IFT6759 Horoma Block 2

Lawrence Abdulnour
Université de Montréal
1089139
lawrence.abdulnour@umontreal.ca

Emilio Botero
Université de Montréal
20120412
emilio.botero@umontreal.ca

1 Introduction

From detecting illegal deforestation and natural disaster response to sustainable land-use and habitat conservation, harnessing deep learning to monitor and better understand our biodiversity is critical. In doing so, one of Horoma's mission is to wider AI development in forestry, by offering solutions such as forest inventory, disaster area segmentation and health and infestation survey. The Horoma project represents one step forward towards these initiatives.

This project is an ambitious task aiming to extract properties from forest canopy satellite images of the Outaouais. More specifically, we are interested in predicting tree species using unsupervised or semi-supervised machine learning techniques, with the challenge of having little labelled data. A first approach towards understanding the specie populations of these forests is to build a baseline model as a reference point for the continuation of this project. The baseline algorithm consists of segmenting our dataset using the k-means clustering algorithm. As we are dealing with high dimensional data, mapping data points to a lower dimensional space is needed. Hence, principal component analysis (PCA) was privileged as a dimensionality reduction method before applying k-means. The next step of this project was to explore advanced clustering algorithms leveraging deep learning methods. In the following sections, we present a way to obtain a reduced latent representation of our data using a convolutional auto-encoder (CAE) and a training approach, alternating between the optimization of a CAE and a clustering module.

2 Methodology

In this block we trained a CAE based on two different encoder architectures and a fixed decoder architecture. At the end of each epoch, clustering with either k-means or Gaussian Mixture models was performed with a variable number of clusters on the embeddings computed by the encoder architectures. A mapping from the clusters to the labels was done based on majority voting: given each cluster, we assign to it the most frequent label occurring in samples from half of the validation set samples assigned to that cluster. Classification accuracy and F1-score are then computed on the remaining half of the validation set samples. The k-means models were trained for 1000 iterations using 3 random restarts, whereas due to time constraints the GMM models were trained once for 100 iterations using a full covariance matrix.

All models were trained on NVIDIA K-80/K-20 GPUs on Compute Canada's Helios cluster. Moreover, models were optimized using the Adam optimizer with a learning rate of 0.001 and batch size of 256.

2.1 Dataset Description

The horoma dataset consists of 3 Digital Surface Models (or images) of forest areas in a Western Quebec region. Human interpreters labeled subsections of an image and 32 x 32 x 3 pixel patches of type RGB were extracted from these image subsections. This data extraction resulted in a training, valid and test set of respectively 150,900, 480 and 498 pixels patches and 17 unique tree specie labels. Pixel patches coming from the same subsection present the same overall properties and therefore offer the same general pixel information. This stresses the need to separate the valid set in 2 subsets in our experiments, in order to avoid overfitting on the validation set. Indeed, the valid set requires a subset to assign clusters to the labels and another to compute performance metrics, making sure that pixels corresponding to the same subset belong in the same subdivision. We thus divide our validation set in 2, leaving 240 samples to evaluate the model at every epoch.

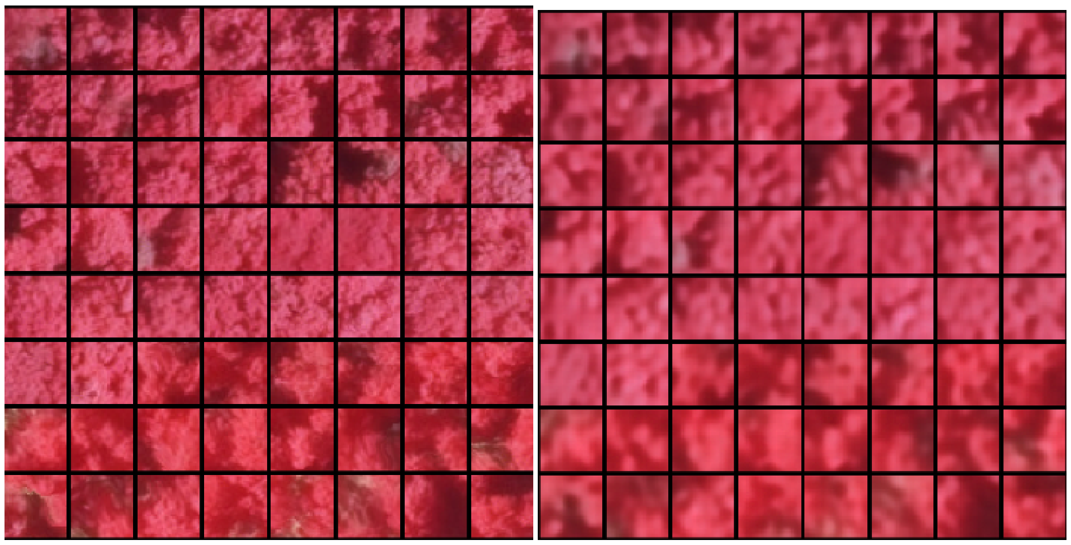
2.2 Experiments

2.2.1 Baseline Model - Block 1

As mentioned previously, PCA is used in the baseline model to get a lower dimensional embedding of the Horoma dataset. Executing PCA such that 90% of the variance of data points is explained, leads to each sample point having a dimension of 3072 to a dimension of 50, the number of principal components. Thus, without too much of a loss of information, the k-means algorithm is then applied using 50 clusters. With these hyperparameters, we obtain an accuracy on the valid set of 36.91% and an F1-score of 33.89%.

2.2.2 Refined Model - Block 2

As mentioned previously, we now obtain a new embedding and a lower feature space by training a convolutional auto-encoder (CAE). Much like fully connected auto-encoders (AEs), CAEs are typically based on the encoder-decoder paradigm, where the encoder maps the inputs to a hidden representation and the decoder reconstructs the inputs from that same hidden representation, as shown in the figure below (more images of our valid embeddings attained by our final CAE model can be examined in appendix A). Figure 1 shows samples of pixel patches from the validation set and their reconstructions obtained by the CAE. The reconstruction manages to recover the overall shape of the patches, but are much less detailed and do not have the fine-grained structure the original patches do.



(a) Original Pixel Patches

(b) Reconstructed Pixel Patches

Figure 1: CAE: before and after reconstruction

However, as opposed to AEs, convolutional auto-encoders preserve spatial locality [3], a quality well-suited for inputs such as images. They are trained by passing through the encoder and decoder layers and thereafter, by computing the standard backpropagation optimization. The labels of the CAE are the initial inputs (images) themselves and we optimize the reconstruction loss by minimizing the mean squared error (MSE):

$$L(\theta) = \frac{1}{n}\sum_{i=1}^{n}(\mathbf{x}_i - \mathbf{y}_i)^2$$

We trained two different architectures for the encoder, detailed in tables 1 and 2 : a basic one based on two sets of convolutional layers, followed by batch normalization, max pooling and ReLU non-linearities and ResNet-18. The decoder architecture, as shown in table 3, was composed of a series of transposed convolution layers followed by batch normalization and ReLU, expanding inputs until we obtain the same output size as the image's. After training the CAE with this setting, we tried the ResNet-18, as it is a deep and expressive structure. Note that both encoders include a max-pooling layer, which helps to introduce sparsity over the encoded input [3]. As for the clustering component of our algorithm, both k-means and Gaussian mixture models (GMM) were considered. k-means with the simpler CAE architecture and 60 clusters is the final model chosen, as it resulted to the highest f1 score (and is less computationally expensive to train). It is important to mention that the f1 score was favoured, due to data imbalance between the different tree species.

Layer	**Kernel**	**Filters**	**Stride**	**Padding**	**Output Size**
Conv/BN/Pool/ReLU	3x3	64	1	0	64x10x10
Conv/BN/Pool/ReLU	3x3	256	1	0	128x2x2
Fully Connected Layer	-	-	-	-	64*

Table 1: Encoder of Best Model Architecture (cnn)

Layer name	**Output size**	**Configuration**
conv1	32x32x64	7x7, 64, stride 1, padding 3
conv2_x	16x16x64	3x3 max pool, stride 2
conv2_x	8x8x128	$\begin{bmatrix} 3x3, 64 \\ 3x3, 64 \end{bmatrix}$x2
conv3_x	4x4x256	$\begin{bmatrix} 3x3, 128 \\ 3x3, 256 \end{bmatrix}$x2
conv4_x	2x2x64	$\begin{bmatrix} 3x3, 256 \\ 3x3, 64 \end{bmatrix}$x2
	1x1x64	average pool, 1-d

Table 2: Encoder of ResNet18 Architecture

Layer	**Kernel**	**Filters**	**Stride**	**Padding**	**Output Size**
TranspConv/BN/ReLU	3x3	256	2	0	256x3x3
TranspConv/BN/ReLU	3x3	128	2	0	128x7x7
TranspConv/BN/ReLU	3x3	64	2	0	64x15x15
TranspConv/BN/ReLU	3x3	32	2	0	32x32x32
MixConv/BN/ReLU	3x3	3	2	1	32x32x3

Table 3: Decoder architecture

A first intuitive consideration would be to train sequentially the CAE and the clustering module, similarly to what was achieved for the baseline model in block 1. However, this procedure might not be best suited for our objective, as we are not necessarily producing k-means-friendly spaces [5]. The last statement can be explained by the fact that the CAE is trained to optimize the reconstruction error and not the cluster's efficiency. A better suited approach would be to simultaneously train the

CAE and the clustering module. Therefore, our training scheme is inspired by [2], where at each epoch, we alternate between training the CAE and k-means. Specifically, throughout one epoch we train the CAE, at the end of which we cluster the embeddings.

It is therefore possible to perform 'early stopping' on the CAE based on the performance of the cluster, but not on the MSE loss, reiterating that our objective is primarily to have good clusters and not a low reconstruction loss. We ran all the experiments for 15 epochs with different values of clusters, as presented in the subsequent section. Finally, we select our final model at the epoch offering the best f1-score.

3 Results and Discussion

Following the methodology described in the previous section, we compare the accuracy and the f1-score using 2 different convolutional auto-encoders, 2 clustering methods and several numbers of clusters (17, 60 and 100). The results shown in Table 4 and 5 present the metrics obtained by evaluating on the validation subset, while table 6 compares our best model with the baseline model.

Table 4: Best results using k-means for 15 epochs

Hyperparameter Settings	Accuracy	F1-score
K-means-resnet-17	0.3296	0.2351
K-means-resnet-60	0.3515	0.3330
K-means-resnet-100	0.3227	0.3168
K-means-cnn-17	0.3051	0.2267
K-means-cnn-60	**0.3640**	**0.3467**
K-means-cnn-100	0.3237	0.3267

*Best model shown in bold

Table 5: Best Results using GMM for 15 epochs

Hyperparameter Settings	Accuracy	F1-score
gmm-resnet-17	0.3586	0.2832
gmm-resnet-60	0.3111	0.3033
gmm-resnet-100	0.3231	0.3069
gmm-cnn-17	0.3661	0.2769
gmm-cnn-60	0.3419	0.3086
gmm-cnn-100	0.3468	0.3436

Tables 4 and 5 show how the differences between each model are relatively small. All of the models achieve accuracy values above 30%, but the f1-score is sometimes significantly lower, as seen in the first row of table 4. Interestingly, all models trained with the number of clusters set to the number of class labels scored below 30% in the f1-score, while all those set to more clusters scored above 30%. This means that by choosing a higher number of clusters and performing majority voting, we were able to retrieve predictions for imbalanced classes. With that being said, the margin between these models is so small that statistical differences may be more difficult to confirm.

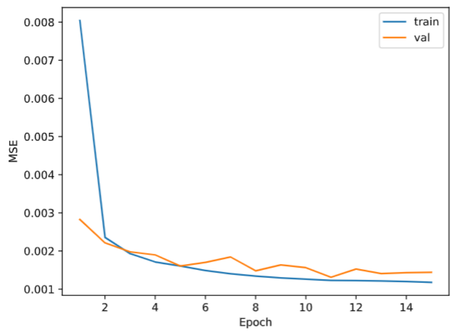
Additionally, we notice that the best model was achieved with a very simple encoder architecture, instead of the more complex ResNet-18, contrary to what we might have expected. This could be due to the fact that the ResNet model manages to extract embeddings that are better for reconstruction, but this comes at the expense of performance in accuracy.

On the other hand, the clustering methods seem to be very similar in terms of accuracy, but the k-means clustering method achieves better F1-scores overall. Furthermore, it is curious to note that the GMM models compete with k-means models, even despite the fact that they are trained for a smaller number of iterations. This hints that results could improve by training the GMM models for a longer amount of time.

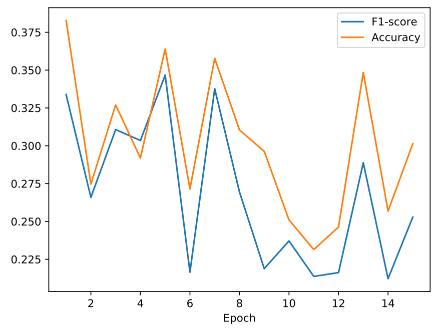
Table 6: Results of Baseline vs Best Model on the Validation Set

Algoritm	Accuracy	F1-score
Baseline	0.3691	0.3389
K-means-cnn-60	**0.3640**	**0.3467**

Figure 2a shows the training and validation curves of the MSE loss functions. The model achieves low values rather quickly, and the reconstructions, as shown in figures 1 and 4 are quite good in terms of overall structure. In that sense, it is clear that the CAE is performing well on its objective function. However, figure 2b shows that this performance does not necessarily translate to the evaluation metrics we are actually interested in. Figure 2b shows a general downward trend in the evaluation metrics. This is due to the fact that the network's weights, both in the encoder and decoder, are being trained for reconstruction and not for classification. In that sense, as the network gets better at reconstruction, the embeddings obtained by the encoder are not necessarily optimal for clustering. Because of this, a better approach would be to optimize explicitly the network's weights to achieve better accuracy when clustering the encoder's embeddings.



(a) Training and validation losses



(b) Evaluation metrics

Figure 2: CAE: MSE objective compared to accuracy and F1-score during training

A useful tool to visualize embeddings and observe some structure that could be promising for clustering is t-SNE. t-SNE maps non-linear representations of embeddings in a 2 or 3 dimensional space, in order to visualize high-dimensional data. We notice in figure 3 that our data presents some kind of skeleton that is promising for using a clustering algorithm. For example, we see some groups of classes such as the points belonging to labels PT, TO, PU and ES, meaning that k-means might be able to capture these clusters. However, tendencies of several other classes are not as obvious as the ones previously listed, explaining perhaps the relatively low performance of our algorithm.

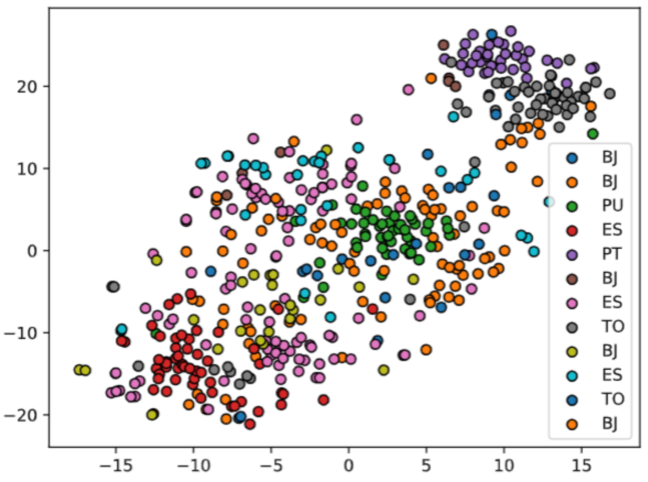


Figure 3: t-SNE Visualization

4 Conclusion

In conclusion, PCA and K-means were used to determine clusters for tree species as a baseline model. Both components were trained sequentially. In this block, a CAE was chosen to extract features from the Horoma dataset and simultaneous training of the mapping and K-means was performed. We achieved a 34.67 f1-score with our best model. Moving forward, the next step of this project is to use semi-supervised techniques, with now a portion of the training set being labeled.

If one were hoping to enhance the performance of our model, further advanced methods could be explored. For instance, in [2], Yang et Al. also trained the 2 elements coincidentally however, they defined a joint optimization problem by formulating a single loss function for both components. Another state-of-the-art technique is one described in [1], where Caron et Al. make the use of cluster centroids as 'pseudo-labels' to train the CAE, unlike our design, where we use the images themselves as labels. Furthermore, instead of the MSE optimization criterion, parameters of CAE are learned with the multinomial logistic loss function.

Another interesting alternative for further experiments would be to use soft assignments in clusters, replacing the hard assignment in k-means. More precisely, one can introduce stochasticity to assign samples to centroids, which is more flexible and might enhance generalization ability. For example, Xie et Al. uses the Student's t-distribution to measure the probability of an example belonging to a cluster and the KL divergence minimization to refine clusters [4].

References

[1] Mathilde Caron et al. "Deep Clustering for Unsupervised Learning of Visual Features". In: *Lecture Notes in Computer Science* (2018), pp. 139–156. ISSN: 1611-3349. DOI: 10.1007/978-3-030-01264-9_9. URL: http://dx.doi.org/10.1007/978-3-030-01264-9_9.

[2] Shlomo E. Chazan, Sharon Gannot, and Jacob Goldberger. *Deep Clustering Based on a Mixture of Autoencoders*. 2018. arXiv: 1812.06535 [cs.LG].

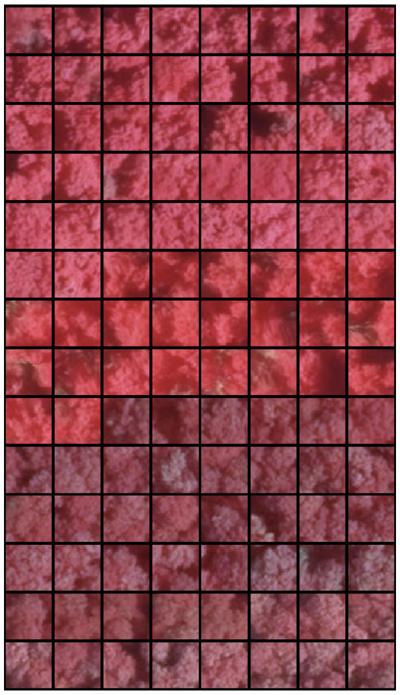
[3] Jonathan Masci et al. "Stacked Convolutional Auto-Encoders for Hierarchical Feature Extraction". In: *Artificial Neural Networks and Machine Learning – ICANN 2011*. Ed. by Timo Honkela et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 52–59. ISBN: 978-3-642-21735-7.

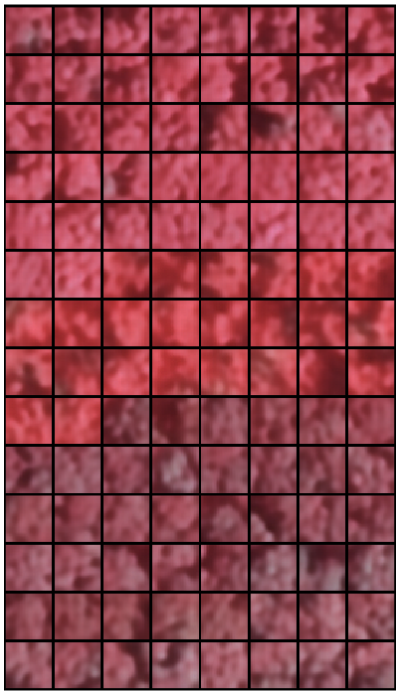
[4] Junyuan Xie, Ross Girshick, and Ali Farhadi. *Unsupervised Deep Embedding for Clustering Analysis*. 2015. arXiv: 1511.06335 [cs.LG].

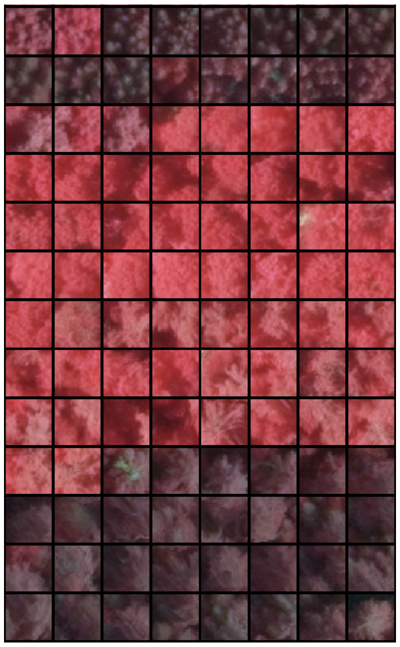
[5] Bo Yang et al. *Towards K-means-friendly Spaces: Simultaneous Deep Learning and Clustering*. 2016. arXiv: 1610.04794 [cs.LG].

Appendix

Appendix A : Image Embeddings of Validation Set


(a) Original Pixel Patches


(b) Reconstructed Pixel Patches


(c) Original Pixel Patches

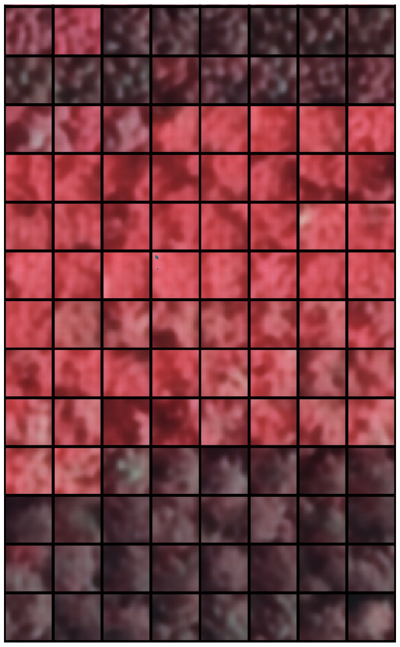

(d) Reconstructed Pixel Patches

Figure 4: CAE: Before and After Reconstruction