Reproducing A Novel Weakly Supervised Clustering Framework and Evaluating Practical Applicability

Sameen Islam, Mohammed Mossuily & Saivignesh Pandian
Department of Electronics and Computer Science
University of Southampton
Southampton, United Kingdom
{si1u19, mtm1g19, ssp1e17}@soton.ac.uk

Abstract

In this investigation we seek to replicate the results of a previously proposed weakly supervised clustering model that exploits unique class count labels. We independently reconstruct the model presented and reproduce the results on the MNIST dataset, achieving a 98.37% clustering accuracy. Furthermore, we bridge the gap between the proposed model and a future practical deployment by exploring practical limitations of the model with training and implementation.

1 Introduction

In Oner et al. (2020), they present a weakly supervised model which is trained to predict the number of unique classes in a dataset. They argue that a model trained to do so must be learning some underlying patterns in the data to inform it of the number of classes. Thus, such a model should after training, be able to be re-purposed to predict the class labels on specific instances of data. This is presented to be of great use in fields such as the medical profession, in which one might know the number of cancerous cells in an image, but not be able to identify which pixels in the image correspond to such a cell.

2 Experimental Methodology

While the specific code for the model is provided and available[1], we found that it was poorly documented and improperly motivated. Thus in this investigation, we completely re-implemented the model [2], which allowed us to explore the architecture and understand the motivations behind certain choices.

We seek to provide some reasoning behind the model choices made in the original paper, as well as bridge the gap in knowledge needed to implement and deploy such a model. For example, as the original paper makes no reference to the training time and difficulty, we seek to analyse whether such a model can be considered practical when the computational costs are also taken into account. Finally, we seek to replicate the results found in the original paper, and compare with other novel techniques to draw conclusions about the efficacy of this presented model.

For this investigation, we have limited the scope to only reproducing and exploring the presented model on the MNIST dataset. Oner et al. (2020) make no mention as to the training difficulty of their model. However, we found that the training times for useful results were unreasonably long on the MNIST dataset, and training on other datasets such as CIFAR or CAMELYON were considerably slower. As such, only MNIST data is used in this project.

3 Reproducing the Model

3.1 Model Architecture

The weakly supervised clustering framework consists of an end-to-end trained deep convolutional autoencoder and a classifier. Figure 1 shows the architecture schematic of this framework, which can roughly be considered in three parts: the encoder, the decoder and the classifier. The encoder exists to extract features from the input images, and consists of several convolutional layers and wide residual layers, described by Zagoruyko & Komodakis (2016). The extracted features are modelled by probability distributions in the Kernel Density Estimation (KDE) layer. These distributions are then fed to an MLP classifier to estimate the number of unique classes in the data. We simultaneously minimise the decoder and classifier loss so that the extracted features are specifically useful for accurate classification. The decoder ultimately produces a reconstruction of the original image, but this is unused as its only used for the purposes of minimising loss. The model minimises a weighted sum of classification and auto-encoder losses with the weight being a hyper-parameter.

[1] http://bit.ly/uniqueclasscount
[2] https://github.com/COMP6248-Reproducability-Challenge/UCC-Classifier

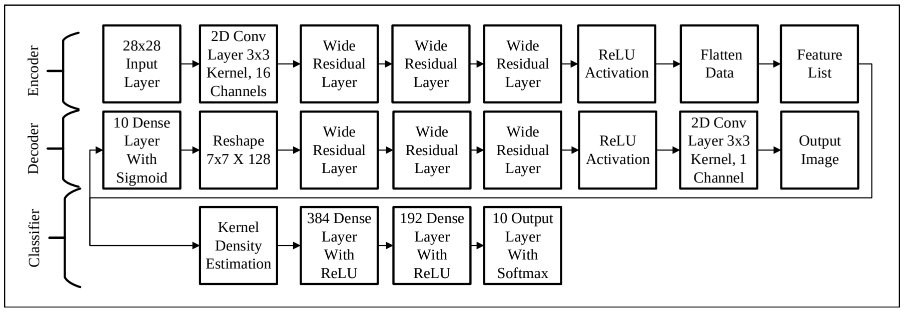

Figure 1: Model architecture showing a flat feature vector is obtained from the encoder, which itself takes in a 28×28 image. This flat vector is then passed to a decoder and a kernel density estimator, which then reconstructs and makes a softmax prediction of the UCC label respectively.

3.2 Training the Model

Under the weakly supervised framework, we train the UCC model to predict the number of unique classes in batches of MNIST data. Our model was trained locally using TensorFlow with Python 3.6 using a NVIDIA RTX 2070 GPU. Oner et al. (2020) train their model for 128,000 epochs, which requires $\sim$ 12 hours of continuous training. When compared to other simple models, this training time is orders of magnitude larger. The UCC model also demands a large amount of memory. During experimentation, we found our system undergoing excessive thrashing, as the model consumed in excess of 60GB of memory.

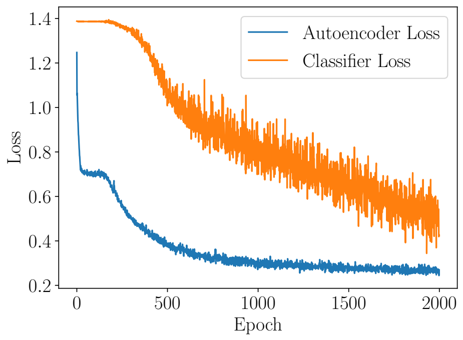

Figure 2: Training loss for UCC model monotonically decreasing. The auto-encoder (blue) converges quicker than the classifier (orange).

Figure 2 shows the training with the described loss criterion where we find both plots are noisy. This is usually caused by a high learning rate causing instability or by a large batch size causing class imbalance as new data is seen. We experimented with both of these hyper-parameters, however this effect still persisted.

4 Applying the Model to MNIST

4.1 Autoencoder Model

As explained in Section 3.1, the total loss of the model is dependent on the weighted sum of losses of the classifier and the autoencoder. This is intuitive, since the autoencoder needs to be able to extract “useful” features that the classifier can use to label the data. In this investigation, the autoencoder was trained to reduce the 28×28 MNIST images into a flattened list of 10 “features”, the encoded data.

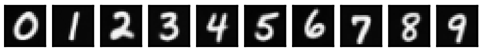

Figure 3: Reconstructed digits from autoencoder, which look very similar to the input images.

The reconstructed digits from the MNIST data are presented in Figure 3. We immediately observe that the digits are very similar to the original MNIST data, implying that the small loss we observed in Figure 2 for the autoencoder model, has led to its ability to properly encode and decode the data.

The encoded data, i.e. the feature list, could be directly fed into an MLP model for classification. However, investigations found that this produced poor results in classification, likely as a result of the small feature list. A solution is to apply certain transformations to the feature list, to obtain better classification scores. This motivates the use of KDE.

4.2 Kernel Density Estimation Layer

The KDE Layer is used to construct a probability distribution of the features extracted by the autoencoder model. In this investigation, a Gaussian kernel was used, since no prior knowledge of the extracted features was known. As such, we argue that through central limit theorem, the output of a sufficiently complex encoder model can be weighted using a Gaussian kernel.

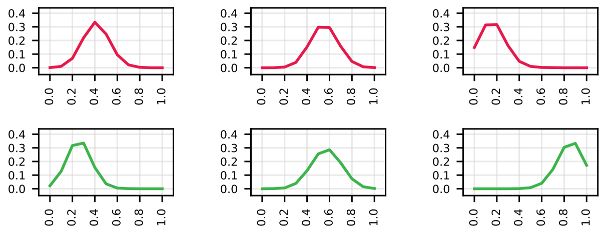

Figure 4: Distributions created by KDE layer using features from Encoder. Red shows three features distributed in class 1, and green shows three features distributed in class 2.

We can construct a plot of the distribution of different extracted features for different classes. This is presented in Figure 4, which shows the way that features 1, 2 and 3 are distributed on classes 1 (red) and 2 (green). We immediately observe that there are clear differences in the distributions of features for different classes. This intuitively informs us that the KDE layer can create distributions that can be easily separated into classes, resulting in higher performance of the classsifier when the KDE layer is used.

We might consider whether during inference, we can forego the MLP classifier in favour of directly clustering the distributions using an unsupervised algorithm such as K-Means. One such model is also investigated in this paper.

4.3 Training and Inference on MNIST

The MNIST images are randomly grouped into batches of 32. We refer to these batches as “bags”. Initially, in the training stage, the model is trained for 128000 epochs to estimate the number of unique classes in the bag. The training set is 50000 images, while the validation step is 10000 images. Early stopping is implemented, but was found to be unnecessary as overfitting on this dataset did not occur. It is possible to not have a separate training and test set, and allow the model to train on the same data upon which it will infer labels, since in the inference stage it is predicting the cluster membership and not the number of unique clusters.

Once trained, the model is used to label given instances of MNIST images into appropriate classes. We find that we achieve a clustering accuracy of **98.37**%. This is close to the result found by Oner et al. (2020). We further analyse this result and compare it with existing models in the following Section.

4.4 Comparison with Other Models

We find that we can reproduce the clustering accuracy of the model created by Oner et al. (2020) with our re-implemented model. The comparisons to existing models are presented in Table 1.

Table 1: Clustering Accuracies for different models on MNIST dataset. *Models developed, trained, and tested in this investigation. Other models from other papers are shown as well.

Model	**Clustering Accuracy**
UCC Classifier*	98.37%
UCC Classifier (Oner et al. (2020))	98.4%
Fully Supervised Model (Oner et al. (2020))	98.8%
K-Means with Extracted Features*	97.73%
K-Means with Raw Data (Wang et al. (2014))	57.2%
IIC (Ji et al. (2019))	98.4%

We observe that with the same number of epochs and identical hyperparameters, we reproduce the original results. When we compare this model to a fully supervised model representing the upper limit of state of the art neural networks to classify MNIST data, we note that the UCC Classifier presented in this paper produces a clustering accuracy on MNIST that is very close to state of the art supervised approaches. This provides motivation for using the UCC classifier, since it can achieve high performance on weakly supervised data, comparable to a supervised model.

As suggested earlier, the classifier section of the total model (which is a simple MLP after a KDE Layer), may be superfluous during inference, as the output of the KDE distribution could be given directly to a unsupervised model to cluster. This is presented in the table, and we observe that the accuracy is very close to the total model. However, this approach still requires the use of an MLP in the training stage. As explained in Section 3.1, the loss of the autoencoder is coupled to the loss of the classifier, to ensure that only features that are useful for classification are extracted by the encoder.

When we compare the results of this weakly supervised approach to a fully unsupervised K-Means algorithm applied directly to the raw data, we observe significant increases in performance, motivating the use of this weakly supervised approach.

Finally, we compare to a state of the art unsupervised approach, presented by Ji et al. (2019). We observe almost identical clustering accuracy compared to their proposed Invariant Information Clustering (IIC) approach, which does not require extensive training, or bag-level labels. This indicates that these novel unsupervised techniques might surpass this UCC Classifier approach in certain domains.

5 Efficacy of Project

Oner et al. (2020) present this weakly supervised clustering approach as a proof of concept. They argue that a model trained to predict the unique class count in a bag is learning patterns in the data that could be used to cluster the data into the classes. In this regard, this investigation supports their conclusions. However, we raise concerns over the practical advantage provided by such a model in a real-world setting.

We argue that the long training times ($\sim$ 12 hours) necessary on commercial hardware to achieve these results limits the efficacy of their presented model. When compared to models such as the IIC model presented by Ji et al. (2019), the clustering accuracy is identical, but the IIC model does not require such extensive training. As such this UCC model's effectiveness is limited somewhat by the computational costs. The effectiveness of this model may vary in different domains, however.

In addition to this, the relative difficulty of reproducing the results presented by Oner et al. (2020) make the use of an identical model challenging. As they do not present specific model sizes, and fail to motivate the reasoning behind the choices made for certain architectures, it is difficult to adapt the model for other domains.

The main claim made by the original paper, however, is valid. And in many instances in which bag-level labels are available, for example in the medical field where the presence of cancerous tissue is known, even if its exact position is not, this model can prove to be superior to unsupervised techniques.

6 Conclusions

In this investigation we have re-implemented the UCC model presented by Oner et al. (2020) and replicated their results for the MNIST Dataset. We have also further investigated the model architecture, including by testing the model without certain components (eg. KDE layer), and provided justification for the choice of certain parts of the model. Finally, we have compared the reproduced results with existing unsupervised and supervised techniques and compared its effectiveness when factors such as the training time and model construction difficulty are considered. We argue that while the model may present benefits on other domains, its practical usefulness is outstripped by state of the art unsupervised techniques on the MNIST dataset.

References

Xu Ji, João F Henriques, and Andrea Vedaldi. Invariant information clustering for unsupervised image classification and segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 9865–9874, 2019.

Mustafa Umit Oner, Hwee Kuan Lee, and Wing-Kin Sung. Weakly supervised clustering by exploiting unique class count. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=B1xIj3VYvr`.

Jianfeng Wang, Jingdong Wang, Jingkuan Song, Xin-Shun Xu, Heng Tao Shen, and Shipeng Li. Optimized cartesian k-means. *IEEE Transactions on Knowledge and Data Engineering*, 27(1):180–192, 2014.

Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.