Horoma Block 2 Team b2phot5

Jonathan Bhimani-Burrows p1211136
Julien Horwood p1198276
Badreddine Sayah p20141352

1 Introduction

With climate change becoming increasingly apparent, more so than ever is it necessary to monitor our changing ecosystems. By quantifying land cover change, deforestation and habitat degradation and disease, satellite imagery interpretation has the potential to provide key insights in solving these problems. However, efficient and intelligent interpretation of this data remains a challenge. In this project block, we attempt to make use of Horoma's satellite forest region image data to interpret and classify tree species in an unsupervised manner.

As was elaborated upon in phase one of the project, this task is particularly challenging. While this is inherently true of unsupervised learning, difficulty is increased in particular due to the very noisy image data. This renders differentiating between tree species based on raw images using unsupervised learning extremely challenging.

In block 1, groups were tasked with building a baseline latent embedding and clustering model of the data set using both Principal Component Analysis (PCA) and K-Means. The latent embedding is necessary in order to make the clustering task computationally feasible. A good embedding should also help the clustering performance by reducing the effects of the curse of dimensionality. While this baseline is a good starting point, changes in the data distribution and prediction task meant this baseline needed to be recalculated for block 2. As such, our progression for this block consisted first in defining the data to be used for training, cluster labelling and prediction validation.

Second, we established new baselines based on PCA and K-Means. We then developed various Auto-Encoder models in order to learn more complex, non-linear embeddings of the dataset. We discuss in this report our findings in using both standard Auto-Encoders and Convolutional Auto-Encoders along with their Variational counterparts. Our initial hypothesis was that Variational Auto-Encoders should outperform standard ones, due to the fact that they are optimized towards preserving a data distribution prior under the encoder. This intuitively would suggest that elements which are close together in latent space are more likely to be generated from the same distribution, and thus be better suited to coherent clustering.
Following the development of these models, we used a two-step greedy approach in optimizing for the clustering task. Since we treat the embedding and clustering tasks separately, this was necessary in order to reduce the complexity and resources necessary for model selection. We thus make the assumption that better reconstruction loss should, in general, correlate with increased clustering accuracy. Model selection was performed over both K-Means and Gaussian Mixture Model (GMM) clustering methods.

2 Methodology

2.1 Data Description and Processing

2.1.1 Dataset and Validation Split

The provided data has many particularities which deserve special attention. Indeed, each image corresponds to an image patch of size 32x32x3 taken from a larger image patch, which we call region, within the original raw image data. This means there is spatial dependence that must be taken into account. Clearly, two adjacent image patches within a region are not independent and have higher probability of sharing the same tree species distribution. Thus we chose to split the cluster labelling

and validation set in such a way as to maintain *region independence* between splits and, to the extent possible, preserving class balance subject to this constraint. In order to do this, we implemented a class-wise approximation to the split, guaranteeing that the training labeled set had at least the defined split proportion of samples for each class label. This means the overall split defined as a hyper-parameter is an approximation to the true split. In practice, this amounted to 72% of samples being used for cluster label assignment, given a hyper-parameter value of 70%, with 28% being used for validation.
Furthermore, two separate versions of the data may be used. The original version contains 150900 unlabeled training samples and 480 total samples for validation, while an overlapped version of the data contains 544749 unlabeled training samples and 1331 total validation samples. This augmented data was generated by shifting the original image patches across the larger source image by 50% translation increments. Given that we defined our validation splits according to region independence, we feel it is beneficial to use the overlapped version of the data as it should provide more information to train a more robust model for both embedding and clustering. We justify this empirically in our model selection.

2.1.2 Pre-processing

Given that processing the data was largely the focus of experimentation from block one and that the primary transformations performed within this block simply involved rescaling the images, we did not explore further pre-processing and chose to focus on model development. That said, re-scaling the images to tensors in the range [0,1] from their original [0,255] range is a crucial pre-processing step, particularly as we are working with the Mean Squared Error for measuring reconstruction loss. Without scaling, the loss quickly diverges or overflows due to the gradients overflowing.

2.2 Proposed methods

In order to render the clustering phase more computationally tractable, dimensionality reduction through unsupervised learning is crucial. Furthermore, clustering algorithms suffer from the curse of dimensionality, where sparsity of points in higher dimensions may lead to severe degradation of relative distance estimates, and thus clustering performance. While PCA is a reasonable baseline to remedy this, deep learning methods, and Auto-Encoders in particular, offer the potential of learning highly informative compact representations of the input space.

Indeed, such networks compress the input representation down to a bottleneck layer, which we refer to as the latent representation, before upsampling this bottleneck back to the original input space. By using the input as the target to the network, it learns to represent the input space through the compressed representation of the bottleneck layer. An example architecture diagram is given in Figure 1.

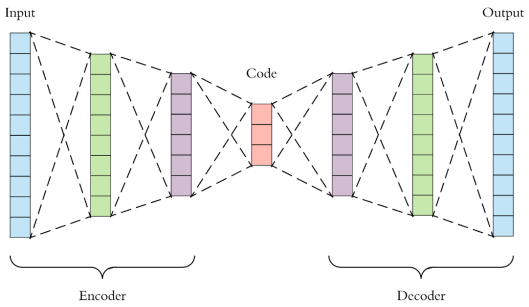

Figure 1: Standard Auto-Encoder Architecture

We attempted four variations of the general auto-encoder framework in order to obtain this latent representation. We refer to the source code for specific architectural details of our models, but present a brief discussion of their respective strengths and salient differences.

In general,we used symmetric architectures for all models, with model sizes of approximately 215 000 parameters for convolution-based models and 3.5 million parameters for feed-forward models.We used ReLU non-linearites throughout the models. Both the encoder and decoder have six layers in the convolutional case compared to three in the feed-forward case.

- **Vanilla Auto-Encoder (AE)** We implemented a Vanilla Auto-Encoder, as described above, as a second "baseline" among our deep models. Thus this simply consists of feed-forward encoding and decoding layers. Note that using such a model requires flattening the input images to 3072-dimensional vectors. This discards the spatial information available within the image.
- **Convolutional Auto-Encoder (CAE)**

  The CAE was the second model proposed for this task. The major difference for this model stems from the use of convolutional layers which allow parameter sharing and local connectivity across the input image while preserving spatial representation throughout the network. As CNNs have been shown to dramatically improve image classification, as seen in [3] and are key components of state-of-the-art computer vision models, our expectation was that autoencoders leveraging convolutional layers should be able to learn better latent representations.
- **Variational Auto-Encoders (VAE and CVAE)**

  We chose to also explore the variational variants to the above models, the VAE and CVAE. As described in [2], a VAE is a generative model, in that it learns a *distribution* over the latent space. It does so by learning the parameters of a diagonal multivariate gaussian distribution instead of directly learning a latent representation. In order to backpropagate through this network, we use the reparameterization trick, which involves adding the mean vector and multiplying the learned standard deviation by samples drawn from a standard gaussian distribution. This effectively allows us to encode an arbitrary gaussian (diagonal) while backpropagating through the parameters.
  Intuitively, this encourages the encoding towards a prior (gaussian) and allows us to sample new examples from the latent space. Thus, nearby examples in latent space should preserve semantic similarities from the input space. We thus implemented the reparameterization trick and redefined the bottleneck layer in order to adapt the previous two models to their variational versions.

2.3 Experiments and Model Selection

In order to perform model selection, we chose to perform a series of "greedy" task oriented modelling decisions based on prior hypotheses and simple empirical comparisons. This was to reduce the amount of tuning done for our final model selection. The first such decision was whether or not to focus on the standard feed-forward networks architectures or convolutional architectures. In order to do so, we attempted training models with identical latent dimensions and compared reconstruction costs. We then used the best performing epoch version from each architecture to compute a sample K-Means clustering. The best encoding model across epochs is saved as a training checkpoint within the source code's training loop, and was computed best on the held-out validation split reconstruction loss. We note that here this validation split is that taken from the unlabeled training set, and not the split alluded to previously. Indeed, this split is to measure reconstruction performance of the auto-encoder as opposed to clustering accuracy. We used an 80/20 split for computing this reconstruction loss. Furthermore, all experiments and hyper-parameters were tracked using Comet ML.
Second, we performed a similar analysis using the same architectures to compare model-wise performance on both the overlapped and non-overlapped datasets. This was used to guide our dataset choice for further model selection, and confirm the type of architecture to use.
Following these choices, we performed model selection by tuning the latent dimension and cluster size. We chose to optimize directly for clustering accuracy during hyper-parameter tuning, as this is ultimately the task we are evaluating. We chose to use a learning rate scheduler and Adam optimization to increase robustness to the initial learning rate hyper-parameter without performing explicit tuning.

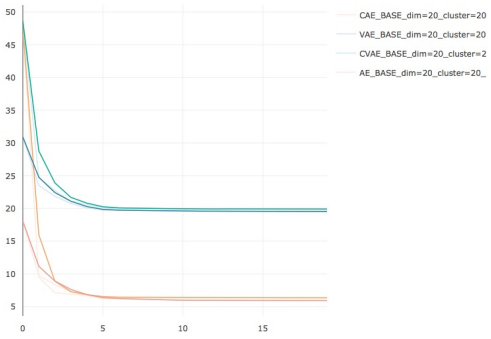

Figure 2: Training loss over epochs

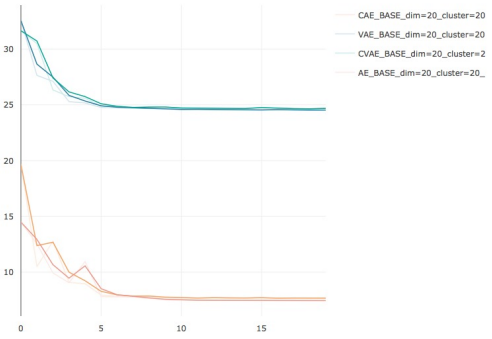

Figure 3: Validation loss over epochs

2.3.1 Initial comparison of Vanilla and Convolutional Models

Figures 2 and 3 show the training and validation reconstruction losses achieved on a first experiment with our models. We notice that convergence is quite fast for all models and that we do observe, as expected, a bias in the training set loss. Surprisingly, the convolutional version of each auto-encoder performs essentially as well as the feed-forward version. We attempt to reason about this by noting that the spatial structure within the input image is not as clear as traditional vision-based tasks. Indeed, the data is extremely noisy and hard to interpret, at least from a human perspective.
Furthermore, we note that the embedding layer remains flat in both the convolutional and feed-forward models. This is in contrast to the alternative of learning a spatially reduced convolutional bottleneck layer before applying a flattening at clustering time. Our reasoning for designing the convolutional model in this way was that the model should learn to flatten the two-dimensional data in an intelligent way during the backpropagation process, as opposed to simply reshaping the obtained output. Further comparison would however be warranted to ratify this choice. This may, however, further explain the similar reconstruction loss. Indeed, both model versions were trained to reproduce the two-dimensional output from what is ultimately a linear embedding.
We also note that comparing losses across regular and variational auto-encoders is pointless, as they optimize for distinct loss functions. Thus the higher reconstruction loss for the variational models in the above figures does not indicate that we should prefer regular auto-encoders.
Despite these figures, our preference remains to keep the convolutional versions of these models, as they are much more efficient in terms of model size and are architecturally designed for structured data. We further confirm this choice below in comparing clustering performance on the different labeled datasets.

2.3.2 Comparison of Regular and Overlapped Datasets

In order to continue our initial comparison across models and define our dataset choice to be used for model tuning, we trained the models described above on the overlapped dataset. We trained over only five epochs, achieving approximate convergence, given that the overlapped dataset is much larger. Comparison of the resulting K-Means clustering accuracy for each model on the labeled validation set split is given in Figure 4.

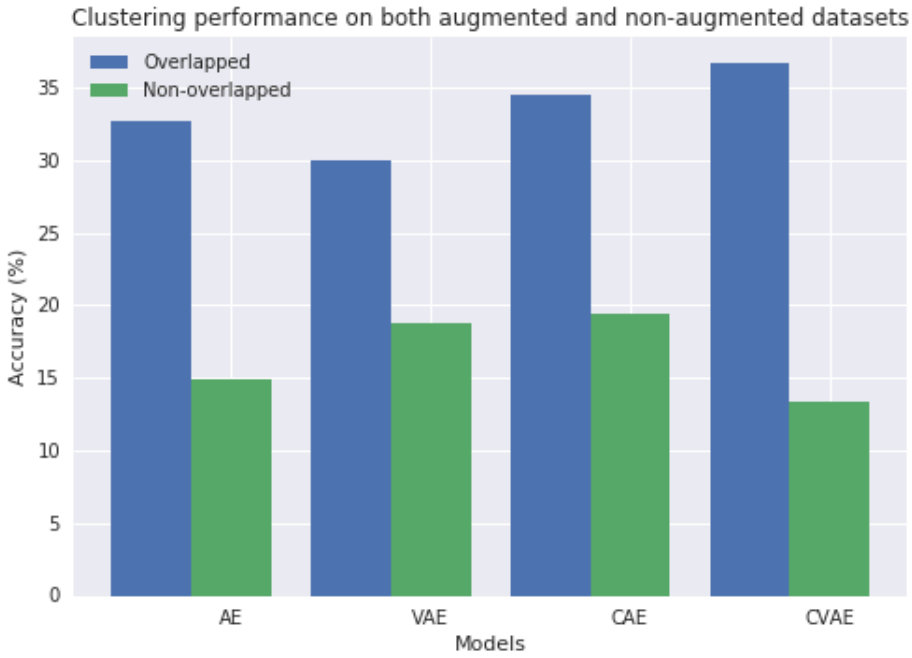

Figure 4: Comparison of model clustering performances using K-Means over both available datasets

Clearly, the augmented, overlapped version of the data leads to much higher clustering accuracies, regardless of the model used. Given that we have split our validation set in such a way as to maintain independence of regions, we consider these performance estimate differences sufficiently reliable and significant to justify using the overlapped dataset for further tuning. Additionally, we notice here that the convolutional versions of each model outperform their feed-forward counterparts, adding empirical support for our initial intuitions that the learned embeddings should be more informative. We thus will perform further tuning on these models alone.

2.3.3 Hyper-parameter tuning

In order to tune our final architectures, we chose to optimize over a small grid search over latent dimensions, number of clusters and clustering using K-Means or Gaussian Mixture Models. While K-Means is considered a simpler clustering model and a special case of GMM, we found it necessary to optimize over both as K-Means often outperformed GMM. We present the results of this hyper-parameter search and related discussion in the following section.

3 Results and Discussion

The results from Figure 5 present the achieved accuracy scores on the labeled validation set across different hyper-parameter settings.We present the clustering accuracy according the the best performing clustering algorithm between K-Means and GMM. In all cases, we found K-Means outperformed GMM.

Latent Dim \ n_cluster	Model	17	21	25
5	CAE	34%	36%	24%
	CVAE	34%	34%	30%
20	CAE	31%	36%	33%
	CVAE	32%	35%	37%
40	CAE	31%	36%	33%
	CVAE	31%	35%	31%

Figure 5: Final Hyperparameter Tuning Values

There were several interesting observations noted during the model selection phase. First, it is interesting that K-Means clustering in general performed better than the Gaussian Mixture Model. Indeed, this is surprising as K-Means is a special case of the GMM and thus has lower capacity in learning complex clusterings. Although we do not have a definitive reason for this observation, we note that as a guiding principle, simpler models tend to perform better when the assumptions of the model correspond well to the underlying data, and present less variance with respect to changing assumptions. Conversely, higher capacity models may degrade when their assumptions are false. Thus if the underlying data is not well behaved according to some underlying Gaussian Mixture distribution, we may begin to see why K-Means performs better. At the very least, this is what empirical results suggest.

Another interesting and very important observation is that the auto-encoder reconstruction error was, on its own, not a significant indicator of the final clustering validation set accuracy. That is, an auto-encoder with low mean-squared-error (MSE) and, by extension, better image reconstruction, didn't necessary imply improved clustering. On the one hand, this is understandable, as it would be theoretically possible to obtain a low loss on various latent space dimensions, yet not all latent dimension embeddings are necessarily suitable for clustering. Indeed, ultimately the auto-encoder does not optimize for clustering accuracy. Obtaining a reasonable latent embedding for reconstruction is thus merely a heuristic to improve the clustering representation. Jointly optimizing these tasks provides an interesting avenue for future work.

4 Conclusion and Future Work

In summary, we improved upon the first block models, which used PCA and K-Means clustering, by training four different types of auto-encoders to learn more complex embeddings before clustering using K-Means and GMMs. In doing so, we achieved our highest validation set performance at 37.27 percent accuracy vs PCA and K-Means, which achieved 32.43 percent accuracy. We further explored the differences between feed-forward and convolutional auto-encoders along with their variational counterparts. We ultimately found that the CVAE with 20 latent dimensions and 25 clusters performed the best, which supports our initial hypothesis that both the convolutional model may better preserve the spatial structure of the data, while the generative aspect of variational models encourages these distributions to preserve distance semantics in the latent space. Of course, these results were only marginally different, so drawing strong conclusions is difficult.

For future work, we believe many improvements are possible for this task. First, due to time constraints we were unable to perform extensive model selection and hyper-parameter tuning over our models. We believe this may have great impact as we found our experiments to present very large variance across different hyper-parameter initializations. In addition, as mentioned above, the current work optimized for reconstruction loss and clustering separately, whereas our observations indicated this was likely a sub-optimal approach. Models which leverage deep learning to simultaneously embed and cluster data, such as that presented in [1], provide a good avenue for improvement in our opinion. Finally, it may be possible to make use of the labelled data to learn labels more effectively than simple cluster label assignment. For example, one could use a soft class probability assignment for each cluster as opposed to assigning the maximum class to each cluster, and then compute a weighted sum over class probabilities for new samples according to cluster

centroid distances. Prior information about class distribution may also be introduced more effectively. Unfortunately, we were unable to explore these options, but we believe the added information from such soft methods should incorporate more information into the labelling process and may thus aid in improving performance.

References

[1] Xifeng Guo et al. “Deep Clustering with Convolutional Autoencoders”. In: National University of Defense Technology. 2017.

[2] Diederik P. Kingma and Max Welling. “Auto-Encoding Variational Bayes”. In: ICLR 2014. 2013.

[3] Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton. “ImageNet Classication with Deep Convolutional Neural Networks”. In: University of Toronto. 2012.