Measuring Disentanglement: A Review of Metrics

Marc-André Carbonneau, Julian Zaïdi, Jonathan Boilard, and Ghyslain Gagnon, Senior Member, IEEE

Abstract—Learning to disentangle and represent factors of variation in data is an important problem in artificial intelligence. While many advances have been made to learn these representations, it is still unclear how to quantify disentanglement. While several metrics exist, little is known on their implicit assumptions, what they truly measure, and their limits. In consequence, it is difficult to interpret results when comparing different representations. In this work, we survey supervised disentanglement metrics and thoroughly analyze them. We propose a new taxonomy in which all metrics fall into one of the three families: intervention-based, predictor-based, and information-based. We conduct extensive experiments in which we isolate properties of disentangled representations, allowing stratified comparison along several axes. From our experiment results and analysis, we provide insights on relations between disentangled representation properties. Finally, we share guidelines on how to measure disentanglement.

Index Terms—Disentanglement, representation learning.

I. INTRODUCTION

In recent years, learning disentangled representations has attracted considerable attention from the machine learning community [1], [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [14], [15], [16], [17], [18], [19], [20], [21], [22], [23], [24], [25]. A disentangled representation independently captures true underlying factors that explain the data. Such representations offer many advantages: when used on downstream tasks, they improve predictive performance [15], [16], reduce sample complexity [6], [18], [26], [27], offer interpretability [1], [26], improve fairness [17], and have been identified as a way to overcome shortcut learning [28].

Originally, disentanglement was evaluated by visual inspection, but recent research efforts have been devoted to propose metrics for more rigorous evaluations [1], [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13]. Frequently, a new metric is proposed alongside a new representation learning method to highlight the benefits not captured by existing metrics. Unfortunately, it is often unclear what these metrics quantify and under which conditions they are appropriate [4], [9], [11], [16], [29]. Fair quantitative evaluation is important to assess research progress by comparing new representation learning methods with the state of the art but also equally important for practitioners when performing model selection and hyperparameter tuning [12].

While most metrics correlate on simple datasets, they do not on more complex and realistic data [16]. Moreover, this correlation does not mean that they lead to the selection of the same model, as observed in [4], [16], and [29]. We highlight this problem in our experiments in Section V-A. Having metrics that lead to different conclusions means that before choosing a model or a hyperparameter setting, one must choose an appropriate metric for the application. This is not a trivial task because existing metrics measure different properties of disentanglement and make different, often implicit, assumptions of these properties. Moreover, these metrics are sometimes complex procedures themselves subject to hyperparameter configuration. The goal of this article is to provide some guidance to practitioners for selecting a metric given an application.

Very few papers discuss how to measure disentanglement. Locatello et al. [16] conducted a large-scale study on disentanglement in the unsupervised setting. Their main conclusion is that disentangling predefined factors is impossible without inductive bias and that random seeds and hyperparameters have a greater impact on performance than the architecture of the studied models. They also conducted experiments to measure the degree of agreement of the six metrics used in this article. They found that five of the six metrics correlate on the simple dSprites dataset [1] but only mildly on other more realistic datasets. Unfortunately, no interpretations are given onto why one of the metrics sometimes inversely correlates with the others or why metrics measuring different properties strongly correlate. Eastwood and Williams [5] proposed a framework for the evaluation of disentangled representations. They identify three desirable properties of a disentangled representation: explicitness, compactness, and modularity. They introduce the idea that these properties should be quantified separately and propose a new metric decomposed into three parts. The key idea of measuring different properties separately is also advocated in [6]. The authors point out that one of the three properties, compactness, is of lesser interest in practical scenarios. We will discuss these properties in detail in Section II.

To our knowledge, work [11] is the only study focusing on comparing metrics. The authors organize metrics based on the basic disentanglement properties they measure. Then, they verify that metrics assign a high score to all perfect representations and a low score to all representations that do not satisfy the measured property. Through demonstrations, they expose failure cases for several metrics. This constitutes a significant step toward the theoretical analysis of metrics in extreme cases.

In this article, we propose an in-depth analysis of supervised disentanglement metrics with real-world applications in mind. We establish a clear taxonomy of metric families and underline their strengths and shortcomings. We compare the metrics with respect to many practical considerations, such as robustness to noise and hidden factors, nonlinear relationships, accuracy, calibration, and computational efficiency. We conduct exper-
ments that abstract the representation learning model and data, which allows us to generate representations for which we can accurately control and isolate the properties under study. Moreover, it also alleviates difficulties related to the identification of ground-truth generative factors in datasets. We focus our analysis on supervised metrics (i.e., metrics that require ground-truth factors) since there exist very few unsupervised metrics [9], [12], [13]. To our knowledge, this is the first time that such extensive and fully controlled experiments are conducted and that metrics are compared in depth.

Here is a summary of our contributions:
1) We carry out an extensive review of disentanglement metrics, where we expose underlying assumptions, implementation complexity, and other practical considerations.
2) We establish a clear taxonomy of metric families and underline their strengths and shortcomings.
3) We conduct experiments that eliminate ambiguities introduced by learning algorithms and datasets to directly measure a metric’s performance.
4) We release our code to allow for the use of our implementation metrics and the reproduction of our experiments.1
5) We provide recommendations for meaningful comparison between representations, as well as guidance for selecting appropriate metrics depending on the application context.

The rest of this article is organized as follows. We start by identifying desirable representation properties that we wish to quantify. In Section III, we define desirable characteristics for a metric. In Section IV, we survey existing metrics and present our taxonomy. In Section V, we present our experiments and their results. In Section VI, we discuss the implications of our findings and provide insight on how to measure disentanglement. We identify possible extensions to this article in the conclusion.

II. PROPERTIES OF A DISENTANGLED REPRESENTATION

Before analyzing metrics, we discuss what constitutes a disentangled representation. While there is no unanimously accepted definition of disentanglement, most agree on two main aspects [9], [14], [18], [26], [30], [31]. First, the representation has to be distributed. This means that an input is a composition of explanatory factors and corresponds to a single point in the representation space. As in [5], we call this point a code in the remainder of this article. Each factor is encoded in separate dimensions of the code. In Section II-A, we further discuss factor independence and its implications. Second, the representation should also encode relevant information for the downstream task. Depending on the application and the representation learning algorithm, the way codes and factors related to each other may vary significantly. We discuss information content in Section II-B.

A. Factor Independence in Representation

Factor independence means that variation in one factor does not affect other factors, i.e., there is no causal effect between them [32]. In a disentangled representation, factors are also independent in the representation space. In other words, a factor affects only a subset of the representation space, and only this factor affects this subspace. Most authors agree on the importance of this property, which has different names (e.g., disentanglement [5] and modularity [6]). In this article, we use the naming convention of [6] and refer to this property as modularity.

Some authors argue that the subset of the representation space affected by a factor should be as small as possible. Ideally, only one dimension completely describes a factor. This property is called completeness in [5] but is called compactness in [6]. In this article, we refer to this property as compactness.

The desirability of compactness relates to the type of factors for a given application. As argued in [6], enforcing compactness may be counterproductive. A group of code dimensions provides more flexibility when describing complex factors [33]. For instance, if an angle is represented as a single value $\theta \in [0, 2\pi]$, there is a discontinuity in the space at $2\pi$. Alternatively, if two dimensions encode the angle (e.g., $\sin \theta$ and $\cos \theta$), continuity is preserved. Furthermore, sometimes factors are complex concepts such as facial expression [19] or phonetic content [21], which are unlikely describable by a 1-D space. As a second argument against enforcing strict compactness, Ridgeway and Mozer [6] explained that allowing redundancy in the learned latent space allows for different equivalent solutions, which facilitates model optimization from a practical standpoint.

All metrics assume that a set of independent factors exists in the problem. However, in practice, identifying useful and interpretable independent factors represents a challenging task [23]. Factors must be conceptually independent but should also be statistically independent [2]. This condition is hard to satisfy in real-world datasets where certain factor realizations tend to co-occur more than others [14], [34]. For example, in a dataset of fruit images, we could be interested in two conceptually different factors: fruit type and color. However, color is statistically dependent on the fruit type. Dependent factors impact modularity score. If two factors were to share information, parts of the representation relate to both. The selection of factors is application-specific and is beyond the scope of this article.

B. Information Content

To be truly useful, a representation should completely describe explanatory factors of interest. In other words, it should be possible to retrieve the complete factor realization from a point in the representation space (code). We call this property explicitness as in [14].

Perfect explicitness entails that a generalizable relation between factors and codes was learned. The nature of this relation may vary and is implicitly assumed by some metrics. A linear relation between factors and codes is the simplest, and arguably most desirable, type of relation [6]. In applications such as user-specified conditioned generation, learning a monotonic relation, even if not linear, allows for intuitive navigation in the representation space.

While a monotonic relation is a desirable characteristic for continuous factors, sometimes they are best described as categorical. In that case, the learning model must partition the representation space in regions corresponding to each category.

1https://github.com/ubisoft/ubisoft-laforge-DisentanglementMetrics
The sample distribution in this space is multimodal. Navigating this representation space by increasing the value of a code makes little sense. Measuring disentanglement with categorical factors necessitates metrics that do not make assumptions on the nature of the factor–code relation.

Bengio et al. [26] argued that a good representation should be invariant to other factors and noise. Unfortunately, it is not always clear which factors are pertinent for the downstream tasks. This is why it is advocated to learn as many factors as possible and discard as little information as possible [19], [26].

### III. Characterization of Disentanglement Metrics

Section II established three main properties of disentangled representations: modularity, compactness, and explicitness. This section enumerates desirable characteristics for a disentanglement metric.

A metric should accurately measure a disentanglement property or a subset of the properties described in Section II. Ideally, the metric should not have failure modes as identified in [2] and [11]. The scoring range of the metric should be calibrated. The metric should attribute the minimum score to a completely random or fully entangled representation and a perfect score to a perfectly disentangled representation. We verify this in the experiment of Section V-B. We provide details on how to normalize the output range for metrics that were not normalized in their original implementation in Section V.

In addition to being calibrated, a metric score should also evolve linearly with the quality of the disentanglement properties that it measures. The worst case scenario would be a metric that acts as a step function, which offers poor score interpretability and makes the comparison between two models with the same score meaningless. Moreover, such behavior renders the metric highly unstable. We evaluate how linearly the metric scores change with respect to explicitness in Section V-B, as well as to compactness and modularity in Section V-C.

As discussed in Section II-B, every metric makes an implicit assumption on the shape of the factor–code relationship. Sometimes, the application may dictate the type of relationship expected. For instance, if a linear relation is expected, metrics, which penalize nonlinear relations [5], [7], are best equipped to assess the quality of a representation. However, in most situations, metrics should not make any assumptions and should capture nonlinear and multimodal relations. In Section V-E, we compare metrics on monotonic and increasingly nonlinear relations.

A metric should not be overly sensitive to hyperparameter configuration. Low parameter sensitivity ensures stability across different configurations. A metric overly sensitive to configuration behaves unpredictably and may lead to inaccurate conclusions when comparing models. Examples of hyperparameters include predictor parameters (e.g., regularization weights), discretization granularity, batch size, and validation protocol. The best way to mitigate hyperparameter sensitivity is to reduce the number of parameters in the first place.

In real-world applications, datasets are likely to be noisy. Metrics that measure compactness or modularity should be tolerant to noise, while explicitness metrics should reflect the amount of noise in the representation. We measure robustness to noise in the experiment of Section V-B. In the context of measuring disentanglement, explaining factors that are not targeted by the metric can also be viewed as sources of noise. In most real-world applications, there will be several factors that will not be identified and measured. We evaluate tolerance to these distracting factors in Section V-F.

Finally, there are practical considerations when choosing a metric. Some metrics have high computational complexity, while others require a large number of data points to yield meaningful results. We briefly discuss these considerations in Sections V-G and VI-B.

### IV. Overview of Metrics

This section surveys existing supervised metrics. We propose a new taxonomy that organizes metrics into three families. A family group’s metrics are based on their underlying working principle. Intervention-based metrics compare codes by creating subsets of data in which one or more factors are kept constant. Predictor-based metrics use regressors or classifiers to predict factors from codes. Information-based metrics leverage information theory principles, such as mutual information (MI), to quantify factor–code relationships. Inspired by [11], we further divide each family in groups based on the disentanglement property that the metrics are designed to measure. Holistic methods capture two or more properties in a single score. Fig. 1 shows all metrics organized following the proposed taxonomy. In the rest of this section, after introducing the notation, we go over all families in greater detail and describe metrics individually.

#### A. Notation

Inspired by [3], we denote a set of $N$ observations as $X = \{x_1, x_2, \ldots, x_N\}$. Each observation is assumed to be completely explained by a set of $M$ factors $V = \{v_1, v_2, \ldots, v_M\}$ through a generative process $g(v) \rightarrow x$. We denote $V = \{v_1, v_2, \ldots, v_N\}$ the set of factor realizations that produced $X$. A representation learning algorithm is a mapping $r(x) \rightarrow z$ where $z \in \mathbb{R}^d$ is a point in the learned code space denoted by $Z = \{z_1, z_2, \ldots, z_d\}$. $Z = \{z_1, z_2, \ldots, z_N\}$ is the set of all points in $X$ projected in the code space by $r(\cdot)$. Supervised disentanglement metrics compute a score by comparing $V$ to $Z$. Fig. 2 shows the notation. Throughout this article, boldface lowercase letters represent vectors.

#### B. Intervention-Based Metrics

The metrics in this family evaluate disentanglement by fixing factors and creating subsets of data points. Codes and factors in the subsets are compared to produce a score. To sample the fixed-size data subsets, these methods discretize the factor space. This sampling procedure necessitates large quantities of diverse data samples to produce a meaningful score. The main advantage is that these metrics do not make any assumptions on the factor–code relations. However, there are several hyperparameters to adjust such as the size and the number of data subsets, the discretization granularity, classifier hyperparameters, or the choice of a distance function. Finally, Kim and Mnih [2] and Seppiarskaia et al. [11] identified several failure modes (i.e., situations where the metrics wrongly score representations).

1) Z-Diff: The Z-diff metric [1], sometimes called the $\beta$-VAE metric, selects pairs of instances to create batches. In a batch, a factor $v_i$ is chosen randomly. Then, a fixed number of pairs are formed with samples $v^1$ and $v^2$ that have

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
the same value for the chosen factor ($v^1 = v^2$). Pairs are represented by the absolute difference of the codes associated with the samples ($p = |z^1 - z^2|$). The intuition is that code dimensions associated with the fixed factor should have the same value, which means a smaller difference than the other code dimensions. The mean of all pair differences in the subset creates a point in a final training set. The process is repeated several times to constitute a sizable training set. Finally, a linear classifier is trained on the dataset to predict which factor was fixed. The accuracy of the classifier is the Z-diff score. For a completely random classifier, we expect an accuracy of $1/M$, where $M$ is the number of factors. This can be used to scale the output closer to the $[0,1]$ range.

2) Z-Min Variance: The Z-min variance\(^2\) metric [2], also called the FactorVAE metric, was introduced to address some of the weaknesses of the Z-diff metric. The intuition is the same as for the Z-diff; code dimensions encoding a factor should be equal if the factor value is the same. First, all codes are normalized by their standard deviation computed over the complete dataset. For a subset, a factor is randomly selected and fixed at a random value. The subset contains sampled instances for which the selected factor is fixed at the selected value. Variance is computed over the normalized codes in the subset. The code dimension with the lowest variance is associated with the fixed factor. Several subsets are created and the factor–code associations are used as data points in a majority vote classifier. The Z-min Variance score is the mean accuracy of the classifier. As for Z-diff, random classifier accuracy of $1/M$ can be used to scale the output closer to the $[0,1]$ range.

3) Z-Max Variance: Z-max Variance\(^2\) metric [3], also known as R-FactorVAE, is similar to Z-min Variance. The main difference is the approach used to collect subsets of samples. Here, all factor values are fixed except one. This time the intuition is that if all factors are the same except one, code dimensions corresponding to the free factor should exhibit higher variance. A majority vote classifier is also used to compute the score, but it is the code dimension with the highest variance that is chosen as a training point.

4) Interventional Robustness Score: Interventional robustness score (IRS) [4] measures the difference in the code dimensions after modifying some factors while keeping others fixed. The intuition is that changes in nuisance factors should not impact code dimensions attributed to factors of interest. The metric revolves around a measure called post interventional disagreement (PIDA), which compares the codes from a set of instances before and after modification of the nuisance factors. In practice, the authors compute the maximum distance (e.g., $\ell_2$) incurred by a modification of the nuisance factors for each factor of interest realization. Then, a weighted average of the maximum distances over all these realizations is computed. The distances are weighted by the frequency of the factor realizations in the dataset. In that paper, they refer to this quantity as the expected maximal PIDA (EMPIDA). Finally, EMPIDA is normalized by the expected maximal deviation from the mean codes when there are not any fixed factors. We obtain a score between 0 and 1, which yields the final IRS score: $\text{IRS} = 1 - \text{normalize}(\text{EMPIDA})$. One caveat in practice when using IRS is that one must identify which codes are associated with each factor.

### C. Predictor-Based Metrics

These metrics train regressors or classifiers to predict factor realizations from codes ($f(z) \mapsto v$). Then, the predictor is analyzed to assess the usefulness of each code dimension in predicting the factors. These methods are naturally suited to measure explicitness. They are typically equipped to deal with continuous factors as well as categorical factors simply by choosing an appropriate predictor. However, compared to information-based metrics, they require more design choices and hyperparameter tuning. This means that a metric is more likely to behave differently from one implementation to another.

1) Disentanglement, Completeness, and Informativeness: Eastwood and Williams [5] proposed a complete framework
to evaluate disentangled representations instead of a single metric. They report separate scores for modularity, compactness, and explicitness, which they call disentanglement, completeness, and informativeness (DCI). Regressors are trained to predict factors from codes. Modularity and compactness are estimated by inspecting the regressor’s inner parameters to infer predictive importance weights $R_{ij}$ for each factor and code dimension pair. They use a linear lasso regressor or a random forest for nonlinear factor–code mappings. For the lasso regressor, the importance weights $R_{ij}$ are the magnitudes of the weights learned by the model, while the Gini importance [35] of code dimensions is used with random forests.

The compactness for factor $v_i$ is given by $C_i = 1 + \sum_{j=1}^{d} p_{ij} \log_d p_{ij}$, where $p_{ij}$ is the probability that code dimension $z_j$ is important to predict $v_i$. These probabilities are obtained by dividing each importance weight by the sum of all importance weights related to this factor: $p_{ij} = R_{ij} / \sum_{k=1}^{M} R_{ik}$. The compactness of the whole representation is the average compactness over all factors.

Similarly, the modularity for code dimension $z_j$ is given by $D_j = 1 + \sum_{i=1}^{M} p_{ij} \log_d p_{ij}$, where $p_{ij}$ is the probability that code dimension $z_j$ is important to predict only $v_i$. This time the importance weights are normalized with respect to codes: $p_{ij} = R_{ij} / \sum_{k=1}^{M} R_{kj}$. The modularity score for the whole representation is a weighted average of the individual code dimension modularity scores $\sum_{ij} p_{ij} D_j$. The scores are weighted by $p_j$ to account for codes that are less important to predict factors. The weight $p_j$ is the total importance for $z_j$ normalized by the sum of all importance weights: $p_j = \frac{\sum_{i=1}^{M} R_{ij}}{\sum_{k=1}^{M} \sum_{i=1}^{M} R_{ik}}$.

The prediction error of the regressor measures the explicitness of the representation. With normalized inputs and outputs, it is possible to compute the estimation error for a completely random mapping and use it to normalize the score between 0 and 1. We postulate that a representation is not explicit if the mean squared error (mse) of the predictor is higher than the expected mse between two uniformly distributed random variables (X and Y). It can be shown that $\text{mse} = \mathbb{E}[(X - Y)^2] = 1/6$. Thus, explicitness can be written as $1 - 6 \cdot \text{mse}$. In our implementation, values under 0 are reported as 0.

2) Explicitness Score: Ridgeway and Mozer [6] proposed to use a classifier trained on the entire latent code to predict factor classes, assuming that factors have discrete values. They suggest using a simple classifier such as logistic regression and report classification performance using the area under the receiver operating characteristic (ROC) curve (AUC-ROC). The final score is the average AUC-ROC over all classes for all factors. The AUC-ROC minimal value is 0.5, which means that the score needs to be normalized to obtain a value between 0 and 1. In our implementation, we balance weights in the loss of the logistic regression to account for class imbalance.

3) Attribute Predictability Score: Attribute predictability score (SAP) [7] attributes a score $S_j$ to all pairs of factor $v_i$ and code dimension $z_j$. A linear regression predicts a continuous factor from each code and $S_j$ is the $R^2$ score of the regression. For categorical factors, it fits a decision tree on codes and reports balanced classification accuracy. Scores corresponding to codes with energy below a user-specified threshold (i.e., dead codes) are set to 0. The final SAP score is obtained by computing the difference between the two highest $S_{ij}$ for all factors

$$\text{SAP} = \frac{1}{M} \sum_{i}^{M} S_i \cdot S_{io}. \quad (1)$$

In this equation, $S_i$ is the highest score for factor $v_i$, while $S_{io}$ is the second highest. $M$ is the number of factors. Similar $S_i$ and $S_{io}$ mean that explicitness is low if both values are low. Two similarly high values indicate that more than one code dimension encodes the factor, which means low compactness. This corresponds to the gap idea in mutual information gap (MIG) (Section IV-D1).

D. Information-Based Metrics

Information-based metrics compute a disentanglement score by estimating the mutual information between the factors and the codes. These methods require fewer hyperparameters than intervention- and predictor-based metrics. Moreover, they do not make assumptions on the nature of the factor–code relations.

While elegant in theory the estimation of entropy and MI is nontrivial in practice. Even assessing the quality of the estimators remains an open problem [36]. Aside from specific cases where the distribution of the spaces is known and simple, it requires quantization of both spaces or a sampling procedure, which needs to be parameterized. Most existing public MI-based metric implementations use the maximum likelihood estimator. For example, in the widely used disentanglement_lib, MI is computed as follows:

$$I(v, z) = \sum_{i=1}^{B_v} \sum_{j=1}^{B_z} \hat{P}(i, j) \log \left( \frac{\hat{P}(i, j)}{P(i) P(j)} \right). \quad (2)$$

Factor and code spaces are discretized in $B_v$ and $B_z$ bins. $P(i)$ and $P(j)$ are estimated as the proportion of samples assigned to bins $i$ and $j$, respectively, over all samples ($N$). Similarly, $\hat{P}(i, j)$ is the proportion of samples assigned to both bins $i$ and $j$. Problems arise when estimating from undersampled high-dimensional data [37]. This is the case when computing MI, or joint entropy, between a factor and more than one dimension of the latent space. Moreover, the estimated MI value is affected by the granularity of the discretization, which makes the metrics sensitive to this parameter.

1) Mutual Information Gap: MIG [8] computes the MI between each code and factor $I(v_i, z_j)$. Then, the code dimension with maximum MI is identified $I(v_i, z_\star)$ for each factor. Next, the second highest MI, $I(v_i, z_\star)$, is subtracted from this maximal value. This difference constitutes the gap. The gap is then normalized by the entropy of the factor

$$\text{MIG} = \frac{I(v_i, z_\star) - I(v_i, z_\star)}{H(v_i)}. \quad (3)$$

The MIG score of all factors is averaged to report one score. Robust MIG (RMIG) was proposed in [9]. It is identical to MIG in essence but proposes a more robust formulation when MI is computed from the input space, which does not apply in our context. For the remainder of this article, we will refer to both MIG and RMIG as MIG-RMIG because our results apply to both in the same way.

---

3https://github.com/google-research/disentanglement_lib
2) Joint Entropy Minus Mutual Information Gap: MIG verifies that the information related to a factor is expressed by only one code dimension (compactness). However, modularity is not directly measured. For instance, a code dimension could contain information about more than one factor. Joint entropy minus mutual information gap (JEMMIG) [9] addresses this drawback by including the joint entropy of the factor and its best code

\[ \text{JEMMIG} = H(v_i, z_\ast) - I(v_i, z_\ast) + I(v_i, z_\ast). \] (4)

As opposed to MIG, this metric indicates a high disentanglement quality with a lower score. The maximum value is bounded by \(H(v_i) + \log(B_z)\), where \(B_z\) is the number of bins used in the code space discretization. This means that JEMMIG can be rewritten as follows to get a score between 0 and 1

\[ \text{JEMMIG} = 1 - \frac{H(v_i, z_\ast) - I(v_i, z_\ast) + I(v_i, z_\ast)}{H(v_i) + \log(B_z)}. \] (5)

As done for MIG, JEMMIG is reported as the average for all factors \(v_i\).

3) MIG-Sup: It is an extension of MIG [10]. As JEMMIG, it addresses the fact that MIG measures compactness but does not measure modularity. It is designed to be used in conjunction with MIG. The idea is similar to MIG except that the MI gap is computed from the code point of view

\[ \text{MIG-sup} = I(z_j, v_\ast) - I(z_j, v_\ast) \] (6)

where \(v_\ast\) is the factor that has the highest MI with code dimension \(z_j\). \(v_\ast\) is the factor that has the second highest MI with code dimension \(z_j\). \(I(z_j, v_i)\) is the MI normalized by the entropy of the factor \(v_i\). MIG-sup is reported as the average gap over all meaningful code dimensions. Meaningful dimensions can be identified by comparing the magnitude of \(I(z_j, v_\ast)\) for every code dimension. In real-world scenarios, a threshold has to be set to decide which code dimension is meaningful. The same representation would obtain largely different scores depending on how selective is the threshold. Unfortunately, there is no objective way of setting this threshold unless we know the correct disentanglement score in advance. The authors of MIG-SUP did not provide detail on the meaningful code dimension selection scheme used in this article. To avoid thresholding, the code dimension selection process could be replaced with a scaling method inspired from DCI [5]. In our implementation, we consider all code dimensions.

4) Modularity Score: To measure modularity, in [6], the factor \(v_\ast\), which shares the maximum MI for each code dimension \(z_j\), is identified. This maximal MI value \(I(v_\ast, z_j)\) is then compared with MI values of all other factors

\[ \text{modularity} = 1 - \frac{\sum_{i \in \mathcal{V}_{zs}} I(i, z_j)^2}{I(v_\ast, z_j)^2(M - 1)}. \] (7)

We denote \(\mathcal{V}_{zs}\) as the set of all factors except \(v_\ast\) and \(M\) as the number of factors. The average modularity score over all codes is reported.

5) DCIMIG: DCIMIG [11] is a metric inspired by DCI and MIG. As MIG, it computes MI gaps between factors and code dimensions. As DCI, it analyzes a factor–code importance matrix. However, unlike DCI, DCIMIG reports a single score for all three disentanglement properties. DCIMIG starts by computing the MI between each factor and code dimension \(I(v_i, z_j)\). Then, the factor with maximum MI, \(I(v_i, z_j)\), is identified for each code. After, the second highest MI, \(I(v_i, z_j)\), is subtracted from this maximal value. Thus, we obtain a gap for each code dimension \(R_j = I(v_i, z_j) - I(v_i, z_j)\). Each of these gaps \(R_j\) relates to a code dimension and the factor for which MI is maximal. For each factor \(v_i\), we find all associated gaps \(R_j\) and use them as score \(S_j\) for this factor. If there are more than one \(R_j\) associated with the factor, \(S_j\) equals the highest \(R_j\). If there are none, \(S_j = 0\). Finally, the metric is the sum of all scores normalized by the total factor entropy

\[ \text{DCIMIG} = \frac{\sum_{j=1}^{M} S_j}{\sum_{i=1}^{M} H(v_i)}. \] (8)

V. Experiments

In the experiments, we abstract the relation \(z = r(g(v))\) by \(z = f(v)\). Except for Section V-A, we do not learn representations on datasets, but instead, we directly define \(f(v)\) as a function that allows for complete control over the parameters of the representation evaluated by the metrics. This removes any ambiguities related to the quality of the dataset, the choice of learning algorithm and its training, as well as the choice of factors to disentangle. We assume that factors are selected as follows \(v_i \parallel v_j, i \neq j\) as prescribed in [4]. The code for the experiments is publicly available.

A. Model Selection

In this experiment, we validate that using different metrics to perform model selection or hyperparameter tuning leads to different outcomes. The experiment emulates practitioners trying to tune model hyperparameters to maximize disentanglement. We perform a grid search using the metric scores as the maximization objective. We arbitrarily chose to optimize two hyperparameters for \(\beta\)-VAE [1]: the regularization strength \((\beta \in (0.001, 0.01, 0.1, 1, 10, 100))\) and the dimensionality of the representation space \((d \in \{2, 4, 8, 16, 32, 64\})\), which leads to 36 different hyperparameter configurations. We use two standard datasets Cars3D [38] and SmallNORB [39]. For each hyperparameter configuration, we train the model for 300k training steps using the Adam optimizer and a batch size of 64 similar to [16]. After training, we obtain 36 learned representations, one for each hyperparameter configuration. We produce a ranking of the learned representations based on the scores obtained with each metric. Then, we measure the agreement between rankings for each pair of metrics with the Kendall rank correlation coefficient [40]. We report the results in Fig. 3.

Our results show that two practitioners would have chosen different models if they measured disentanglement with different metrics, even if they were designed to quantify the same properties. This is consistent with the results obtained in [4], [16], and [29]. In Fig. 3, we observe that for the same dataset, some metrics correlate, but often, correlation is weak or even inverse. When comparing correlations across the two datasets, we observe a general correspondence in correlation directions, especially for strong correlations. This is reassuring since it indicates that the metrics are quantifying the same properties somewhat consistently across datasets. However, the magnitudes of these correlations vary indicating that there is an interplay between the nature of the data, the learned representations, and the metric behaviors.
B. Perfect Disentangled Representation With Noise

In this section, we evaluate how metrics behave in a scenario where we gradually depart from a perfect disentangled representation to a completely random representation. In a perfect representation, factors completely describe the data and have a one-to-one relation with codes. This scenario shows how metrics behave as explicitness decreases under perfect compactness and modularity. We also verify that metrics are well calibrated (i.e., attribute a perfect score to perfect representation and a low score to noise). The factor–code relation is defined by

\[ z = f(v) = (1 - \alpha)v + \mathbf{n} \]  

where \( \mathbf{n} \sim \mathcal{U}(0, 1) \), \( \alpha \in [0, 1] \), and \( \mathbf{v}, \mathbf{z} \in \mathbb{R}^{M \times d} \). We simulate a problem with eight factors \( (d = M = 8) \). We tried a different number of factors and found conclusions to be similar. Given a set of factor realizations \( V \), we use \( f(\cdot) \) to obtain its representation in the code space \( Z \). The set \( V \) contains \( N = 20k \) samples from the uniform distribution. We use the same 20k samples for all metrics. When necessary, factor values are discretized into ten equal bins. We evaluate \( \alpha \) at \( \{0.0, 0.2, 0.4, \ldots, 1.0\} \). We repeat the experiment with 100 different sampled versions of \( V \) using 100 random seeds and report the average result. Fig. 4 shows the mean score for all metrics as the noise level \( (\alpha) \) increases.

Most metrics recognize a perfect representation and attribute a perfect score. There are three exceptions. IRS is unlikely to produce a perfect score for any representation because it computes a distance between codes for factors that are binned together. Factors in the same bin are likely to differ within the range of the bin, which in turn results in small distances in code values for the same factor bin. This explains why IRS cannot attribute a perfect score to a perfect representation. To circumvent this problem, smaller discretization needs to be applied if the number of samples is large enough for the given application. The explicitness score is the average of AUC-ROC for \( M \times 10 = 80 \) logistic regression classifiers trained in a one-versus-the-rest strategy. One classifier is trained for each bin value per factor. The optimizer does not consistently find the optimal solution for all classifiers, which leads to an AUC-ROC under 1. Z-max Variance requires a dense combination of factor values to sample meaningful batches for the majority vote classifier. The 20k examples used in the experiment, when discretized in 10 bins, do not provide enough examples for the same factors’ realization. The direct consequence is a biased estimation of the variance, which causes a score under 1 for a perfect representation and a score higher than 0 for a completely random representation. To circumvent this problem, coarser discretization needs to be applied, which in turn might lead to an overestimation of the scores.

The majority of metrics attribute a score near 0 to complete noise. However, DCI for modularity and compactness scores the representation over 0.3 when using a lasso regressor. Even if the regressor accuracy is low, weights are still learned and compared to compute compactness and modularity scores. The regularization term in lasso pushes some weights toward 0, and thus, sizable differences between them will be observed. This leads to observing random isolated factor–code relations that drive the score up. When using the modularity score, the MI between each factor and code dimension should be similar. However, a maximal MI value normalizes the score, which leads to a wrongfully optimistic value in most experiments in this article.

When measuring explicitness under noise, an ideal metric score should steadily decrease as the noise level increases. IRS is a perfect example of a score that decreases linearly with...
noise. In fact, most metrics that focus on explicitness perform adequately. If explicitness metric scores should decrease in the presence of noise, we expect a different behavior from modularity or compactness metrics. Ideally, a metric should recognize these disentanglement properties, even in noisy representations. The predictor-based DCI exhibits high noise robustness, which makes sense since predictors naturally discard noise information to improve generalization. This being said, their tendency to observe random isolated factor–code relations discussed above inflates this perception of noise robustness. In addition to being well calibrated, intervention-based metrics Z-diff and Z-min Variance also proved to be quite robust. Inversely, this experiment exposes the vulnerability to noise of information-based metrics. Noise causes codes to be assigned to neighboring bins, which decreases the observed MI between factors and codes.

C. Decreasing Compactness and Modularity

In Section V-B, we observed how metrics behave as explicitness decreases. Now, we study what happens as we gradually decrease compactness and modularity, while explicitness remains perfect. The embedding function is constructed given by $z = f(v) = vR$. The projection matrix $R$ is defined by

$$R = \begin{bmatrix}
1 - \alpha & \alpha & 0 & \cdots & 0 \\
0 & 1 - \alpha & \alpha & \cdots & 0 \\
0 & 0 & 1 - \alpha & \cdots & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
\alpha & 0 & 0 & \cdots & 1 - \alpha
\end{bmatrix}.
$$

When $\alpha = 0$, $R$ is the identity matrix and the representation is perfectly compact and modular. As $\alpha$ increases, all factors are represented by two code dimensions and each code dimension relates to two factors. Fig. 5 shows how metric scores evolve as the representation becomes less modular and less compact.

Results from this experiment reveal several differences among metrics. Since the representation allows for complete recovery factor values, explicitness metrics maintain a high score as expected. We would expect modularity and compactness metric scores to linearly decrease as $\alpha$ increases. This is the case for most information-based metrics and predictor-based metrics. Interestingly, some metrics output a 0 score when a code dimension relates to two factors or vice versa. This means that these metrics make no distinction between a representation where a code dimension relates to two factors.

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
and a representation where a code dimension relates to all factors. DCI is the best equipped metric to quantify this distinction because it will never yield a zero score unless all codes are equally relevant to predict the factors. The experiment also reveals a failure mode of intervention-based metrics as identified in [11]. These metrics consistently attribute a high score to the representations even when it is imperfect. The worst case is Z-diff that attributes a perfect score even when $\alpha = 0.5$. It is always trivial for the classifier to identify a factor by finding the distinct combination of two code dimensions with the lowest difference.

D. Modular but not Compact

Here, we evaluate how the metrics behave when the representation is perfectly explicit and modular but not compact. As discussed in Section II-A, compactness is of lesser interest than modularity in many real-world applications. Thus, it is important to assess the ability of the metrics to recognize modularity even when several code dimensions are used to describe a single factor.

The first experiment of this section emulates a model that has learned a decomposed representation of angles. When a scalar defines an angle, the representation space has a discontinuity at $2\pi$. Decomposing angles in sine and cosine values ensures that the space is continuous, which is preferred in many applications. Here, each factor represents an angle $\theta \in [0, 2\pi]$. Codes represent angles as $\cos \theta$ and $\sin \theta$. Factor realizations define four angles: $v = [\theta_1, \theta_2, \theta_3, \theta_4]$ and the corresponding codes are given by $z = [\cos \theta_1, \sin \theta_1, \cos \theta_2, \ldots, \sin \theta_4]$. Factor values are discretized to 10 bins ($v_i \in [0, \pi/5, 2\pi/5, \ldots, 9\pi/5]$).

Following the same idea, we create a second dataset where factors are encoded by two code dimensions. However, this time, factor–code relations are linear. This corresponds to a scenario where the representation learning algorithm has learned redundant codes. This scenario allows for comparison of results obtained in the previous experiment without having to account for the nonlinear relations (sine and cosine). We keep the same four factors, but we use linear relations: $v = [\theta_1, \theta_2, \theta_3, \theta_4]$ corresponds to $z = [\theta_1, \theta_1, \theta_2, \ldots, \theta_4]$.

Finally, we repeat the same experiment except that there are only two factors associated with four code dimensions each: $v = [\theta_1, \theta_2]$ corresponds to $z = [\theta_1, \theta_1, \theta_2, \ldots, \theta_2]$. Following the sampling methodology described in Section V-B, we compute the metrics and report scores in Table I.

On the left of the result table, we can see that none of the intervention-based metrics penalizes representations for not being compact. This is in accordance with the results from the previous experiment. Predictor-based metrics exhibit different behaviors depending on the type of predictor used. The lasso predictor, unsurprisingly, has trouble dealing with the nonlinear sine and cosine relations. More interestingly, it has problems dealing with redundant codes. Since only one code dimension is necessary to predict a factor, the information from the duplicated code dimensions is discarded, encouraged by the regularization term. This falsely leads the metric to think that only one code dimension is associated with the factor, hence the perfect compactness for all experiments. Using a random forest predictor overcomes this problem. As observed in the preceding experiment, SAP and MIG, which measure compactness, cannot express to what degree a representation is not compact. This is because they compute a gap that subtracts the two most significant terms and ignores all of the others. The same can be said for JEMMIG. DCIMIG while intended as a holistic method does not penalize noncompactness in this experiment. Finally, we can observe that information-based metrics have trouble dealing with nonlinear relations. This will be discussed in greater detail in the next experiment.

E. Nonlinear Relations

Here, we explore representations with nonlinear relations between factors and codes. The representation is kept perfectly compact and modular and should receive a perfect score from all metrics. The mapping function becomes increasingly nonlinear as $\alpha$ increases but is always monotonic for $v \in [0, 1]$.

$$z = f(v) = 1000^{-\alpha+0.25} \tan(\omega(v - 0.5)) + 0.5 \quad (10)$$

where $\omega = 2 \arctan(1000^{\alpha-0.25}/2)$. When $\alpha = 0$, the relation is practically linear, and when $\alpha = 1$, the relation takes the shape of a tangent function, as shown in Fig. 6(a). This relation is interesting because it highlights potential problems with using a linear regressor to compute scores, as well as potential problems inherent to discretization. The results are reported in Fig. 7.

As expected, predictor-based metrics using linear regression to measure explicitness, DCI lasso and SAP, underperform as the factor–code relation becomes less linear. The monotonic nature of the relation allows DCI lasso to accurately score modularity and compactness. Naturally, a more expressive predictor makes the metric robust to more complex relationships.

| TABLE I |
|---|
| SCORES ATTRIBUTED TO DISENTANGLEMENT WHERE A FACTOR IS ENCODED WITH MORE THAN ONE CODE |
| $\theta \rightarrow [\cos \theta, \sin \theta]$ | $\cos \theta$, $\sin \theta$, $\cos \theta$, $\sin \theta$ | $\cos \theta$, $\sin \theta$, $\cos \theta$, $\sin \theta$ | $\cos \theta$, $\sin \theta$, $\cos \theta$, $\sin \theta$ | $\cos \theta$, $\sin \theta$, $\cos \theta$, $\sin \theta$ | $\cos \theta$, $\sin \theta$, $\cos \theta$, $\sin \theta$ | $\cos \theta$, $\sin \theta$, $\cos \theta$, $\sin \theta$ |
| $\theta \rightarrow [\theta, \theta]$ | $\cos \theta$, $\sin \theta$, $\cos \theta$, $\sin \theta$ | $\cos \theta$, $\sin \theta$, $\cos \theta$, $\sin \theta$ | $\cos \theta$, $\sin \theta$, $\cos \theta$, $\sin \theta$ | $\cos \theta$, $\sin \theta$, $\cos \theta$, $\sin \theta$ | $\cos \theta$, $\sin \theta$, $\cos \theta$, $\sin \theta$ | $\cos \theta$, $\sin \theta$, $\cos \theta$, $\sin \theta$ |
| $\theta \rightarrow [\theta, \theta, \theta]$ | $\cos \theta$, $\sin \theta$, $\cos \theta$, $\sin \theta$ | $\cos \theta$, $\sin \theta$, $\cos \theta$, $\sin \theta$ | $\cos \theta$, $\sin \theta$, $\cos \theta$, $\sin \theta$ | $\cos \theta$, $\sin \theta$, $\cos \theta$, $\sin \theta$ | $\cos \theta$, $\sin \theta$, $\cos \theta$, $\sin \theta$ | $\cos \theta$, $\sin \theta$, $\cos \theta$, $\sin \theta$ |
This experiment highlights potential problems with discretization, which is at the center of information-based metrics, as well as intervention-based metrics and even some predictor-based metrics such as the explicitness score. Equal binning of the code space results in a larger amount of the population being assigned to the middle bins. Fig. 6(b) shows the proportion of samples assigned to each discretization bin when $\alpha = 0$ and $\alpha = 1$. This uneven population distribution lowers the code space entropy and in turn affects MI computation. Similarly, it affects how subsets are created in intervention-based metrics. This explains why a large proportion of metrics fail to properly score the perfectly modular, compact, and explicit representation.

F. When Factors Partially Describe Data

This experiment simulates the case where metrics measure only a fraction of all the generative factors. This frequently happens in real-world scenarios because it is difficult to identify all generative factors in a dataset. For instance, channel noise may corrupt data and get modeled in some dimensions of the code as in [41]. These nonmeasured factors still need to be encoded to preserve explicitness because they can be useful for downstream tasks.

From the metric point of view, code dimensions corresponding to nonmeasured factors are seen as noise or dead codes [5], which affects metric scoring. We generate perfect representations in the same way as in Section V-B but without noise ($\alpha = 0$). The relation between factors and codes becomes the identity $z = f(v) = v$. Then, we apply the metrics to these perfect representations and vary the proportion of measured factors. In other words, we ignore the existence of some of the factors when computing the metric scores.

When metrics measure all of the eight generative factors captured by the perfect representation, their score should be maximal. Metrics should maintain that the maximal value as the proportion of measured factors decreases because the representation does not change. Fig. 8 shows how metric scores evolve as the proportion of measured factors increases.

Most metrics are equipped to deal with nonmeasured factors, except for Z-max Variance, IRS, MIG-sup, and the modularity score. This limits their relevance in contexts outside of academic toy problems. Successful methods that measure disentanglement from the code point of view implement a mechanism to discard dead codes [5]. A dead code is a code dimension that does not inform on any factor. The modularity score does not provide such mechanisms. Also, our implementation of MIG-sup does not account for dead codes because it requires tuning a rejection threshold, which is not applicable in practice. Borrowing strategy to deal with dead codes from other metrics would be beneficial, as discussed in Section IV-D3. When sampling to create subsets, IRS and Z-max Variance implicitly assume that when all known factors are fixed, the corresponding codes are also fixed. This assumption is violated when there are other sources of variation for the code than the known factors.

G. Sample Efficiency

This section studies how many data points the metrics need to get a fair estimation of the representation score. The intuition is that an ideal metric should attribute the same score regardless of the number of samples observed for the same representation. While it does not make sense to expect a fair estimation of the true score without a minimal quantity of samples, this minimal quantity differs across metrics.

In the experiment, we create random representations: $z = f(v) = vR$, where the projection matrix $R$ is filled by sampling in the uniform distribution and each element is replaced by 0 with a 0.75 probability. We create these random representations to avoid biases that could be created by perfect or completely noisy representations. We generate 100k samples for representation. Then, we apply each metric to the first 100, 1k, and 10k samples and finally to the whole 100k sample set. We compute the absolute difference between each score and the full 100k samples score. We repeat this process 100 times and average the differences. We report the results in Fig. 9.

All metrics provide a fair estimate of the disentanglement score 1k samples except for Z-max Variance and the explicitness score. The Z-max Variance metric requires a large amount of data because of the way it constructs batches on which it measures code variance. In a batch, all factors except one are fixed. This entails that the dataset must contain several examples in which the majority of the code is the same, which is only possible with extremely large datasets. In our experiment, there are seven fixed code dimensions each quantized to 10 bins. It is difficult to cover the $10^7$ code possibilities with only $10^5$ samples and obtain a reliable...
Estimate. In the explicitness score, the distribution of samples across its internal classes affects the shape of the multiple ROC curves leading to an unstable estimation without large datasets.

Intervention-based metrics rely on sampling to create batches, which necessitates a large number of samples to provide a reliable estimate. Predictor-based metrics necessitate sufficient data to train a predictor that generalizes.
The quantity of needed data points depends on the predictor model complexity. Simpler models, such as linear regressions, require less data than more complex models such as random forests. This explains the gap observed between the DCI Lasso and DCI RF. On the other hand, information-based metrics are generally more sample efficient because they are free from stochastic components. For all metrics, the minimal quantity of data needed to get a fair estimate depends on the number factor and their distribution in the dataset, as well as the latent space dimensionality.

VI. DISCUSSION

This section summarizes our learning from the experiment results and insights from relevant papers. After discussing relations between representation properties, we identify best practices for measuring disentanglement in real-world applications. Finally, we provide recommendations on how measurements should be reported.

A. Relations Between Representation Properties

While disentanglement properties can be measured separately, they are implicitly linked together. This makes the analysis of disentanglement more difficult and might have motivated the holistic approach of some metrics.

Evidently, some degree of explicitness is necessary to observe modularity or compactness; otherwise, it would not be relevant to compare factor–code relations. However, as shown in the experiment of Section V-B, a high level of modularity and compactness can be observed, even when explicitness is minimal. In other words, explicitness is a necessary condition for modularity and compactness, but the magnitude of these properties does not inform on the magnitude of the explicitness.

Modularity and compactness are linked together by the size of the code space. When all factors are represented, if the code space is the same dimensionality as the number of factors, perfect modularity necessarily implies perfect compactness. This relation is not symmetric. Perfect compactness does not necessarily mean perfect modularity in this situation. A code dimension could encode two factors even if each factor is encoded by only one dimension. This would, however, mean that there are one or more dead codes. As the code space increases, under perfect modularity, imperfect compactness is possible. The ratio between the code space dimensionality and the number of factors determines how much compactness is allowed to deteriorate. As a general rule, the code space should be larger than the measured factor space to allow for composite factors and nonmeasured factors. The fact that the code space size links compactness and modularity could explain correlations sometimes observed between metrics that focus on only one of these properties as in [16].

Modularity is more important in practice than compactness. This has already been stated in [14]. There are several reasons why researchers and practitioners should focus on modularity instead of compactness. The main reason is that measuring compactness is desirable only if one can identify atomic (1-D) factors, which is often difficult or impossible in real-world applications. As mentioned earlier, some basic concepts, such as angle or color, are best represented in a 2-D or 3-D space. In addition, any composite factors (e.g., object type in images or speaker identity in speech segments) are more meaningfully represented in a multidimensional space, where each dimension represents an atomic factor. These atomic factors are sometimes concepts difficult to identify, describe, and measure. For example, if one wants to disentangle speaker identity from speech segments. Many atomic factors define a voice print. Some are simpler to identify and measure such as pitch and speech rate, but they do not paint the whole picture. The complete set of atomic factors for voice print remains elusive, even for speech experts. Nonetheless, they must be encoded to successfully perform a downstream task such as speaker identification or conditioned speech synthesis. The same goes for illumination in a picture. In practice, one might want to isolate the effect of a light source. However, light sources have many attributes, such as 3-D position, direction, color, shape, size, and intensity. All these factors have to be explicitly identified and quantified to measure compactness. For both these example applications, useful disentanglement is best measured through modularity. A high modularity score indicates that atomic factors of interest are contained in a defined subset of the code space.

The factor space and the code space need to be aligned to accurately measure disentanglement. Perfect compactness and modularity entails complete disentanglement of generative factors. However, it is possible to learn a representation where factors are completely disentangled and yet measure low compactness and modularity scores because of a misalignment between space axes. As a thought experiment, if we take a perfectly disentangled representation, where each factor corresponds to only one code, and rotate this representation space around any axis, the resulting rotated representation space maintains the independence between factors. However, existing metrics will fail to capture perfect modularity or compactness because variations in one factor will cause variation on several dimensions of the code space and vice versa. We believe that a metric should be robust to this kind of misalignment and be able to evaluate representations by looking at them from the right “point of view.” In practice, axis alignment can be enforced during learning through supervision or indirectly encouraged as in VAEs [22], [42] but cannot be guaranteed in most unsupervised learning settings [16].

B. Practical Considerations for Choosing a Metric

In this section, we extract conclusions from our analysis and experimental results. We provide guidance for choosing an appropriate metric for real-world applications, and we highlight practical considerations when measuring disentanglement.

Table II compiles our experimentation results and analysis. For a metric to possess a characteristic ($\phi$), it has to be true by design and not disproven experimentally. For instance, DCI with lasso regressor is marked with ($\beta$) because it has a failure mode when measuring compactness as shown in Section V-D, even if, in theory, it can measure the property. The same goes for metrics necessitating discretization when dealing with nonlinear relations.

Metrics affected by nonmeasured factors should be avoided in real-world scenarios. As discussed in Section II-A, identifying factors in practice is challenging. Identifying all factors is even more difficult. Moreover, when identified, factors must be measured, which is sometimes impossible. This means that for most applications, there will exist unidentified factors explaining the data, which will cause some metrics to underestimate modularity, as shown in Section V-F.
Using discretization is not trivial and has an impact on score. As shown in Section V-E, discretization of the code and the factor space has a considerable impact on the ability of metrics to deal with nonlinear relations.

The granularity of the discretization has an impact on the estimated MI, which is the centerpiece of information-based metrics. Intuitively, MI informs on how easy it is to predict a variable \( A \) knowing \( B \). Suppose that \( A \) is a random variable and \( B = A + \sigma \), where \( \sigma \) is random noise. On one extreme, if both variables are discretized in 1 bin, then the MI is maximum. On the other end of the spectrum, \( A \) and \( B \) are discretized in a large number of narrow bins. If the number of samples is limited, it is unlikely that \( B \) will help predict the exact bin of \( A \). In that case, MI will appear to be low even if there exists a strong relation between \( A \) and \( B \). This being said, when representation distributions are simple, MI can be analytically computed and these considerations can be avoided.

In intervention-based metrics, the discretization granularity determines the degree of similarity/dissimilarity of examples grouped in the same subset. A too coarse discretization creates heterogeneous groups that are considered homogeneous, which biases results. A too fine discretization makes it impossible to create large enough subsets of data points with the same fixed value. To our knowledge, no procedure has been proposed yet to strike the right balance between coarse and fine discretization for any type of metric.

DCI implemented with random forest is the best all-around metric. Measuring disentanglement properties separately allows for accurate scoring. Because random forest is an expressive model, it can discover nonlinear relationships and does not suffer from problems related to discretization. Moreover, random forests can be used as classifiers and regressors, which makes them appropriate for applications mixing continuous and categorical factors. DCI implements a weighting scheme that accounts for dead codes in problems where not all factors can be identified. However, there are three disadvantages to DCI. First, modeling relations with random forests requires a bit of expertise to set the hyperparameters and determine a relevant criterion for code dimension importance. The hyperparameters must be tuned using an appropriate cross-validation procedure, to ensure proper regularization of the model; otherwise, it will overfit, which results in an overestimation of explicitness as well as an underestimation of modularity and compactness. This cross-validation procedure is time-consuming, which is the second main disadvantage of the method. In fact, DCI with random forest is by far the most computationally expensive of all metrics implemented in this article. Finally, training reliable RF models requires appreciable quantity of data points when compared to some other metrics.

In their current state, metrics in the intervention-based family should be used with great caution. They require large quantities of data to create subsets with fixed values. This prohibits their application in problems with limited quantities of data with labeled factors. They are subject to vulnerabilities associated with discretization. Moreover, they are prone to failure modes, which limits their reliability. Finally, unlike most metrics from other families, they do not produce a factor–code relation matrix, which makes their results difficult to interpret and less helpful when debugging.

Information-based metrics are in theory flexible and elegant. They can measure factor–code relations of any shape, continuous or categorical, with a minimal amount of hyperparameter tuning and few data points. However, the aforementioned challenges with discretization limit their universality and make them vulnerable to noise. Also, metrics based on information gaps, such as MIG, only consider the difference between the two best candidates. This limits their expressiveness. For instance, in the experiment of Section V-D, MIG attributes the same compactness score (0.0) to representations where a factor corresponds to two and four code dimensions. We believe that if these limitations were addressed, information-based metrics would be more interesting solutions.
C. Reporting Results

Disentanglement properties should be measured separately. We share this opinion with [5] and [6]. In our experiments, we showed that we could vary properties independently and get the same overall score in very different situations. Metrics measuring all at once make the analysis and comparison of algorithms imprecise. This is particularly true in cases where a parameter balances reconstruction error and factor separation (e.g., β-VAE [1]). Using a single metric to measure both explicitness and modularity makes it impossible to determine the contribution of each property to the score.

Disentanglement should be measured for each factor independently. While global scores give a quick impression on disentanglement quality, they do not paint the whole picture and can be deceiving. It is impossible to tell from a single number if a model performs generally well except on a few problematic factors or equally badly on all of them. The first case might indicate a problem with the data or the choice of factors, while in the second, it indicates poor performance of the representation model.

Metrics should be run several times on the same representation. One should report average scores alongside standard deviation. Some metrics implement stochastic components. For instance, intervention-based metrics sample subsets on which they rest their analysis. Predictor-based metrics create validation sets to perform hyperparameter tuning. Moreover, in applications with large datasets, representations are evaluated on a subset of samples for efficiency. This sampling process adds to the stochasticity of the evaluation even for stable metrics. Performing several measurement runs allows performing statistical significance tests on results to ascertain conclusions from experiments, which should be standard practice when comparing different solutions.

A minimal sample set size is required to get an accurate estimation of the disentanglement score of a learned representation. As explained in Section V-G, this minimal quantity depends on each metric, factor distribution, and code space dimensionality. For instance, in our experiment, metrics, such as DCI RF and Z-min Variance, provide an estimate of the true score that most likely differs by ±0.03 from the representation true score, even with N = 10,000, while information-based metrics fare better than their counterparts when fewer samples are available. This score estimation error should be considered when comparing representations and calls for caution when drawing conclusions.

VII. Conclusion

In this work, we studied how to quantify disentanglement in representations. We conducted an extensive review of supervised disentanglement metrics. We analyzed and compared them experimentally with real-world applications in mind. We reviewed definitions of disentanglement and proposed a new taxonomy organizing the metrics into three families: intervention-based, predictor-based, and information-based.

We highlighted the lack of correlation between the different metric scores and exposed their differences in a series of fully controlled experiments on the robustness to noise, modularity, compactness, hidden factors, calibration, and nonlinear relationships. Our experiments revealed different limitations for each metric. We showed how discretization hinders reliability under limited amount of data, noise, and nonlinear factor–code relations. We found that predictor-based metrics, when parameterized with caution, were the best performing family of solutions. We discussed the importance of modularity over compactness for practical applications. We concluded, perhaps unsurprisingly, that each disentanglement property should be measured separately for better interpretability.

While we shed some light on the inner working assumption of supervised metrics, several open questions remain. We think that some of the limits exposed in the study can be solved, and thus, some metrics, notably from the information-based family, could prove to be stronger solutions than they are now. Also, supervised metrics necessitate factors to be identified and measured, which is not always possible when dealing with real-world data. This is why research efforts are now increasingly focused on measuring disentanglement without ground-truth factors. This study intentionally left out unsupervised metrics, which is open for future work.

References

[1] I. Higgins et al., “β-VAE: Learning basic visual concepts with a constrained variational framework,” in Proc. Int. Conf. Learn. Represent., 2017, pp. 1–22.
[2] H. Kim and A. Mnih, “Disentangling by factorising,” in Proc. Int. Conf. Mach. Learn., 2018, pp. 2649–2658.
[3] M. Kim, Y. Wang, P. Sahu, and V. Pavlovic, “Relevance factor VAE: Learning and identifying disentangled factors,” 2019, arXiv:1902.01568.
[4] R. Suter, D. Miladinovic, B. Schölkopf, and S. Bauer, “Robustly disentangled causal mechanisms: Validating deep representations for interventional robustness,” in Proc. Int. Conf. Mach. Learn., 2019, pp. 6056–6065.
[5] C. Eastwood and C. K. I. Williams, “A framework for the quantitative evaluation of disentangled representations,” in Proc. Int. Conf. Learn. Represent., 2018, pp. 1–15.
[6] K. Ridgeway and M. C. Mozer, “Learning deep disentangled embeddings with the F-statistic loss,” in Proc. Adv. Neural Inf. Process. Syst., 2018, pp. 1–10.
[7] A. Kumar, P. Sattigeri, and A. Balakrishnan, “Variational inference of disentangled latent concepts from unlabeled observations,” in Proc. Int. Conf. Learn. Represent., 2018.
[8] R. T. Q. Chen, X. Li, R. B. Grosse, and D. K. Duvenaud, “Isolating sources of disentanglement in variational autoencoders,” in Proc. Adv. Neural Inf. Process. Syst., 2018, pp. 1–16.
[9] K. Do and T. Tran, “Theory and evaluation metrics for learning disentangled representations,” in Proc. Int. Conf. Learn. Represent., 2020, pp. 1–30.
[10] Z. Li, J. V. Murkute, P. K. Gyawali, and L. Wang, “Progressive learning and disentanglement of hierarchical representations,” in Proc. Int. Conf. Learn. Represent., 2020, pp. 1–14.
[11] A. Sepliarskaia, J. Kiseleva, and M. de Rijke, “How to not measure disentanglement,” 2019, arXiv:1910.05587.
[12] S. Duan et al., “Unsupervised model selection for variational disentangled representation learning,” in Proc. Int. Conf. Learn. Represent., 2020, pp. 1–29.
[13] X. Liu, S. Thomos, G. Valvano, A. Chartsias, A. O’Neill, and S. A. Tsafarlis, “Measuring the biases and effectiveness of content-style disentanglement,” 2020, arXiv:2008.12378.
[14] K. Ridgeway, “A survey of inductive biases for factorial representation-learning,” 2016, arXiv:1612.05299.
[15] F. Locatello, M. Tschannen, S. Bauer, G. Rätsch, B. Schölkopf, and O. Bachem, “Disentangling factors of variations using few labels,” in Proc. Int. Conf. Learn. Represent., 2020, pp. 1–24.
[16] F. Locatello et al., “Challenging common assumptions in the unsupervised learning of disentangled representations,” in Proc. Int. Conf. Mach. Learn., 2019, pp. 4114–4124.
[17] F. Locatello, G. Abbati, T. Rainforth, S. Bauer, B. Schölkopf, and O. Bachem, “On the fairness of disentangled representations,” in Proc. Adv. Neural Inf. Process. Syst., 2019, pp. 1–14.
[18] S. van Steenkiste, F. Locatello, J. Schmidhuber, and O. Bachem, “Are disentangled representations helpful for abstract visual reasoning?” in Proc. Adv. Neural Inf. Process. Syst., 2019, pp. 1–14.
G. Desjardins, A. Courville, and Y. Bengio, “Disentangling factors of variation via generative entangling,” 2012, arXiv:1210.5474.

X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel, “InfoGAN: Interpretable representation learning by information maximizing generative adversarial nets,” in Proc. Adv. Neural Inf. Process. Syst., 2016, pp. 1–9.

G. Boulianne, “A study of inductive biases for unsupervised speech representation learning,” IEEE/ACM Trans. Audio, Speech, Language Process., vol. 28, pp. 2781–2795, 2020.

C. P. Burgess et al., “Understanding disentangling in β-VAE,” in Proc. Workshop Learn. Disentangled Represent. 31st Conf. Neural Inf. Process. Syst., 2017, pp. 1–11.

E. Mathieu, T. Rainforth, N. Siddharth, and Y. W. Teh, “Deep visual analogy-making,” in Proc. 36th Int. Conf. Mach. Learn., 2019, pp. 4402–4412.

V. Thomas et al., “Disentangling the independently controllable factors of variation by interacting with the world,” in Proc. Workshop Learn. Disentangled Represent. 31st Conf. Neural Inf. Process. Syst., 2017, pp. 1–9.

O. Press, T. Galanti, S. Benaim, and L. Wolf, “Emerging disentanglement in auto-encoder based unsupervised image content transfer,” in Proc. Int. Conf. Learn. Represent., 2019, pp. 1–26.

Y. Bengio, A. Courville, and P. Vincent, “Representation learning: A review and new perspectives,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 35, no. 8, pp. 1798–1828, Aug. 2013.

B. Schölkopf, D. Janzing, J. Peters, E. Sgouritsa, K. Zhang, and J. Mooij, “On causal and anticausal learning,” in Proc. Int. Conf. Mach. Learn., 2012, pp. 1–8.

R. Geirhos et al., “Shortcut learning in deep neural networks,” 2020, arXiv:2004.07780.

A. H. Abdi, P. Abolmaesumi, and S. Fels, “A preliminary study of disentanglement with insights on the inadequacy of metrics,” 2019, arXiv:1911.11791.

J. Schmidhuber, “Learning factorial codes by predictability minimization,” Neural Comput., vol. 4, no. 6, pp. 863–879, 1992.

D. Bouchacourt, R. Tomioka, and S. Nowozin, “Multi-level variational autoencoder: Learning disentangled representations from grouped observations,” in Proc. 32nd AAAI Conf. Artif. Intell., 2018, pp. 1–8.

J. Peters, D. Janzing, and B. Schölkopf, Elements of Causal Inference: Foundations and Learning Algorithms. Cambridge, MA, USA: MIT Press, 2017.

B. Esmaeili et al., “Structured disentangled representations,” in Proc. Int. Conf. Artif. Intell. Statist., 2019, pp. 2525–2534.

F. Triebel et al., “On disentangled representations learned from correlated data,” 2020, arXiv:2006.07886.

L. Breiman, “Random forests,” Mach. Learn., vol. 45, no. 1, pp. 5–32, 2001.

L. Paninski, “Estimation of entropy and mutual information,” Neural Comput., vol. 15, no. 6, pp. 1191–1253, 2003.

J. Haussler and K. Strimmer, “Entropy inference and the James–Stein estimator, with application to nonlinear gene association networks,” J. Mach. Learn. Res., vol. 10, no. 50, pp. 1469–1484, 2009.

S. E. Reed, Y. Zhang, Y. Zhang, and H. Lee, “Deep visual analogy-making,” in Proc. Adv. Neural Inf. Process. Syst., 2015, pp. 1–9.

Y. LeCun, F. Jie Huang, and L. Bottou, “Learning methods for generic object recognition with invariance to pose and lighting,” in Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2004, pp. 1–8.

M. G. Kendall, Rank Correlation Methods, London, U.K., Charles Griffin, 1948.

W.-N. Hsu et al., “Hierarchical generative modeling for controllable speech synthesis,” in Proc. Int. Conf. Learn. Represent., 2019, pp. 1–27.

M. Rolínek, D. Zietlow, and G. Martius, “Variational autoencoders pursue PCA directions (by Accident),” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2019, pp. 12406–12415.

Marc-André Carbonneau received the Ph.D. degree from the École de technologie supérieure (Université du Québec), Montreal, QC, Canada, in 2017, for his work on multiple instance learning. He currently works as a Research Scientist at the La Forge Research Laboratory, Ubisoft La Forge, Montreal. He leads a group of researchers applying machine learning to video games. His research interests include machine learning, weakly supervised learning, active learning, speech, and signal processing.

Julian Zaïdi received the bachelor’s degree in aeronautical engineering from ISAE ENSICA, Toulouse, France, in 2015, and the master’s degree in mechanical engineering from Polytechnique Montreal, Montreal, in 2017.

He worked for five years as a Researcher at La Forge Research Laboratory, Ubisoft Montreal, where he applied machine learning techniques for speech synthesis in video games. He currently works at Meta, Montreal, QC, Canada, as a Machine Learning Engineer, where he uses machine learning to improve integrity on Facebook.

Jonathan Boilard received the B.Eng. degree in electrical engineering at the University of Sherbrooke, Sherbrooke, QC, Canada, in 2018, and the M.Sc. degree from the École de technologie supérieure, Montreal, QC, Canada, in 2022, with a focus on disentangled representations for audio applications.

He is a young professional currently focusing on his data science career. He is currently working at Canam Group, Boucherville, QC, Canada, where he applies data science in the steel manufacturing industry. He worked on multiple proofs of concept, such as a vision system for welding inspection based on object detection, machine learning-driven costing models for manufacturing time prediction, and structural steel detailed design prediction using recommender systems.

Ghyslain Gagnon (Senior Member, IEEE) received the Ph.D. degree in electrical engineering from Carleton University, Ottawa, ON, Canada, in 2008.

He is currently the Dean of research and a Professor at the École de Technologie Supérieure, Montreal, QC, Canada. He is a Board Member of ReSMiQ, Montreal, QC, and was the Director of research laboratory LACIME, Montreal, QC, from 2013 to 2020, a group of 15 faculties and nearly 150 highly dedicated students and researchers in microelectronics, digital signal processing, and wireless communications. Highly inclined toward research partnerships with industry, his research aims at microelectronics, digital signal processing, and machine learning with various applications, including health care, media art, and building energy management.