FIB: A Method for Evaluation of Feature Impact Balance in Multi-Dimensional Data

Xavier F. Cadet∗
Imperial College London

Sara Ahmadi-Abhari
Imperial College London

Hamed Haddadi
Imperial College London

Abstract

Errors might not have the same consequences depending on the task at hand. Nevertheless, there is limited research investigating the impact of imbalance in the contribution of different features in an error vector. Therefore, we propose the Feature Impact Balance (FIB) score. It measures whether there is a balanced impact of features in the discrepancies between two vectors. We designed the FIB score to lie in [0, 1]. Scores close to 0 indicate that a small number of features contribute to most of the error, and scores close to 1 indicate that most features contribute to the error equally. We experimentally study the FIB on different datasets, using AutoEncoders and Variational AutoEncoders. We show how the feature impact balance varies during training and showcase its usability to support model selection for single output and multi-output tasks.

1 Introduction

In an healthcare-related problem failing to correctly predict a single piece of information could have dramatic consequences. On the other hand, poorly predicting a single pixel in the background of an image could be acceptable. Such discrepancies between the two scenarios highlight that even a single feature can make a difference.

While there are plenty of metrics to evaluate model performance, to the best of our knowledge, no metric is concerned with the balance of features impact on a given error. Therefore, we introduce the Feature Impact Balance (FIB) score, a metric that quantifies to which extent errors between two vectors depend on a small number of features or many of them. In this work, we look into different properties of models and representations and their effect on the FIB score.

In this paper, we attempt to answer the following questions: (i) How are AutoEncoders equilibrating their errors during training? (ii) Can we use the FIB score to find the best models? (iii) Is the representation learned by these better suited for downstream tasks such as classification regression?

To answer these questions we focus on a specific type of models, namely AutoEncoders (AE) and Variational AutoEncoders (VAE) [Kingma & Welling (2014)] commonly used for unsupervised learning task. We use shallow AutoEncoders and relatively low dimensional datasets for visualization purposes and to help build intuition. We look into learning and re-purposing a representation for a given task to another task, commonly considered as Transfer Learning problem. Transfer Learning helped advance numerous fields such as drug discovery [Ramsundar et al. (2015); Yang et al. (2021); Min et al. (2021)] or natural language processing [Liu et al. (2019); Lee et al. (2021)]. One approach to Transfer Learning relies on the assumption that a common representation exists suitable for multiple tasks [Du et al. (2021); Tripuraneni et al. (2021)]. We also assume that there exists a representation shareable across multiple tasks.

The contributions of this paper are:

∗xfc17@ic.ac.uk

Preprint. Under review.
• **Feature Impact Balance (FIB) score** a mathematically-grounded quantification of the balance of the contribution of each feature in the difference between two vectors.

• Empirical study of the learning pattern and feature balance during training of specific AutoEncoders architectures.

2 Background

Here we introduce the problem and illustrate it through a comparison to Mean Squared Error (MSE), we then cover AutoEncoders, and Representation learning.

**Problem setting.** When comparing multidimensional continuous outputs, various methods exist based on the data type. A common method to quantify dissimilarity between two continuous multidimensional objects is the Mean Squared Error (MSE). When two vectors \( x \) and \( y \) represent multiple observations of their respective variables, the MSE quantifies the average difference between each observation \( x_i \) and \( y_i \). Nonetheless, if we consider these vectors as single observations with multiple properties, the MSE quantifies, on average, how each observation’s properties are dissimilar from one another. In this paper, we consider the MSE as the latter version.

One of the shortcomings of the MSE is that it can associate the same value to vectors with drastically different feature impacts as illustrated in Figure 1. The MSE does not distinguish the cases where a single feature, a subgroup of features, or all the features, carry the errors.

![Figure 1: Comparison between MSE and FIB](image)

Figure 1: Comparison between MSE and FIB: Vectors \( x \) represent the ground truth, vectors \( y_1, y_2, y_3 \) represent hypothetical predictions from models. The errors between \( x \) and the \( y \)s are carried by different columns. The MSE yields the same value for all combinations of \( x \) and \( y \)s. The FIB indicates (from top to bottom) an important imbalance when looking at \( y_1 \), moderated imbalance using \( y_2 \), and balanced error contribution considering \( y_3 \).

**Representation Learning.** Over the past years, the interest in Representation Learning has grown, becoming a field on its own [Bengio et al., 2014]. Representation Learning aims at finding a description of data, a new representation, which makes subsequent tasks simpler to solve [Goodfellow et al., 2016]. A common trade-off between preserving information about the input and obtaining properties of interest exists in Representation Learning problems [Goodfellow et al., 2016].

**AutoEncoder.** AutoEncoders are Neural Network architectures that are extensively used for Unsupervised Machine Learning tasks. Their aim is to learn a new representation of their input, such that they can reproduce their inputs [Rumelhart & McClelland, 1987; Baldi & Hornik, 1989; Hinton & Zemel, 1993; Baldi, 2011]. AutoEncoders are composed of 2 sections: an encoder function \( f \) and a decoder function \( g \). The encoder maps the input \( x \) from the input space to the code space resulting in a code \( h \). The decoder maps the code \( h \) from the code space to the output space and produces \( \hat{x} \), which stands for the reconstruction of \( x \).
3 Feature Impact Balance (FIB)

Let's compare a vector $x$ to a group of vectors $\{y_1, y_2, y_3\}$. We might obtain the same MSE values even if the dissimilarities per feature are drastically different, as illustrated in Figure 1. Having errors evenly spread across all features can be beneficial if we do not know which feature is the most important. On the contrary, if we know that a feature is important, we should pick a model that does not fail to predict this feature.

To quantify the balance between the contributions of each feature in the error, we introduce the Feature Impact Balance (FIB) score. In this section, we define the different components leading to the computation of the FIB, from the Internal Error (IE) quantification, the Feature Impact (FI), the Feature Impact Imbalance (FII), and the proofs that guarantee that the FIB score can take values in $[0, 1]$. Having values ranging in $[0, 1]$ allows comparing models across experiments, similarly to the accuracy score in classification tasks.

We define the FIB in its simplest form as:

$$FIB(x, y) = 1 - \frac{M}{M - 1} \sum_{k=1}^{M} \frac{|x_k - y_k|}{||x - y||_1} - \frac{1}{M} \in [0, 1] \tag{1}$$

With $x, y \in \mathbb{R}^M$ and $||x - y||_1 = \sum_{k=1}^{M} |x_k - y_k|$.

The Feature Impact Balance quantifies errors at two levels, the Internal Error (IE) and Balance Error (BE). The Internal Error quantifies the error between the two entries $x$ and $y$. The Balance Error quantifies the error between the contribution of each entries in the Internal error and a balance vector. The balance vector represents the scenario where each component contributed equally to the error.

In the following, we detail the steps to compute the FIB score between two vectors; the computations can be adapted to matrices. Using matrices, we evaluate properties at different levels. For instance, we could consider subsets of a given dataset, features associated with a specific group, or even mini-batch when training a Neural Network.

Let $x, y \in \mathbb{R}^M$ where $M$ is the number of features. We define the Internal Error (IE) as a function that takes two entries $x$ and $y$ and results in a vector $e \in \mathbb{R}^K$ ($e_k$ indicates the $k$-th element of the vector $e$):

$$IE(x, y) = e \in \mathbb{R}^K \tag{2}$$

where $K$ is the dimension of the error vector. We allow the dimension of the Internal Error to be different from those of the inputs, such that we can consider feature groups in the next sections.

In the following steps, we consider the Absolute Error as Internal Error function, yielding a vector of the same dimensions as the inputs ($i.e K = M$). We expect the Internal Error function to be constituted of positive or null values (null values when no error is made). Therefore, we use:

$$IE(x, y) = AE(x, y) = [|x_1 - y_1|, ..., |x_M - y_M]|^T \in \mathbb{R}^M \tag{3}$$

$$AE(x, y) = [|x_1 - y_1|, ..., |x_M - y_M]|^T \in \mathbb{R}^M \tag{4}$$

We then define the Feature Impact (FI) as the Internal Error vector divided by the sum of its elements:

$$FI(x, y) = \frac{e}{\sum_{k=1}^{M} e_k} \in [0, 1]^M \tag{5}$$

where $e = IE(x, y)$. It follows that the sum of its elements equals to 1.

The elements of the Feature Impact vector indicates the contribution of each feature to the errors.

In case there are no errors, (the errors between x and y sum to 0), we substitute $FI(x, y)$ by $\frac{1}{M}$ where $1_M = [1, ..., 1]^T \in \mathbb{R}^M$.

As such, we consider that when no errors are done, each component contributed equally.

To quantify the imbalance of the contributions of each feature, we define the Balance Error (BE). When each feature participates equally (namely contributes to $\frac{1}{M}$ of the sum of the errors), the BE equals 0. Otherwise, it quantifies how far the contributions are from the expected contributions.
Then, we define the Feature Impact Imbalance (FII) as the Balance Error between the Feature Impact vector from equation 5 and a Balance vector. The Balance vector is \( b = \frac{1}{M} \mathbf{1} \in \mathbb{R}^M \). Therefore, we define the FII as:

\[
FII(x, y) = BE(FI(x, y), \frac{1}{M})
\]  

(6)

We use the MSE to compute the Balance Error in the following experiments. Nonetheless, we could also use the Mean Absolute Error (MAE).

By computing the Squared Error between the Feature Impact vector and the balance vector, we evaluate the error between the Feature Impact observed and the Feature Impact achieved when every feature contributed equally. Finally, we take the mean of these balance errors to obtain a single value to quantify the imbalance.

The shortcoming from the MSE persists here, but at a different level, the FII will not distinguish between different combinations of feature impact imbalances (i.e \([1,0,0]^T\) against \([0,0,1]^T\) in \(\mathbb{R}^3\)), but it will be sensible to the differences at the input level \(x\) and \(y\).

We demonstrate that when using the MAE and MSE as Balance error, the FII can reach a maximum via Theorem 3.2 and its value using Theorem 3.3.

Let \((c_k)_{k \in \{1, \ldots, M\}}\) be the values of \(FI\) such that \(FI(x, y) = [c_1, \ldots, c_M]^T\). By construction \(\forall k \in \{1, \ldots, M\}, 0 \leq c_k \leq 1\) and \(\sum_{k=1}^{M} c_k = 1\). The right hand part of the equation 6 when using the MAE can be rewritten as a function of \(M\) variables noted \(\varphi(c_1, \ldots, c_M) : \mathbb{R}^M \rightarrow \mathbb{R}\)

\[
\varphi(c_1, \ldots, c_M) = \frac{1}{M} \sum_{k=1}^{M} |c_k - \frac{1}{M}|
\]  

(7)

Similarly when using the MSE, we now consider \(\psi(c_1, \ldots, c_M) : \mathbb{R}^M \rightarrow \mathbb{R}\)

\[
\psi(c_1, \ldots, c_M) = \frac{1}{M} \sum_{k=1}^{M} (c_k - \frac{1}{M})^2
\]  

(8)

In order to compute the maximum of \(\varphi\) and \(\psi\), we first give the following definition.

**Definition 3.1.** [Ziegler (1995)] A \(M\)-simplex in \(\mathbb{R}^M, M \geq 1\), is the convex hull of \(M + 1\) affinely independent points.

A \(M\)-simplex is the intersection of all convex sets containing these points; it is also the smallest convex set containing these points. It finally follows from a theorem due to Carathéodory Ziegler (1995) that the \(M\)-simplex \(S\) defined by the points \(a_1, \ldots, a_{M+1}\) is characterized as follows: a point \(a\) belongs to \(S\) if and only if there exists \(\lambda_i, i = 1, \ldots, M + 1\), \(\lambda_i \in [0,1], i = 1, \ldots, M + 1, \sum_{i=1}^{M+1} \lambda_i = 1\), such that

\[
a = \sum_{i=1}^{M+1} \lambda_i a_i.
\]

This yields that a \(M\)-simplex is closed and bounded, and thus compact, in \(\mathbb{R}^M\); it is also, by definition, convex.

**Theorem 3.2.** Let \(f : S \rightarrow \mathbb{R}\) be continuous and convex, where \(S \subset \mathbb{R}^M\) is the \(M\)-simplex defined by the points \(a_1, \ldots, a_{M+1}\), \(M \geq 1\). Then \(f\) reaches its maximum at one of the \(a_i\)’s.

**Theorem 3.3.** Let \(A = \{(c_1, \ldots, c_M) \in \mathbb{R}^M, c_i \in [0,1], i = 1, \ldots, M, \sum_{i=1}^{M} c_i = 1\}\). We set \(\varphi(c_1, \ldots, c_M) = \sum_{i=1}^{M} |c_i - \frac{1}{M}|\) and \(\psi(c_1, \ldots, c_M) = \sum_{i=1}^{M} (c_i - \frac{1}{M})^2\). Then, \(\max_A \varphi = \frac{2M-1}{M}\) and \(\max_A \psi = \frac{M-1}{M^2}\).

We provide the proofs for both theorems in the Appendix B. When using the MSE as Balance Error, we know from Theorem 3.2 that:

\[
0 \leq FII(x, y) \leq \frac{M-1}{M^2}
\]  

(9)
where the value 0 means that the feature impact is balanced, as it is achieved in the case where \( \forall k \in 1, \ldots, M, c_k = \frac{1}{M} \).

Using the maximal values of the FII, we can now normalize the FII scores and obtain values in \([0, 1]\). For a more straightforward interpretation of the FII values, we define the Normalized Feature Impact Imbalance (NFII) as:

\[
NFII(x, y) = \frac{M^2}{M - 1} \cdot FII(x, y)
\]  

From equations (9) and (10) it comes that:

\[
0 \leq \frac{M^2}{M - 1} \cdot FII(x, y) \leq 1
\]  

Finally, we introduce the Feature Impact Balance (FIB). For conciseness, we omit the mention that it is normalized, as one minus the NFII, defined as:

\[
FIB(x, y) = 1 - NFII(x, y) \in [0, 1]
\]

A value of 1 indicates that each of the features contributed equally to the errors, and a value of 0 indicates that a single feature is responsible for the error.

### 3.1 Adaptation to higher number of features.

The more features there are, the higher the chances that the FIB score will be high. To reach a balanced impact each feature needs to contribute to \( \frac{1}{M} \). Therefore, to obtain a score of 0, it requires that a single feature carries all the error. Furthermore, as the number of features increases the contribution required for balance gets close to 0.

In order to use the FIB on high dimensional data, we propose to group features. For instance, considering an image dataset such as MNIST with images composed of 28x28 pixels, namely 784 features, we can regroup these features into 10 groups, and compute the FIB score over these 10 groups instead of the 784 features.

In Figure 2, we give an example with \( M = 1024 \), comparing using no groups (Figure 2a) and 10 groups (Figure 2b). After injecting noise in 10% of the 1024 features, we obtain a score that is already close to 1.0, i.e. \( \epsilon = 0.99 \) (99% in the plot). Nonetheless, after modifying a single feature, the score drops to 0. While it can be an interesting property, for instance when each feature is critical, we might be interested in imbalance between groups of features. For instance, we could consider that anywhere between 1 feature and 10% of the features should result in a FIB score of 0.0 (0% in Figure 2b). To do so, we can specify a different Internal Error as defined in equation 2. For instance, we can group the \( M \) features into \( K \) groups. Then, we chose an aggregation strategy (sorting, random selection, predefined groups, etc...), and for each group we can pick a reduction strategy to bring each group to a single value (similarly to Mean Pooling or Max Pooling in Convolutional Neural Networks).

We provide an example of grouping using sorting and splitting as an aggregation method and summation as the reduction operation in Appendix C.

### 4 Experiments

We ran experiments over different datasets (Section 4.2), we show the impact of Feature Grouping (Section 4.3) using different types of AutoEncoders (Section 4.4), and show how FIB can be used to support model selection (Section 4.5).

#### 4.1 Experimental Setup

**Datasets.** We chose three datasets to run experiments on: 1) Iris [P. Fisher (1936)], 2) SARCOS [Vijayakumar & Schaal (2000)] and 3) MNIST [LeCun & Cortes (2005)].

**Feature Grouping.** We computed grouped FIB scores on SARCOS and MNIST for group number ranging from 2 to 10.
Figure 2: We add noise drawn from an Uniform distribution into a 1024-Dimensional vector for different proportions of perturbation (10% to 100%). We repeat this operation 10000 times. In 2a we compute the FIB without grouping. In 2b we compute the FIB over 10 groups. While for the FIB without groups most scores are close to 1.0, for the 10 Groups FIB, the scores are more spread over [0, 1].

**AutoEncoders.** We experiment with two types of AutoEncoders. These are: Classical AutoEncoders (AE) and Variational AutoEncoders (VAE). In both cases, we considered only AutoEncoders based on Fully Connected layers.

**Using FIB to support Model Selection.** We chose the best performing epoch of the classical AutoEncoders based on their validation loss. We use these models as feature extractors. We then train Logistic Regressions and Linear regression using these features and evaluate their performance on a test set.

### 4.2 Different Datasets

We computed the FIB score on the Iris, SARCOS, MNIST datasets. For each of the Dataset, we tried Fully Connected AutoEncoders. Figure 3a shows the evolution of the FIB scores over 1000 models for 1000 epochs for Iris and SARCOS and 300 for MNIST. Result are reported based on the validation set performances. Figure 3a shows the evolution using an AutoEncoder with layers of size 4, 2 on Iris. Figure 3b shows the evolution with sizes 21, 16, 4 for SARCOS. And Figure 3c shows the evolution for sizes 784, 512, 256 for MNIST. In the case of Iris, the FIB scores vary between 0.80 and 1.0 as the models train. The pattern highlights that while the majority of the models started with high FIB score on the validation as the model trains, the balance is not maintained. Table 1 highlights that after training, models tend to reach FIB scores above 0.90. As the capacity of the models increase, the FIB score is higher.
Table 1: MSE and FIB scores obtained at the final epoch over two model designs for Iris, SARCOS and MNIST

| Dataset | Iris [4, 1] | Iris [4, 2] | SARCOS [21, 16, 4] | SARCOS [21, 16, 8] | MNIST [784, 512, 128] | MNIST [784, 512, 256] |
|---------|-------------|-------------|--------------------|--------------------|-----------------------|----------------------|
| Layers  |             |             |                    |                    |                       |                      |
| MSE     | 0.344 ± 0.170 | 0.151 ± 0.102 | 0.168 ± 0.041 | 0.068 ± 0.006 | 0.004 ± 0.004 | 0.003 ± 0.000 |
| FIB     | 0.934 ± 0.022 | 0.947 ± 0.018 | 0.997 ± 0.000 | 0.998 ± 0.000 | 0.999 ± 0.004 | 0.999 ± 0.000 |

Table 2: FIB computed over different numbers of groups on SARCOS and MNIST. A number of group equal to $M$ (the number of features) is the same as computing the FIB without grouping.

| #Groups | 2 | 3 | 5 | 7 | 10 | $M$ |
|---------|---|---|---|---|----|-----|
| SARCOS  | 0.965 ± 0.006 | 0.975 ± 0.004 | 0.988 ± 0.002 | 0.990 ± 0.001 | 0.994 ± 0.001 | 0.997 ± 0.000 |
| MNIST   | 0.472 ± 0.011 | 0.697 ± 0.008 | 0.839 ± 0.004 | 0.889 ± 0.003 | 0.926 ± 0.002 | 0.999 ± 0.000 |

4.3 Feature Grouping

Table 2 contains the FIB scores using Feature Grouping on SARCOS with an AutoEncoder with layers 21, 16, 4 and, MNIST with layers 784, 512, 256. Results are averaged over 100 models after 1000 epochs. We grouped the features using sorting and splitting as aggregation and averaging as reduction. We computed grouped FIB score for group sizes 2 to 10. For the SARCOS Dataset, moving from 21 features to 2 groups of 10 features yields FIB score that are close to 1.0 with a drop of 0.032 in the FIB score. Nevertheless, on MNIST dataset, using 2 groups of 392 features leads to drop of FIB score by 0.527 score, which indicates that close to 50% of the features contribute to most of the error.

4.4 Different AutoEncoders

We compare FIB scores obtained using AutoEncoders and Variational AutoEncoders on the SARCOS and MNIST datasets. Table 3 shows that FIB scores do not drastically change between AE and VAE if we consider all features. We considered the results after training 100 models for 1000 epochs on SARCOS and 300 for MNIST. For the SARCOS dataset, the sizes of the layers of the AutoEncoders are 21,16 and 4. For MNIST the sizes are 784, 512, 256. We see that both architectures yield high a FIB score in Table 3. We further compare performance based on the number of groups used, and in the case of SARCOS the VAE tends to have slightly higher, 0.004 for 2 groups and 0.003 for 3 groups, while for MNIST the VAE drops by 0.291 points and 0.132 for 2 and 3 groups FIB respectively.

![Figure 4](image)

Figure 4: We compare the FIB scores for AutoEncoders and Variational AutoEncoders for different numbers of groups of features on SARCOS and MNIST. (a) Using an AutoEncoder with latent dimension 4, (b) Using a Variational AutoEncoder with latent dimension 4 for SARCOS, and with latent dimension 256 for MNIST.
Table 3: FIB and grouped FIB scores obtained at the final epoch on AutoEncoders and Variational AutoEncoders for SARCOS and MNIST

| Dataset | Type | SARCOS | SARCOS | MNIST | MNIST |
|---------|------|--------|--------|-------|-------|
|         |      | AE     | VAE    | AE    | VAE   |
| FIB     |      | 0.997 ± 0.000 | 0.997 ± 0.004 | 0.999 ± 0.000 | 0.999 ± 0.000 |
| FIB 2   |      | 0.965 ± 0.006 | 0.969 ± 0.002 | 0.472 ± 0.011 | 0.281 ± 0.003 |
| FIB 3   |      | 0.975 ± 0.004 | 0.978 ± 0.001 | 0.697 ± 0.008 | 0.565 ± 0.002 |

4.5 Model Selection and FIB

For each of the 100 models trained in section 4.2, we use the weights associated with the best performing epoch with respect to the loss over the validation set. Using these versions of our models, we use the encoder section to extract representations from our train, validation and test splits.

After extracting the representations, we train 3 Logistic Regression for Iris, 7 Linear Regression for SARCOS to predict each dataset's respective property of interest. For instance, on Iris each Logistic Regression predicts a single class among "Setosa", "Versicolour" and "Virginica". For Iris, we consider the AutoEncoders with sizes 4, 2, for SARCOS we consider the AutoEncoders with sizes 21, 16, 4. The performances reported here are obtained on the test set. Figure 5 shows the performance obtained by the models using the learned representations. In the case of Iris, no model was able to rank in the top 3 of all 3 models at once. In the case of SARCOS no models were able to rank in the top 3 of 4 to 7 models. In average models that ranked in the top 3 on most classifiers (resp. regressors) have a higher FIB score.

Therefore, representations from models with higher FIB score are better suited to cover more downstream tasks.

5 Limitations and Restrictions

Depending on the function used to compute the Balance Error, it is possible to have the same value of FIB for different combinations of imbalance. Nonetheless, using the Feature Impact vector from equation 5, we can obtain a fine grain overview of the contributions of each feature.

Furthermore, the FIB score only gives information about the balance between the impact of the features in the error and does not provide information about the intensity of said error. Therefore, the FIB score should be used in combination with other metrics to assess the performance of models.
We only considered Fully Connected AutoEncoders in this work, further work could be done to analyze the impact of using AutoEncoders using Convolutional Layers.

6 Conclusion and Future work

We introduced a method to quantify the balance of the contribution of features to an error between two vectors, called Feature Impact Balance (FIB) score. FIB relies on an Internal Error function to evaluate the error between two vectors, then use a Balance Error function to quantify how balanced the contributions of each feature are with respect to the errors. The FIB score identifies whether a single feature or a group of features contributes to most errors. We demonstrated that the FIB score can be adapted to different configurations to assess imbalance within the contribution of individual features or groups of features in the errors. We show that AutoEncoders tend to balance the impact of the features during the training phase. We observe that representations obtained from models with higher FIB scores tend to be beneficial for multiple tasks.

Future Work.

- **Training Using FIB.** It could be possible to train models using the FIB score. One way would be to add it as a penalization term. Nevertheless, as it is possible to achieve high FIB with high reconstruction error in the case of AutoEncoders, optimizing for the FIB score might not lead to a stable learning, one way to overcome this issue could be to increase the importance of the loss due to the FIB score after some epoch, or after lowering the reconstruction loss first.

- **FIB and Multimodal Machine Learning.** Feature Grouping could be used to study imbalance between modalities. If we were to consider the multimodal representation problems [Baltrušaitis et al. (2019)], we could monitor which models spread their error contribution equally over multiple modalities.

- **Different Data types.** We could leverage the Internal Error function to obtain FIB score based on different data types. If we are able to quantify the error produced by a specific feature, we should be able to compute the impact of each feature on the total error and obtain a FIB score.

- **FIB and Difference between clusters.** FIB score could be computed over averaged features from a specific class in the case of classification, for instance we could look at the difference between the average vectors of class A versus the average vector of another class B.

Acknowledgments and Disclosure of Funding

The authors would like to thank Ranya Aloufi, Anastasia Borovykh, Alain Miranville, Yuchen Zhao for numerous comments, technical questions, references, and invaluable suggestions for presentation that led to an improved text.

Xavier F. Cadet is supported by UK Research and Innovation [UKRI Centre for Doctoral Training in AI for Healthcare grant number EP/S023283/1].

References

Baldi, P. Autoencoders, unsupervised learning and deep architectures. In Proceedings of the 2011 International Conference on Unsupervised and Transfer Learning workshop - Volume 27, UTLW'11, pp. 37–50, Washington, USA, July 2011. JMLR.org.

Baldi, P. and Hornik, K. Neural networks and principal component analysis: Learning from examples without local minima. Neural Networks, 2(1):53–58, January 1989. ISSN 0893-6080. doi: 10.1016/0893-6080(89)90014-2. URL https://www.sciencedirect.com/science/article/pii/0893608089900142.

Baltrušaitis, T., Ahuja, C., and Morency, L.-P. Multimodal Machine Learning: A Survey and Taxonomy. IEEE Transactions on Pattern Analysis and Machine Intelligence, 41(2):423–443,
Bengio, Y., Courville, A., and Vincent, P. Representation Learning: A Review and New Perspectives. arXiv:1206.5538 [cs], April 2014. URL http://arxiv.org/abs/1206.5538 arXiv:1206.5538 version: 2.

Du, S. S., Hu, W., Kakade, S. M., Lee, J. D., and Lei, Q. Few-Shot Learning via Learning the Representation, Provably. arXiv:2002.09434 [cs, math, stat], March 2021. URL http://arxiv.org/abs/2002.09434.

Fisher, R. A. The Use of Multiple Measurements in Taxonomic Problems. Annals of Eugenics, 7(2):179–188, 1936. ISSN 2050-1439. doi: 10.1111/j.1469-1809.1936.tb02137.x. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1469-1809.1936.tb02137.x. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1469-1809.1936.tb02137.x.

Goodfellow, I., Bengio, Y., and Courville, A. Deep learning. Adaptive computation and machine learning. The MIT Press, Cambridge, Massachusetts, 2016. ISBN 978-0-262-03561-3.

Hinton, G. E. and Zemel, R. S. Autoencoders, minimum description length and Helmholtz free energy. In Proceedings of the 6th International Conference on Neural Information Processing Systems, NIPS’93, pp. 3–10, San Francisco, CA, USA, November 1993. Morgan Kaufmann Publishers Inc.

Kingma, D. P. and Welling, M. Auto-Encoding Variational Bayes. Technical Report arXiv:1312.6114, arXiv, May 2014. URL http://arxiv.org/abs/1312.6114 arXiv:1312.6114 [cs, stat] type: article.

LeCun, Y. and Cortes, C. The mnist database of handwritten digits, undefined, 2005. URL https://www.semanticscholar.org/paper/The-mnist-database-of-handwritten-digits-LeCun-Cortes/d52d1de1b90b69d296bc5b34c9310b7eaa99a2.

Lee, J. D., Lei, Q., Saunshi, N., and Zhuo, J. Predicting What You Already Know Helps: Provable Self-Supervised Learning. arXiv:2008.01064 [cs, stat], November 2021. URL http://arxiv.org/abs/2008.01064 arXiv:2008.01064.

Liu, X., He, P., Chen, W., and Gao, J. Multi-Task Deep Neural Networks for Natural Language Understanding. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pp. 4487–4496, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1441. URL https://aclanthology.org/P19-1441.

Min, S., Kim, H., Lee, B., and Yoon, S. Protein transfer learning improves identification of heat shock protein families. PloS One, 16(5):e0251865, 2021. ISSN 1932-6203. doi: 10.1371/journal.pone.0251865.

Ramsundar, B., Kearnes, S., Riley, P., Webster, D., Konerding, D., and Pande, V. Massively Multitask Networks for Drug Discovery. arXiv:1502.02072 [cs, stat], February 2015. URL http://arxiv.org/abs/1502.02072 arXiv:1502.02072.

Rasmussen, C. E. and Williams, C. K. Gaussian Processes for Machine Learning, 2019. ISBN 978-0-262-25683-4. URL http://www.vlebooks.com/vleweb/product/openreader?id=none&isbn=9780262256834 OCLC: 1178958074.

Rumelhart, D. E. and McClelland, J. L. Learning internal representations by error propagation. In Parallel distributed processing: Explorations in the microstructure of cognition: Foundations, pp. 318–362. 1987.

Tripuraneni, N., Jin, C., and Jordan, M. I. Provable Meta-Learning of Linear Representations. arXiv:2002.11684 [cs, stat], December 2021. URL http://arxiv.org/abs/2002.11684 arXiv:2002.11684.

Vijayakumar, S. and Schaal, S. Locally Weighted Projection Regression: An O(n) Algorithm for Incremental Real Time Learning in High Dimensional Space. In in Proceedings of the Seventeenth International Conference on Machine Learning (ICML 2000, pp. 1079–1086, 2000.
7 Appendix

A Training Methodology

Dataset Information. We conducted our experiments on 3 datasets:

- Iris: We use the version provided by Scikit-Learn. We apply z-score normalization after splitting the data.
- SARCOS: Using the dataset from the companion website of [Rasmussen & Williams] (2019). We only use the training set, as the original test sets overlaps with the original training set. We apply Z-score normalization after splitting the dataset.
- MNIST: Using the version provided by the torchvision package. We merged both the train and test sets into a single one splitting. We do not preprocess the data otherwise.

We split each of the dataset into 3 sets, a test set 20%, 24% validation, 56% training set. We use the training set and validation to train the AutoEncoders and Variational AutoEncoders (Sections 4.2, 4.4, 4.3). We use all three sets when considering Classifications and Regression tasks using the learned representations (Section 4.5).

Training Details. Models were trained on an internal cluster with various computing power allocations. We train our models using ADAM optimizer with learning rate 1e-3 for AutoEncoders and 1e-4 for the Variational AutoEncoders. We use an internal cluster composed of Nvidia RTX 6000 GPUs. For SARCOS, one model trained for 1000 epochs takes in average 1h-2h, leading to 100h-200h for a given architecture.

B Theorems and their proofs

Theorem 3.2 Let \( f : S \rightarrow \mathbb{R} \) be continuous and convex, where \( S \subseteq \mathbb{R}^M \) is the \( M \)-simplex defined by the points \( a_1, \ldots, a_{M+1}, M \geq 1 \). Then \( f \) reaches its maximum at one of the \( a_i \)'s.

Proof. First note that \( f \) is continuous on the nonempty compact set \( S \). It thus reaches its maximum. Let then \( a \in S \). We can thus write

\[
a = \sum_{i=1}^{M+1} \lambda_i a_i,
\]

where \( \lambda_i \in [0, 1], i = 1, \ldots, M+1, \sum_{i=1}^{M+1} \lambda_i = 1 \). Since \( f \) is convex, we have

\[
f(a) = f(\sum_{i=1}^{M+1} \lambda_i a_i) \leq \sum_{i=1}^{M+1} \lambda_i f(a_i) \\
\leq \max(f(a_1), \ldots, f(a_{M+1})),
\]

which proves the theorem. \( \Box \)
Theorem 3.3 Let $A = \{(c_1, ..., c_M) \in \mathbb{R}^M, c_i \in [0, 1], i = 1, ..., M, \sum_{i=1}^M c_i = 1\}$. We set

$\varphi(c_1, ..., c_M) = \sum_{i=1}^M |c_i - \frac{1}{M}|$ and $\psi(c_1, ..., c_M) = \sum_{i=1}^M (c_i - \frac{1}{M})^2$. Then, $\max_A \varphi = 2\frac{M-1}{M}$ and $\max_A \psi = \frac{M-1}{M}$.

Proof. For $M = 1$ we have $c_1 = 1$, so that both $\varphi$ and $\psi$ vanish and the assertion of the theorem holds true.

Let us now assume that $M \geq 2$. Note that $\varphi$ and $\psi$ are continuous and convex (as sums of continuous and convex functions). Then note that we wish to maximize $\varphi$ and $\psi$ on the $(M-1)$-simplex defined by the points (in $\mathbb{R}^{M+1}$) $(1, 0, \ldots, 0), \ldots, (0, 0, \ldots, 0, 1)$ in the affine hyperplane $\sum_{i=1}^M c_i = 1$. The result then follows from Theorem 3.3.

C Example of Feature Grouping Algorithm

D Feature Grouping

Figures 6, 7, 8, 9 showcase the difference between FIB scores and FIB scores obtained using Feature Grouping. From top to bottom, the figures represent the effects of FIB and Grouped FIB over 128, 256, 512 and 1024 Features. As the number of features grows, the noise injection packs the FIB (right) score closer to the maximal value. Nonetheless, using Feature Grouping leads to FIB scores spreading from the minimal score to the maximal score. The values are reported based on results from 10000 noise injections on the synthetic data discussed in section 4.3.

Figure 6: Uniform noise injection in 128 dimensional vectors using (left) FIB and (right) Grouped FIB with 10 groups. Colors are associated with % of dimensions with noise injection. After grouping, varying percentages of perturbation lead to different FIB scores.
Figure 7: Uniform noise injection in 256 dimensional vectors using (left) FIB and (right) Grouped FIB with 10 groups. Colors are associated with % of dimensions with noise injection. After grouping, varying percentages of perturbation lead to different FIB scores.

Figure 8: Uniform noise injection in 512 dimensional vectors using (left) FIB and (right) Grouped FIB with 10 groups. Colors are associated with % of dimensions with noise injection. After grouping, varying percentages of perturbation lead to different FIB scores.
Figure 9: Uniform noise injection in 1024 dimensional vectors using (left) FIB and (right) Grouped FIB with 10 groups. Colors are associated with % of dimensions with noise injection. After grouping, varying percentages of perturbation lead to different FIB scores.