Journal of Data Science, Statistics, and Visualisation

April 2025, Volume V, Issue II. *doi: 10.52933/jdssv.v5i2.118*

Permutation-Based Visualisation of Input and Encoded Space in Autoencoders

Alan Inglis
Maynooth University

Andrew Parnell
University College Dublin

Abstract

Autoencoders, known for their capability to compress and reconstruct data, play a pivotal role in unsupervised learning tasks. Deciphering the importance of input features and encoded dimensions can enhance the interpretability of the black-box nature of autoencoders. This paper introduces a permutation importance method tailored to evaluating the importance of raw pixel values and encoded dimensions in image data processed by autoencoders. We apply permutation importance in two stages: first, on the original image data to assess the impact of each pixel on the encoded representations; and second, on the encoded space to determine the importance of each dimension in the reconstruction process for different image classes. Our approach reveals how variations in input feature importance affect the encoded representations, shedding light on the encoder's focus and potential biases. Experimental results on benchmark image datasets and on a larger case study, concerning audio samples, demonstrate the efficacy of our method, providing a novel perspective on evaluating feature importance in unsupervised learning scenarios and offering greater interpretability of the inner workings of autoencoders. Our approach is implemented in the R package **aim** (Autoencoder Importance Mapping).

Keywords: Model visualisation, autoencoders, neural networks, variable importance.

1. Introduction

Autoencoders (AEs) are a type of neural network (NN) which have gained popularity due to their strong predictive power and flexibility across different types of data (Chen and Guo 2023). They are widely used in fields such as financial modelling for detecting anomalies and assessing risks (Nguyen et al. 2021), healthcare for analysing medical images and predicting diseases (Chen et al. 2017), natural language processing (NLP) (Dai and Le 2015), data denoising (Zhang et al. 2017), and image recognition (Pu et al. 2016; Amiriparian et al. 2017). AEs have proven particularly effective in image compression/dimension reduction and reconstruction (Theis et al. 2016), which are the central topics of this study. Dimension reduction helps machines process lower-dimensional data efficiently, which is crucial as handling high-dimensional data can be resource-intensive (Yang et al. 2016).

Fundamentally, AEs consist of an encoder that compresses the input into a lower-dimensional latent-space representation and a decoder that reconstructs the input from this encoded form. In the context of image compression and reconstruction, the goal is to closely match the original input by capturing essential data characteristics and ignoring less significant features. Activation functions introduce non-linearity, enabling the model to learn complex patterns. For image tasks, common choices are the Rectified Linear Unit (ReLU) and sigmoid functions, with ReLU outputting the input if positive and zero otherwise, while the sigmoid function maps inputs to values between 0 and 1. Performance is measured using loss functions that quantify the difference between the original input and the reconstructed output. Common choices include Mean Squared Error (MSE) and Binary Cross-Entropy (BCE). Optimisers, such as Adam (Adaptive moment estimation) (Kingma and Ba 2014), adjust the model's weights to minimise the loss function, improving training efficiency and effectiveness. An example of the basic architecture is shown in Figure 1. For more details on AE architecture, see Baldi (2012).

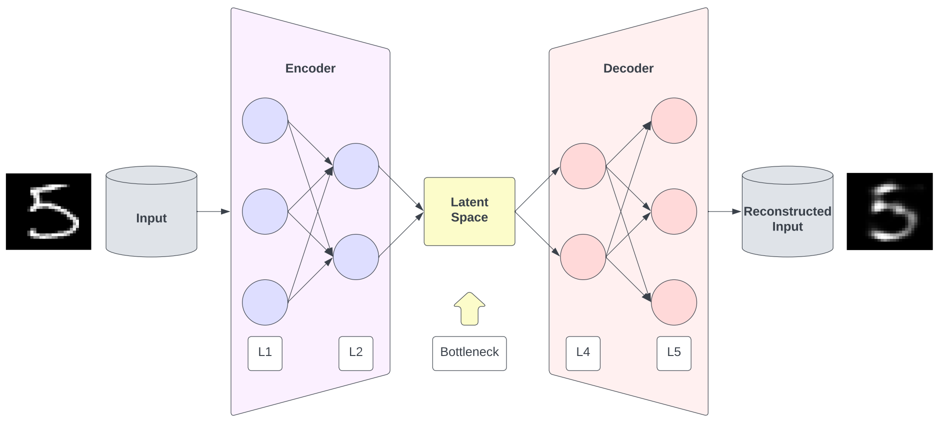


Figure 1: Basic autoencoder architecture. The process starts with an input image, which undergoes compression by the encoder through layers L1 and L2 into a lower-dimensional latent space representation, called the bottleneck. The decoder then reconstructs the image from the latent space through layers L4 and L5, resulting in the output that closely approximates the original input.

Given the critical applications of autoencoders in sectors such as healthcare and finance, ensuring their decisions are trustworthy is a key concern (Liu et al. 2022). However, the interpretability of these models presents a significant challenge. AEs, like many deep learning models, operate as *black-box* systems where the internal workings and the logic behind decisions remain largely opaque. This opacity is primarily due to their reliance on complex non-linear transformations, and the presence of numerous parameters, which complicate the tracing and understanding of how inputs are transformed into outputs. Furthermore, the difficulty in interpreting these models also hinders the ability to diagnose and rectify flaws in their architecture or training data (Molnar 2020). For example, biases embedded in the training data can inadvertently be learned by the model, propagating these biases in its outputs. Without a clear understanding of the model's decision-making process, identifying and correcting these biases becomes a complex challenge (Geman et al. 1992).

In our work, we evaluate various aspects of an AE model by shedding a light on the data's underlying structure and the network's learning processes through a dual permutation-based interpretability framework. Our approach analyses both the input data and the encoded latent space. By permuting input features and latent dimensions, we uncover their respective roles in shaping latent representations and reconstruction quality. Additionally, we visualise the importance of each input pixel within the latent space and the importance of latent dimensions, providing intuitive insights into the model's behaviour. Furthermore, by incorporating regression-based directional importance analysis, our method provides deeper insights into how specific inputs influence encoded representations. Our approach is divided into three distinct parts, which we summarise here, but for an in-depth explanation see Section 3:

1. **Input Pixel Importance Method:** We measure the importance of input pixels on the latent space by permuting the values of each pixel in the input data and observing the impact on the AE's performance. This allows us to gauge the relative importance of each pixel within the learned representation. Additionally, we fit a regression model for each pixel against the encoded dimension to determine the directional importance by examining the sign of the slope. Figure 2, steps 1 and 2 shows the basic workflow process.

2. **Latent Space Importance Method:** We apply permutation importance to the latent space data to identify which latent dimensions are important for reconstructing images of specific classes. Figure 2, step 3 shows the basic workflow of this process.

3. **Analysis and Visualisation:** To enable a coherent picture of what the AE is doing, we combine the findings from both input pixel importance and latent space importance methods into a single visualisation. This combination allows us to comprehensively understand how individual input pixels and latent dimensions contribute to the AE's performance and reconstruction accuracy.

We have developed an R package, called **aim** (Autoencoder Importance Mapping) that includes our permutation methods and various plotting functions and can be found at `https://github.com/AlanInglis/AIM`. This package also features a Shiny (Chang

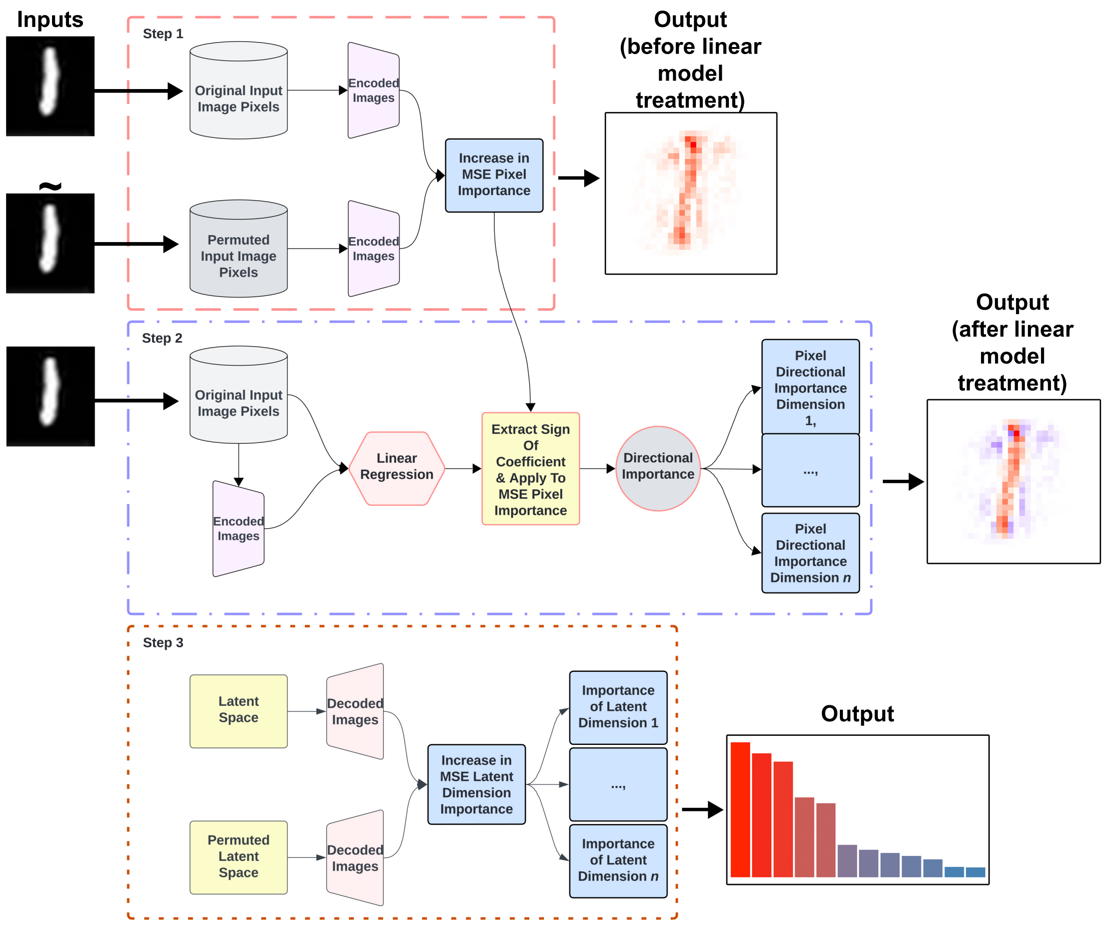

Inputs
Step 1
Original Input Image Pixels
Encoded Images
Permuted Input Image Pixels
Encoded Images
Increase in MSE Pixel Importance
Output (before linear model treatment)
Step 2
Original Input Image Pixels
Encoded Images
Linear Regression
Extract Sign Of Coefficient & Apply To MSE Pixel Importance
Directional Importance
Pixel Directional Importance Dimension 1,
...,
Pixel Directional Importance Dimension *n*
Output (after linear model treatment)
Step 3
Latent Space
Decoded Images
Permuted Latent Space
Decoded Images
Increase in MSE Latent Dimension Importance
Importance of Latent Dimension 1
...,
Importance of Latent Dimension *n*
Output

Figure 2: Workflow of our pixel and latent space importance methods using a handwritten digit ‘1’ as an example. In Step 1, pixel importance is calculated by encoding both original and pixel-permuted images and computing their difference using mean square error (MSE), with the importance values visualised as a heatmap. In Step 2, linear models are fitted to each input pixel using the encoded representations, where the sign of each model’s coefficient determines the direction of pixel influence on the encoding, shown as red-blue directional importance maps for each output dimension. In Step 3, latent space importance is calculated by first decoding from the encoded latent space to establish a baseline, then individually permuting each latent dimension and decoding again, with the MSE between baseline and permuted decodings quantifying each dimension’s importance, displayed as a bar plot of importance values.

et al. 2023) app that showcases both directional pixel importance and encoded dimension importance jointly. All plots in Sections 4 and 5 were generated using the **aim** package. A demonstration of the Shiny app is shown in Figure 3, with a more detailed explanation of the app available in Appendix B.

The layout of our paper is as follows. In Section 2 we discuss previous related work with regard to interpreting AEs. In Section 3 we describe the permutation importance algorithm and our methods and visualisations. In Section 4, we examine two benchmark image datasets and demonstrate our new techniques. In Section 5, we apply our methods to a case study using more complex audio data. In Section 6 we discuss some of the practical applications of our methods. Finally, in Section 7, we provide some concluding remarks and discuss limitations of our proposed methods. This study was performed using the **keras** (Allaire and Chollet 2023) and **tensorflow** (Allaire and Tang 2023) packages in the R (R Core Team 2023) language environment. To reproduce the examples in this text, all the code has been made available at `https://github.com/AlanInglis/AutoEncoder_Paper`.

2. Background

AEs have been widely studied for their flexibility and effectiveness in various tasks, yet their interpretability and methodological advancements remain areas of active research. Efforts to address the interpretability of AEs include methods that simplify their understanding or provide insights into their internal processes. For example, Aguilar et al. (2022) introduced an AE based on decision trees for categorical data, eliminating the need for transformation. Additionally, Shankaranarayana and Runje (2019) adapted the Local Interpretable Model-Agnostic Explanations (LIME) framework (Ribeiro et al. 2016) to work with AEs. Traditionally, LIME generates explanations for individual instances by creating a synthetic dataset through random sampling and perturbations around a specific instance, followed by training a local, interpretable linear model. In the adaptation by Shankaranarayana and Runje (2019), an AE is used to improve the weighting function of the local model.

The selection of latent space dimensionality is an important aspect of AE design. Mai Ngoc and Hwang (2020) proposed methods to determine the optimal dimensionality by balancing reconstruction accuracy with compression. Similarly, Gadirov et al. (2021) assessed different AE architectures for spatial ensemble dimensionality reduction, focusing on reconstruction quality and feature preservation. In the broader context of unsupervised representation learning, Cavallari et al. (2018) explored convolutional and stacked AEs, analysing feature spaces in domain and cross-domain contexts. Their work highlighted the role of architectural components, such as dense and convolutional layers, in determining learned representations. Variational Autoencoders (VAEs), introduced by Kingma and Welling (2013), advanced the field by integrating probabilistic modelling into AEs, enabling them to learn meaningful latent representations. Building on this, Burda et al. (2015) proposed the Importance Weighted Autoencoder (IWAE), which uses a stricter log-likelihood lower bound. This enhancement allows the recognition network to better model complex posteriors, resulting in richer and more expressive latent space representations compared to traditional VAEs. Extensions of VAEs to specialised domains have also emerged, such as the permutation-invariant VAE proposed by Winter et al. (2021) for graph-level unsupervised representation learning. This model addresses the high complexity of graph representations by indirectly matching node orderings in input and output graphs, enabling effective graph-level reconstruction and representation learning.

Visualisation techniques have also been pivotal in interpreting AEs. Dimensionality reduction methods such as t-SNE (van der Maaten and Hinton 2008) and UMAP (McInnes et al. 2018) project high-dimensional latent spaces into two or three dimensions, revealing clustering and structure within the encoded data. Two key approaches for understanding AEs, especially in image recognition, are feature visualisation and pixel attribution. Feature visualisation aims to understand what features the network has learned by finding inputs that maximise the activation of specific network components, such as neurons, channels, or layers (Olah et al. 2017). Pixel attribution determines how much each input pixel contributes to the network's output, commonly visualised through saliency maps (Simonyan et al. 2013) that highlight important pixels, and can be categorised into two main approaches: gradient-based and perturbation-based methods (Molnar 2020).

Gradient-based methods compute the influence of input features by analysing gradients of the outputs with respect to the inputs. Several techniques have been developed to enhance these gradient-based interpretations, notably Integrated Gradients (Sundararajan et al. 2017), which provides a path-integral-based measure of feature importance, and SmoothGrad (Smilkov et al. 2017), which improves interpretability by reducing noise in saliency maps through gradient smoothing. Perturbation-based methods, such as LIME, take a different approach by systematically modifying input values (for example, by masking or adding noise to image regions, distinct from permutation which involves reordering values) to understand how these alterations affect the output. These interpretation methods have been successfully applied to AEs in various domains. For example, in Uzunova et al. (2019), the authors use a perturbation approach in medical image classification where an AE is trained to differentiate between different images of unhealthy and healthy tissues. Additionally, Krishnan et al. (2022) use a perturbation approach to explore the latent space of a VAE used in molecular design of protein inhibitors. For more information on AE and NN interpretability in general, see Zhang et al. (2021), Fan et al. (2021), or Zeiler and Fergus (2014). For image recognition, see Zhang et al. (2018), in which the authors propose a general method to modify convolutional NNs to increase interpretability. Finally, in Agarwal et al. (2021) the authors propose a neural additive model which joins the interpretability of GAMs with the complexity of NN.

3. Visualising Feature Importance in Autoencoders

In our work, we apply the permutation feature importance technique to both the input pixels and the encoded representations of the AE. Specifically, we assess the importance of each input pixel, denoted as $\mathbf{X}_i$ for pixel i, on the encoded dimensions $\mathbf{Z}_j$ ($j = 1, \dots, d$), as well as assessing the importance of each encoding dimension d in the encoded space $\mathbf{Z}$, and visualise the results. These visualisations not only enhance understanding of the model's performance across various processing stages but also provide insights into the influence of specific features and dimensions on image reconstruction, enabling a deeper exploration of the model's behaviour. In this section, we describe our methods to obtain the importance scores and demonstrate our visualisations using an AE specifically fit on the digits zero and one from the MNIST data (Yann 1998) for the purpose of illustration and simplicity, with the encoding dimensions, d, set to a size of 12. This approach was chosen to highlight the methods under discussion. The results are shown in Figure 3 via the use of our Shiny app (see Appendix B for a demonstration of the app's features) and are discussed in more detail in the subsequent subsections. For the full algorithm of our methods, see Algorithm 1. To start, we provide a brief overview of permutation importance, as understanding this will help the reader grasp our approach.

3.1. Permutation importance

Permutation importance, first proposed by Breiman (2001), is a technique used to determine which input variables $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ have the greatest effect on the predicted outcome of a machine learning model. The permutation algorithm begins by measuring the model's predictive accuracy using the baseline data. For each variable $\mathbf{x}_i$, the algorithm shuffles the variable's values across the dataset, creating a permuted dataset $\tilde{\mathbf{X}} = \{\mathbf{x}_1, \ldots, \text{shuffle}(\mathbf{x}_i), \ldots \mathbf{x}_n\}$. The predictive accuracy is then updated using this revised dataset. This shuffling breaks any relationship between $\mathbf{x}_i$ and the response variable, allowing us to assess how much the model depends on $\mathbf{x}_i$. The difference between the performance of the permuted model and the baseline model (e.g., via the increase in MSE) constitutes the permutation-variable importance score, V_i for $\mathbf{x}_i$. In practice the permutation is performed repeatedly for each variable to ensure robustness, thus creating multiple $\tilde{\mathbf{X}}^{(k)}$ data sets where $k = 1, \ldots, n_{\text{perm}}$.

3.2. Importance in encoded representations

To assess the importance of the encoded representations $\mathbf{Z}$, we apply the permutation feature importance method to the encoder component of the AE by permuting each of the input pixels $\mathbf{X}_i$ (see Figure 2). This approach allows us to identify which pixels significantly influence the encoded representations and, consequently, the quality of the reconstructed images. Using the encoder model E, we first encode the original unaltered test images to obtain the baseline encoded representations, $\mathbf{Z} = E(\mathbf{X})$. For each pixel i, we perform n_{perm} permutations, generating permuted datasets $\tilde{\mathbf{X}}^{(k)}$ and their corresponding encoded representations, $\tilde{\mathbf{Z}}^{(k)} = E(\tilde{\mathbf{X}}^{(k)})$. We then compute the MSE between the original and permuted encodings for each latent dimension j:

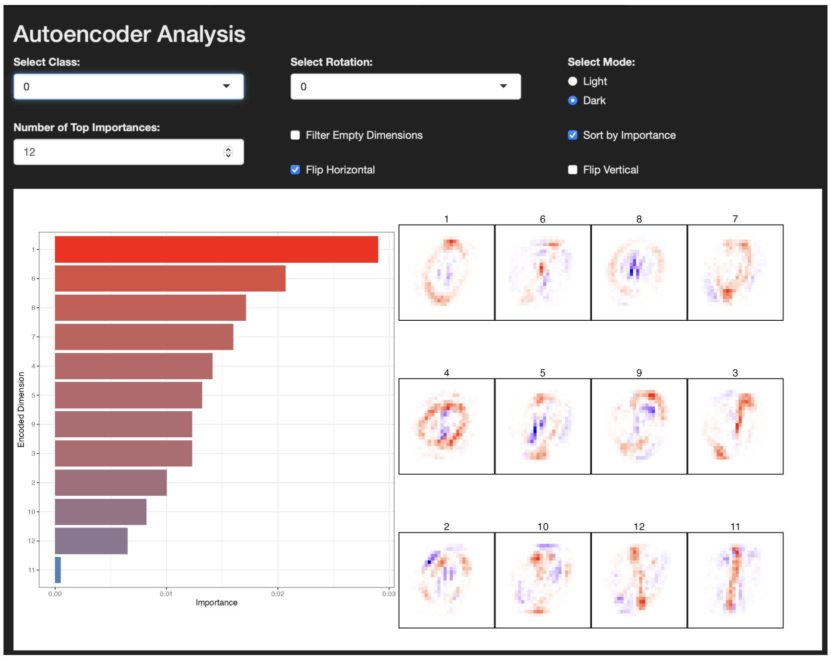
$$\mathbf{V}_{i,j} = \mathbf{V}_{i,j} + M\left(\mathbf{Z}_{[1:n],j},\ \tilde{\mathbf{Z}}^{(k)}_{[1:n],j}\right),$$

where n is the number of input images, M is the error metric (that is, MSE) and $\mathbf{V}_{i,j}$ is the resulting variable importance for pixel i in dimension j. We then average over the number of permutations to obtain the importance scores matrix $\mathbf{V}$.

To further refine our understanding, we adjust these scores by determining the direction of each pixel's influence on the encoded dimensions. This is achieved by fitting a linear regression model for each pixel $\mathbf{X}_i$ against each encoded dimension $\mathbf{Z}_j$:

$$\mathbf{Z}_{[1:n],j} = \beta_0^{(i,j)} + \beta_1^{(i,j)}\,\mathbf{X}_{[1:n],i} + \boldsymbol{\epsilon}.$$

We extract the sign of the regression coefficient $\beta_1^{(i,j)}$ to classify the importance as negative, neutral, or positive. Neutral values correspond to $\beta_1 = 0$ and are common when that pixel plays no role in the latent dimension, for example if it is constant. These directional signs are then combined with the permutation importance scores to produce the adjusted importance matrix $\mathbf{IP}$, revealing both the importance and direction of each pixel's impact. The $\mathbf{IP}$ values are then visualised as heatmaps, as



Autoencoder Analysis

Select Class: 0

Select Rotation: 0

Select Mode: Light [✓ Dark]

Number of Top Importances: 12

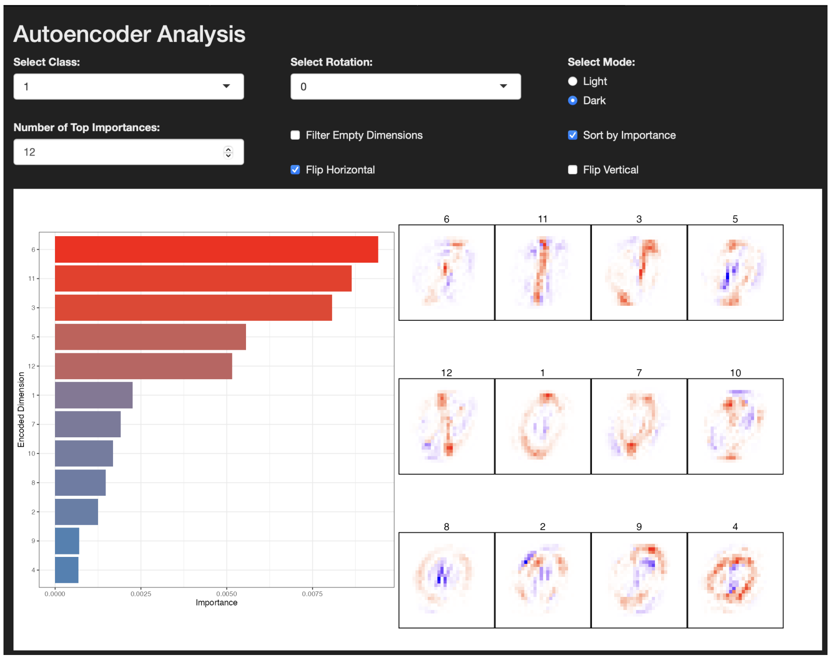
☐ Filter Empty Dimensions

☑ Sort by Importance

☑ Flip Horizontal

☐ Flip Vertical

(a) Digit 0



Autoencoder Analysis

Select Class: 1

Select Rotation: 0

Select Mode: Light [✓ Dark]

Number of Top Importances: 12

☐ Filter Empty Dimensions

☑ Sort by Importance

☑ Flip Horizontal

☐ Flip Vertical

(b) Digit 1

Figure 3: Shiny application permutation importance plot from the **aim** package, using the digits 0 and 1 from the MNIST data. Panel (a) shows, for the digit 0, the sorted importance of each encoded dimension in the image reconstruction in the left-hand barplot and the directional importance of the input pixels in the right-hand heatmaps. Panel (b) shows the importance for the same metrics for digit 1. In panel (a) we can see that latent dimension 1 has the most importance, which corresponds to a circular structure representing a zero in the right hand heat maps. In panel (b) we can see that latent dimensions 6, 11 and 3 are the most important, all of which represent different version of drawing a 1 in the right hand heat maps. Images can be rotated or flipped to allow for the correct orientation to be displayed.

seen in Figures 3a and 3b. An example of how this process aids in highlighting the patterns learned in the encoded dimensions is also shown in Appendix A.

3.3. Encoded dimension importance in image reconstruction

To assess the importance of individual encoded dimensions on image reconstruction quality, we apply the permutation feature importance method to the decoder component D, as outlined in Algorithm 1. For each class $c \in C$, we extract the class-specific samples $\mathbf{X}_c$ and obtain their encoded representations $\mathbf{Z}_c = E(\mathbf{X}_c)$. The baseline reconstruction $\hat{\mathbf{X}}_c = D(\mathbf{Z}_c)$ is then generated. Next, for each encoded dimension j, we perform n_{perm} permutations by shuffling the values of $\mathbf{Z}_{[1:n],j}$ across all samples to create permuted representations $\tilde{\mathbf{Z}}_c^{(k)}$. These permuted representations are then decoded to produce $\hat{\mathbf{X}}_c^{(k)}$, and the MSE between $\hat{\mathbf{X}}_c$ and $\hat{\mathbf{X}}_c^{(k)}$ is computed. Averaging these MSEs over all permutations yields the importance score $\mathbf{IE}_{c,j}$ for each dimension j. The resulting importance scores are visualised as barplots, as seen in Figures 3a and 3b.

3.4. Visualising feature and dimension importance

To visualise the encoded dimension importance in image reconstruction, we use a bar chart to depict the importance scores across different encoded dimensions. The barplots in Figure 3a and 3b show examples for the digits zero and one, respectively, with the x-axis displaying the importance scores and the y-axis displaying the encoded dimensions, sorted by importance. The importance of encoded dimension representations is further visualised in Figures 3a and 3b as heatmaps for each of the 12 encoding dimensions of our AE, also for the digits zero and one, respectively. These heatmaps are constructed by reshaping the importance scores of each pixel into the original image dimensions. Each panel of the heatmaps represent one dimension of the encoded space. For a direct comparison, the heatmaps are sorted according to the most important encoded dimensions. Each pixel is coloured on a diverging colour scale where blue represents negative values, red represents positive values, and the intensity of each colour indicates the importance. The mid-point is displayed as white and this represents when a pixel has a constant value and has no bearing on the latent dimension.

In Figure 3, we observe that the top three most important encoded dimensions for the digit zero are dimensions 1, 6, and 8, while for the digit one, they are dimensions 6, 11, and 3. This suggests that these specific encoded dimensions capture critical features necessary for accurate reconstruction. Further analysis of the heatmaps support this, showing varied patterns of importance across different dimensions. Specifically, in encoding dimension 1, the red-coloured pixels distinctly outline the shape of the digit zero, indicating its crucial role in recognising and reconstructing zeros. In contrast, in dimension 11, the red pixels form the straight line characteristic of the digit one, highlighting its importance for identifying and reconstructing ones. It is noteworthy that dimension 6 is among the top three most important encoded dimensions for both digits zero and one. This suggests that dimension 6 captures fundamental features that are important for the reconstruction of both digits. The shared importance of this dimension indicates that it may be encoding common structural elements or characteristics essential for distinguishing and reconstructing these digits accurately.

Algorithm 1: Permutation Importance for Encoded and Decoded Dimensions

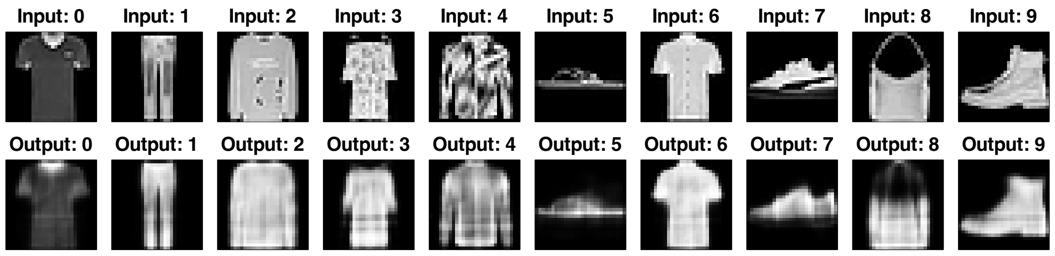
```
Input: Encoder model $E$; Decoder model $D$; Test labels $Y$; Set of classes $C$;
       Test data matrix $\mathbf{X}_{(n \times p)}$, where $n$ is the number of input images and $p$
       is the number of pixels; Number of latent dimensions $d$; Number of
       permutations $n_{\text{perm}}$; Error metric $M$.
Output: Adjusted pixel importance scores matrix $\mathbf{IP}_{(p \times d)}$; Encoded dimension
        importance scores vector $\mathbf{IE}_{(C \times d)}$.
1  Step 1: Permutation Pixel Importance
2  Initialise matrix of importance scores $\mathbf{V}_{(p \times d)}$ to zero.
3  Get original encoded representation: $\mathbf{Z}_{(n \times d)} = E(\mathbf{X})$.
4  for $i = 1$ to $p$ do
5      for $k = 1$ to $n_{perm}$ do
6          Create permuted data $\tilde{\mathbf{X}}^{(k)}$ by shuffling pixel $i$ values across samples.
7          Get permuted encoded representation: $\tilde{\mathbf{Z}}^{(k)} = E(\tilde{\mathbf{X}}^{(k)})$.
8          for $j = 1$ to $d$ do
9              Calculate MSE between original and permuted encodings:
                $\mathbf{V}_{i,j} = \mathbf{V}_{i,j} + M\left(\mathbf{Z}_{[1:n],j},\ \tilde{\mathbf{Z}}^{(k)}_{[1:n],j}\right)$.
10 Average importance over permutations: $\mathbf{V} = \mathbf{V}/n_{\text{perm}}$.
11 Step 2: Direction Adjustment
12 for $j = 1$ to $d$ do
13     for $i = 1$ to $p$ do
14         Fit linear regression between pixel $i$ and encoded dimension $j$:
15         $\mathbf{Z}_{[1:n],j} = \beta_0^{(i,j)} + \beta_1^{(i,j)} \mathbf{X}_{[1:n],i} + \boldsymbol{\epsilon}$.
16         Extract sign of regression coefficient: $\mathbf{S}_{i,j} = \text{sign}\left(\beta_1^{(i,j)}\right)$.
17 Adjust importance by direction sign: $\mathbf{IP} = \mathbf{V} \circ \mathbf{S}$ (where $\circ$ is the Hadamard
     product).
18 Step 3: Encoded Dimension Importance
19 for $c \in C$ do
20     Extract class-specific samples: $\mathbf{X}_c = \{\mathbf{x}_i \in \mathbf{X} \mid Y_i = c\}$.
21     Get encoded representation: $\mathbf{Z}_c = E(\mathbf{X}_c)$.
22     Get decoded reconstruction: $\hat{\mathbf{X}}_c = D(\mathbf{Z}_c)$.
23     Initialise matrix of importance scores $\mathbf{IE}_{C \times d}$ to zero.
24     for $j = 1$ to $d$ do
25         for $k = 1$ to $n_{perm}$ do
26             Create permuted encoding: $\tilde{\mathbf{Z}}_c^{(k)}$ by shuffling dimension $j$ values in
                $\mathbf{Z}_c$.
27             Get reconstruction from permuted encoding: $\hat{\mathbf{X}}_c^{(k)} = D\left(\tilde{\mathbf{Z}}_c^{(k)}\right)$.
28             Calculate MSE: $\mathbf{IE}_{cj} = \mathbf{IE}_{cj} + M\left(\hat{\mathbf{X}}_c,\ \hat{\mathbf{X}}_c^{(k)}\right)$.
29 Average importance over permutations: $\mathbf{IE} = \mathbf{IE}/n_{\text{perm}}$.
30 return  Adjusted pixel importance scores $\mathbf{IP}$.
31         Encoded dimension importance scores $\mathbf{IE}$.
```

Some dimensions, like 9 and 10, display seemingly random clusters of red or blue pixels that do not form any specific shape, likely encoding more abstract or less specific features. These abstract features contribute to the overall variability of the AE's representations and might capture more general characteristics of the digits or variations in handwriting styles. In Figure 3, there appears to be an inverse relationship between the important and unimportant dimensions. For example, in panel (a), dimension 8 is one of the most important dimensions for encoding the digit zero, while dimension 11 is the least important. In contrast, panel (b) shows the opposite, with dimension 11 being important for encoding the digit one, and dimension 8 is among the least important.

4. Method Demonstration with Example Datasets

Here we apply our methods on two benchmark data sets, both of which are variants of the the well known MNIST (Modified National Institute of Standards and Technology) dataset (Yann 1998). For each dataset, we fit an AE model following a consistent workflow. This included splitting the dataset into training and test (85-15), normalising and reshaping images, defining the model with 32 latent dimensions and 100 epochs, compiling with the Adam optimiser and binary crossentropy loss, and training with a batch size of 256. We then apply our permutation importance method setting the number of permutations to be four for each dataset. After testing, we found that four permutations allows us to capture a balance between computational efficiency and the stability of the importance estimates, ensuring reliable results without excessive computational cost.

The first data set used in this Section is the Extended-MNIST dataset (EMNIST) (Cohen et al. 2017) and consists of 28x28 pixel grayscale images of handwritten letters. In our work we select only the vowels (in English) from the dataset. This results in 16,800 training images and 2,800 test images. The second data set is the Fashion-MNIST dataset (FMNIST) (Xiao et al. 2017) and consists of 60,000 training and 10,000 testing 28x28 pixel grayscale images, each representing one of ten fashion categories and are described in the table in Figure 4. The FMNIST data can be obtained directly from the **keras** package.



0: T-shirt/top	1: Trouser	2: Pullover	3: Dress	4: Coat
5: Sandal	6: Shirt	7: Sneaker	8: Bag	9: Ankle boot

Figure 4: A selection of original (input - top row) and reconstructed (output - bottom row) images from an autoencoder using the FMNIST data. The table shows their labels.

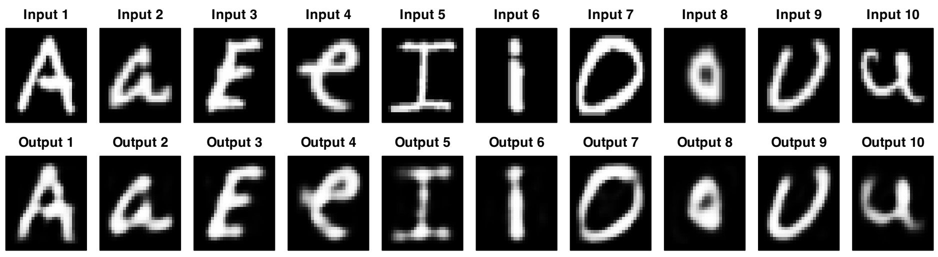
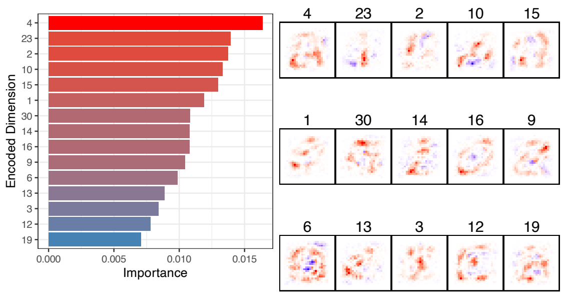
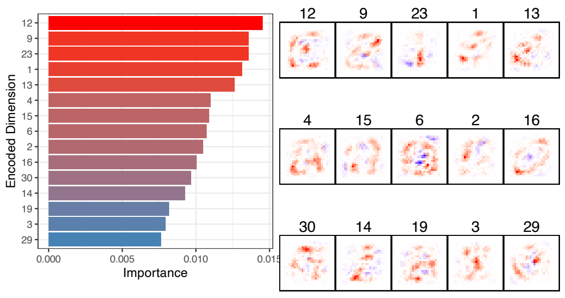


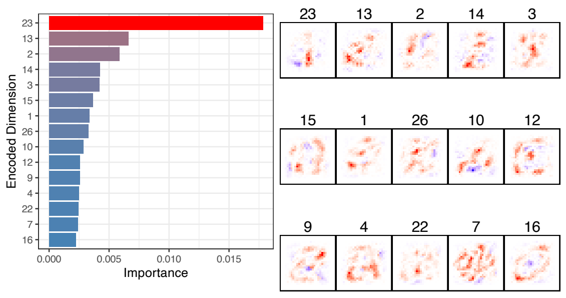
Figure 5: A selection of original (input - top row) and reconstructed (output - bottom row) images from at autoencoder using the EMNIST data.



(a) Class A



(b) Class E



(c) Class I

Figure 6: Top ten most important dimensions for encoded dimension and pixel importance, sorted by encoded dimension importance, across various classes of the EMNIST data. We can see that the most important encoded dimension typically outlines the shape of the respective class.

4.1. EMNIST data

For the EMNIST data, we build an AE by following the workflow previously outlined. Here we only train the model using only the uppercase and lowercase vowels and configure the AE as outlined previously. Figure 5 shows a selection of the original input images against their reconstructed output counterparts.

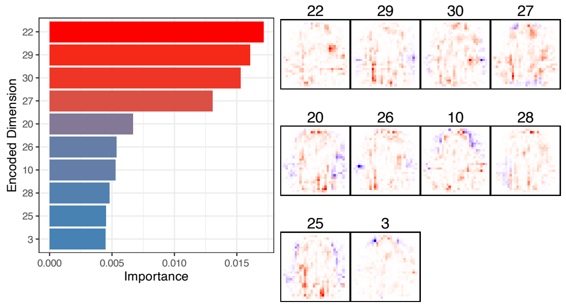
Figure 6 displays the top 15 most important encoded dimensions and the corresponding pixel importance for the classes A, E, and I, shown in 6a, 6b, and 6c, respectively. For clarity and size considerations, the remainder of the plots are not presented via our Shiny app. For plots of all classes, see Appendix C. In Figure 6, the red pixels in the heatmaps for each class's most important encoded dimension typically outline the shape of the respective class, highlighting the regions of the input that are most important for encoding and reconstruction. For class A, dimension 4 is the most important. The corresponding heatmap highlights features resembling both the uppercase and lowercase forms of the letter. Specifically, it captures the *"bowl"* (loop) of the lowercase letter on the left and the tail extending to the right, aligned with the crossbar of the uppercase A. For class I, dimension 23 stands out as the most important by a large margin, emphasising the vertical lines characteristic of the letter I. Interestingly, dimension 23 is among the top three most important encoded dimensions across all three classes, suggesting it represents fundamental features common to different digit structures. For class E, dimension 12 primarily encodes the distinctive structure of the uppercase letter E. Dimension 9, the second most important dimension for class E, appears to capture the characteristic curvature seen in some handwritten variations of the letter. Notably, dimension 9 has low importance within the top 15 encoded dimensions for the other classes, indicating that this feature is unique to the representation of E and less relevant for distinguishing other characters.

4.2. FMNIST data

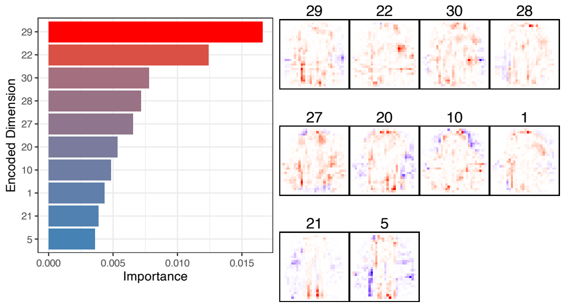
The FMNIST data contains images of 10 different types of fashion article classes and is commonly used as a more complex alternative to the traditional MNIST data. Each clothing class label is shown in the table in Figure 4. An AE was built, using the FMNIST data, as previously outlined, with 100 epochs. Figure 4 shows a sample of the original input images compared to the reconstructed images for each class.

Figure 7 shows the top ten most important dimensions for both the encoded dimension and pixel importance, sorted by the importance of the encoded dimension, across various classes. For brevity, we only show a selection of five classes (see Appendix D for plots of all classes). These are; *Shirt* in panel (a), *T-Shirt* in (b), *Sandal* in (c), *Ankle Boot*, in (d), and *Trouser* in (e). We can observe that items of clothing with similar physical characteristics also share the same most important encoded dimension. For example, *Shirt* and *T-Shirt* both share dimensions 22, 29, and 30 as their top 3 most important encoded dimensions. Also, *Sandal* and *Ankle Boot* (both being footwear) both share dimensions 9, 16, and 27 in their top four most important dimensions. This suggests that these dimensions play a crucial role in determining the basic shape of these classes.

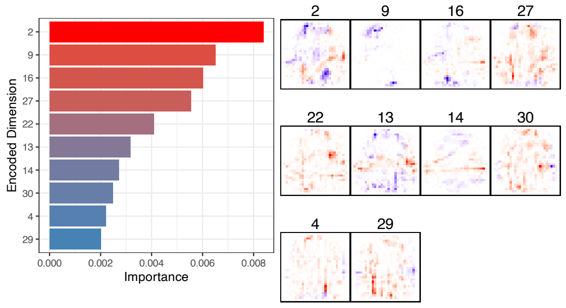
The heatmaps reveal distinct patterns of pixel importance across encoded dimensions for each class. For *Shirt* and *T-Shirt*, the heatmaps highlight the central region corresponding to the torso area, closely resembling the structure of these clothing items. In the case of *Trouser*, dimension 21 is uniquely the most important, followed by dimension 5. These dimensions display a series of straight lines in the highlighted pixels. Notably, these dimensions show minimal importance for other classes (see Appendix D for heatmaps of all 32 dimensions), indicating their particular sensitivity to the linear patterns characteristic of trousers. For the *Ankle Boot* class, dimension 15, the third most important, clearly outlines the shape of a boot, suggesting that this dimension effectively captures the overall form of this class. Additionally, dimension 22 consistently appears among the top 10 across all classes, hinting at its role in encoding a fundamental feature shared across different types of clothing.



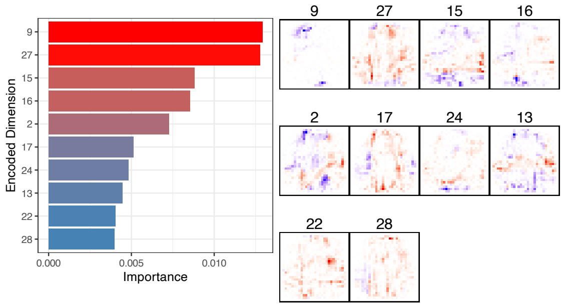
(a) Class *Shirt*



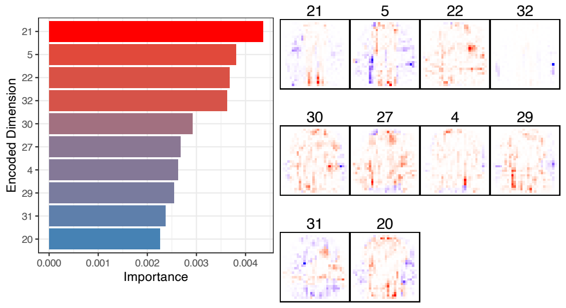
(b) Class *T-Shirt*



(c) Class *Sandal*



(d) Class *Ankle Boot*



(e) Class *Trouser*

Figure 7: Top ten most important dimensions for encoded dimension and pixel importance, sorted by encoded dimension importance, across various clothing classes from the FMNIST data.

5. Case Study: Analysing Audio MNIST Data

In this section, we apply our methods on a case study with more complex data. The dataset used in this study consists of 30,000 audio recordings from 60 individuals from various countries, each vocalising the digits from 0 to 9. It was originally studied by Becker et al. (2023) and the raw data can be found at `https://github.com/soerenab/AudioMNIST`. These audio samples are transformed into visual representations called spectrograms, an example of which is shown in Figure 8. The axes of a spectrogram represent frequency versus time, while the amplitude of the audio signal at different frequencies and times is depicted through varying colours in these images. In our case, the size of each image is set to 32x32 pixels. The following subsections detail the methodology employed in preparing the audio samples and the specific architecture of the autoencoder.

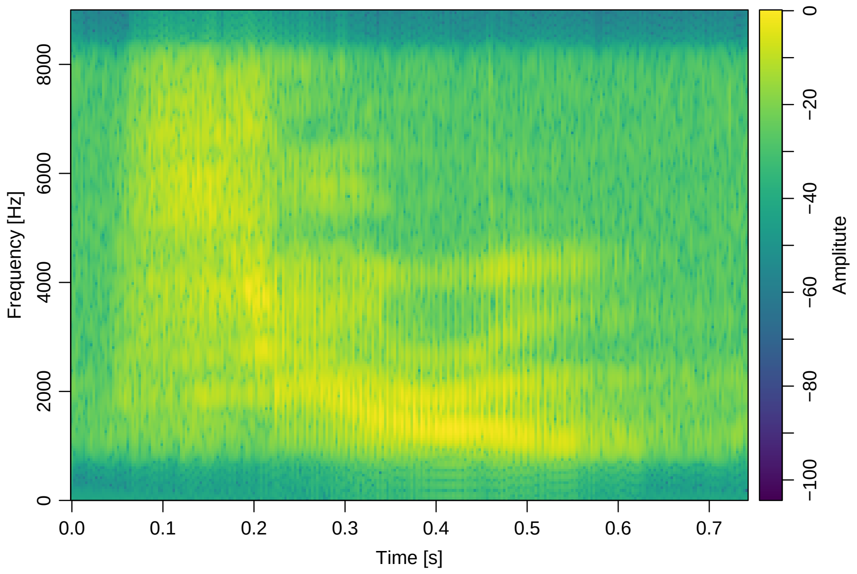


Figure 8: Example spectrogram of a person saying the number zero.

5.1. Data preparation

The raw audio data, visualised as spectrograms, are first pre-processed by smoothing the RGB colour channels to enhance the signal and reduce noise. The raw images have dimensions of 32x32x4, where the four channels represent the red, green, and blue colour channels and an alpha channel for transparency. Since the alpha channel is not needed for our analysis, it is discarded. Subsequently, using a kernel smoother for irregular 2D data, the RGB channels are merged into a single channel to represent the amplitude. Following this, the data is normalised to the range [0,1].

5.2. Autoencoder architecture

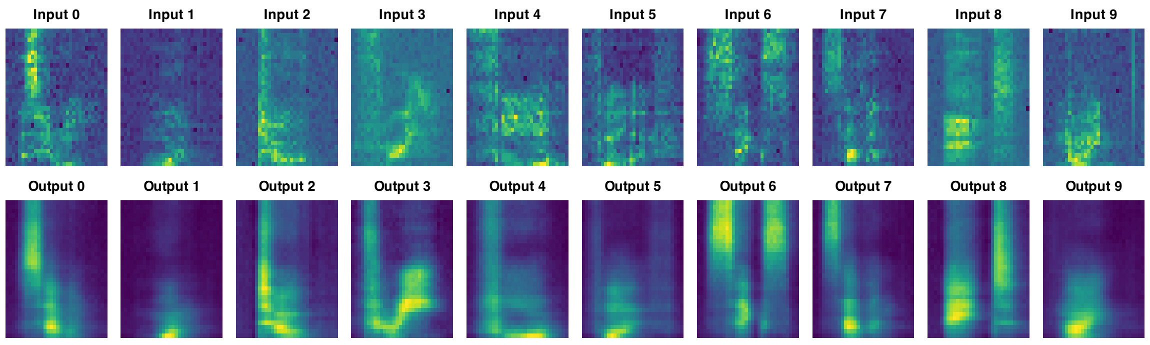


Figure 9: Autoencoder input (original spectrograms) and reconstructed outputs for digits 0–9. Each image depicts the spectrogram of the spoken digi. Digits with fricative sounds (for example, 0, 6, 7) show prominent high-frequency components.

The encoder of the autoencoder compresses the input, which consists of 1,024 features (flattened 32x32 images), into a compact representation of 32 features through eight dense layers. The number of units in these layers progressively decreases, before reaching the bottleneck layer of 32 units. All layers employ ReLU activation functions. The decoder reconstructs the original image from this compressed representation by mirroring the encoding process, expanding the 32 features back up to 1,024 features through a reverse sequence of dense layers, also using ReLU activation functions. The final output layer uses a sigmoid activation function to reconstruct the image. The autoencoder is compiled with the Adam optimiser and a binary cross-entropy loss function, reflecting the binary nature of the normalised pixel values. It is trained over 100 epochs with a batch size of 256 and validation with test images. Figure 9 shows the original versus reconstructed spectrograms for selected spoken digits, with each image corresponding to the specific sequential number. The autoencoder had the effect of smoothing and de-noising the original images. Of note here is that certain digits (for example, 0, 6, 7) exhibit pronounced high-frequency components, whereas digits such as 1 or 9 display stronger low-frequency energy.

5.3. Analysis of results

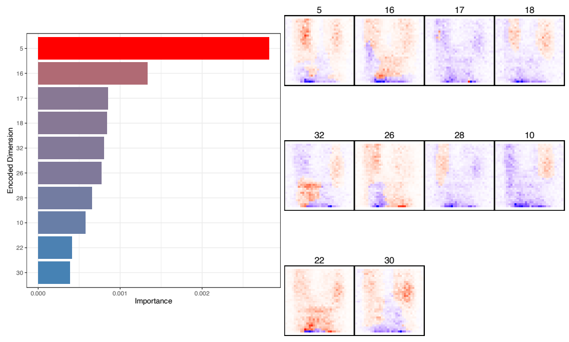
Encoded dimensions and pixel importance

In Figure 10, we show the top ten most important encoded dimensions and their corresponding pixel importance plots for classes 0, 4, 5, 6, 7, and 9 (see Appendix E for all classes). Here, we can see several patterns emerging. For example, certain dimensions, such as 16 and 26, consistently appear among the top ten across all selected classes. As indicated by the heatmaps, these dimensions capture features spanning both high and low-frequency ranges, suggesting that they encode acoustic characteristics across a broad frequency spectrum.

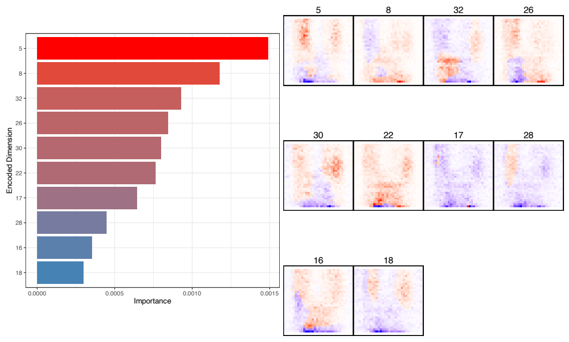
For classes 0, 6, and 7, dimension 5 is notably more important than the others. This may reflect unique pronunciation features of these digits, such as the fricative sounds ‘S’ in ‘six’ and ‘seven’ and ‘Z’ in ‘zero’, which require capturing distinct high-frequency bands and temporal patterns (this can be seen in Figure 9). Class 4 also has dimension 5 as its most important, possibly due to the acoustic similarity between ‘F’ and ‘S’, which both emphasise high-frequency turbulence, albeit in slightly different ranges. For class 5, dimension 23 is the most important, closely followed by dimension 30. Both highlight activity in two distinct bursts within the high-frequency range. These bursts may correspond to the fricatives ‘F’ and ‘V’ in ‘five’, as encoding their spectral and temporal characteristics requires focusing on high-frequency features. In contrast, class 9 shows dimension 32 as the most important, with its heatmap emphasising lower frequency bands. This is consistent with the nasal consonant ‘N’ in ‘nine’, which is characterised by strong low-frequency energy (which can be seen in Figure 9). Notably, classes 0, 4, 5, 6, and 7 consistently highlight higher frequency bands as important. For example, the fricatives ‘F’ in ‘four’ and ‘S’ in ‘six’ contribute high-frequency energy, reinforcing the need for dimensions capturing these regions.

Concluding remarks

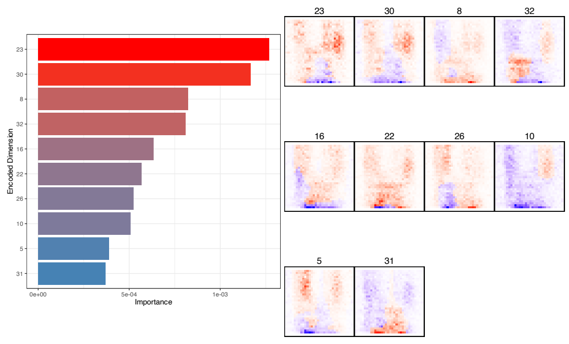
This case study demonstrates the effectiveness of using a permutation importance approach to analyse the reconstruction of audio data from spectrograms. By examining



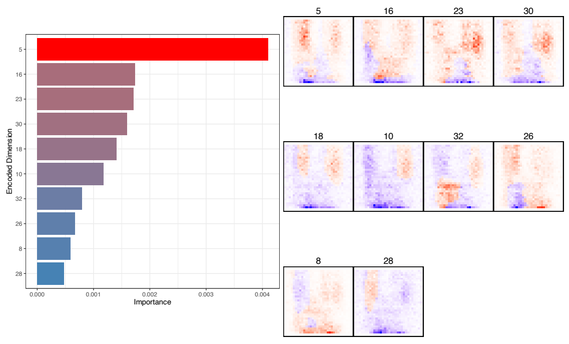
(a) Class 0



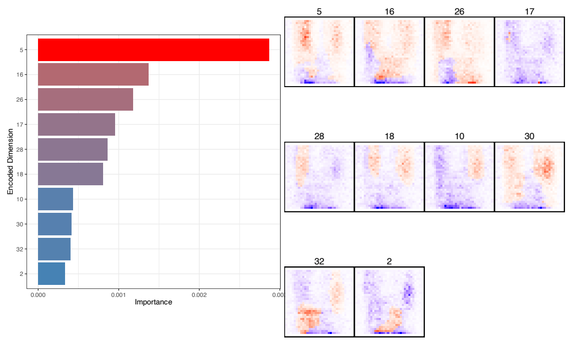
(b) Class 4



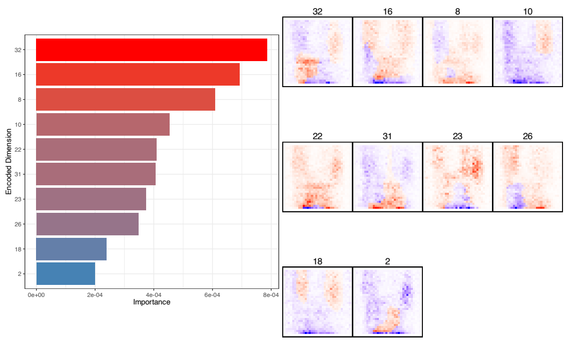
(c) Class 5



(d) Class 6



(e) Class 7



(f) Class 9

Figure 10: Top ten most important encoded and pixel dimensions, ranked by encoded dimension importance, across selected audio MNIST classes.

the importance of both raw input pixels and encoded dimensions, we have identified key features that are critical for accurate reconstruction. Heatmap visualisations revealed that certain encoded dimensions capture shared acoustic features across multiple digits, such as common phonetic properties, while others highlight unique characteristics specific to certain digits. For example, digits 0, 6, and 7 share many of the same importance profiles and highlight similar frequency bands in their pixel importance heatmaps, suggesting common acoustic properties and distinct phonetic features.

UMAP

Here, we used Uniform Manifold Approximation and Projection (UMAP) (McInnes et al. 2018) to visualise the encoded dimensions of the audio MNIST data. This helps us assess how effectively the AE separates the different classes in the encoded space. Using UMAP serves as a sanity check, ensuring that our subsequent results are consistent with the visual separation observed in the UMAP plot. The UMAP is a dimensionality reduction technique that is particularly well-suited for visualising high-dimensional data. It reduces the dimensions of the data to two or three, making it possible to plot the data points in a scatterplot. The technique aims to preserve both the local and global structure of the data, meaning that points that are close together in the high-dimensional space should remain close together in the low-dimensional space, while also maintaining the overall shape and distribution of the data. The UMAP plot in Figure 11 shows the resulting 2D visualisation of the AE applied to the audio MNIST data.

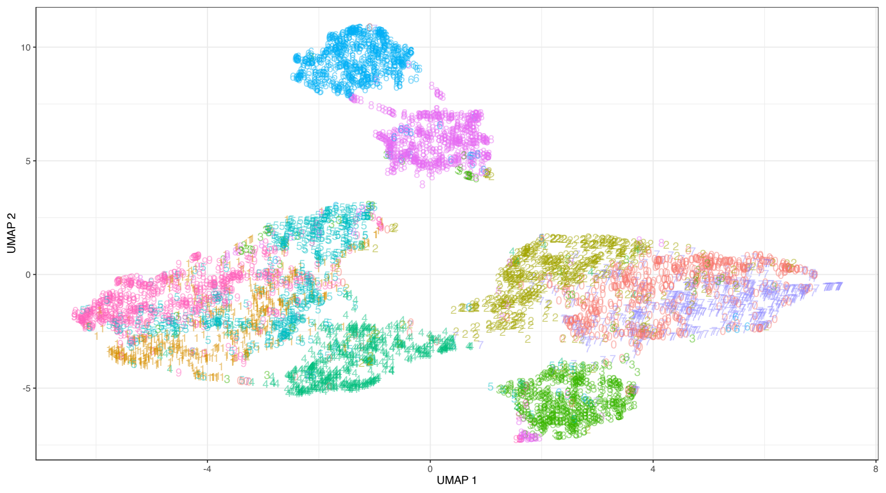


Figure 11: UMAP plot of the encoded dimensions where the label represents the number being spoken. We can see moderate class separation and noticeable clusters, particularly for digits 6 and 8.

Referring back to Figure 11, the overlapping regions of $\{0, 7\}$ in the UMAP plot can also be observed in Figure 10, as these classes share many of the same important dimensions.

In Figure 11, we can see a moderate separation of the classes, with some noticeable clusters corresponding to specific digits. For instance, digits such as $\{3, 4, 6, 8\}$ show well-defined clusters (with minimal cross-over of other digits), indicating that the AE has learned to encode these digits in a way that makes them distinct in the latent space. However, there are also regions with significant overlap, such as between digits $\{0, 2, 7\}$, or $\{1, 5, 9\}$, suggesting that the AE may struggle to distinguish between these digits as effectively. The overlapping regions of $\{0, 7\}$ in the UMAP plot can also be observed in Figure 10, as these classes share many of the same important dimensions. An interesting note here is that some classes appear as outliers. For example, at the edge of cluster 8 we can see a few instances of the classes 2 and 3. This may be due to mis-labelling or suggest that the feature representation of these particular digits share similarities with the 8 cluster, indicating ambiguity in the digit's shape or style.

6. Practical Implications

The findings from our permutation-based analysis have significant practical implications, demonstrating how knowledge about biases and key dimensions can be used to improve model performance and address potential biases in the data. For example, in our analysis of the Fashion-MNIST dataset (Section 4), we observed that the encoded dimensions important for the *Shirt* and *T-Shirt* classes captured similar features. This overlap suggests that the AE is encoding these two classes in a similar manner, potentially leading to difficulties in accurately reconstructing and differentiating between them. Recognising this issue, we can take practical steps to enhance the model's ability to distinguish between these classes.

One approach is to adjust the AE's architecture to better capture the unique characteristics of each class. By introducing additional layers or modifying existing ones to focus on features that differentiate *Shirts* from *T-Shirts*, we can encourage the model to learn more discriminative representations. For example, incorporating convolutional layers that specialise in detecting specific patterns associated with each class can improve the model's sensitivity to subtle differences. Another strategy involves augmenting the training data with variations that highlight distinguishing features between similar classes. By enriching the dataset with images that emphasise the unique aspects of *Shirts* and *T-Shirts*—such as variations in sleeve length or collar style etc., we provide the model with a broader range of examples from which to learn. This enhanced diversity helps the AE to form more distinct encoded representations for each class, improving reconstruction accuracy and class differentiation.

Understanding which encoded dimensions are most important for reconstruction also allows for targeted model optimisation. If certain dimensions are found to be less important, we can reduce the size of the latent space without significantly impacting performance. This reduction leads to a more efficient model with decreased computational requirements and a lower risk of overfitting. Simplifying the model in this way can also make it more interpretable, aiding in the identification and correction of biases. Moreover, preprocessing techniques can be guided by knowledge of important input features. For example, applying attention mechanisms or region-of-interest pooling can ensure that the model concentrates on critical areas such as the neckline or sleeve pattern, which are key to differentiating *Shirts* from *T-Shirts*.

The approach demonstrated here extends beyond the Fashion-MNIST dataset and can be applied to various domains where biases may exist. In healthcare, for example, certain dominant features in medical images might overshadow subtle but clinically significant information. By identifying and adjusting for this imbalance, we can improve diagnostic accuracy. In financial anomaly detection, understanding which features contribute most to reconstruction error can help identify systemic biases, leading to better risk assessment. In natural language processing, recognising key features in encoded representations can enhance language models by highlighting important linguistic structures or detecting biases in language usage.

7. Conclusion

In this paper, we presented a novel permutation importance method for evaluating the significance of raw pixel values and encoded dimensions in autoencoders applied to image data. By applying permutation importance at two stages (that is, on the original image data and the encoded space), we provided a detailed analysis of the feature importance in the encoding and reconstruction processes. Our approach reveals how variations in input feature importance affect the encoded representations, shedding light on the encoder's focus and potential biases. Additionally, we identified the key encoded dimensions that significantly impact the reconstruction quality for different image classes. Experimental results on benchmark image datasets, including Fashion-MNIST and EMNIST, demonstrated the efficacy of our method. We observed diverse patterns of pixel importance across different encoded dimensions, indicating that autoencoders learn a variety of features from the data. The case study on Audio MNIST data further validated our approach, showing that specific encoded dimensions capture crucial acoustic features for accurate reconstruction.

Our method enhances the interpretability of autoencoders, providing deeper insights into their inner workings. By understanding which features and dimensions are most important, we can improve model transparency and potentially address biases inherent in the training data. By identifying biases and key dimensions, practitioners can make informed adjustments to their models or data, leading to more robust and trustworthy applications. The practical examples demonstrated in Section 6 illustrate how knowledge about biases and key dimensions can be used to address potential issues and optimise models in real-world scenarios.

Future work could extend this methodology to other types of data and autoencoder architectures, such as variational autoencoders (VAE) or Large Language Models (LLMs), further broadening the scope and applicability of permutation importance in unsupervised learning scenarios. For VAEs, understanding the importance of encoded dimensions can aid in improving latent space representations and enhancing the quality of generated data. Similarly, for LLMs, which often rely on intricate encoding mechanisms to understand and generate human language, our approach could be adapted to provide valuable insights into which aspects of the input data are most important for generating coherent and contextually accurate outputs.

While our approach offers significant insights, it also has limitations. The computational cost of permutation importance can be high, especially for large datasets and complex models. Future work could include parallelisation of the permutation process to decrease computational time. Additionally, the method assumes that the importance of features is independent, which may not always hold true in practice. Addressing these limitations in future research could lead to more efficient and comprehensive interpretability methods for autoencoders and other deep learning models.

Computational Details

The results in this paper were obtained using R 4.3.1 and the R package **aim** available at: `https://github.com/AlanInglis/aim`. All additional packages used are available from the Comprehensive R Archive Network (CRAN) at `https://CRAN.R-project.`

org/. The source code and data for generating the experimental results is available at https://github.com/AlanInglis/autoencoder.

Acknowledgments

Alan Inglis's work is kindly supported by the Department of Agriculture, Food, and the Marine (DAFM) and the Department of Agriculture, Environment, and Rural Affairs (DAERA), grant number 2021R460 as part of the Mycotox-I project. Andrew Parnell's work was supported by: the UCD-Met Éireann Research Professorship Programme (28-UCDNWPAI); a Research Ireland, Northern Ireland's Department of Agriculture, Environment and Rural Affairs (DAERA), UK Research and Innovation (UKRI) via the International Science Partnerships Fund (ISPF) under Grant number [22/CC/11103] at the Co-Centre for Climate + Biodiversity + Water; the Research Ireland Centre for Research Training 18/CRT/6049; and Research Ireland Co-Centre Climate+ in Climate Biodiversity and Water award 22/CC/11103. For the purpose of Open Access, the authors have applied a CC BY public copyright licence to any Author Accepted Manuscript version arising from this submission.

References

Agarwal, R., Melnick, L., Frosst, N., Zhang, X., Lengerich, B., Caruana, R., and Hinton, G. E. (2021). Neural additive models: Interpretable machine learning with neural nets. 34:4699–4711, https://proceedings.neurips.cc/paper_files/paper/2021/file/251bd0442dfcc53b5a761e050f8022b8-Paper.pdf.

Aguilar, D. L., Medina-Pérez, M. A., Loyola-Gonzalez, O., Choo, K.-K. R., and Bucheli-Susarrey, E. (2022). Towards an interpretable autoencoder: A decision-tree-based autoencoder and its application in anomaly detection. *IEEE Transactions on Dependable and Secure Computing*, 20(2):1048–1059. DOI: 10.1109/TDSC.2022.3148331.

Allaire, J. and Chollet, F. (2023). ***keras**: R Interface to '**Keras**'*, https://CRAN.R-project.org/package=keras. R package version 2.13.0.

Allaire, J. and Tang, Y. (2023). ***tensorflow**: R Interface to '**TensorFlow**'*, https://CRAN.R-project.org/package=tensorflow. R package version 2.13.0.

Amiriparian, S., Gerczuk, M., Ottl, S., Cummins, N., Freitag, M., Pugachevskiy, S., Baird, A., and Schuller, B. (2017). Snore sound classification using image-based deep spectrum features. In *Proceedings of Interspeech 2017*, pages 3512–3516. DOI: 10.21437/Interspeech.2017-434.

Baldi, P. (2012). Autoencoders, unsupervised learning, and deep architectures. In *Proceedings of ICML Workshop on Unsupervised and Transfer Learning*, pages 37–49. JMLR Workshop and Conference Proceedings, https://proceedings.mlr.press/v27/baldi12a.html.

Becker, S., Vielhaben, J., Ackermann, M., Müller, K.-R., Lapuschkin, S., and Samek, W. (2023). AudioMNIST: Exploring explainable artificial intelligence for audio analysis on a simple benchmark. *Journal of the Franklin Institute*, 361(1):418–428, ISSN: 0016-0032, DOI: 10.1016/j.jfranklin.2023.11.038.

Breiman, L. (2001). Random forests. *Machine Learning*, 45:5–32, DOI: 10.1023/A:1010933404324.

Burda, Y., Grosse, R., and Salakhutdinov, R. (2015). Importance weighted autoencoders. *arXiv preprint arXiv:1509.00519*, DOI: 10.48550/arXiv.1509.00519.

Cavallari, G. B., Ribeiro, L. S.F., and Ponti, M. A. (2018). Unsupervised representation learning using convolutional and stacked auto-encoders: A domain and cross-domain feature space analysis. In *2018 31st SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*, pages 440–446. IEEE, DOI: 10.1109/SIBGRAPI.2018.00063.

Chang, W., Cheng, J., Allaire, J., Sievert, C., Schloerke, B., Xie, Y., Allen, J., McPherson, J., Dipert, A., and Borges, B. (2023). ***shiny**: Web Application Framework for R*, https://CRAN.R-project.org/package=shiny. R package version 1.7.5.

Chen, M., Shi, X., Zhang, Y., Wu, D., and Guizani, M. (2017). Deep feature learning for medical image analysis with convolutional autoencoder neural network. *IEEE Transactions on Big Data*, 7(4):750–758, DOI: 10.1109/TBDATA.2017.2717439.

Chen, S. and Guo, W. (2023). Auto-encoders in deep learning—a review with new perspectives. *Mathematics*, 11(8):1777, DOI: 10.3390/math11081777.

Cohen, G., Afshar, S., Tapson, J., and Van Schaik, A. (2017). EMNIST: Extending MNIST to handwritten letters. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 2921–2926. IEEE, DOI: 10.1109/IJCNN.2017.7966217.

Dai, A. M. and Le, Q. V. (2015). Semi-supervised sequence learning. *Advances in Neural Information Processing Systems*, 28, DOI: 10.48550/arXiv.1511.01432.

Fan, F.-L., Xiong, J., Li, M., and Wang, G. (2021). On interpretability of artificial neural networks: A survey. *IEEE Transactions on Radiation and Plasma Medical Sciences*, 5(6):741–760, DOI: 10.1109/TRPMS.2021.3066428.

Gadirov, H., Tkachev, G., Ertl, T., and Frey, S. (2021). Evaluation and selection of autoencoders for expressive dimensionality reduction of spatial ensembles. In *International Symposium on Visual Computing*, pages 222–234. Springer, DOI: 10.1007/978-3-030-90439-5_18.

Geman, S., Bienenstock, E., and Doursat, R. (1992). Neural networks and the bias/variance dilemma. *Neural Computation*, 4(1):1–58, DOI: 10.1162/neco.1992.4.1.1.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, DOI: 10.48550/arXiv.1412.6980.

Kingma, D. P. and Welling, M. (2013). Auto-encoding variational Bayes. *arXiv preprint arXiv:1312.6114*, DOI: 10.48550/arXiv.1312.6114.

Krishnan, K., Kassab, R., Agajanian, S., and Verkhivker, G. (2022). Interpretable machine learning models for molecular design of tyrosine kinase inhibitors using variational autoencoders and perturbation-based approach of chemical space exploration. *International Journal of Molecular Sciences*, 23(19):11262, DOI: 10.3390/ijms231911262.

Liu, H., Wang, Y., Fan, W., Liu, X., Li, Y., Jain, S., Liu, Y., Jain, A., and Tang, J. (2022). Trustworthy AI: A computational perspective. *ACM Transactions on Intelligent Systems and Technology*, 14(1):1–59, DOI: 10.48550/arXiv.2107.06641.

Mai Ngoc, K. and Hwang, M. (2020). Finding the best k for the dimension of the latent space in autoencoders. In *Computational Collective Intelligence: 12th International Conference, ICCCI 2020, Da Nang, Vietnam, November 30–December 3, 2020, Proceedings 12*, pages 453–464. Springer, DOI: 10.1007/978-3-030-63007-2_35.

McInnes, L., Healy, J., and Melville, J. (2018). UMAP: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, DOI: 10.48550/arXiv.1802.03426.

Molnar, C. (2020). *Interpretable Machine Learning*. Lulu.com. https://christophm.github.io/interpretable-ml-book/.

Nguyen, H. D., Tran, K. P., Thomassey, S., and Hamad, M. (2021). Forecasting and anomaly detection approaches using LSTM and LSTM autoencoder techniques with the applications in supply chain management. *International Journal of Information Management*, 57:102282, DOI: 10.1016/j.ijinfomgt.2020.102282.

Olah, C., Mordvintsev, A., and Schubert, L. (2017). Feature visualization. *Distill*, 2(11):e7, DOI: 10.23915/distill.00007.

Pu, Y., Gan, Z., Henao, R., Yuan, X., Li, C., Stevens, A., and Carin, L. (2016). Variational autoencoder for deep learning of images, labels and captions. *Advances in Neural Information Processing Systems*, 29, DOI: 10.48550/arXiv.1609.08976.

R Core Team (2023). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, https://www.R-project.org/.

Ribeiro, M. T., Singh, S., and Guestrin, C. (2016). "Why should I trust you?" Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1135–1144. DOI: 10.1145/2939672.2939778.

Shankaranarayana, S. M. and Runje, D. (2019). ALIME: Autoencoder based approach for local interpretability. In *Intelligent Data Engineering and Automated Learning–IDEAL 2019: 20th International Conference, Manchester, UK, November 14–16, 2019, Proceedings, Part I 20*, pages 454–463. Springer, DOI: 10.48550/arXiv.1909.02437.

Simonyan, K., Vedaldi, A., and Zisserman, A. (2013). Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, DOI: 10.48550/arXiv.1312.6034.

Smilkov, D., Thorat, N., Kim, B., Viégas, F., and Wattenberg, M. (2017). SmoothGrad: Removing noise by adding noise. *arXiv preprint arXiv:1706.03825*, DOI: 10.48550/arXiv.1706.03825.

Sundararajan, M., Taly, A., and Yan, Q. (2017). Axiomatic attribution for deep networks. In *International Conference on Machine Learning*, pages 3319–3328. PMLR, https://proceedings.mlr.press/v70/sundararajan17a/sundararajan17a.pdf.

Theis, L., Shi, W., Cunningham, A., and Huszár, F. (2016). Lossy image compression with compressive autoencoders. In *International Conference on Learning Representations*. DOI: 10.48550/arXiv.1703.00395.

Uzunova, H., Ehrhardt, J., Kepp, T., and Handels, H. (2019). Interpretable explanations of black box classifiers applied on medical images by meaningful perturbations using variational autoencoders. In *Medical Imaging 2019: Image Processing*, volume 10949, pages 264–271. SPIE, DOI: 10.1117/12.2511964.

van der Maaten, L. and Hinton, G. (2008). Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9(11):2579–2605, http://www.jmlr.org/papers/v9/vandermaaten08a.html.

Winter, R., Noé, F., and Clevert, D.-A. (2021). Permutation-invariant variational autoencoder for graph-level representation learning. *Advances in Neural Information Processing Systems*, 34:9559–9573, https://proceedings.neurips.cc/paper_files/paper/2021/file/4f3d7d38d24b740c95da2b03dc3a2333-Paper.pdf.

Xiao, H., Rasul, K., and Vollgraf, R. (2017). Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*. DOI: 10.48550/arXiv.1708.07747.

Yang, Y., Wu, Q. M. J., and Wang, Y. (2016). Autoencoder with invertible functions for dimension reduction and image reconstruction. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 48(7):1065–1079, DOI: 10.1109/TSMC.2016.2637279.

Yann, L. (1998). The MNIST database of handwritten digits. *R*.

Zeiler, M. D. and Fergus, R. (2014). Visualizing and understanding convolutional networks. In *European Conference on Computer Vision*, volume 8689, pages 818–833. Springer, DOI: 10.1007/978-3-319-10590-1_53.

Zhang, K., Zuo, W., Chen, Y., Meng, D., and Zhang, L. (2017). Beyond a Gaussian denoiser: Residual learning of deep cnn for image denoising. *IEEE Transactions on Image Processing*, 26(7):3142–3155, DOI: 10.1109/TIP.2017.2662206.

Zhang, Q., Wu, Y. N., and Zhu, S.-C. (2018). Interpretable convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8827–8836. DOI: 10.1109/CVPR.2018.00920.

Zhang, Y., Tiňo, P., Leonardis, A., and Tang, K. (2021). A survey on neural network interpretability. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 5(5):726–742, DOI: 10.1109/TETCI.2021.3100641.

Appendix

In this Appendix, we show a a practical example of the how the regression, step outlined in Section 3.2, can be used to highlight important pixels for a specific class. Additionally, we provide both a demonstration of our Shiny app and display plots of all the classes from the autoencoder models used in Sections 4 and 5.

A. Visualisation of the Regression Step

In Figure 12, we provide an example of how applying the linear model can help highlight patterns learned by each dimension. Both panels display heatmaps of importance values for dimension 11 from the example shown in Figure 3. In the left panel, pixel importance is shown on a gradient from white (low) to red (high). While a generally linear pattern is visible, the image is noisy and appears somewhat random. In the right panel, pixels are classified as either positive or negative using the linear regression model method. This reveals the vertical, linear shape of the digit ‘one’ as areas of positive importance in red, with the surrounding negative space highlighted in blue.

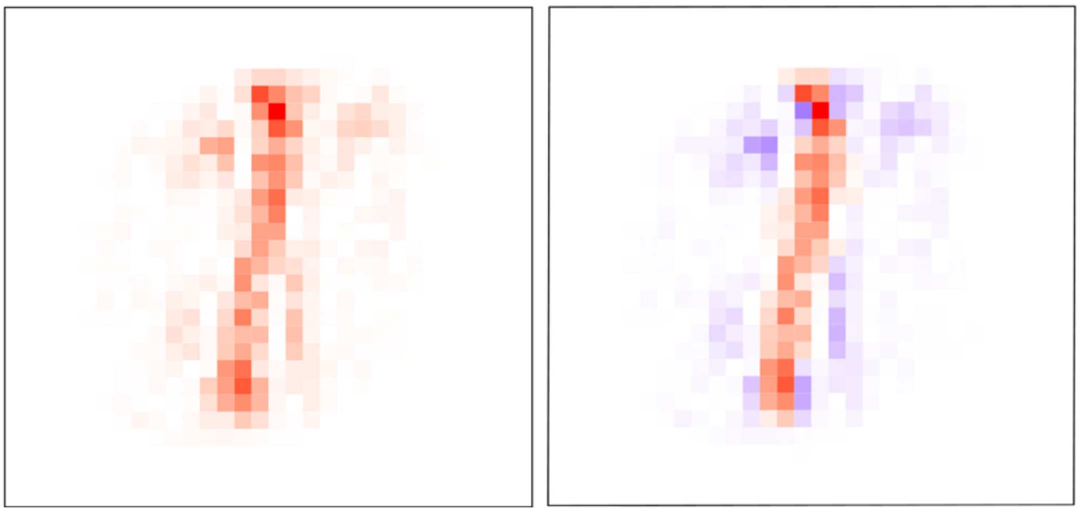
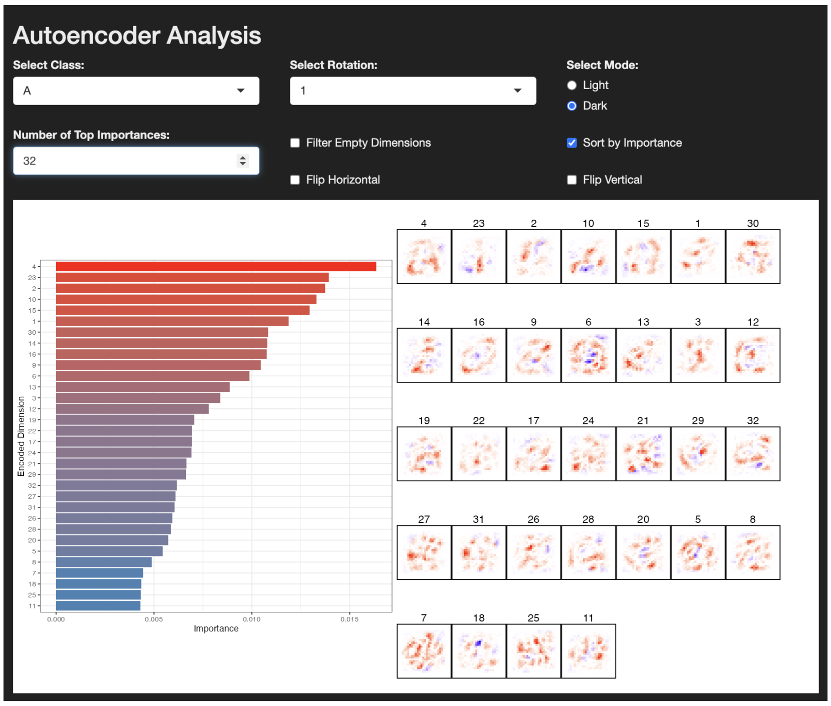


Figure 12: Heatmaps of importance values for dimension 11. The left panel shows pixel importance on a gradient from white (low) to red (high), while the right panel uses the linear model method to highlight positive importance in red and negative importance in blue, revealing the shape of the digit one.

B. Shiny App

In this section, we show a brief demonstration of the Shiny app developed in our R package **aim** (Autoencoder Importance Mapping), using the EMNIST vowel data. In Figure 13, we can see the layout of the app.

In the main plot window we show both the encoded dimension importance and the pixel importance side-by-side. Additionally, we provide multiple user interfaces, starting from the top left, these are:



Autoencoder Analysis

Select Class: A

Select Rotation: 1

Select Mode: Light [✓ Dark]

Number of Top Importances: 32

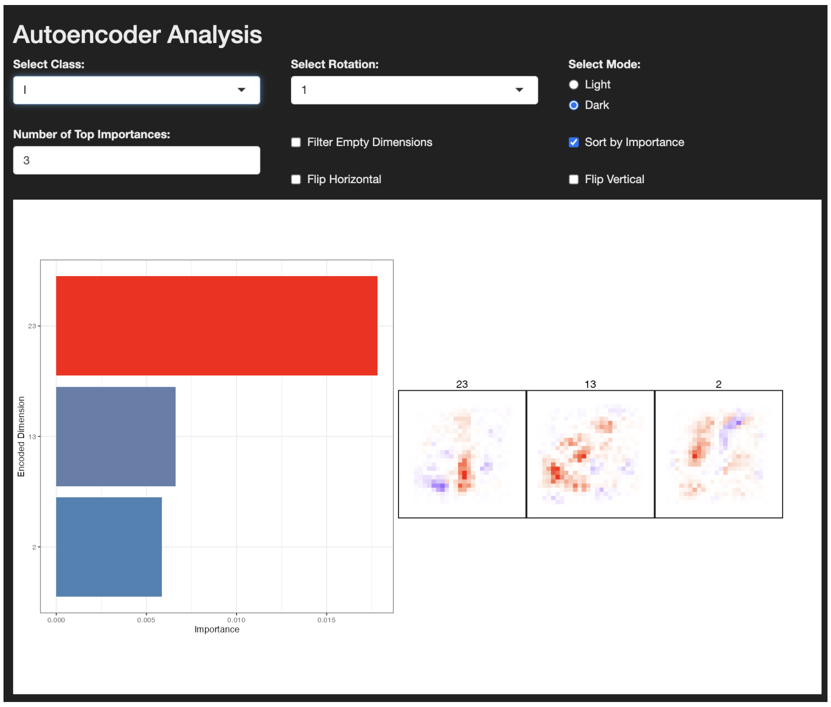
☐ Filter Empty Dimensions

☑ Sort by Importance

☐ Flip Horizontal

☐ Flip Vertical

(a)



Autoencoder Analysis

Select Class: I

Select Rotation: 1

Select Mode: Light [✓ Dark]

Number of Top Importances: 3

☐ Filter Empty Dimensions

☑ Sort by Importance

☐ Flip Horizontal

☐ Flip Vertical

(b)

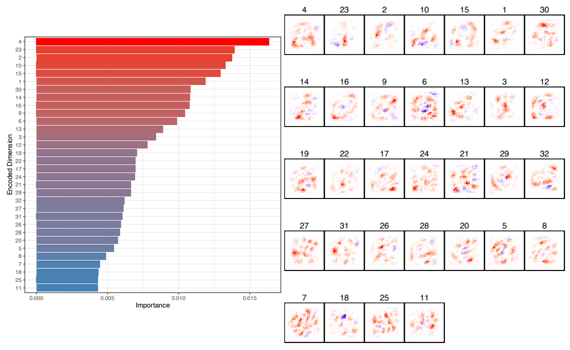
Figure 13: Demonstrations of the Autoencoder analysis Shiny app using EMNIST vowel data. Panel (a) shows all the dimensions for class A. Panel (b) filters the important dimensions from 32 to 3 for the selected class I.

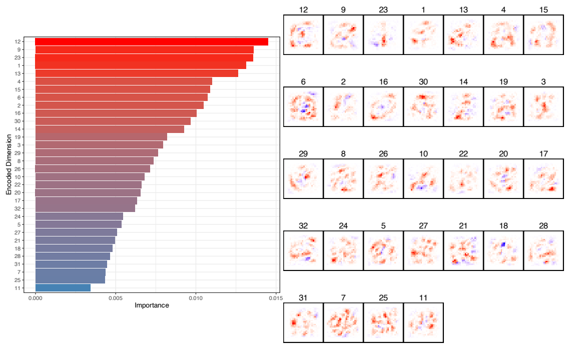
- **Select Class:** This allows the user to select and display a specific class from the data.
- **Number of Top Importances:** Allows the user to filter the plot to display the top x most important dimensions.
- **Select Rotation** Specifies the rotation of the heatmaps, where $0 =$ no rotation, $1 = 90°$, $2 = 180°$, $3 = 270°$.
- **Filter Empty Dimensions:** This filters any dimensions that are not used by the autoencoder.
- **Sort by Importance:** When checked, this sorts the encoded dimension importance from high to low and additionally sorts the pixel importance plots to correspond with the encoded dimensions.
- **Flip Horizontal:** Flips the heatmaps horizontally.
- **Flip Vertical:** Flips the heatmaps vertically.
- **Select Mode:** Allows the user to switch between light and dark modes on the app.

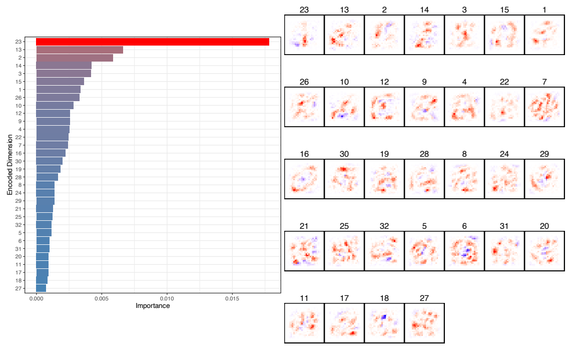
In Figure 13 (a), we can see that the selected class is A, and we are displaying all the 32 dimensions used in building the AE model from Section 3.A. As there are no unused dimensions, the `Filter Empty Dimensions` option is unchecked. We are additionally sorting the plots by the encoded dimension importance values. We have also selected the first rotation (that is 90°) and have not flipped the image horizontally or vertically. The inclusion of rotation and flipping options for heatmaps in the Shiny app addresses the need for correct image orientation and enhanced visual clarity. Sometimes the initial orientation of an image is incorrect, and these features allow users to easily adjust the heatmaps to the correct orientation by rotating and flipping. This ensures that the visualisations are displayed in the most readable and interpretable manner. The final option is a mode button which allows a user to select either a ‘Dark’ mode (with darker surrounding colours) or a ‘Light’ mode (which has lighter surrounding colours). In this case ‘Dark’ mode is selected. In panel (b) the selected class has been changed to I and the number of top importances has been set to three. This filters the plot to display the top 3 most important encoded dimensions. All other input remain the same as in panel (a).

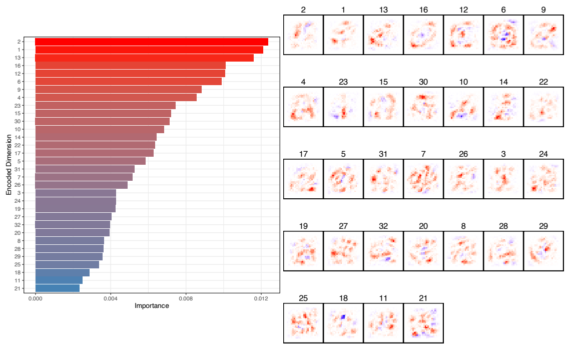
C. EMNIST Importance Plots For All Classes

In this appendix, we present the EMNIST importance plots for all classes. Each plot shows the most important dimensions for encoded dimension and pixel importance, sorted by encoded dimension importance.


(a) Class A


(b) Class E


(c) Class I


(d) Class O

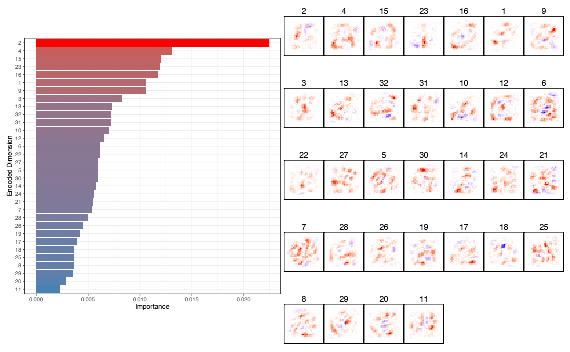
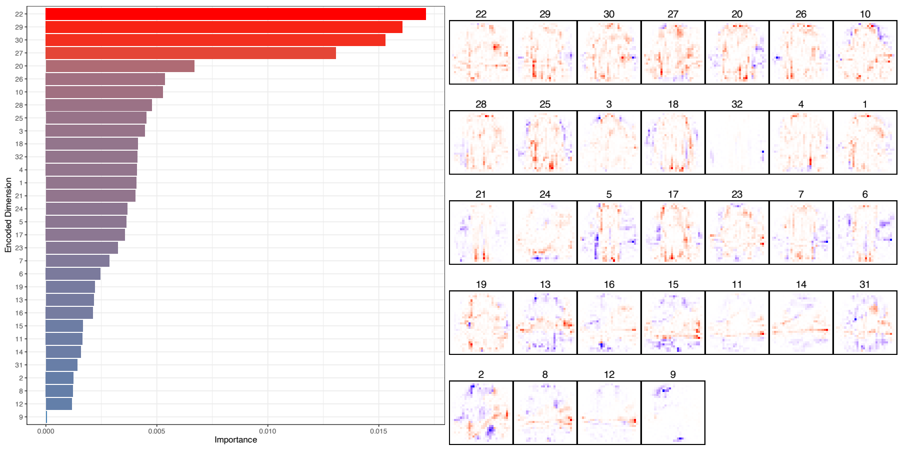

(e) Class U

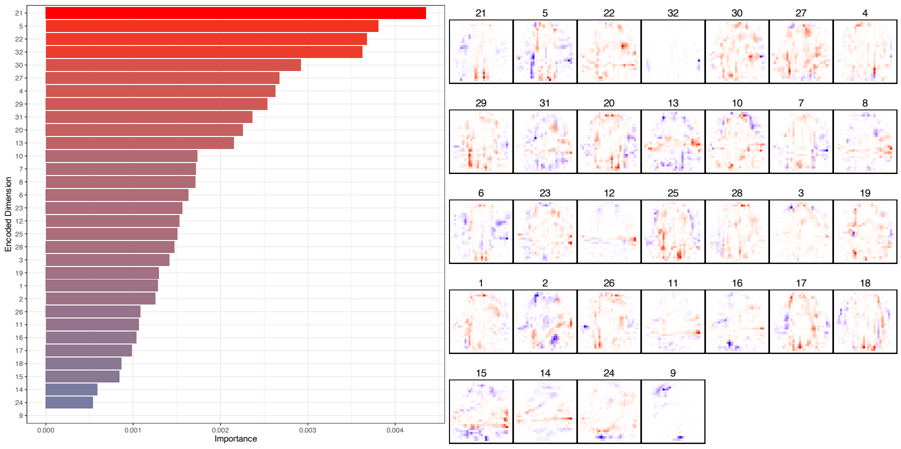
Figure 14: The most important dimensions for encoded dimension and pixel importance, sorted by encoded dimension importance, across all classes of the EMNIST data.

D. FMNIST Importance Plots For All Classes

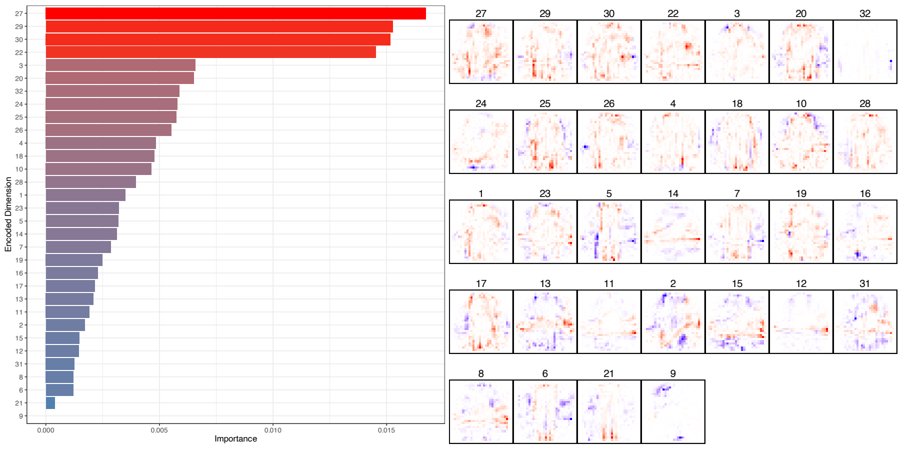
In this appendix, we present the FMNIST importance plots for all classes. Each plot shows the most important dimensions for encoded dimension and pixel importance, sorted by encoded dimension importance.



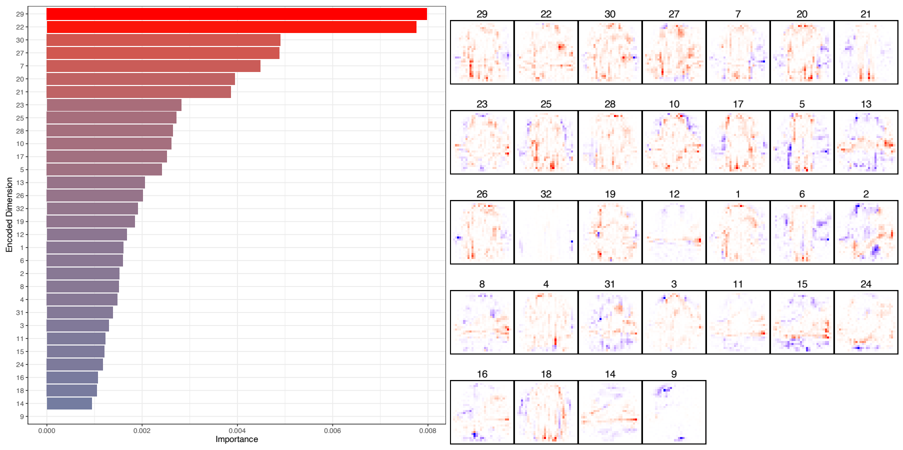
(a) Shirt



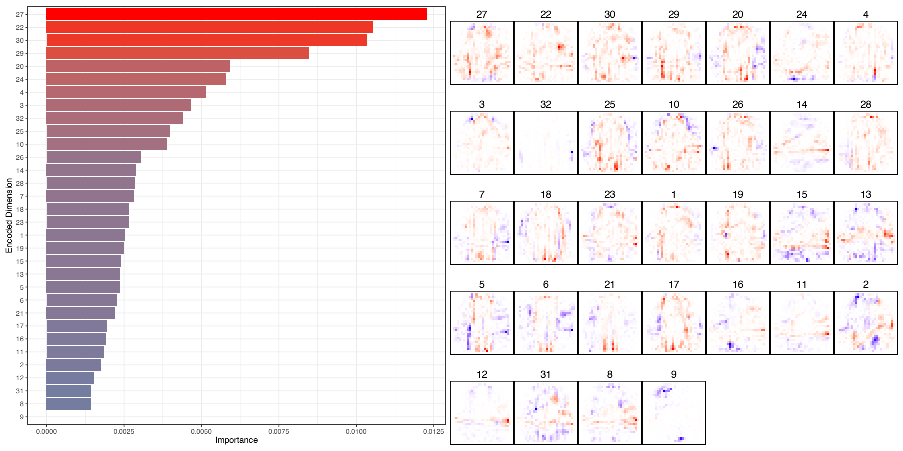
(b) Trouser



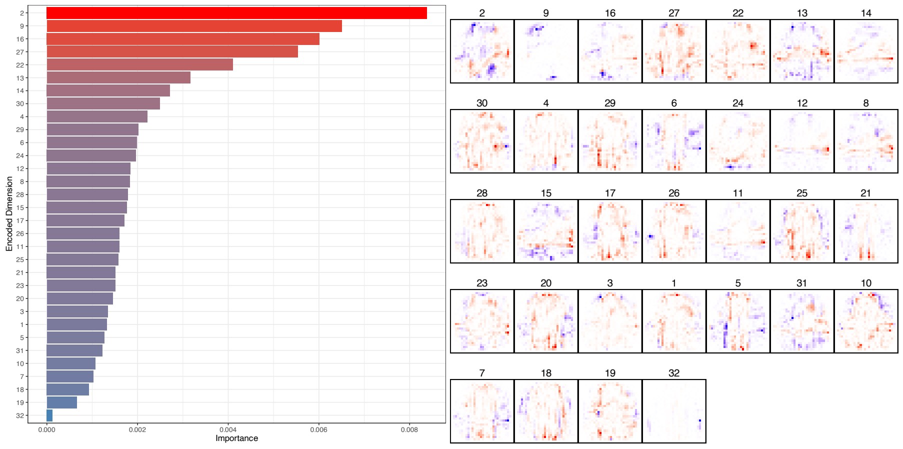
(c) Pullover



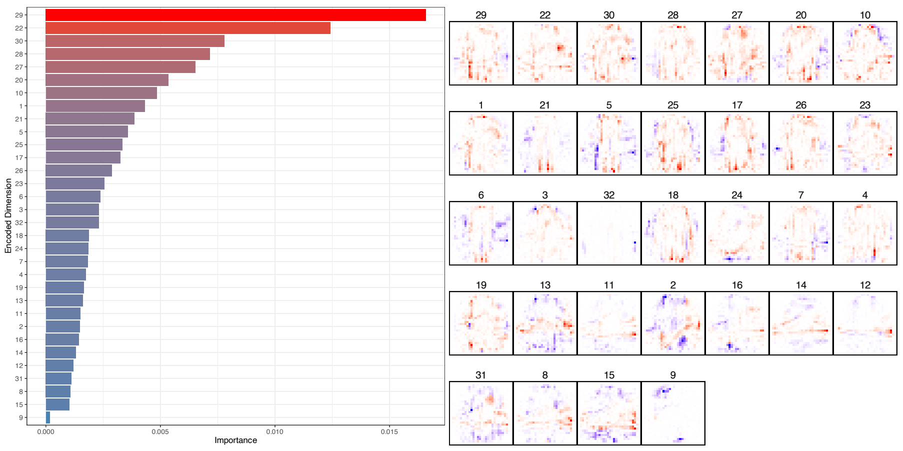
(d) Dress



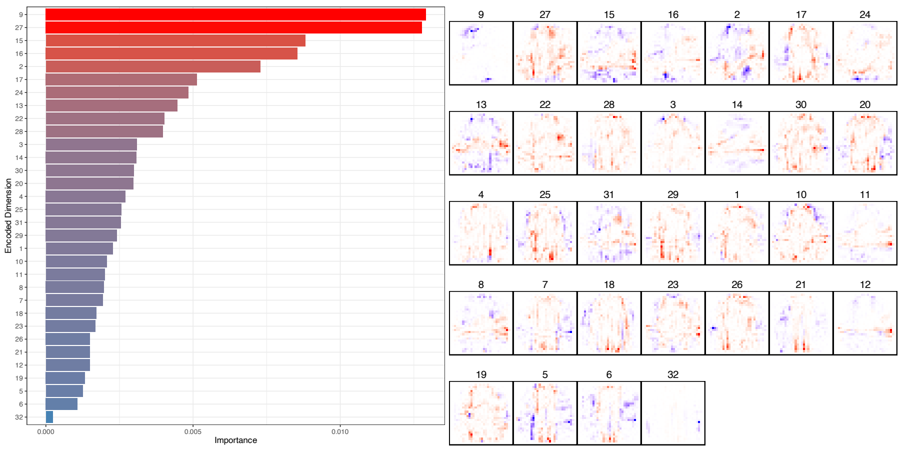
(e) Coat



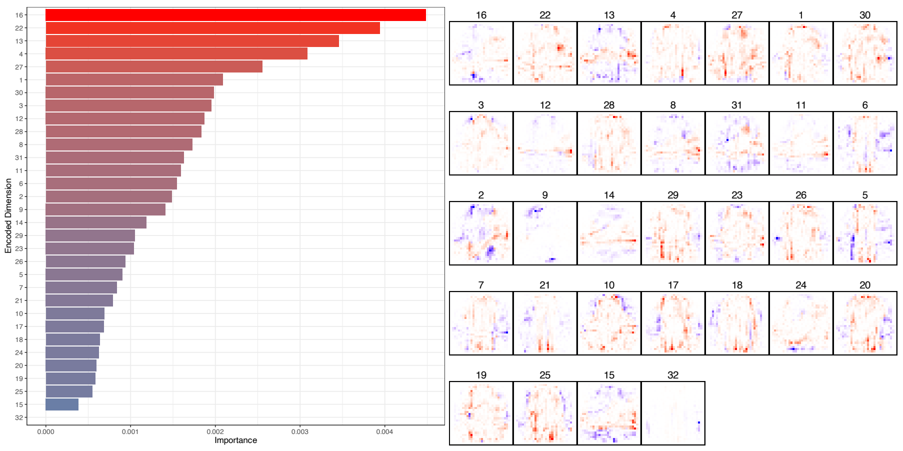
(f) Sandal



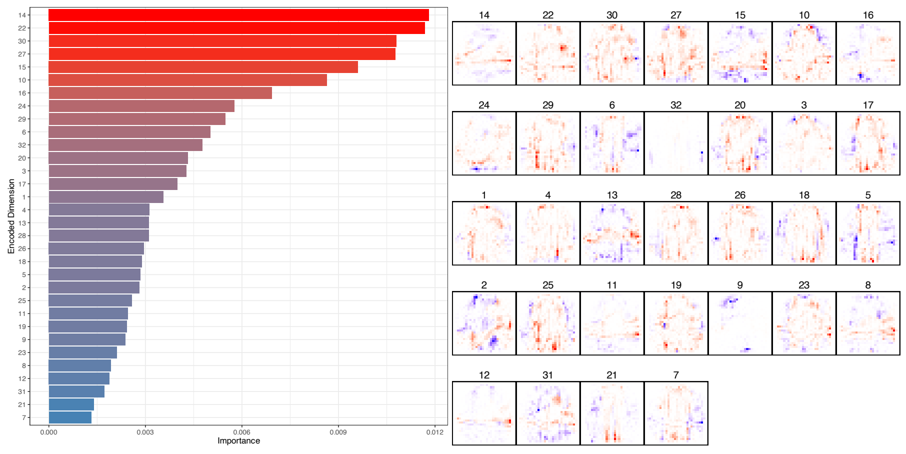
(g) T-Shirt



(h) Ankle Boot



(i) Sneaker

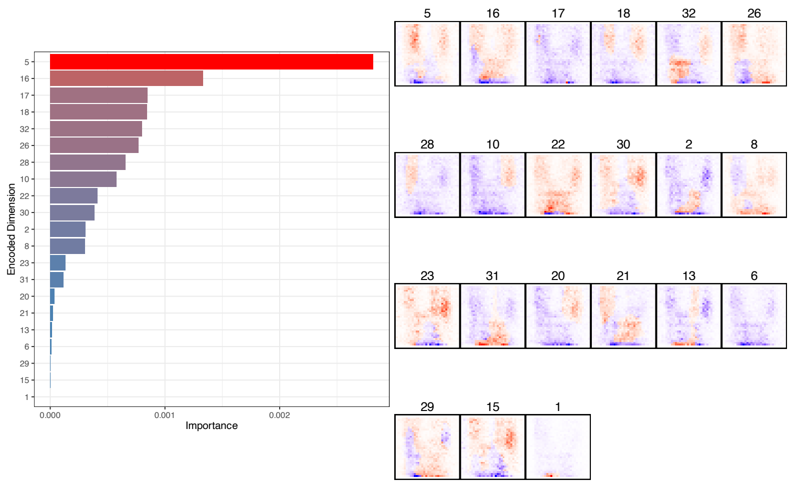


(j) Bag

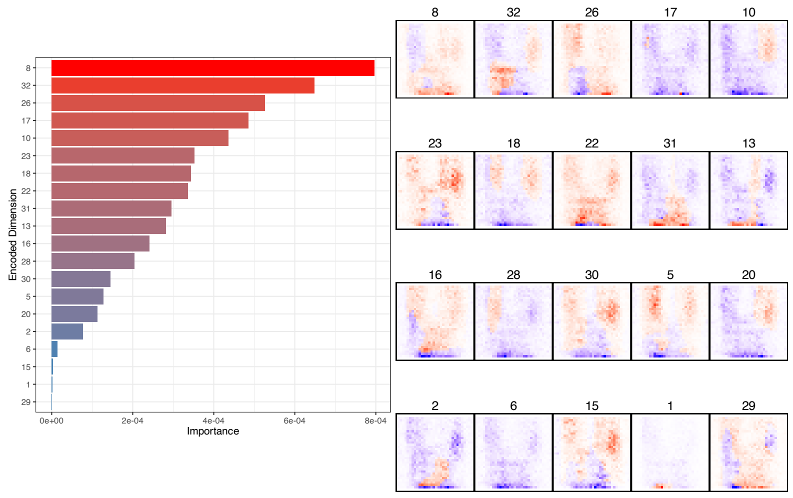
Figure 15: The most important dimensions for encoded dimension and pixel importance, sorted by encoded dimension importance, across all classes of the FMNIST data.

E. Audio MNIST Importance Plots For All Classes

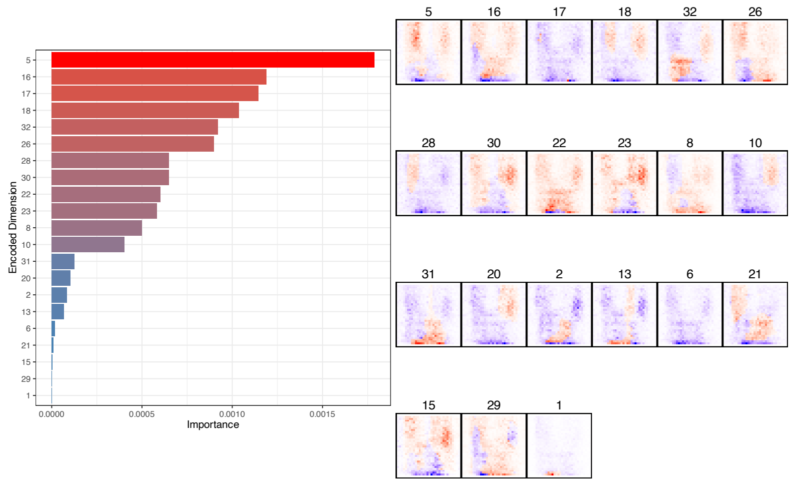
In this appendix, we present the audio MNIST importance plots for all classes. Each plot shows the most important dimensions for encoded dimension and pixel importance, sorted by encoded dimension importance.



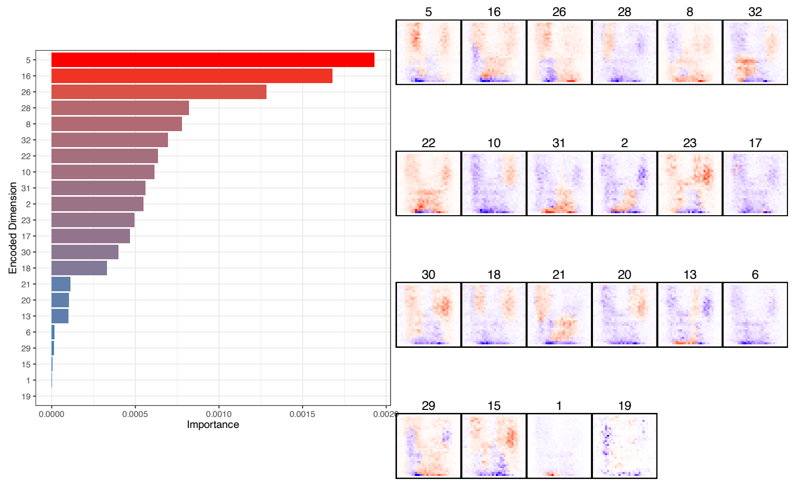
(a) Class 0



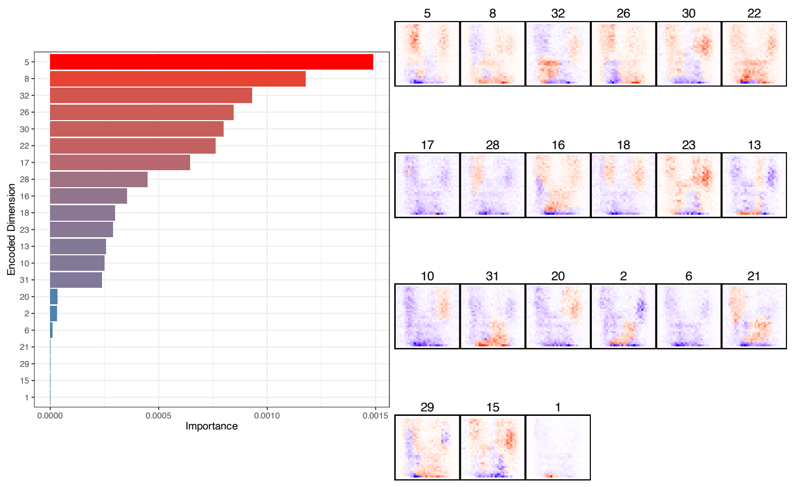
(b) Class 1



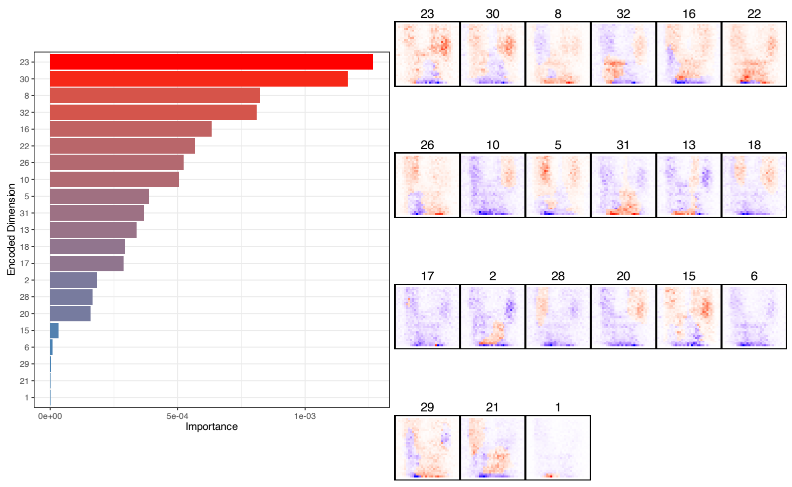
(c) Class 2



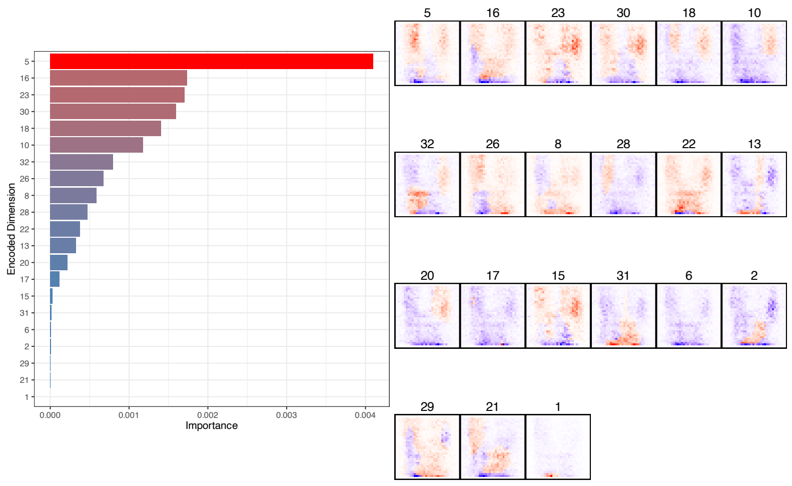
(d) Class 3



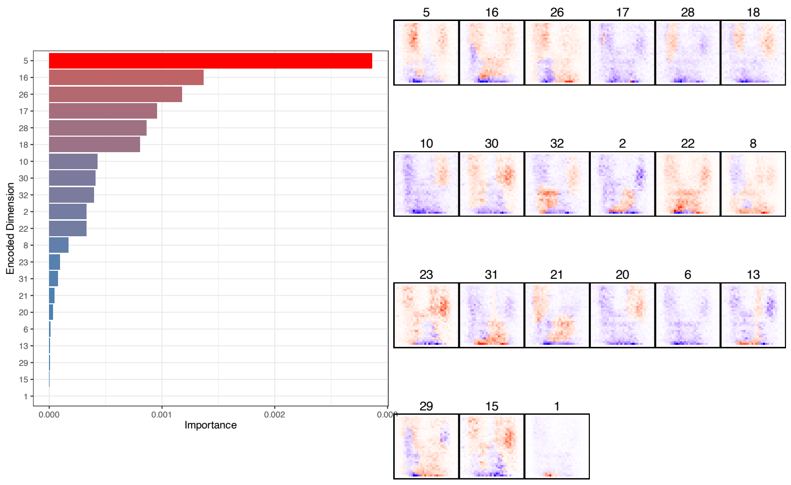
(e) Class 4



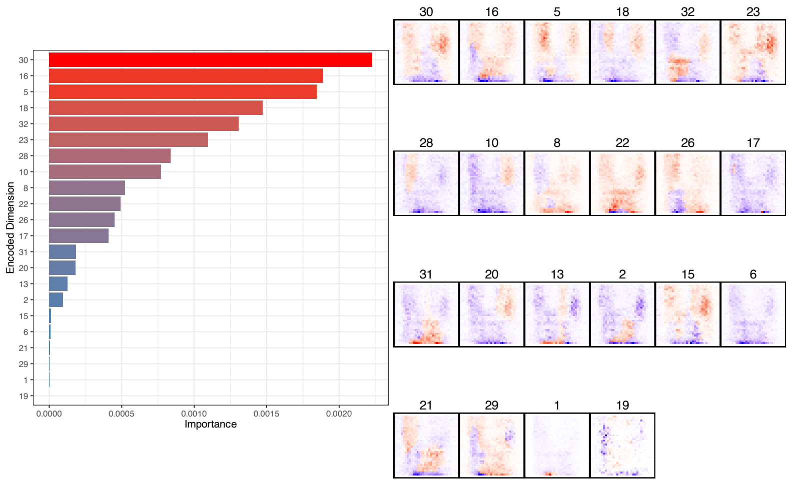
(f) Class 5



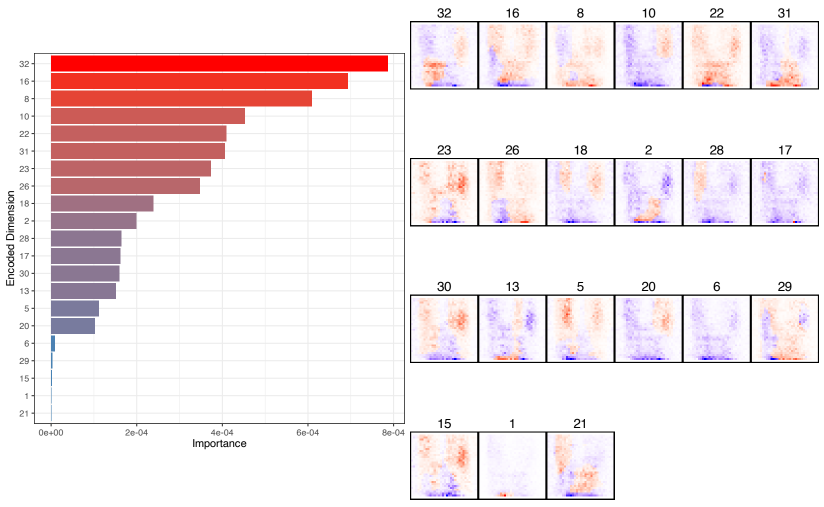
(g) Class 6



(h) Class 7



(i) Class 8



(j) Class 9

Figure 16: Top most important dimensions for encoded dimension and pixel importance, sorted by encoded dimension importance, across all classes of the audio MNIST data.

Affiliation:

Alan Inglis
Hamilton Institute
Maynooth University
Maynooth
Co.Kildare
Ireland
E-mail: alan.n.inglis@gmail.com

Andrew Parnell
School of Mathematics and Statistics
University College Dublin
Science Centre - South Belfield
Co.Dublin
Ireland
E-mail: Andrew.parnell1@ucd.ie

Journal of Data Science, Statistics, and Visualisation https://jdssv.org/
published by the International Association for Statistical Computing
http://iasc-isi.org/

ISSN 2773-0689
April 2025, Volume V, Issue II *Submitted:* 2024-09-09
doi:10.52933/jdssv.v5i2.118 *Accepted:* 2025-03-14