Towards Financially Inclusive Credit Products Through Financial Time Series Clustering

Tristan Bester, Benjamin Rosman

School of Computer Science and Applied Mathematics
University of the Witwatersrand
Johannesburg, South Africa
tristanbester@gmail.com, benjamin.rosman1@wits.ac.za

Abstract

Financial inclusion ensures that individuals have access to financial products and services that meet their needs. As a key contributing factor to economic growth and investment opportunity, financial inclusion increases consumer spending and consequently business development. It has been shown that institutions are more profitable when they provide marginalised social groups access to financial services. Customer segmentation based on consumer transaction data is a well-known strategy used to promote financial inclusion. While the required data is available to modern institutions, the challenge remains that segment annotations are usually difficult and/or expensive to obtain. This prevents the usage of time series classification models for customer segmentation based on domain expert knowledge. As a result, clustering is an attractive alternative to partition customers into homogeneous groups based on the spending behaviour encoded within their transaction data. In this paper, we present a solution to one of the key challenges preventing modern financial institutions from providing financially inclusive credit, savings and insurance products: the inability to understand consumer financial behaviour, and hence risk, without the introduction of restrictive conventional credit scoring techniques. We present a novel time series clustering algorithm that allows institutions to understand the financial behaviour of their customers. This enables unique product offerings to be provided based on the needs of the customer, without reliance on restrictive credit practices.

Introduction

Financial inclusion means that individuals have access to financial products and services that meet their needs. Notably, financial inclusion is a strong contributing factor to economic growth as it stimulates entrepreneurship while expanding investment opportunities. It boosts consumer spending and business development, leading to job creation and improved productivity. Financial institutions are strongly incentivised to support the positive social impact brought about by a financially inclusive market (Moin and Ahmed 2012). This is a consequence of the fact that institutions are more profitable when they make use of modern technological strategies that provide marginalised so-

cial groups access to financial services (Alshehadeh and Al-Khawaja 2022).

A well-known strategy used to enable financial inclusion is customer segmentation. A key result in service marketing research indicates that companies should not market the same set of services to all customers, but rather target products at specific segments of the customer base (Ansari, Riasi et al. 2016). Customer segmentation can be achieved through the analysis of bank transaction data, which has become available as a result of the large-scale automated data acquisition systems present in modern financial institutions (Taifi 2023). Unfortunately, the challenge remains that data annotations are difficult and/or expensive to obtain from domain experts. This lack of annotations prevents supervised learning models from being used to perform customer segmentation based on domain expert knowledge (Dempster, Petitjean, and Webb 2020; Ismail Fawaz et al. 2019). As a result, unsupervised clustering is an attractive alternative to partition customers into homogeneous groups based on the spending behaviours encoded within their transaction data.

In this paper, we present a solution to one of the key challenges preventing modern financial institutions from providing financially inclusive credit, savings and insurance products. This is the inability to understand consumer financial behaviour, and hence risk, without the introduction of restrictive conventional credit scoring techniques. To this end, we extend the evaluation of time series clustering algorithms by measuring their performance on financial data. Guided by these results, we present a novel approach to financial time series clustering that is shown to outperform state-of-the-art techniques. Our method enables institutions to understand their customer base as a set of homogeneous groups, each with similar financial behaviour. By understanding the group to which a customer belongs, a product offering can be tailored to suit the needs of the customer in accordance with the risk appetite of the institution.

Background

Clustering

Cluster analysis or clustering is the task of partitioning a set of objects such that the elements within each partition are more similar to one another than those in other partitions. The notion of similarity in the space is defined both in

terms of the distance metric as well as the algorithm responsible for the calculation of the clustering. As a result, cluster analysis is not comprised of a single algorithm, but rather a collection of techniques. The definition of what constitutes a cluster differs significantly between algorithms, and consequently, the approaches followed to efficiently calculate them.

Cluster analysis is an unsupervised learning technique. Consequently, the ideal cluster associated with each sample in the dataset is unknown. This aspect of clustering makes it difficult to quantify the quality of a given clustering. This follows from the fact that the information required to compute such a quantity directly is not available. This has led to the introduction of proxy objectives such as the *Silhouette Coefficient (SC)* and *Davies-Bouldin index (DBI)* which measure the quality of clusterings based on assumed properties of high-quality clusterings. The SC quantifies the degree of similarity between elements within each cluster in comparison to that of other clusters. The SC ranges from -1 to 1, where a high value indicates that the objected is well suited to its cluster and poorly suited to neighbouring clusters. The DBI is a measure of similarity between the elements of a cluster and those of the closest neighbouring cluster. The similarity is defined as the ratio of within-cluster distances to between-cluster distances. The minimum DBI value is zero with lower values indicating better clusterings.

Specificity of the Time Dimension

In this paper we focus on *time series* data, where a time series is comprised of a collection of data points that are indexed through time. Each data point is a vector of feature values at a specific instant in time. It has been shown that conventional clustering algorithms perform poorly on this type of data, as they make use of distance metrics that are incompatible with the time series representation (Lafabregue et al. 2022). In response, several dedicated time series clustering algorithms have been developed. These algorithms either make use of distance metrics specifically designed for time series (Müller 2007) or introduce alternative data representations (Madiraju 2018). The latter approach, known as *representation learning*, is the primary focus of this study.

Neural Network Architectures

The architecture of a neural network refers to the number, types and sizes of the layers from which it is composed. Throughout the literature, several variations have been proposed for deep neural network layers. These layers can be organised into three families: *fully connected, convolutional* and *recurrent layers*.

Fully Connected Neural Network Layers (FCNN)   Each neuron in a fully connected layer is connected to all neurons in the following layer (Block, Knight Jr, and Rosenblatt 1962). Fully connected layers apply a linear transformation to the input data before passing the result through a non-linear activation function.

Convolutional Neural Network Layers (CNN)   Convolutional neural network layers are well known as a result of their widespread adoption in the field of computer vision (He et al. 2016). A key property of the convolutional kernels used within each layer is shift-invariance. This property is a consequence of the shared-weight architecture on which the design is based. While the most commonly used convolutional layers, designed for image processing, make use of two-dimensional convolutions, one-dimensional variants exist (Wang, Yan, and Oates 2017). These variants have been adapted to capture temporal rather than spatial patterns in the data.

Recurrent Neural Network Layers (RNN)   The recurrent neural network layer has been specifically designed for applications involving the time dimension (Hopfield 1982). Recurrent layers introduce cyclical connections, allowing neuron outputs to propagate through the inference process, subsequently affecting future inputs to the layer. These layers can effectively capture temporal dynamic behaviour. An RNN layer variant known as the *Long Short-Term Memory (LSTM)* layer has been proposed in (Hochreiter and Schmidhuber 1997) as a solution to the vanishing gradient problem which hinders the performance of RNNs.

Deep Representation Learning

The objective of representation learning is to produce an alternate representation of the raw data. This representation is intended to increase the likelihood of obtaining desirable clustering results when used in conjunction with clustering algorithms that have not been designed for time series data. Representation learning is a key component of most deep learning-based clustering frameworks. Generally, the primary role of deep learning within these frameworks is to produce an alternate representation of the data. This is achieved through the use of a deep neural network known as the encoder. The encoder is a non-linear mapping $E_\phi : \mathcal{X} \rightarrow \mathcal{Z}$, parameterised by ϕ, that maps elements from the original data space $\mathcal{X}$ to the space of encoded representations $\mathcal{Z}$. The encoded representation of $x \in \mathcal{X}$, denoted $z = E_\phi(x)$, is referred to as the latent representation. The objective of deep representation learning is to learn a mapping E_ϕ that can produce a latent representation of the data that facilitates the desired clustering when used in combination with clustering algorithms that have not been designed for time series data.

We seek to optimise the parameters of the encoder such that the learned representation favours the clustering of data points into homogeneous groups. In this study, such a clustering corresponds to one in which individuals with similar transaction data, measured in terms of size and frequency of transaction amounts, are clustered together. However, as the desired clusters are unknown, we must optimise the encoder through the introduction of an auxiliary objective. Consequently, a second neural network known as the decoder is introduced along with a self-supervised objective function. Similarly to the encoder, the decoder $D_\theta : \mathcal{Z} \rightarrow \mathcal{X}$ is a non-linear mapping, parameterised by θ, that maps elements from the latent space $\mathcal{Z}$ back to the original data space $\mathcal{X}$. The encoder and decoder functions form what is known as an autoencoder. Forward propagation in an autoencoder consists of two stages. Firstly, the input feature vector x is em-

bedded into the latent space through the use of the encoder network. Secondly, a reconstruction of x, denoted x', is produced by passing the latent representation through the decoder network. The introduction of the decoder network facilitates the usage of a self-supervised objective.

Autoencoder Architectures The architecture of an autoencoder consists of two main components, the encoder and decoder. As described, the encoder and decoder are both neural networks which function as non-linear mappings between vector spaces. In general, the decoder is constructed as a mirror of the encoder with the exception of the embedding layer (Lafabregue et al. 2022). That is, the architecture of the decoder is obtained by arranging the layers of the encoder in reverse order. Depending on the application, any of the previously described neural network layers can form part of an autoencoder's architecture.

Autoencoder Training As described, the objective of representation learning is to produce an alternate representation of the raw data that facilitates the desired clustering. This requires the parameters of the encoder to be optimised towards this objective. As the desired cluster labels are unavailable, preventing the usage of supervised objectives, many auxiliary objectives have been used throughout the literature. While several approaches have been developed to train the parameters of the encoder, we focus on those that rely on the tasks of data reconstruction and generation.

In the first approach to autoencoder training, the parameters of the encoder and decoder networks are optimised such that the network can encode and decode data while incurring minimal loss of information. More precisely, the network is trained to minimise the error between the input and reconstructed representations with respect to some loss function. The most commonly used objective function for this task is the classical reconstruction loss (Meng et al. 2017). The classical reconstruction loss is defined as the mean squared error between the original and reconstructed representations:

$$\mathcal{L}_{\mathcal{R}} = \frac{1}{N} \sum_{i=1}^{N} ||x_i - D_\theta(E_\phi(x_i))||_2^2 \qquad (1)$$

This loss was extended by Ghasedi Dizaji et al. (2017) in which a layerwise objective is introduced. That is, the loss is defined as the sum of reconstruction losses associated with each depth in the autoencoder network:

$$\mathcal{L}_{\mathcal{LR}} = \frac{1}{N} \sum_{i=1}^{N} \sum_{l=1}^{L} \frac{1}{|z_i^l|} (z_i^l - \hat{z}_i^l)^2 \qquad (2)$$

where L is the number of encoder and decoder layers, z_i^l is the output of the lth encoder layer, $\hat{z}_i^l$ is the output of the lth decoder layer and $|z_i^l|$ is the number of elements in the output of the lth encoder layer.

In contrast to reconstructive approaches, generative methods optimise the parameters of the encoder for the generation of realistic data. The *variational autoencoder (VAE)* introduced by Kingma and Welling (2013) is one of the most commonly used generative methods. While this method is based on the autoencoder architecture, a significantly different training regime is introduced for parameter optimisation. The input is passed through the encoder in which it is mapped onto a Gaussian distribution $q_\phi(z|x)$. Samples are then drawn from this distribution and passed through the decoder to obtain the distribution $p_\theta(x|z)$. The parameters of the generative model D_θ (the decoder) are optimised to reduce the reconstruction error between the input and output representations. The parameters of the encoder network E_ϕ are optimised to minimise the distance between the two distributions $q_\phi(z|x)$ and $p_\theta(z|x)$. The proposed loss function to perform this task is known as the *evidence lower bound (ELBO)*:

$$\mathcal{L}_{\mathcal{V}} = \sum_{i=1}^{N} \mathop{\mathbb{E}}_{z \sim q_\phi(\cdot|x_i)} \ln \left(\frac{p_\theta(x_i, z)}{q_\phi(z|x_i)} \right) \qquad (3)$$

Encoder Optimisation for Clustering The aforementioned optimisation tasks have focused on the development of a meaningful latent space for specific tasks. While this approach provides good performance in some cases (Barkhordar, Shirali-Shahreza, and Sadeghi 2021), there is no guarantee that the learned representation will be well suited to the clustering task. In response to this challenge, several methods have been proposed that modify the learned latent space to increase compatibility with the clustering task (Madiraju 2018; Xie, Girshick, and Farhadi 2016). Generally, this is achieved through the introduction of a complementary loss function, used to increase separability in the latent space. This loss function is referred to as the clustering loss.

Deep Embedded Clustering (DEC) is one of the first conventional clustering algorithms presented in this space (Xie, Girshick, and Farhadi 2016). The proposed time series variant is known as *Deep Temporal Clustering (DTC)* (Madiraju 2018). Learning in the encoder layers of the DTC model is driven by the interleaved optimisation of two loss functions, namely, the classical reconstruction loss, defined in equation (1), as well as the introduced clustering loss, defined in equation (4). The DTC model contains a learnable set of cluster centroids. These centroids are used to perform clustering during forward propagation and are updated based on the clustering loss.

The training procedure used in these models consists of two distinct phases. Firstly, the autoencoder is pretrained to reconstruct the input data. After pretraining is complete, the training data is embedded into the latent space, allowing for the initialisation of the model's cluster centroids. The latent representations are clustered through the use of hierarchical clustering with complete linkage (Hubert 1974), and the centroids are initialised as the average of all elements in each cluster. The pretraining phase is used to ensure a meaningful initial set of cluster centroids. In the second phase of training, the clustering loss is introduced. The interleaved optimisation of two loss functions takes place in this phase. The first loss function is the classical reconstruction loss between the input and reconstructed representations, defined in equation (1). The second loss is the clustering loss. In each forward pass, the distance from the latent representation z_i to each cluster centroid ω_j is calculated as $d_{ij}(z_i, \omega_j)$, where

$d : \mathcal{Z} \times \mathcal{Z} \to \mathbb{R}$ is the distance metric used in the space and $\mathcal{Z}$ is the set of all latent representations. Once calculated, the distances are then normalized to probability assignments q_{ij} using the Student's t-distribution kernel:

$$q_{ij} = \frac{\left(1 + \frac{d(z_i, \omega_j)}{\alpha}\right)^{\frac{-\alpha+1}{2}}}{\sum\limits_{j=1}^{k} \left(1 + \frac{d(z_i, \omega_j)}{\alpha}\right)^{\frac{-\alpha+1}{2}}}$$

where q_{ij} is the probability input i belongs to cluster j, α is the number of degrees of freedom, set to one by convention and k is the number of clusters. The clustering loss is formulated as the Kullback–Leibler divergence between the assignment probabilities q_{ij} and the target distribution p_{ij}. The target distribution P, used in the loss function, is chosen to strengthen high-confidence predictions and normalise losses to prevent distortion of the latent representations:

$$p_{ij} = \frac{\frac{q_{ij}^2}{\sum\limits_{i=1}^{n} q_{ij}}}{\sum\limits_{j=1}^{k} \frac{q_{ij}^2}{\sum\limits_{i=1}^{n} q_{ij}}}$$

This gives the overall loss used in the second phase of training:

$$\mathcal{L}_{\mathcal{D}} = \frac{1}{N} \sum_{i=1}^{N} ||x_i - D_\theta(E_\phi(x_i))||_2^2 + D_{KL}(Q||P) \quad (4)$$

where Q is the cluster assignment distribution, P is the target distribution and D_{KL} is the Kullback–Leibler divergence.

As the latent representations produced by the DTC encoder are time series as opposed to vectors, the *Complexity Invariant Distance (CID)* is commonly used as a metric function for the latent space. We use $\mathcal{L}_{\mathcal{DE}}$ to denote the loss function in which Euclidean distance is used as the metric function and similarly, $\mathcal{L}_{\mathcal{DC}}$ for the case where CID is used as the metric function.

Dimensionality Reduction

Dimensionality reduction is the transformation of data from a high-dimensional space to a lower-dimensional space. The objective of dimensionality reduction techniques is to maximally preserve the meaningful properties of the original data after transformation. Dimensionality reduction techniques are commonly divided into linear and non-linear approaches.

Principal Component Analysis (PCA) Principal component analysis is the most widely used linear dimensionality reduction technique. PCA maps the data to a lower-dimensional space in such a way that the variance of the data in the low-dimensional representation is maximized (Hotelling 1933).

Uniform Manifold Approximation and Projection (UMAP) The UMAP algorithm is a general-purpose non-linear dimensionality reduction technique that utilises a theoretical framework based on Riemannian geometry and algebraic topology (McInnes, Healy, and Melville 2018).

Methodology

In this paper, we decompose deep representation learning-based clustering methods into their constituent components before performing an in-depth performance analysis. The objective of this analysis is to find associations between architectural components and good clustering performance when applied in the financial domain. Finally, guided by the results of our performance analysis, we present a novel deep representation learning-based clustering algorithm *Financial Transaction History Clustering (FTHC)*. This algorithm is compared to the current state-of-the-art approaches in the field.

Component Selection

Deep representation learning-based methods are decomposed into four distinct component classes. These component classes consist of (i) the autoencoder architecture, (ii) the dimensionality reduction technique, (iii) the pretext loss function and (iv) the clustering loss function. The full set of components considered is listed in Table 2 with more detailed configuration information provided in Appendix A.

Component	Option 1	Option 2	Option 3	Option 4
Architecture	FCNN	CNN	LSTM	DTC
Dimensionality Reduction	PCA	UMAP	NONE	
Pretext Loss	$\mathcal{L}_{\mathcal{R}}$	$\mathcal{L}_{\mathcal{LR}}$	$\mathcal{L}_{\mathcal{V}}$	NONE
Clustering Loss	$\mathcal{L}_{\mathcal{DE}}$	$\mathcal{L}_{\mathcal{DC}}$	NONE	

Table 1: Component Configurations

Component Combinations

A component combination is defined as a set of components including exactly one component from each of the four classes. The performance of each compatible combination is evaluated. A combination is defined as compatible if the components within the combination can function without the violation of any assumptions. For example, the DTC autoencoder architecture is incompatible with the variational autoencoder loss, equation (3), as the latent representation produced by the encoder is a time series. Several component incompatibilities exist between architectures and loss functions. Loss function support is described in Table 2.

Dataset

The Berka[1] dataset consists of a collection of financial records from a bank in Czechoslovakia (Berka et al. 2000). Specifically, this dataset is an aggregation of the financial information of approximately 5,300 bank customers. The dataset includes over 1,000,000 transactions, close to 700 loans and nearly 900 credit card applications.

---

[1]https://data.world/lpetrocelli/czech-financial-dataset-real-anonymized-transactions

		Pretext Loss			Clustering Loss	
		$\mathcal{L}_{\mathcal{R}}$	$\mathcal{L}_{\mathcal{LR}}$	$\mathcal{L}_{\mathcal{V}}$	$\mathcal{L}_{\mathcal{DE}}$	$\mathcal{L}_{\mathcal{DC}}$
Autoencoder	FCNN	✓	✓	✓	✓	✓
	CNN	✓	✓	✓	✓	✓
	LSTM	✓	✗	✓	✓	✓
	DTC	✓	✗	✗	✓	✓

Table 2: Architecture compatibility with loss functions. The LSTM and DTC architectures are incompatible with the extended reconstruction loss, equation (2), as the decoders are structured differently from the encoders. This prevents the calculation of the layerwise reconstruction error required by the loss. The DTC architecture is incompatible with the variational autoencoder loss, equation (3), as the latent representation produced by the encoder is a time series.

Evaluation Procedure

The training procedure[2] used for deep representation learning-based models consists of two distinct phases. In the first phase, pretraining, the autoencoder of the model is trained to minimise the pretext loss function. This is followed by the cluster optimisation phase to improve cluster assignments through the minimisation of the clustering loss function. All compatible component combinations were tested.

The dataset was randomly partitioned to form non-overlapping training and testing datasets. Each dataset was composed of 50% of the available data. Component combinations were trained using the training dataset and evaluated based on the clustering produced on the testing dataset. As described by Ma et al. (2019); Xiao et al. (2021), although clustering is an unsupervised learning technique, combinations must be evaluated on unseen data to ensure that the learned latent spaces can generalise. Clustering performance is measured in terms of the Silhouette Coefficient and the Davies-Bouldin index. The final performance scores associated with each combination were calculated as the average score achieved across five independent trials.

Results

In this section, we discuss the results[3] obtained from the various analyses carried out in this study. We begin with a discussion relating to the stability of models which incorporate a clustering loss. This is followed by the introduction of a training heuristic for this class of models. Next, we present the performance results for each component class. Finally, we show that our formulation outperforms the current state-of-the-art methods in the domain.

Model Stability

Throughout the development of the experimental framework, it was observed that invalid clusterings were produced significantly more frequently by certain component combinations than others. A clustering was deemed invalid if one

[2]Training details included in Appendix B.

[3]Source code and results available at https://github.com/TristanBester/berka_clustering

of the following two cases occurred: a degenerate cluster was produced containing all data points or the model was unable to produce cluster assignments as a result of divergence during the training process. In the first case both the SC and DBI are undefined. In the second case, the clusters cannot be computed as a result of numerical errors. The model cannot be evaluated if either of the cases occurs. Upon investigation, it was determined that the only common element between the affected combinations was the inclusion of a clustering loss. This divergent behaviour has previously been attributed to vanishing or exploding gradients (Lafabregue et al. 2022). For combinations that do not incorporate a clustering loss component, we agree with these findings. However, the majority of invalid clusterings are produced by combinations that incorporate a clustering loss as shown in Figure 1.
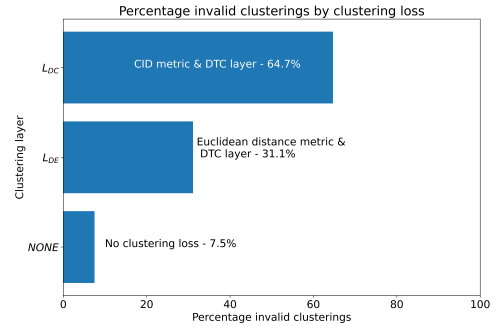

Figure 1: Percentage of invalid clusterings produced across all combinations. Results are shown for each clustering layer variant.

An investigation was carried out to determine the factors leading to the observed behaviour. The clustering loss component used in this study is that of the DTC model proposed by Madiraju (2018). To ensure the observed behaviour was not a consequence of the combination of disparate model components, the investigation was carried out using the exact DTC architecture described in the original paper. This restriction would verify that the behaviour was a property of the clustering layer rather than a consequence of the model combination strategy.

An investigation was conducted through the use of a synthetic dataset to test the relationship between the learning rates used in the pretraining and cluster optimisation phases. The learning rate used in the pretraining phase is that which is used to pretrain the autoencoder before cluster optimisation. Similarly, the learning rate used in the cluster optimisation phase is that which is used to minimise the clustering loss function. In the first stage of the investigation, the DTC autoencoder was pretrained on the experimental dataset. After this stage had been completed, the cluster centroids required to initialise the clustering layer are calculated as previously described. As the DTC autoencoder is being used for the experiment, both the latent representations and cluster centroids are time series with a reduced number of time steps. At this point, the autoencoder parameters and cluster centroids were fixed. This allowed for the initial state of

the clustering layer to be consistent across all subsequent experiments. Following this step, the learning rates used to train the clustering layers were varied and the results were recorded for analysis.

Polynomials of varying degrees were used to generate synthetic time series for the experimental dataset. Cluster labels were assigned to each of the time series in the dataset based on the degree of the polynomial from which it was generated. The pretraining phase was carried out on the dataset, followed by the centroid initialisation process. The latent space produced by the encoder at this point is illustrated in Appendix C. A learning rate of η_{pre} was used in the pretraining phase and η_{cls} in the cluster optimisation phase. The effect of varying the learning rate used in the cluster optimisation phase is illustrated in Figure 2.
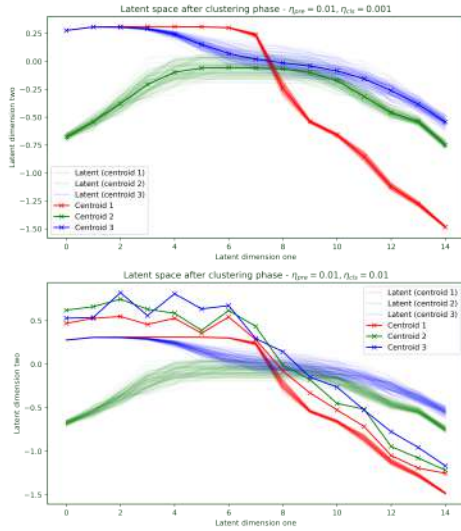
Figure 2: The effect of varied learning rates in the cluster optimisation phase. In the lower plot, it can be seen that the cluster centroids remain in their initial positions while the latent representations all converge to a single representation. Consequently, all data points are assigned to the same cluster. The stability of the upper model is clear from the converged latent space representation.[4]

After analysis of the recorded results, a subset of which have been presented, the cause for the behaviour was identified. If the learning rate used in the cluster optimisation phase was greater than or equal to that of the pretraining phase, the model was significantly more likely to produce an invalid clustering. Moreover, it was observed that model stability could be significantly improved by setting the learning

---

[4]The loss curve of the upper model flattened out towards the end of training indicating that the model had converged to a stable solution rather than requiring more iterations to diverge as a result of the smaller learning rate.

rate used in the cluster optimisation phase an order of magnitude smaller than that used in the pretraining phase.

Performance Results

An analysis was conducted to assess whether or not clusters separate consumers based on human-understandable properties of transaction histories. It was found that in general clusters associate consumers with similar transaction behaviour (i.e. salaried employees are separated from individuals with irregular income) however counterexamples exist within the clusters. As a result, we feel that research into human-interpretable performance metrics for transaction history clustering is an interesting avenue for future work.

From the results, it is clear that no single combination outperforms all others in each test. That is, for each performance metric and number of clusters, there is no single best algorithm. However, certain combinations are more commonly associated with higher performance than others. The performance associated with each component is calculated as the average performance achieved by all combinations in which the component is used. The results obtained for each component class are described below.

Autoencoder Architecture The results obtained for the autoencoder architecture component are consistent across both performance metrics. This is illustrated in Figure 3. For this component class, the CNN-based component is associated with the highest performance. This is consistent with the results obtained by Lafabregue et al. (2022) on the univariate UCR Archive[5].
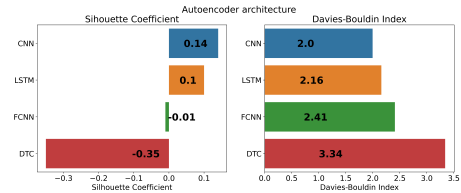

Figure 3: Average clustering performance associated with each autoencoder architecture.

Pretext Loss Function The classical reconstruction loss, equation (1), is associated with the highest average performance across all combinations. The results are illustrated in Figure 4. Our results are in agreement with Lafabregue et al. (2022) in which it is stated that the representations learned for the generation task, equation (3), are incompatible with clustering.

Clustering Loss Function Higher performance is associated with combinations in which a clustering loss is not used. As illustrated in Figure 5, a significant increase in average performance is obtained with the removal of the clustering loss. Contrary to the results presented by Lafabregue et al. (2022), 28 out of the 30 highest performance combinations made use of a clustering loss. It was previously concluded

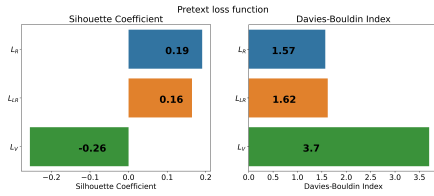---

[5]https://www.cs.ucr.edu/~eamonn/time_series_data

Figure 4: Average clustering performance associated with each pretext loss function.

that the use of existing clustering losses is not relevant for time series. This is likely a result of the fact that the learning rate used in the clustering phase was not tailored to increase model stability. Consequently, the models performed poorly or produced invalid clusterings. We rely on these results in the construction of the FTHC formulation. That is when a stable solution is obtained with the use of a clustering loss, the combination will perform better than if the component had been omitted.
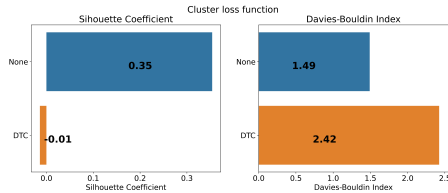

Figure 5: Average clustering performance associated with each clustering loss function.

Clustering Distance Metrics Of the two functions used as metrics in the latent space, the Euclidean distance was associated with the highest performance. On average when Euclidean distance was used as the clustering layer metric function, the performance was significantly higher than that associated with complexity invariant distance. These results are shown in Figure 6.
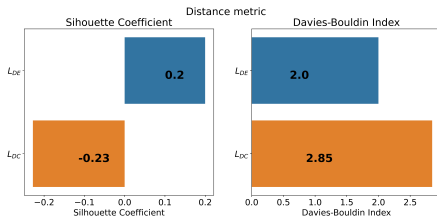

Figure 6: Average performance associated with clustering loss function metrics.

Dimensionality reduction

Across all component combinations, principal component analysis and the omission of dimensionality reduction exhibit similar average performance characteristics. The incorporation of UMAP as a dimensionality reduction technique shows slightly decreased performance as illustrated in Figure 7.
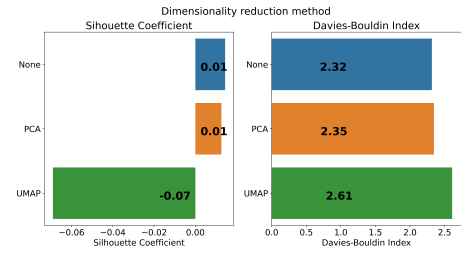

Figure 7: Average clustering performance associated with dimensionality reduction techniques.

Financial Transaction History Clustering (FTHC)

In this section, we define a novel approach designed for financial time series clustering before establishing a strong performance baseline for the domain. Guided by the previously discussed results, we combine the following components to produce a novel approach known as *Financial Transaction History Clustering*. We make use of the CNN-based autoencoder architecture based on ResNet. The autoencoder is trained using the classical reconstruction loss as a pretext loss function. The encoder is trained with the DTC clustering loss function with Euclidean distance as the metric. We compare our approach to the current state-of-the-art techniques in financial time series clustering presented by Lafabregue et al. (2022); Barkhordar, Shirali-Shahreza, and Sadeghi (2021). From the results presented in Figure 8, it can be seen that FTHC outperforms all of the current state-of-the-art approaches both in terms of the SC as well as the DBI.
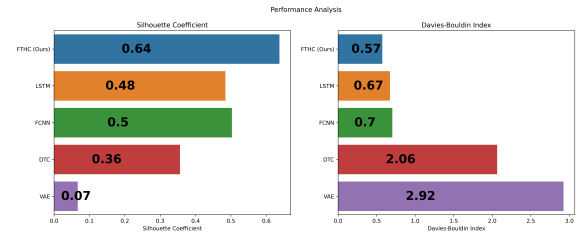

Figure 8: Performance results comparing our approach to the current state-of-the-art methods. The K-means clustering algorithm is used to form clusters in the latent space of the *LSTM, FCNN* and *VAE* autoencoders. The *DTC* approach forms clusters through the use of a clustering layer. Results were averaged across five independent trials.

Conclusion

In this paper, we have conducted a study to compare a variety of time series clustering algorithms based on deep representation learning. The algorithms are compared based on their ability to cluster univariate time series of consumer bank transaction data. We have shown that deep learning-based clustering methods can be decomposed into four components, namely, the architecture, dimensionality reduction

technique, pretext loss function and clustering loss function. Based on the proposed taxonomy, we have conducted a cross-comparison to evaluate each component's influence on clustering performance. We combine the top-performing components to create a novel clustering algorithm which is shown to outperform the current state-of-the-art approaches both in terms of the silhouette coefficient as well as the Davies-Bouldin index.

References

Alshehadeh, A. R.; and Al-Khawaja, H. A. 2022. Financial Technology as a Basis for Financial Inclusion and its Impact on Profitability: Evidence from Commercial Banks. *Int. J. Advance Soft Compu. Appl*, 14(2).

Ansari, A.; Riasi, A.; et al. 2016. Customer clustering using a combination of fuzzy c-means and genetic algorithms. *International Journal of Business and Management*, 11(7): 59–66.

Barkhordar, E.; Shirali-Shahreza, M. H.; and Sadeghi, H. R. 2021. Clustering of Bank Customers using LSTM-based encoder-decoder and Dynamic Time Warping. *arXiv preprint arXiv:2110.11769*.

Berka, P.; et al. 2000. Guide to the financial data set. *PKDD2000 discovery challenge*.

Block, H. D.; Knight Jr, B.; and Rosenblatt, F. 1962. Analysis of a four-layer series-coupled perceptron. II. *Reviews of Modern Physics*, 34(1): 135.

Dempster, A.; Petitjean, F.; and Webb, G. I. 2020. ROCKET: exceptionally fast and accurate time series classification using random convolutional kernels. *Data Mining and Knowledge Discovery*, 34(5): 1454–1495.

Ghasedi Dizaji, K.; Herandi, A.; Deng, C.; Cai, W.; and Huang, H. 2017. Deep clustering via joint convolutional autoencoder embedding and relative entropy minimization. In *Proceedings of the IEEE international conference on computer vision*, 5736–5745.

Guo, X.; Gao, L.; Liu, X.; and Yin, J. 2017. Improved deep embedded clustering with local structure preservation. In *Ijcai*, volume 17, 1753–1759.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

Hochreiter, S.; and Schmidhuber, J. 1997. Long short-term memory. *Neural computation*, 9(8): 1735–1780.

Hopfield, J. J. 1982. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, 79(8): 2554–2558.

Hotelling, H. 1933. Analysis of a complex of statistical variables into principal components. *Journal of educational psychology*, 24(6): 417.

Hubert, L. 1974. Approximate evaluation techniques for the single-link and complete-link hierarchical clustering procedures. *Journal of the American Statistical Association*, 69(347): 698–704.

Ismail Fawaz, H.; Forestier, G.; Weber, J.; Idoumghar, L.; and Muller, P.-A. 2019. Deep learning for time series classification: a review. *Data mining and knowledge discovery*, 33(4): 917–963.

Kingma, D. P.; and Welling, M. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Lafabregue, B.; Weber, J.; Gançarski, P.; and Forestier, G. 2022. End-to-end deep representation learning for time series clustering: a comparative study. *Data Mining and Knowledge Discovery*, 36(1): 29–81.

Ma, Q.; Zheng, J.; Li, S.; and Cottrell, G. W. 2019. Learning representations for time series clustering. *Advances in neural information processing systems*, 32.

MacQueen, J.; et al. 1967. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, 281–297. Oakland, CA, USA.

Madiraju, N. S. 2018. *Deep temporal clustering: Fully unsupervised learning of time-domain features*. Ph.D. thesis, Arizona State University.

McInnes, L.; Healy, J.; and Melville, J. 2018. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*.

Meng, Q.; Catchpoole, D.; Skillicom, D.; and Kennedy, P. J. 2017. Relational autoencoder for feature extraction. In *2017 International joint conference on neural networks (IJCNN)*, 364–371. IEEE.

Moin, K. I.; and Ahmed, D. Q. B. 2012. Use of data mining in banking. *International Journal of Engineering Research and Applications*, 2(2): 738–742.

Müller, M. 2007. Dynamic time warping. *Information retrieval for music and motion*, 69–84.

Taifi, N. 2023. The Intertwined Role of Big Data with Business Intelligence and Real Time Analysis: A Finance Sector Perspective. In *2023 International Conference on Digital Age & Technological Advances for Sustainable Development (ICDATA)*, 17–23. IEEE.

Wang, Z.; Yan, W.; and Oates, T. 2017. Time series classification from scratch with deep neural networks: A strong baseline. In *2017 International joint conference on neural networks (IJCNN)*, 1578–1585. IEEE.

Xiao, Z.; Xu, X.; Xing, H.; Luo, S.; Dai, P.; and Zhan, D. 2021. RTFN: A robust temporal feature network for time series classification. *Information sciences*, 571: 65–86.

Xie, J.; Girshick, R.; and Farhadi, A. 2016. Unsupervised deep embedding for clustering analysis. In *International conference on machine learning*, 478–487. PMLR.

Appendix

Appendix A: Component Configurations

(i) Autoencoder Architecture The configuration of each architecture depends on a large number of hyperparameters. As a result, we have decided to use the configurations used in other articles for our experiments as described below.

- *FCNN* – The fully connected neural network architecture proposed in (Xie, Girshick, and Farhadi 2016; Guo et al. 2017) will be used to represent this class of autoencoder architectures. The encoder is composed of three FCNN layers. The number of neurons in each layer is 500, 500 and 2,000 respectively. An embedding layer is used to map the data into the latent space. The decoder is constructed as a mirror of the encoder with the exception of the embedding layer.

- *CNN* – The convolutional neural network architecture is based on the ResNet architecture described in (He et al. 2016). The encoder is composed of three residual blocks followed by a global average pooling layer. A fully connected layer is used as the embedding layer. Similar to the FCNN, the decoder is constructed as a mirror of the encoder with the exception of the embedding layer.

- *LSTM* – The first recurrent neural network architecture is based on the LSTM. The encoder is comprised of a stacked, two-layer bidirectional LSTM. The network outputs associated with the forward and backward passes at each time step are concatenated to form the final output. The final hidden state of the network is used as the latent representation. The decoder is constructed as a mirror of the encoder and latent representation is upsampled to ensure that the input and output sequence lengths are identical.

- *DTC* – The second recurrent neural network architecture has been proposed in Madiraju (2018). Notably, the latent representation produced by this network is a time series with a reduced number of time steps. The encoder is composed of a convolutional layer followed by a max-pooling layer with a kernel size of 10. The output of this layer is then passed to a stacked, two-layer bidirectional LSTM with a hidden size of 50. Once again, forward and backward pass outputs are concatenated at each time step. The latent representation of the input time series is equal to the hidden state sequence produced by the encoder. The decoder is comprised of an upsampling layer followed by a deconvolutional layer to reconstruct the input time series.

(ii) Dimensionality Reduction

- *PCA* – Principal component analysis is used to reduce the dimensionality of the latent vectors before the clustering operation.

- *UMAP* – The UMAP algorithm is used to reduce the dimensionality of the latent vectors before clustering.

- *NONE* – In this case, no dimensionality reduction is performed. As a result, the latent representations are clustered directly.

(iii) Pretext Loss Function

- $\mathcal{L}_{\mathcal{R}}$ – Equation (1)
- $\mathcal{L}_{\mathcal{LR}}$ – Equation (2)
- $\mathcal{L}_{\mathcal{V}}$ – Equation (3)

(iv) Clustering Loss Function

- $\mathcal{L}_{\mathcal{DE}}$ – This is the DTC clustering loss function in which the Euclidean distance is used as the metric function. That is, the Euclidean distance is used to measure distances in the latent space of the autoencoder.

- $\mathcal{L}_{\mathcal{DC}}$ – This is the DTC clustering loss function in which Complexity Invariant Distance is used as the metric function. That is, the CID is used to measure distances in the latent space of the autoencoder.

- *NONE* – In this case, no clustering loss function is used. As a result, the model is an autoencoder in which clustering is performed in the latent space.

Appendix B: Model Training Procedure

All compatible component combinations were tested. Each combination was trained for 1,000 batch iterations in the pretraining phase and 1,000 batch iterations in the cluster optimisation phase. A constant learning rate of $\eta_{pre} = 10^{-2}$ was used in the pretraining phase and $\eta_{cls} = 10^{-3}$ in the cluster optimisation phase. The selection of these learning rates is motivated in the next section. For combinations in which a clustering loss is not included, training is terminated at the end of the pretraining phase.

For combinations that do not make use of a clustering layer, clusters are calculated using the K-Means algorithm (MacQueen et al. 1967) to cluster the latent representations produced by the model. This is not necessary with the inclusion of a clustering layer, as the layer produces cluster assignments directly.

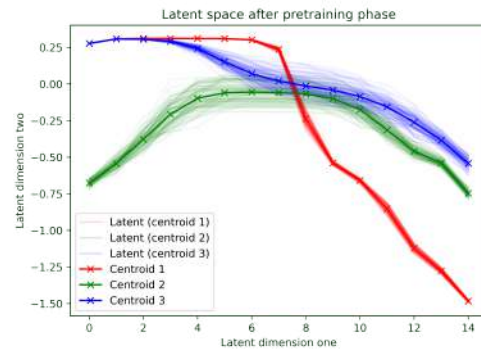## Appendix C: Latent Space After Pretraining

Figure 9: Visualisation of the autoencoder latent space after the pertaining phase.