):

	0	1	2	3	4
0	10	12	5	0	0
1	11	72	9	0	0
2	0	17	44	0	0
3	4	8	4	0	0
4	0	1	3	0	0

Confusion matrix for the benchmark model on the 200 roofs sample labelled dataset (rows correspond to ground truth labels):

	0	1	2	3	4
0	0	25	2	0	0
1	0	86	6	0	0
2	0	5	56	0	0
3	0	11	5	0	0
4	0	3	1	0	0

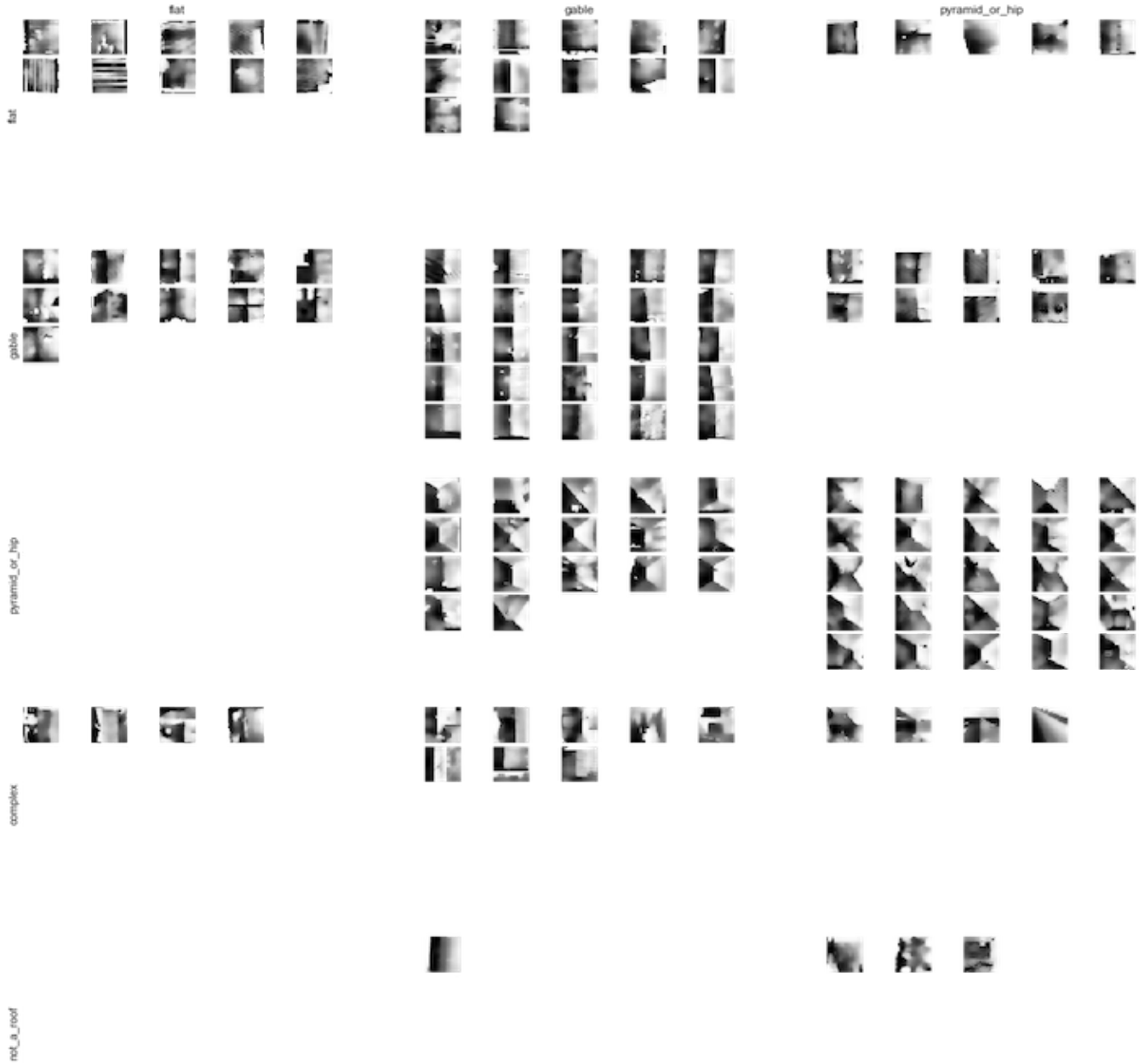


figure 26: confusion table of images for the DEC model on the labelled sample

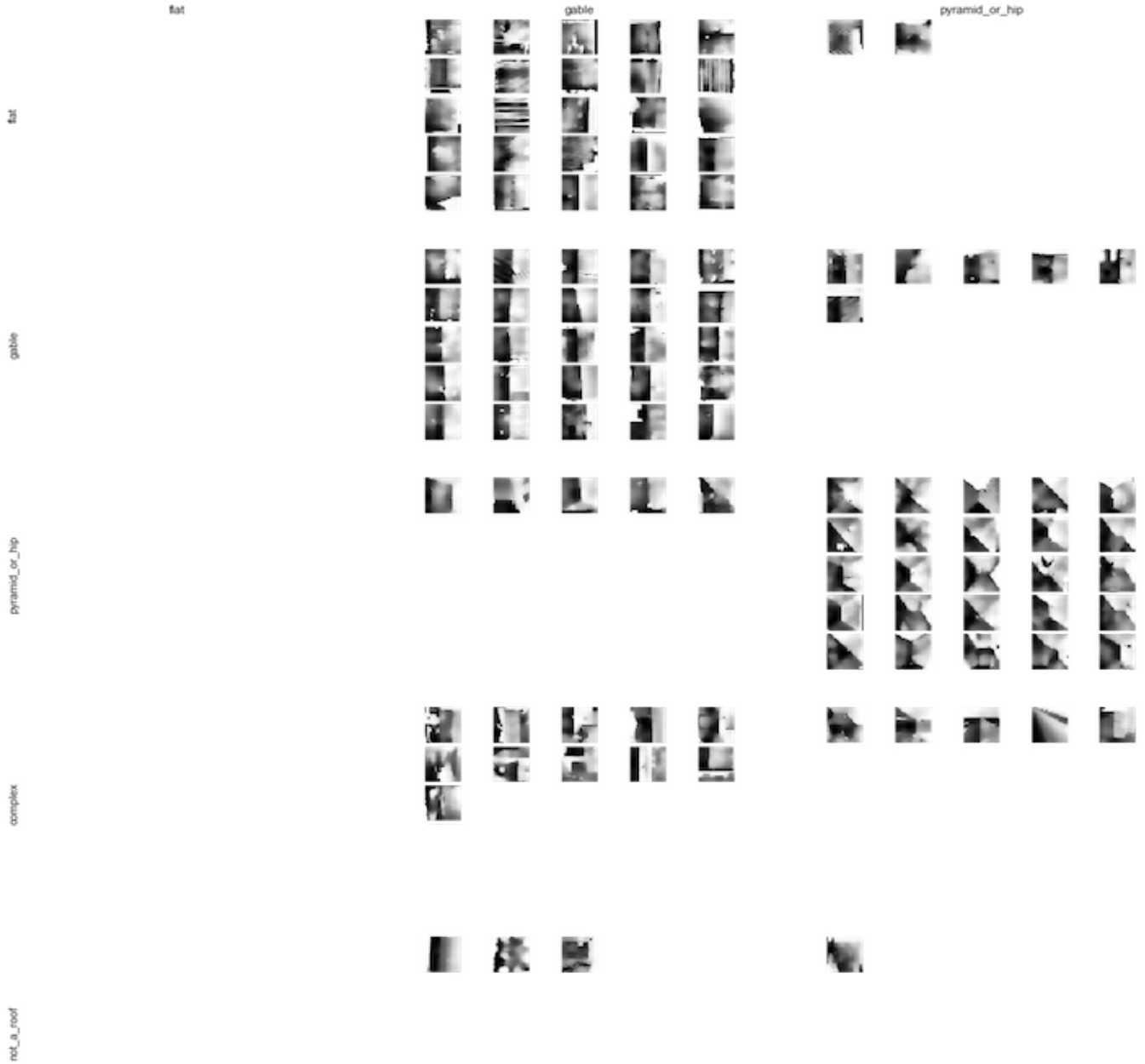


figure 27: confusion table of images for the benchmark model on the labelled sample

The roofs that we have labelled 'complex' and 'not a roof' cannot be assigned to any specific cluster, it seems clear that it is quite hard to discriminate these labels visually. Furthermore, the benchmark model does not take into account the 'flat' roofs and nevertheless has a better clustering accuracy. The more frequent labels, 'gable' and 'hip' roofs are the more important for the clustering. An important proportion of the 'hip' roofs are labelled as 'gable' by the DEC model.

We have calculated the classification accuracy on the 200 roofs sample for the following models:

- constant labelling as the most frequent type of roof - gable: 46%,
- K-means on the points represented with $Hsym$ and $Vsym$: 53%,
- benchmark model: 71%,
- DEC model: 63%.

63% is not a high enough score and is not as good as t-SNE. To be wrong 37% of the time for the final task we aim could lead to bad photovoltaic production estimations and it is not acceptable. Nevertheless, as we will highlight in the following, the soft assignment values available for the DEC model might help to spot roofs for which the labelling clear and those that raise questions.

V. Conclusion

Free-Form Visualization

Before merging, the DEC model provides 6 roof clusters:

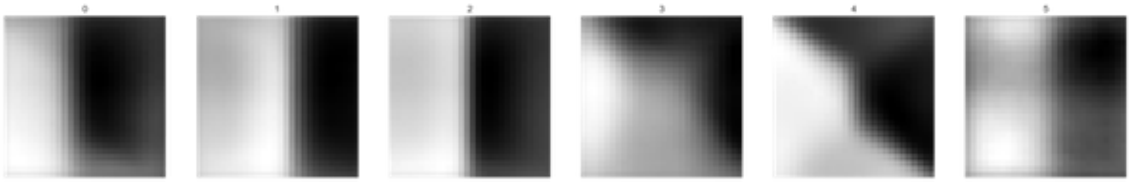


figure 28: the average roof image for each cluster before merging

The merging that gives the best accuracy over the labelled dataset in to have [0,1,2]: gable roof, [3,4]: hip roof, and [5]: flat roof.

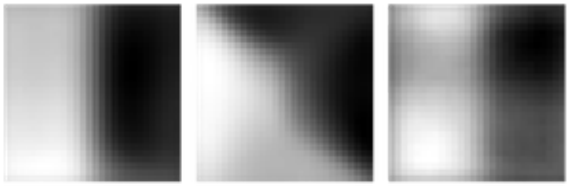


figure 29: the average roof image for each cluster after merging

As can be seen on the confusion table above, the DEC model with merging leads to misclassifying a strong proportion of hip roofs as gable roofs. Since the DEC provides soft assignments (probabilities between 0 and 1), it can be interesting to check the strength of the assignments for the hip roofs misclassified as gable.

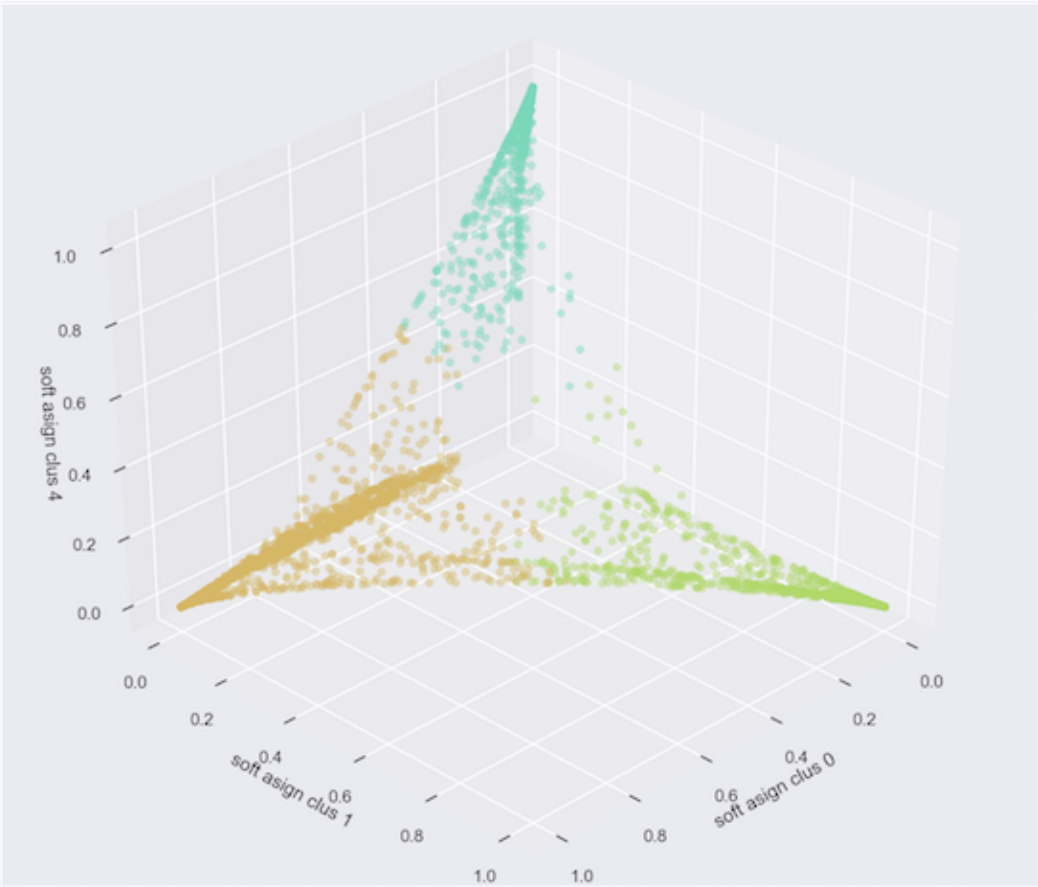
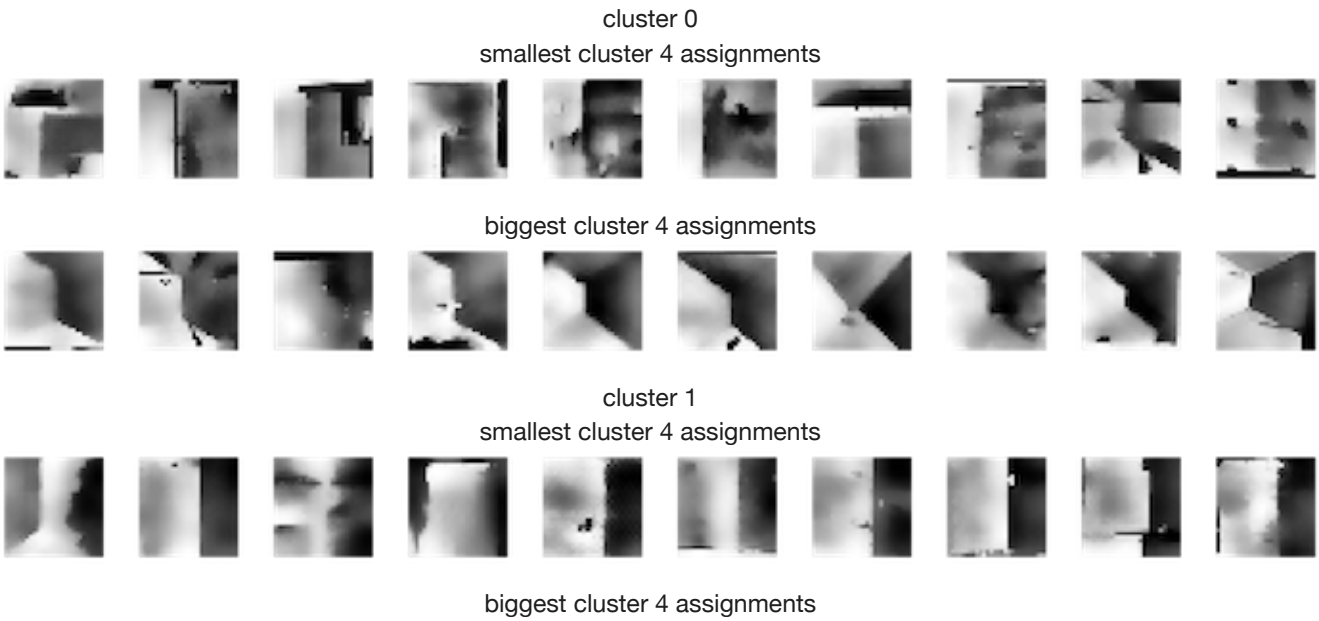


figure 30: a 3D representation of the soft assignments for clusters (before merging) 1, 2 and 4

We notice that eventhough the DEC model aims at strengthening the separation between clusters, we can still identify the points for which the assignments are unclear. We propose to visualize datapoints from clusters [0,1,2] (gable roofs) relatively to their assignment to cluster 4 (one of the two clusters corresponding to hip roofs). For each cluster, we first represent the 10 points with the smallest assignment value for cluster 4 and then the 10 points with the biggest assignment value.



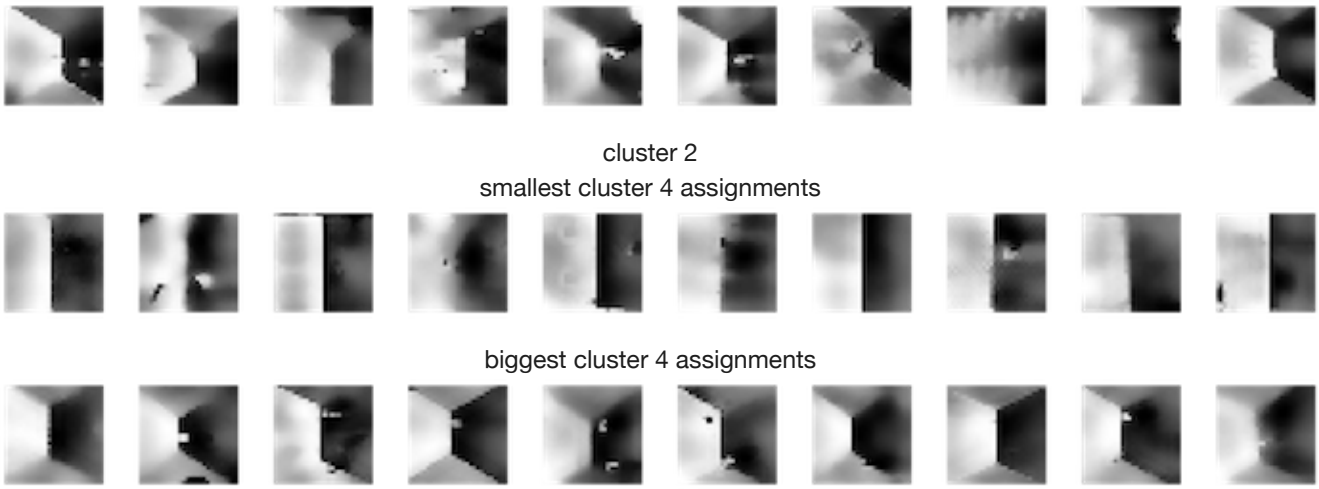


figure 31: 10 samples of each 'gable' cluster for minimum and maximum assignments to a 'hip cluster

It is clear that a lot of the roofs labelled as 'gable' with the largest assignments to cluster 4 are in fact hip roofs wheras it is not the case when the assignments to cluster 4 are the smallest. **To help with the task of manually labelling a dataset, the values of the soft assignments can be a key element to consider.**

Reflection

Starting from aerial images and building footprints for which a human eye could recognize some patterns, we aimed at a machine learning solution that would help us with a segmentation of roofs according to some identified types. The data preparation phase has been very important first to create a dataset of square roof images and then to process these images so as to reduce the dimension while strengthening the information regarding the shape of the roofs.

Since the DEC model has prooved to be efficient at clustering the MNIST dataset, we wanted to train in on the roofs dataset. One major difficulty that arised was to measure the accuracy of the model for different numbers of clusters without any ground truth available. Hence we have labelled a sample dataset and used it with a custom metric, we have had to take into account the possible merging of clusters since datapoints with different characteristics could share the same labelling.

With this new metric, we have been able to choose the number of clusters that performed the best segmentation for our purpose. Eventhough the clustering accuracy is not satisfying, we have noticed that the use of soft assignment values could help us to identify the most problematic datapoints.

What was quite interesting with this project was to use unsupervised learning techniques for labelling images for which supervised models like convolutionnal neural networks would perform better but require labelled data. I think that the DEC model can be very usefull to help in the task of labelling testing data before training a CNN model. I also believe that the quality of the initial data could be improved in order to get a better clustering. Indeed, the roofs images are often truncated because there is a shift between the footprints and aerial images in the first place.

Improvement

A large amount of the roof images of the created dataset are not correctly framed because the footprint layer and the aerial images are not perfectly well aligned. To improve the quality of the dataset, we could train a supervised learning model that learns to identify an adjusted footprint from the original footprint and the images. For this, we would need to define manually the adjusted corners for a sample of images and use these corrected footprints as training data for our model.

figure 32: the four original footprint corners in red, three adjusted corners in blue

A Convolutional Neural Network would be suited for this task. With the adjusted footprints, we could improve the quality of the roof images dataset and hence the quality of clustering for labelling. If the DEC model can be useful to define a reference labelling of the dataset, a supervised learning model could perform better at labelling new datapoints. Therefore, we could use the DEC model to create a labelled dataset of roofs (including soft assignment analysis) and use this a training dataset for a classifier based on a Deep Convolutional Neuron Network.

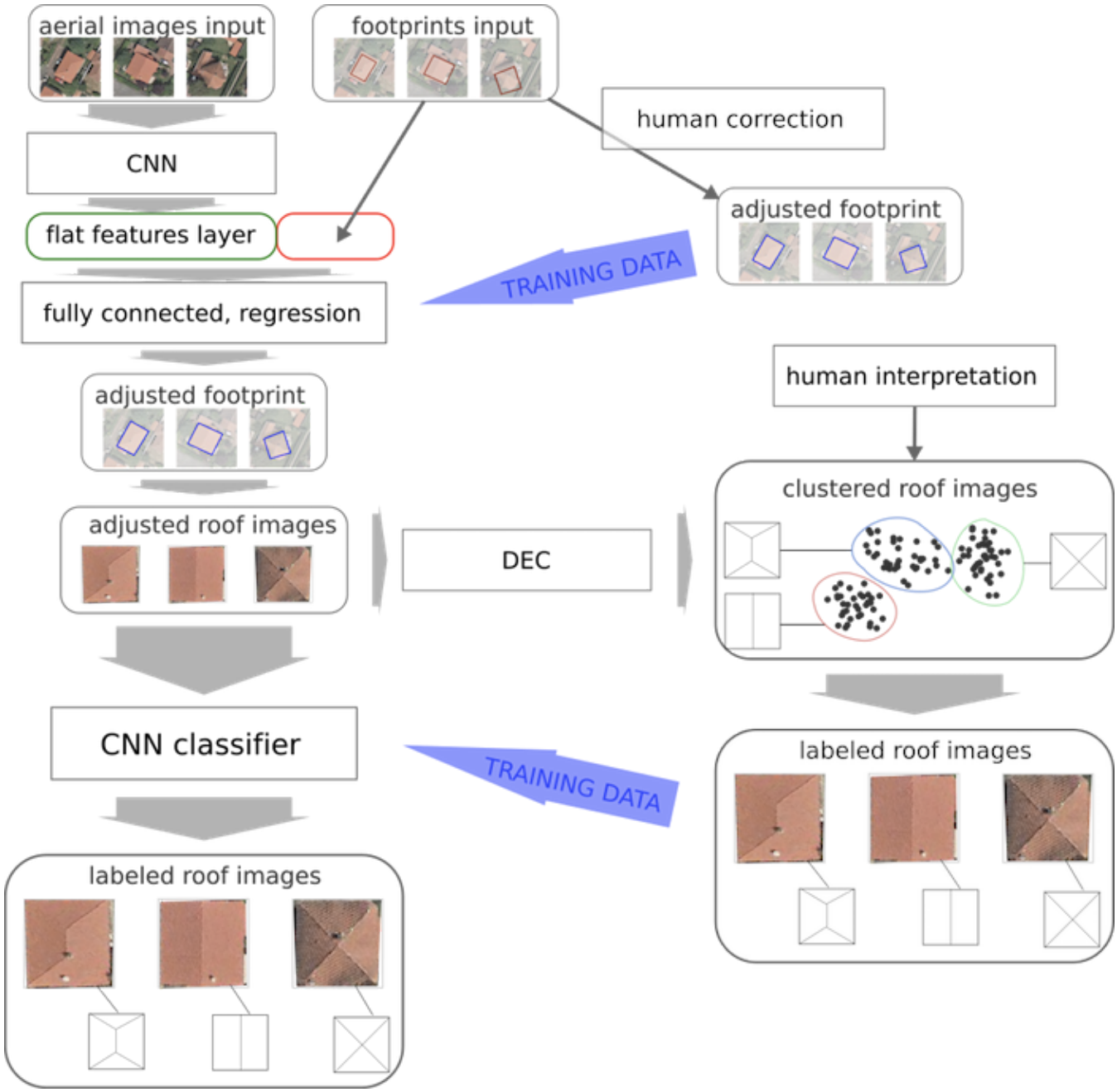


figure 33: the proposed improvement of the roof labelling process