Sample and Predict Your Latent: Modality-free Sequential Disentanglement via Contrastive Estimation

Ilan Naiman [*1] **Nimrod Berman** [*1] **Omri Azencot** [1]

Abstract

Unsupervised disentanglement is a long-standing challenge in representation learning. Recently, self-supervised techniques achieved impressive results in the sequential setting, where data is time-dependent. However, the latter methods employ modality-based data augmentations and random sampling or solve auxiliary tasks. In this work, we propose to avoid that by generating, sampling, and comparing empirical distributions from the underlying variational model. Unlike existing work, we introduce a self-supervised sequential disentanglement framework based on contrastive estimation with *no* external signals, while using common batch sizes and samples from the latent space itself. In practice, we propose a unified, efficient, and easy-to-code sampling strategy for semantically similar and dissimilar views of the data. We evaluate our approach on video, audio, and time series benchmarks. Our method presents state-of-the-art results in comparison to existing techniques. The code is available at GitHub.

1. Introduction

One of the main challenges for modern learning frameworks in tackling new tasks is the lack of high-quality real-world labeled data. Unfortunately, labeling massive amounts of data is a time-consuming process that typically requires expert knowledge. Unsupervised learning is a modeling paradigm for learning without labels, and thus it gained increased attention in recent years (Sohl-Dickstein et al., 2015). Recent approaches utilize the inputs as supervisory signals (Chen et al., 2020a) and use pretext tasks (Misra & Maaten, 2020), yielding highly-competitive self-supervised learning (SSL) frameworks (Caron et al., 2020). The goal of this paper is to study the effect of a novel SSL approach on sequential disentanglement problems.

[*]Equal contribution [1]Department of Computer Science, Ben-Gurion University of the Negev, Beer-Sheva, Israel. Correspondence to: Ilan Naiman <naimani@post.bgu.ac.il>.

Proceedings of the 40th International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

Data disentanglement is related to representation learning, where semantic latent representations are sought, to be used in various downstream tasks. A common goal in sequential disentanglement is the factorization of data to time-invariant (i.e., static) and time-variant (i.e., dynamic) features (Hsu et al., 2017). Most sequential disentanglement approaches for arbitrary data modalities such as video, audio, and time series are unsupervised, modeling the task via variational autoencoders (VAE) (Hsu et al., 2017; Yingzhen & Mandt, 2018; Han et al., 2021). Effectively, the static and dynamic factors are obtained via separate posterior distributions.

Self-supervised learning appeared only recently in sequence disentanglement works via supervisory signals, pretext tasks, and contrastive estimation. However, existing SSL introduces several shortcomings as it depends on the underlying modality. In this work, modality refers to the properties of the data or task. For instance, (Zhu et al., 2020) design auxiliary tasks per data type, e.g., predict silence in audio segments or detect a face in an image. Similarly, (Bai et al., 2021) require positive and negative samples with respect to the input, i.e., same-class and different-class examples, respectively. In practice, positive views are obtained via data-dependent data augmentation transformations such as rotations and cropping, whereas negative views are selected randomly from the batch. To increase the variability in the batch, common solutions address these issues by increasing the batch or creating a memory bank, resulting in high memory costs. In this work, we refer to the above approaches as modality-based supervision methods, and we argue that they can be avoided if the underlying model is generative.

To alleviate the above disadvantages, we design a novel sampling technique, yielding a new contrastive learning framework for disentanglement tasks of *arbitrary* sequential data that is based on the following insights. First, variational autoencoders naturally support the comparison of empirical distributions and their sampling. Second, we observe that a sample may be contrasted with its subsequent VAE prediction, leading to an increase in batch variability while keeping its size fixed. Based on these observations, we will show that we generate good positive and negative views. We evaluate our method on several challenging disentanglement problems and downstream tasks, and we achieve beyond state-of-the-art (SOTA) performance in comparison to several strong baseline methods.

Contributions. Our main contributions are listed below.

1. We observe that VAEs allow to compare and construct empirical distributions while facilitating a contrastive evaluation of samples and their prediction.
2. We propose a novel similarity and sampling technique that exploits inherent properties of VAEs, is modality-free, and it yields good views during training.
3. We present SOTA results on video, audio, as well as time series data; our empirical evaluation includes data and task modalities for which existing SSL approaches are not necessarily effective, such as the Physionet ICU dataset (Goldberger et al., 2000).

2. Related Work

Contrastive learning. Drawing two semantic views of the same object closer already appeared in signature verification systems (Bromley et al., 1993). However, issues such as *catastrophic collapse* were identified in (Chopra et al., 2005) i.e., learned views may "collapse" to a constant function, yielding zero similarity but alas, non-useful representations. To avoid collapse, they draw similar inputs closer, while separating dissimilar inputs via a contrastive term. Gutmann et al. (2010) suggested a popular and theoretically-sound framework to discriminate between observed data and artificial noise, known as noise contrastive estimation. In Contrastive Predictive Coding (CPC) (Oord et al., 2018), the authors generate robust representations by contrasting the predicted next frame. These results have been integrated in vision tasks with frameworks such as SimCLR (Chen et al., 2020a) and InfoMin (Tian et al., 2020). Recent works suggest regularizing the loss (Tsai et al., 2021) and utilizing contrastive learning losses (Zhu et al., 2021).

Contrastive sampling. Contrastive learning techniques require semantically similar samples, as well as semantically dissimilar samples. These examples are often referred to as positive and negative examples, respectively. Often, positive samples are obtained via data augmentation tools, whereas negative samples are selected randomly (Le-Khac et al., 2020). A recent study (Tian et al., 2020) devises optimality conditions on positive views. Specifically, they show that one should reduce the mutual information (MI) between views while keeping task-relevant information intact. We now describe several sampling approaches for positive and negative views that can be considered as reducing MI.

As mentioned above, the majority of approaches use data augmentation for positive sampling (Bachman et al., 2019; Chen et al., 2020a). In (Sermanet et al., 2018), positive examples are obtained from video frames of different views for the same action, e.g., pouring coffee into a cup. A related idea appeared in (Han et al., 2019), where given a collection of videos, they generate samples by considering different videos, and different locations/times within a video. (Ho & Vasconcelos, 2020) use positive adversarial samples to further enhance the effect of contrastive learning.

While significant attention was given to positive sampling techniques, recent approaches focus on negative sampling that goes beyond random selection from the batch (Doersch & Zisserman, 2017) or from a memory bank (Wu et al., 2018; Misra & Maaten, 2020; He et al., 2020). The main issue with random sampling is that it may yield negative examples which are actually positive, an issue known as sampling bias (Chuang et al., 2020). To address the latter problem, Kalantidis et al. (2020); Robinson et al. (2020) construct negative samples by measuring their similarity to the current sample. Recently, Ge et al. (2021) generate negative examples with superfluous features, and similarly, Huynh et al. (2022) aim at discarding semantic information from negative samples. Ash et al. (2021) studied the effect of the number of negative samples on performance. Finally, a few techniques showed impressive results with no negative examples at all (Chuang et al., 2020; Grill et al., 2020).

Disentanglement methods. Separating the underlying factors of variation is a well-established research problem on static image data (Kulkarni et al., 2015; Higgins et al., 2016; Kim & Mnih, 2018; Chen et al., 2018). Disentanglement of sequential data is an emerging field, and it focuses on data factorization to static and dynamic factors. Hsu et al. (2017) introduced unsupervised disentanglement of sequential data via an LSTM-based model on audio data. Later, Yingzhen & Mandt (2018) suggested DSVAE using a similar LSTM architecture while adding a heuristic in which the dynamic features' dimension is small compared to the static features' size. Further, Tulyakov et al. (2018) proposed an adversarial setup. S3VAE (Zhu et al., 2020) improves DSVAE by adding mutual information penalties on the relation between the static and dynamic features and the input, and in addition, they used auxiliary signals. Han et al. (2021) also suggested improving DSVAE by replacing the Euclidean distance with a Wasserstein distance. Toneka-boni et al. (2022) use a VAE model to disentangle arbitrary time series data. C-DSVAE (Bai et al., 2021) includes a contrastive estimation of the mutual information losses introduced by S3VAE. They employ data augmentations for contrastive loss estimation, using a similar architecture as S3VAE and DSVAE. Recent work by Berman et al. (2023) developed structured Koopman autoencoders to promote multifactor disentanglement of the sequential data to more than two semantic components. Our work builds on the architecture and objective of C-DSVAE while overcoming some of its shortcomings. Specifically, we design a simple framework for sampling good positive and negative samples. Our approach is modality-free, i.e., it does not depend on the data domain (video, audio, or time series), nor does it depend on the task (e.g., images of faces or letter images).

Contrastive disentanglement. Several works considered contrastive estimation in the context of disentanglement of latent factors (Lin et al., 2020; Li et al., 2021; Wang et al., 2021). Here, we focus on disentanglement of sequential data. For instance, Wei et al. (2022) employ a contrastive triplet loss for unsupervised video domain adaptation. Self-supervision in sequential disentanglement of arbitrary data appeared only recently. Zhu et al. (2020) utilize auxiliary tasks and supervisory signals, whereas Bai et al. (2021) use contrastive estimation, following the standard augmentation and random sampling for constructing positive and negative examples, respectively, and using the `infoNCE` loss.

3. Background

Problem formulation. Given a dataset $\mathcal{D} = \{x^j_{1:T}\}_{j=1}^N$ of time series sequences $x_{1:T} = \{x_1, \dots, x_T\}$ where the index j is omitted for brevity, our goal is to find a posterior distribution $p(s, d_{1:T} \,|\, x_{1:T})$ of *disentangled* static and dynamic latent representations, s and $d_{1:T}$ respectively, such that $x_{1:T} \sim p(x_{1:T} \,|\, s, d_{1:T})$. We elaborate below on the constraints and assumptions related to the factors and data.

Probabilistic modeling. Our discussion follows closely existing works such as (Yingzhen & Mandt, 2018; Bai et al., 2021). The static factor s and dynamic factors $d_{1:T}$ are assumed to be independent, x_i depends only on s and d_i, and d_i depends on the previous dynamic factors $d_{<i} = \{d_1, \dots, d_{i-1}\}$. Under these assumptions, we consider the following joint distribution

$$p(x_{1:T}, z) = \left[p(s) \prod_{i=1}^{T} p(d_i \,|\, d_{<i}) \right] \cdot \prod_{i=1}^{T} p(x_i \,|\, s, d_i) \;, \quad (1)$$

where $z = (s, d_{1:T})$. The prior distributions $p(s)$ and $p(d_i \,|\, d_{<i})$ are taken to be Gaussian with $p(s) := \mathcal{N}(0, I)$ and $p(d_i \,|\, d_{<i}) := \mathcal{N}(\mu(d_{<i}), \sigma^2(d_{<i}))$.

The posterior distribution $p(s, d_{1:T} \,|\, x_{1:T})$ disentangles static from dynamic, and it is approximated via

$$q(z \,|\, x_{1:T}) = q(s \,|\, x_{1:T}) \prod_{i=1}^{T} q(d_i \,|\, d_{<i}, x_{\leq i}) \;, \quad (2)$$

i.e., s is conditioned on the entire sequence, whereas d_i depends on previous d_j and inputs, and current inputs.

The variational autoencoder (VAE) (Kingma & Welling, 2014) relates the prior and approximate posterior distributions in a regularized reconstruction loss. For mutually independent s and $d_{1:T}$ this loss takes the following form,

$$\begin{aligned} \mathcal{L}_{\text{VAE}} = \lambda_1 \, &\mathbb{E}_{q(z \,|\, x_{1:T})} \log p(x_{1:T} \,|\, z) \\ - \lambda_2 \, &\text{KL}[q(s \,|\, x_{1:T}) \,\|\, p(s)] \\ - \lambda_3 \, &\text{KL}[q(d_{1:T} \,|\, x_{1:T}) \,\|\, p(d_{1:T})] \;, \end{aligned} \quad (3)$$

where $\text{KL}[q \,\|\, p]$ is the Kullback–Leibler divergence that computes the distance between distributions q and p, and $\lambda_1, \lambda_2, \lambda_3 \in \mathbb{R}^+$ are weight hyperparameters.

In practice, the likelihood $p(x_i \,|\, s, d_i)$ in Eq. (1), $p(d_i \,|\, d_{<i})$ and the terms $q(s \,|\, x_{1:T})$ and $q(d_i \,|\, d_{<i}, x_{\leq i})$ in Eq. (2) are all obtained via separate LSTM modules. Sampling from the sequential distribution $p(d_i \,|\, d_{<i})$ is achieved by using the mean and variance the LSTM outputs when feeding d_{i-1}, and similarly for $q(d_i \,|\, d_{<i}, x_{\leq i})$. Finally, we use the mean squared error (MSE) for reconstruction in Eq. (3) and the KL terms are computed analytically. Further network architectural details are given in App. A.3.

Mutual information disentanglement. Similar to the catastrophic collapse observed in (Chopra et al., 2005), VAE models may produce non-informative latent factors (Bowman et al., 2016). In sequential disentanglement tasks, this issue manifests itself by condensing the static and dynamic information into $d_{1:T}$. An empirical heuristic has been partially successful in mitigating this issue, where a low-dimensional d_i and a high-dimensional s are used (Yingzhen & Mandt, 2018), thus d_i is less expressive by construction. However, a recent theoretical result (Locatello et al., 2019) shows that unsupervised disentanglement is impossible if no inductive biases are imposed on models and data. Thus, to alleviate these challenges, several existing works (Zhu et al., 2020; Han et al., 2021; Bai et al., 2021) augmented model (3) with *mutual information* terms.

The main idea in introducing mutual information (MI) terms is to separately maximize the relation in pairs $(s, x_{1:T})$ and $(d_{1:T}, x_{1:T})$, while minimizing the relation of $(s, d_{1:T})$. This idea is realized formally as follows (Bai et al., 2021),

$$\mathcal{L}_{\text{MI}} = \lambda_4 I_q(s; x_{1:T}) + \lambda_4 I_q(d_{1:T}; x_{1:T}) - \lambda_5 I_q(s; d_{1:T}) \;, \quad (4)$$

where $I_q(u; v) = \mathbb{E}_{q(u,v)} \log \frac{q(u \,|\, v)}{q(u)}$. Combining the above losses (3) and (4), the disentanglement model reads

$$\max_{p,q} \mathbb{E}_{x_{1:T} \sim p_{\mathcal{D}}} \mathcal{L}_{\text{VAE}} + \mathcal{L}_{\text{MI}} \;, \quad (5)$$

where $p_{\mathcal{D}}$ is the empirical distribution of the dataset $\mathcal{D}$. Bai et al. (2021) shows that problem (5) is a proper evidence lower bound (ELBO) of the log-likelihood of (1).

Estimating the MI terms is not straightforward. A standard approach uses mini-batch weighted sampling (MWS) (Chen et al., 2018). In contrast, Bai et al. (2021) approximated MI terms via a contrastive estimation known as `infoNCE`,

$$\mathcal{L}_{\text{iNCE}} = \log \frac{\phi(u, v^+)}{\phi(u, v^+) + \sum_{j=1}^{M} \phi(u, v^j)} \;, \quad (6)$$

where u is either the static factor or the dynamic features, i.e., $u \in \{s, d_{1:T}\}$. The samples v^+ and v^j correspond to

positive and negative views with respect to u. For instance, if $u := s$, then the static features $v^+ := s^+$ are similar to s. The function $\phi(u, v) = \exp(u^T v/\tau|u||v|)$ measures the similarity between examples u and v, with $\tau = 0.5$ being a temperature parameter (Chen et al., 2020a). It has been shown that $I_q(u; x_{1:T}) \approx \mathcal{L}_{\text{iNCE}}(u)$ under relatively mild conditions (Oord et al., 2018). Our model architecture and objective function follow C-DSVAE (Bai et al., 2021), while significantly improving their contrastive estimation by proposing a novel sampling procedure as we detail below.

4. Method

Employing contrastive estimation (6) within a sequential disentanglement framework requires positive and negative views of s and $d_{1:T}$ for a given input $x_{1:T}$. In practice, the positive samples are obtained via *modality-based* data augmentations such as cropping and color distortion for images, voice conversion for audio data, and shuffling of frames for general static augmentation. Negative views are obtained by randomly sampling from the batch. See, for instance, (Zhu et al., 2020; Bai et al., 2021). In this work, we argue that while random sampling and data augmentations with the `infoNCE` loss are popular tools for unsupervised learning (Chen et al., 2020a; Tian et al., 2020), one should revisit the core components of sequential contrastive learning. We will show that existing practices for sampling of views and for increasing the batch variation can be improved.

Shortcomings of views' sampling. Creating semantically similar and dissimilar examples is a challenging problem. We distinguish between *domain-modality* and *task-modality* sampling. In general, we will use the following definition of X-modality. A method Y is X-modality-dependent if Y depends on the characteristics of X. For instance, image rotation (Y) is domain-modality-dependent (X) as it will probably be less effective for audio sequences. Similarly, cropping of images (Y) may not be effective for images of letters, and thus it is task-modality-dependent (X). Namely, task-modality-dependent approaches may require separate sampling methods for the same data domain. In summary, modality may mean multiple concepts depending on the particular context, including the format of the data or its statistical features. In general, we argue below that existing disentanglement approaches are modality-based.

Existing studies show that the particular choice of DA can significantly affect results (Chen et al., 2020b; Tian et al., 2020; Zhang & Ma, 2022). Even shuffling of frames which may seem robust, can yield wrong views in critical healthcare applications involving data with the vital measurements of a patient. In conclusion, DA may heavily depend on domain knowledge and task expertise. DA which falls into one of the categories above is referred to as *modality-based* augmentations. To the best of our knowledge, the majority of data augmentation tools are modality-based.

Constructing negative views may seem conceptually simpler in comparison to positive views, however, it bears its own challenges. Common methods select randomly from the dataset (Doersch & Zisserman, 2017). To reduce the sampling bias of false negative views (Chuang et al., 2020), existing works suggest increasing batch sizes (Chen et al., 2020a) or using a memory bank (Wu et al., 2018). Yet, the memory footprint of these methods is limiting. To conclude, both data augmentation and randomness should be avoided in the construction of positive and negative views.

VAE-based sampling. Motivated by the above discussion, we opt for an efficient modality-free sampling approach. We make the following key observation:

variational autoencoders inherently support the formation, comparison, and sampling of empirical distributions

Essentially, given a dataset $\mathcal{D} = \{x^j_{1:T}\}_{j=1}^N$ of time series sequences and model (3), we can generate the individual posterior distributions $\{q(z^j \mid x^j_{1:T})\}_{j=1}^N$, compare them via the Kullback–Leibler divergence, and sample $z^j \sim q(z^j \mid x^j_{1:T})$.

We denote by $x_{1:T}$ the input for which we seek a positive $x^+_{1:T}$ and several negative $x^{-,j}_{1:T}, j = 1, \dots, M$, examples. Our discussion focuses on sampling static views $\{s^+, s^{-,j}\}$, however, a similar process can be performed for sampling dynamic features. Intuitively, s^+ is the factor such that the distance $\text{KL}[q(s \mid x_{1:T}) \,\|\, q(s^+ \mid x^+_{1:T})]$ is minimal, where $x^+_{1:T} \sim p_{\mathcal{D}}$. Similarly, $s^{-,j}$ are the features with maximal KL value. However, a subtle yet important aspect of views is the distinction between *soft and hard* samples. Soft negative examples are those which contribute less to learning as they are too dissimilar to the current sample, whereas hard views are the semantically-dissimilar examples that are close in latent space to $x_{1:T}$ (Kalantidis et al., 2020; Robinson et al., 2020). How would one obtain good views with large batch variability given the above observation?

To increase variation while avoiding memory banks and large batch sizes, we suggest to use the (non-increased) batch itself. Let $\tilde{q}_k(s \mid x_{1:T})$ denote the partially-trained posterior after k epochs of training. We denote by $D \in \mathbb{R}^{n\times n}$ the pairwise KL divergence distances matrix for a batch of size n, $\{x^j_{1:T}\}_{j=1}^n$. Namely,

$$D_{ij} := \text{KL}[\tilde{q}(s^i \mid x^i_{1:T}) \,\|\, \tilde{q}(s^j \mid x^j_{1:T})] \,, \tag{7}$$

where $i, j \in \{1, \dots, n\}$ and we omit the training epoch for brevity. For a particular example in the batch $x^i_{1:T}$, we generate good views based on the following heuristic. We sort the row $D_{i:}$ in ascending order, and we sample positive views from the *first* third of distributions, whereas negative views are sampled from the *last* third. We denote by $S^+(i) = \{\tilde{q}(s^j \mid x^j_{1:T})\}$ the set of positive distributions,

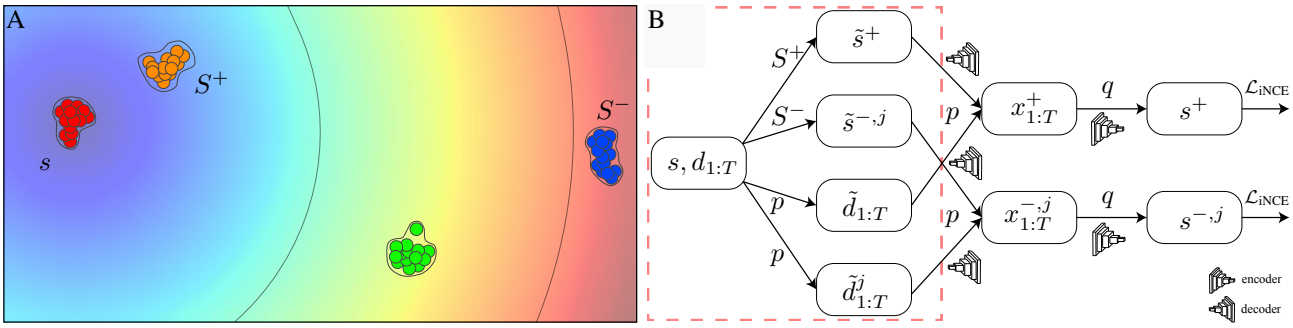

Figure 1. A) To generate positive and negative static views of s, we collect the closest S^+ and farthest S^- distributions in the batch. We sample from these distributions using the reparameterization trick. B) Unfortunately, samples from the batch have limited variation (dashed red rectangle), and thus we use our predictive sampling trick, generating samples by using the posterior of the sampled prior.

and similarly, $S^-(i)$ holds the negative distributions. See Fig. 1 for an illustration of these definitions.

Predictive sampling trick. Unfortunately, as usual batch sizes are relatively small, it may occur that variability is limited in the original batch. Notice that soft positive views always exist via the posterior of the sample itself, and soft negatives probably exist as well for moderate batch sizes, e.g., for $n = 16, 32$. However, it is not clear whether hard views exist in the batch, and thus its variability may need to be increased. To improve variability, we introduce our *predictive sampling trick*.

Again, w.l.o.g. we focus on the setting of sampling static views of a given example $x_{1:T}$ with its static and dynamic features s and $d_{1:T}$. To increase the variability in the views, our predictive sampling trick generates these examples from *the posterior of the sampled prior*. For instance, to produce a positive static view, we denote $\tilde{s}^+ \sim S^+$. The dynamic features can be arbitrary, and thus we sample from the prior $\tilde{d}_{1:T} \sim p(d_{1:T})$. The positive instance $x^+_{1:T}$ is defined via $x^+_{1:T} \sim p(x^+_{1:T} \,|\, \tilde{s}^+, \tilde{d}_{1:T})$. We obtain the positive static view by sampling the posterior, i.e.,

$$s^+ \sim q(s^+ \,|\, x^+_{1:T}) \,. \tag{8}$$

A similar process is used to compute $s^{-,j}, j = 1, \dots, M$. These views $\{s^+, s^{-,j}\}$ are utilized in $\mathcal{L}_{\text{iNCE}}(s, s^+, s^{-,j})$, see the diagram of our predictive sampling in Fig. 1B. We find that our views' heuristic and predictive sampling trick yield soft to semi-soft positive examples and semi-hard to hard negative examples, see Sec. 5.6. For additional implementation details, see App. B.

Our approach is based on the implicit assumption that the underlying model (3) encourages similar examples to be close and dissimilar views to be farther apart. Indeed, previous work on this model (Yingzhen & Mandt, 2018) showed this tendency when using large s and small d_i. Thus, our approach can be viewed as promoting the natural tendency of the model to separate positive and negative views.

5. Results

5.1. Datasets and Methods

In our evaluation, we consider several datasets of different modalities, and we compare our results with several state-of-the-art approaches. Specifically, we test on video datasets such as Sprites (Reed et al., 2015) and MUG (Aifanti et al., 2010) containing animated cartoon characters and subjects performing facial expressions, respectively. Moreover, we also use the Jester dataset (Materzynska et al., 2019) with videos of hand gestures, and the Letters corpus (Ibrahim et al., 2019) with handwritten text. For audio, we experiment with TIMIT (Garofolo et al., 1992). Finally, we also explore time series datasets including Physionet (Goldberger et al., 2000) with individual medical records and Air Quality (Zhang et al., 2017) with measurements of multiple air pollutants. We compare our results to sequential disentanglement frameworks including MoCoGan (Tulyakov et al., 2018), FHVAE (Hsu et al., 2017), DSVAE (Yingzhen & Mandt, 2018), R-WAE (Han et al., 2021), S3VAE (Zhu et al., 2020), SKD (Berman et al., 2023), C-DSVAE (Bai et al., 2021), and GLR (Tonekaboni et al., 2022). See App. A for details.

5.2. Hyperparameters

To control the contribution of each loss component we add a λ_1 coefficient to the reconstruction loss, a λ_2 to the static KL term, a λ_3 to the dynamic KL term, and finally λ_4, λ_5 to the contrastive terms. The hyperparameter λ_1 is tuned over $\{1, 2.5, 5, 10\}$, λ_2 is tuned over $\{1, 3, 5, 7, 9\}$, and λ_4 and λ_5 are tuned over $\{0.1, 0.5, 1, 2.5, 5\}$ while λ_3 is fixed to 1. We used Adam optimizer (Kingma & Ba, 2014) with the learning rate chosen from $\{0.001, 0.0015, 0.002\}$. The static and dynamic features' dimensions are selected from $\{128, 256\}$ and $\{32, 64\}$, respectively. These dimensions are similar or sometimes smaller in comparison to all other benchmark models such as C-DSVAE, S3VAE, DSVAE, R-WAE. We highlight that tuning multiple hyperparameters is often challenging. Hence, we utilize automatic tuning tools,

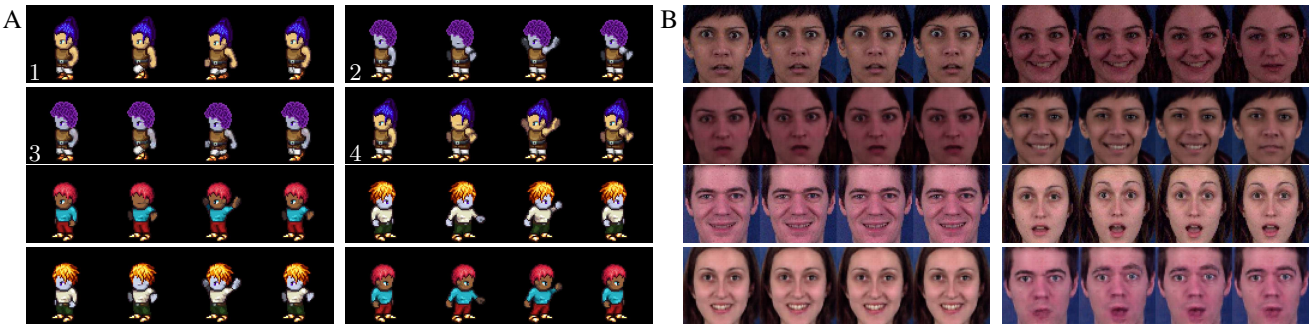

Figure 2. Content and pose swap results in Sprites (A) and MUG (B) datasets. See the text for additional details.

using 5 to 10 runs for each dataset. The hyperparameters for each task and dataset are given in Tab. 6 in the Appendix. All the tasks were trained for at most 600 epochs.

5.3. Qualitative Results

We begin our evaluation with qualitative examples showing the disentanglement capabilities of our approach in Fig. 2. Specifically, given source and target samples, $x^{\text{src}}_{1:T}, x^{\text{tgt}}_{1:T}$, we swap the static and dynamic features between the source and the target. In practice, swapping the content (static) information corresponds to generating an image with factors $(s^{\text{tgt}}, d^{\text{src}}_{1:T})$, i.e., fix the source dynamics and use the static factor of the target. For instance, a perfect content swap in Sprites yields different characters with the same pose. The opposite swap of pose (dynamic) information is obtained with $(s^{\text{src}}, d^{\text{tgt}}_{1:T})$. Fig. 2 shows two separate examples of Sprites and MUG (rows $1, 2$ and rows $3, 4$), where each example is organized in blocks of four panels. For instance on the Sprites example, panel no. 1 is the source and panel no. 2 is the target. Panel no. 3 shows a content (static) swap, and Panel no. 4 shows a pose (dynamic) swap.

5.4. Quantitative Results: Common Benchmarks

Image data. Similar to previous work (Zhu et al., 2020; Bai et al., 2021), we test our model disentanglement and generative abilities on the Sprites and MUG datasets, and we compare our results with state-of-the-art (SOTA) methods. The evaluation protocol takes a sample $x_{1:T}$ with its static and dynamic factors, s and $d_{1:T}$, and generates a new sample $\tilde{x}_{1:T}$ with the original dynamic features, and a new static component sampled from the prior, $\tilde{s} \sim p(s)$. Ideally, we expect that $x_{1:T}, \tilde{x}_{1:T}$ share the dynamic labels, e.g., happy in MUG, whereas, their static classes match with probability close to random guess. To verify that this is indeed the case, we use a pre-trained classifier to predict the dynamic label of $\tilde{x}_{1:T}$, and we compare it to the true label of $x_{1:T}$.

We utilize these labels on several different error metrics: label accuracy (Acc), inception score (IS) that estimates the generator performance, intra-entropy $H(y|x)$ that shows how confident the classifier is regarding its prediction, and inter-entropy $H(y)$ that measures diversity in generated samples, see also App. A. Our results are provided in Tab. 1, alongside the results of previous SOTA approaches. The arrows $\uparrow, \downarrow$ next to the metrics denote which metric is expected to be higher or lower in value, respectively. Notably, our method outperforms existing work on Sprites and MUG datasets with respect to all metrics. Finally, one may also consider the opposite test where the static factor is fixed and the dynamic features are sampled. However, the SOTA methods achieve near perfect accuracy in this setting, and thus, we do not show these results here.

Audio data. Another common benchmark demonstrates the effectiveness of sequential disentanglement frameworks on a different data modality (Hsu et al., 2017; Yingzhen & Mandt, 2018). Specifically, we consider speaker verification on the TIMIT dataset. The main objective is to distinguish between different speakers, independently of the text they read. For a sample $x_{1:T}$, we expect that its static factor s represents the speaker identity, whereas $d_{1:T}$ should not be related to that information. We use the Equal Error Rate (EER) metric where we compute the cosine similarity between all s instances and independently for $d_{1:T}$ instances. Two static vectors encode the same speaker if their cosine similarity is higher than a threshold $\epsilon \in [0, 1]$, and different speakers otherwise. The threshold ϵ needs to be calibrated to receive the EER (Chenafa et al., 2008). Tab. 1 shows that our approach improves SOTA results by a margin of 0.62% and 1.41% for the static and dynamic EER. For additional info on this benchmark, see App. B.5.

Time series data. Recently, (Tonekaboni et al., 2022) explored their approach on downstream tasks with time series data. Sequential information different from image and audio is an ideal test case for our framework as we lift the dependency on data augmentation (DA) techniques. Indeed, while DA is common for image/audio data, it is less available for arbitrary time series data. We follow the evaluation setup in (Tonekaboni et al., 2022) to study the latent representations learned by our method. Specifically,

Table 1. Disentanglement metrics on Sprites, MUG, and TIMIT. Results with standard deviation appear in B.6.

Method	Sprites Acc↑	Sprites IS↑	Sprites $H(y\|x)$↓	Sprites $H(y)$↑	MUG Acc↑	MUG IS↑	MUG $H(y\|x)$↓	MUG $H(y)$↑
MoCoGAN	92.89%	8.461	0.090	2.192	63.12%	4.332	0.183	1.721
DSVAE	90.73%	8.384	0.072	2.192	54.29%	3.608	0.374	1.657
R-WAE	98.98%	8.516	0.055	**2.197**	71.25%	5.149	0.131	1.771
S3VAE	99.49%	8.637	0.041	**2.197**	70.51%	5.136	0.135	1.760
SKD	**100%**	**8.999**	**1.6e−7**	**2.197**	77.45%	**5.569**	**0.052**	1.769
C-DSVAE	99.99%	8.871	0.014	**2.197**	81.16%	5.341	0.092	1.775
Ours	**100%**	8.942	0.006	**2.197**	**85.71%**	5.548	0.066	**1.779**

Method	TIMIT static EER↓	TIMIT dynamic EER ↑
FHVAE	5.06%	22.77%
DSVAE	5.64%	19.20%
R-WAE	4.73%	23.41%
S3VAE	5.02%	25.51%
SKD	4.46%	26.78%
C-DSVAE	4.03%	31.81%
Ours	**3.41%**	**33.22%**

we used an encoder and a decoder to compute codes of consecutive time series windows, and we extract these codes on non-stationary datasets such as Physionet and air quality.

We consider the following tasks: 1. prediction of the risk of in-hospital mortality, and 2. estimation of the average daily rain level. For each task, we train a simple RNN classifier in which we utilize the latent representations from the above autoencoder. For comparison, Tonekaboni et al. (2022) used C-DSVAE without data augmentation and thus with no contrastive estimation losses. However, as noted in the above paragraph, our approach does not have this limitation, and thus we can utilize the entire model (5). Tab. 2 shows the results on the mortality rate and daily rain downstream tasks. Our method performs on par with GLR on the mortality rate task, and it comes second for daily rain estimation. However, it is important to emphasize that GLR was designed specifically for time series data with statistical properties as in Physionet and Air Quality datasets. In contrast, our method is not tuned to specifically handle time series data, and it can work on multiple data modalities such as video, audio, and time series data. Further, In our experimental setup, we were not able to re-produce the baseline results for the daily rain task. We leave this direction for further exploration. For more details regarding the evaluation setup and tasks, we refer to (Tonekaboni et al., 2022).

5.5. Quantitative Results: New Benchmarks

The standard sequential disentanglement benchmark tests include classification of conditionally generated images and speaker verification. Here, we propose a new benchmark that quantifies the quality of the learned representations. For this evaluation, we consider the MUG dataset, and two challenging video datasets with hand writing (Letters) and hand gestures (Jesters). In this experiment, we explore a common framework to evaluate the disentangled codes. First, we compute the static $\{s^j\}$ and dynamic $\{d^j_{1:T}\}$ codes of the test set. Then, we define train and test sets via an $80-20$ split of the test set, and we train four classifiers. The first classifier takes s vectors as inputs, and it tries to predict the static label. The second classifier takes s and it predicts the dynamic label. Similarly, the third classifier takes $d_{1:T}$ and predicts the static label, and the fourth classifier takes $d_{1:T}$ and predicts the dynamic label. An ideal result with input s is a prefect classification score in the first classifier, and a random guess in the second classifier. Additional details on this experiment appear in B.9. Our results are summarized in Tab. 3, where we outperform C-DSVAE often by a large gap in accuracy. The Jesters dataset does not include static labels, and thus we only have partial results. Further, this dataset is extremely challenging due to low-quality images and complex gestures, and currently, C-DSVAE and our approach obtain low scores, where our approach attains $> 7\%$ improvement over a random guess. These results can be improved by integrating recent VAEs (Razavi et al., 2019; Vahdat & Kautz, 2020), as we observe low quality reconstruction, which may effect disentanglement abilities.

5.6. Analysis of Positive and Negative Views

Sec. 3 details how to incorporate contrastive learning in a sequential disentanglement setting, and in Sec. 4, we list some of the challenges such as sampling wrong positive and negative views. Here, we would like to empirically compare the views generated by C-DSVAE and our approach. For instance, we show a qualitative example in Fig. 3A of views used in C-DSVAE and obtained with SimCLR (Chen et al., 2020a), where positive dynamic examples are generated via e.g., color distortion while supposedly keeping the dynamic features fixed. Unfurtunately, not all DA preserve the facial expressions. Beyond these qualitative examples, we also adapt the analysis (Tian et al., 2020) as detailed below.

Table 2. Error metrics on Physionet and Air quality datasets

Method	ICU Mortality Prediction AUPRC	ICU Mortality Prediction AUROC	Avg. Daily Rain MAE
VAE	0.157 ± 0.053	0.564 ± 0.044	1.831 ± 0.005
GPVAE	0.282 ± 0.086	0.699 ± 0.018	1.826 ± 0.001
C-DSVAE	0.158 ± 0.005	0.565 ± 0.007	$\mathbf{1.806 \pm 0.012}$
GLR	0.365 ± 0.092	0.752 ± 0.011	1.824 ± 0.001
Ours	$\mathbf{0.367 \pm 0.015}$	$\mathbf{0.764 \pm 0.040}$	1.823 ± 0.001

Table 3. Downstream classification task on latent static and dynamic features. Results with standard deviation appear in B.9.

Dataset	Method	Static features			Dynamic features		
		Static L-Acc ↑	Dynamic L-Acc ↓	Gap ↑	Static L-Acc ↓	Dynamic L-Acc ↑	Gap ↑
MUG	random	1.92%	16.66%	-	1.92%	16.66%	-
	C-DSVAE	**98.75%**	76.25%	22.25%	26.25%	82.50%	56.25%
	Ours	98.12%	**68.75%**	**29.37%**	**10.00%**	**85.62%**	**75.62%**
Letters	random	1.65%	3.84%	-	1.65%	3.84%	-
	C-DSVAE	95.47%	13.0%	82.47%	**2.79%**	66.35%	63.56%
	Ours	**100%**	**12.16%**	**87.84%**	3.06%	**69.75%**	**66.69%**
Jesters	random	-	-	-	-	20%	-
	C-DSVAE	-	-	-	-	21.88%	-
	Ours	-	-	-	-	**27.70%**	-

Let $x_{1:T} \sim p_{\mathcal{D}}$ denote a data sample with its static and dynamic factors, $(s, d_{1:T})$. A positive static example $x^+_{1:T}$ is generated using the pair $(s^+, \tilde{d}_{1:T})$ where s^+ is similar to s, and $\tilde{d}_{1:T}$ is different from $d_{1:T}$. In the opposite case of a positive dynamic sample, the dynamic features $d^+_{1:T}$ are similar to $d_{1:T}$ and $\tilde{s}$ is different from s. Following Tian et al. (2020), a good view is such that the mutual information $I_q(s^+; y)$ is high, whereas $I_q(\tilde{d}_{1:T}; y)$ is low, where y is the task label. For example, the identity of the person is kept, while its facial expression has changed. To estimate these MI terms, we use the latent codes in classification tasks as in Sec. 5.5 where the static and dynamic factors are predicted. Namely, we use s^+ to predict the static labels, and similarly, we use $\tilde{d}_{1:T}$ to predict the dynamic labels. Good views will yield high static classification scores and low dynamic classification scores.

We show in Fig. 3B the classification results when using $(s^+, \tilde{d}_{1:T})$ with C-DSVAE and with our method, and Fig. 3C shows the opposite case, i.e., $(\tilde{s}, d^+_{1:T})$. For both plots, blue curves are related to our approach and red curves correspond to C-DSVAE. We focus on the test which uses $(s^+, \tilde{d}_{1:T})$, Fig. 3B. The blue and red curves show the classification accuracy when using s^+ to predict the static label, and thus, they should be high. In contrast, the light blue and orange curves arise from using $\tilde{d}_{1:T}$ to predict the dynamic labels, and thus, they should be close to a random guess for semi-hard views (16.66% in MUG). However, the orange curve is around 70%, whereas the light blue attains $\approx 30\%$. These results indicate that our views are semi-hard as they yield accuracy results close to a random guess. In the opposite scenario, Fig. 3C, we use the pair $(\tilde{s}, d^+_{1:T})$ with different static and similar dynamic factors. Here, the blue and red curves should be close to a random guess (1.92%), and the light blue and orange plots should present high accuracy values. However, the orange curve presents $\approx 25\%$ accuracy, whereas ours is around 70%. We conclude that our dynamic features better preserve the underlying action. Additional analysis and results are provided in App. B.4.

5.7. Ablation Study: Negative Views Sampling

The previous evaluation in Sec. 5.6 focused on the quality of positive views. Here, we explore the effect of utilizing various negative sampling rules. Ultimately, we would like to empirically motivate the heuristic we introduce in Sec. 4 where we propose to only consider 33.3% of the farthest distributions as measured by the KL divergence distance. An inferior heuristic is one that produces confusing negative views, i.e., examples that are semantically similar to the current data, instead of being dissimilar. However, choosing "right" negative views is important to the overall behavior of the approach, and thus we explore other sampling policies in the following ablation study. For a meaningful comparison, we fix the hyperparameters of the approach, and we train several models that only differ in their selection strategy of the negative views. Let n the number of inputs in the batch; we define a pool of size $\lfloor n/3 \rfloor$ of negative distributions taken from: 1) the middle third, 2) the farthest third, and 3) both middle and farthest thirds, and 4) random sampling. From these distributions we sample $2n$ views. We present in Tab. 4 the results of our ablation study on the MUG and TIMIT datasets. For MUG, we show the accuracy score, and we display the EER gap in TIMIT. Notably, all sampling strategies attain SOTA results on these tasks, cf. Tab. 1. However, the farthest third yields the best results consistently across tasks, and thus, these results support our heuristic. In App. B.2, we conduct an analysis that motivates and justify our heuristic by showing the similarity distribution of the thirds.

Table 4. Negatives Ablation Study.

Negatives Mode	Acc MUG↑	EER gap TIMIT↑
Random	84.18%	28.53%
Middle Third	84.43%	29.11%
Middle+Farthest	84.96%	29.53%
Farthest Third	**85.71%**	**29.81%**

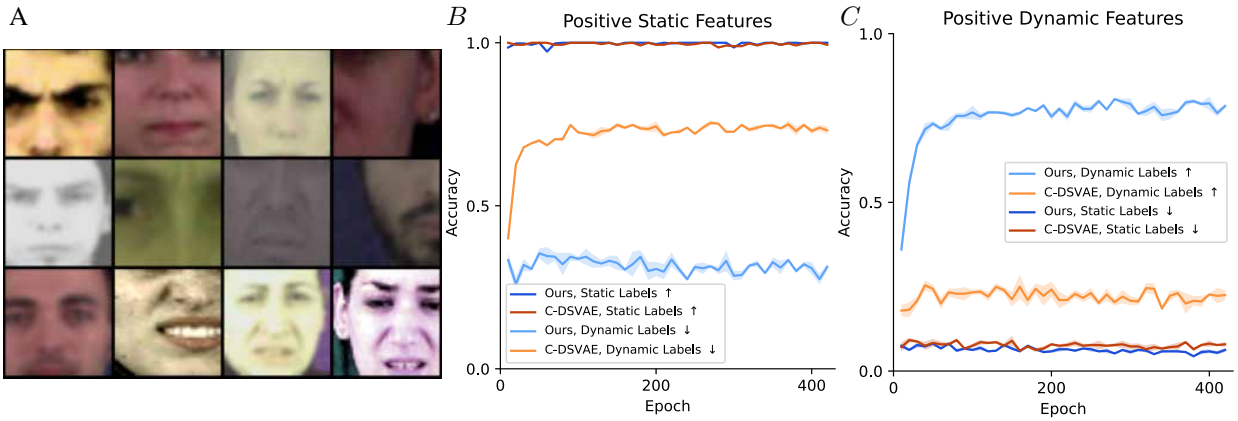

Figure 3. We present randomly selected views of SimCLR on MUG (A). In addition, we compare the quality of views obtained with C-DSVAE and our approach when classifying the static and dynamic labels (B, C). See the text for more details.

6. Limitations

Our model achieves SOTA results on several sequential disentanglement benchmarks. While the method relies on heuristics such as initial disentanglement by restricting the dimensions of s and $d_{1:T}$, and the methodology of selecting negative and positive samples, it is backed up with extensive empirical results that show the significance of each component and the robustness of the method to different modalities. Our model uses a similar number of hyperparameters as existing work. Tuning several hyperparameters may be challenging in practice. Nevertheless, we utilized automatic tuning tools, such as hyperopt, to search for the best parameter values within the predefined hyperparameter space. Finally, similar to existing disentanglement works, we used pre-trained classifiers to evaluate our approach. In general, we believe that the sequential disentanglement community will benefit from new challenging benchmarks that depend on improved evaluation metrics.

7. Discussion

In this work, we investigated the problem of unsupervised disentanglement of sequential data. Recent SOTA methods employ self-supervision via auxiliary tasks and DA which unfortunately, are modality-based. Namely, they depend on the domain-modality (e.g., videos), on the task-modality (e.g., classifying expressions), or on both. In contrast, we propose a contrastive estimation framework that is free of external signals, and thus is applicable to arbitrary sequential data and tasks. Key to our approach is the observation that VAEs naturally support the generation, comparison, and sampling of distributions. Therefore, effective sampling strategies for generating positive and negative views can be devised based solely on the batch inputs. Our method is easy to code, efficient, and it uniformly treats similar and dissimilar views. Our extensive evaluation shows new SOTA results on multiple datasets including video, audio and arbitrary time series and on downstream tasks as speaker verification, unconditional generation, and prediction.

In the future, we would like to explore the interplay between the mutual information loss components. Essentially, these terms are contradicting in nature, and thus, it motivates us to find improved formulations. Moreover, we would like to investigate whether sampling strategies as our method can be effective for non-sequential contrastive estimation on e.g., static images. We believe that this is a very interesting direction for future research and that with some adaptions, our method can contribute to contrastive learning of static information as well. Finally, we aim to tackle challenging datasets as Jesters using improved VAE pipelines.

Acknowledgements

This research was partially supported by the Lynn and William Frankel Center of the Computer Science Department, Ben-Gurion University of the Negev, an ISF grant 668/21, an ISF equipment grant, and by the Israeli Council for Higher Education (CHE) via the Data Science Research Center, Ben-Gurion University of the Negev, Israel.

References

Aifanti, N., Papachristou, C., and Delopoulos, A. The MUG facial expression database. In *11th International Workshop on Image Analysis for Multimedia Interactive Services WIAMIS 10*, pp. 1–4, 2010.

Ash, J. T., Goel, S., Krishnamurthy, A., and Misra, D. Investigating the role of negatives in contrastive representation learning. *arXiv preprint arXiv:2106.09943*, 2021.

Bachman, P., Hjelm, R. D., and Buchwalter, W. Learning representations by maximizing mutual information across views. *Advances in neural information processing systems*, 32, 2019.

Bai, J., Wang, W., and Gomes, C. P. Contrastively disentangled sequential variational autoencoder. *Advances in Neural Information Processing Systems*, 34, 2021.

Berman, N., Naiman, I., and Azencot, O. Multifactor sequential disentanglement via structured Koopman autoencoders. In *International Conference on Learning Representations, ICLR*, 2023.

Bowman, S. R., Vilnis, L., Vinyals, O., Dai, A. M., Józefowicz, R., and Bengio, S. Generating sentences from a continuous space. In *Conference on Computational Natural Language Learning*, pp. 10–21. ACL, 2016.

Bromley, J., Guyon, I., LeCun, Y., Säckinger, E., and Shah, R. Signature verification using a "Siamese" time delay neural network. *Advances in neural information processing systems*, 6, 1993.

Caron, M., Misra, I., Mairal, J., Goyal, P., Bojanowski, P., and Joulin, A. Unsupervised learning of visual features by contrasting cluster assignments. *Advances in Neural Information Processing Systems*, 33:9912–9924, 2020.

Chen, R. T., Li, X., Grosse, R. B., and Duvenaud, D. K. Isolating sources of disentanglement in variational autoencoders. *Advances in neural information processing systems*, 31, 2018.

Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pp. 1597–1607. PMLR, 2020a.

Chen, X., Fan, H., Girshick, R., and He, K. Improved baselines with momentum contrastive learning. *arXiv preprint arXiv:2003.04297*, 2020b.

Chenafa, M., Istrate, D., Vrabie, V., and Herbin, M. Biometric system based on voice recognition using multiclassifiers. In *European Workshop on Biometrics and Identity Management*, pp. 206–215. Springer, 2008.

Chopra, S., Hadsell, R., and LeCun, Y. Learning a similarity metric discriminatively, with application to face verification. In *Conference on Computer Vision and Pattern Recognition CVPR*, volume 1, pp. 539–546. IEEE, 2005.

Chuang, C.-Y., Robinson, J., Lin, Y.-C., Torralba, A., and Jegelka, S. Debiased contrastive learning. *Advances in neural information processing systems*, 33:8765–8775, 2020.

Doersch, C. and Zisserman, A. Multi-task self-supervised visual learning. In *international conference on computer vision*, pp. 2051–2060, 2017.

Garofolo, J., Lamel, L., Fisher, W., Fiscus, J., Pallett, D., Dahlgren, N., and Zue, V. TIMIT acoustic-phonetic continuous speech corpus. *Linguistic Data Consortium*, 11 1992.

Ge, S., Mishra, S., Li, C.-L., Wang, H., and Jacobs, D. Robust contrastive learning using negative samples with diminished semantics. *Advances in Neural Information Processing Systems*, 34:27356–27368, 2021.

Goldberger, A. L., Amaral, L. A., Glass, L., Hausdorff, J. M., Ivanov, P. C., Mark, R. G., Mietus, J. E., Moody, G. B., Peng, C.-K., and Stanley, H. E. Physiobank, physiotoolkit, and physionet: components of a new research resource for complex physiologic signals. *circulation*, 101(23): e215–e220, 2000.

Grill, J.-B., Strub, F., Altché, F., Tallec, C., Richemond, P., Buchatskaya, E., Doersch, C., Avila Pires, B., Guo, Z., Gheshlaghi Azar, M., et al. Bootstrap your own latent-a new approach to self-supervised learning. *Advances in neural information processing systems*, 33:21271–21284, 2020.

Gutmann, M. and Hyvärinen, A. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 297–304, 2010.

Han, J., Min, M. R., Han, L., Li, L. E., and Zhang, X. Disentangled Recurrent Wasserstein Autoencoder. *International Conference on Learning Representations ICLR*, 2021.

Han, T., Xie, W., and Zisserman, A. Video representation learning by dense predictive coding. In *International Conference on Computer Vision Workshops*, pp. 0–0, 2019.

He, K., Fan, H., Wu, Y., Xie, S., and Girshick, R. Momentum contrast for unsupervised visual representation learning. In *conference on computer vision and pattern recognition CVPR*, pp. 9729–9738, 2020.

Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., Mohamed, S., and Lerchner, A. beta-VAE: Learning basic visual concepts with a constrained variational framework. In *International Conference on Learning Representations ICLR*, 2016.

Ho, C.-H. and Vasconcelos, N. Contrastive learning with adversarial examples. *Advances in Neural Information Processing Systems*, 33:17081–17093, 2020.

Hsu, W.-N., Zhang, Y., and Glass, J. Unsupervised learning of disentangled and interpretable representations from sequential data. *Advances in neural information processing systems*, 30, 2017.

Huynh, T., Kornblith, S., Walter, M. R., Maire, M., and Khademi, M. Boosting contrastive self-supervised learning with false negative cancellation. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pp. 2785–2795, 2022.

Ibrahim, A., Elijah, O., Olusayo, F., Omodolapo, B., and Opeyemi, D. Isgl online and offline character recognition dataset. https://data.mendeley.com/datasets/n7kmd7t7yx/1, 2019.

Kalantidis, Y., Sariyildiz, M. B., Pion, N., Weinzaepfel, P., and Larlus, D. Hard negative mixing for contrastive learning. *Advances in Neural Information Processing Systems*, 33:21798–21809, 2020.

Kim, H. and Mnih, A. Disentangling by factorising. In *International Conference on Machine Learning*, pp. 2649–2658. PMLR, 2018.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2014.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. In *International Conference on Learning Representations, ICLR*, 2014.

Kulkarni, T. D., Whitney, W. F., Kohli, P., and Tenenbaum, J. Deep convolutional inverse graphics network. *Advances in neural information processing systems*, 28, 2015.

Le-Khac, P. H., Healy, G., and Smeaton, A. F. Contrastive representation learning: A framework and review. *IEEE Access*, 8:193907–193934, 2020.

Li, H., Wang, X., Zhang, Z., Yuan, Z., Li, H., and Zhu, W. Disentangled contrastive learning on graphs. *Advances in Neural Information Processing Systems*, 34:21872–21884, 2021.

Lin, Z., Thekumparampil, K., Fanti, G., and Oh, S. Infogan-cr and modelcentrality: Self-supervised model training and selection for disentangling GANs. In *international conference on machine learning*, pp. 6127–6139. PMLR, 2020.

Locatello, F., Bauer, S., Lucic, M., Raetsch, G., Gelly, S., Schölkopf, B., and Bachem, O. Challenging common assumptions in the unsupervised learning of disentangled representations. In *international conference on machine learning*, pp. 4114–4124. PMLR, 2019.

Materzynska, J., Berger, G., Bax, I., and Memisevic, R. The jester dataset: A large-scale video dataset of human gestures. In *International Conference on Computer Vision Workshops*, pp. 0–0, 2019.

Misra, I. and Maaten, L. v. d. Self-supervised learning of pretext-invariant representations. In *Conference on Computer Vision and Pattern Recognition CVPR*, pp. 6707–6717, 2020.

Oord, A. v. d., Li, Y., and Vinyals, O. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. PyTorch: An imperative style, high-performance deep learning library. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 32*, pp. 8024–8035. Curran Associates, Inc., 2019.

Razavi, A., Van den Oord, A., and Vinyals, O. Generating diverse high-fidelity images with VQ-VAE-2. *Advances in neural information processing systems*, 32, 2019.

Reed, S. E., Zhang, Y., Zhang, Y., and Lee, H. Deep visual analogy-making. *Advances in neural information processing systems*, 28, 2015.

Robinson, J., Chuang, C.-Y., Sra, S., and Jegelka, S. Contrastive learning with hard negative samples. *Conference on Learning Representations ICLR*, 2020.

Sermanet, P., Lynch, C., Chebotar, Y., Hsu, J., Jang, E., Schaal, S., Levine, S., and Brain, G. Time-contrastive networks: Self-supervised learning from video. In *IEEE international conference on robotics and automation (ICRA)*, pp. 1134–1141. IEEE, 2018.

Sohl-Dickstein, J., Weiss, E., Maheswaranathan, N., and Ganguli, S. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*, pp. 2256–2265. PMLR, 2015.

Tian, Y., Sun, C., Poole, B., Krishnan, D., Schmid, C., and Isola, P. What makes for good views for contrastive learning? *Advances in Neural Information Processing Systems*, 33:6827–6839, 2020.

Tonekaboni, S., Li, C.-L., Arik, S. O., Goldenberg, A., and Pfister, T. Decoupling local and global representations of time series. In *International Conference on Artificial Intelligence and Statistics*, pp. 8700–8714. PMLR, 2022.

Tsai, Y.-H. H., Ma, M. Q., Yang, M., Zhao, H., Morency, L.-P., and Salakhutdinov, R. Self-supervised representation learning with relative predictive coding. *arXiv preprint arXiv:2103.11275*, 2021.

Tulyakov, S., Liu, M.-Y., Yang, X., and Kautz, J. MoCoGAN: Decomposing motion and content for video generation. In *conference on computer vision and pattern recognition CVPR*, pp. 1526–1535, 2018.

Vahdat, A. and Kautz, J. NVAE: A deep hierarchical variational autoencoder. *Advances in Neural Information Processing Systems*, 33:19667–19679, 2020.

Van der Maaten, L. and Hinton, G. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

Wang, T., Yue, Z., Huang, J., Sun, Q., and Zhang, H. Self-supervised learning disentangled group representation as feature. *Advances in Neural Information Processing Systems*, 34:18225–18240, 2021.

Wei, P., Kong, L., Qu, X., Yin, X., Xu, Z., Jiang, J., and Ma, Z. Unsupervised video domain adaptation: A disentanglement perspective. *arXiv preprint arXiv:2208.07365*, 2022.

Wu, Z., Xiong, Y., Yu, S. X., and Lin, D. Unsupervised feature learning via non-parametric instance discrimination. In *conference on computer vision and pattern recognition CVPR*, pp. 3733–3742, 2018.

Yingzhen, L. and Mandt, S. Disentangled sequential autoencoder. In *International Conference on Machine Learning*, pp. 5670–5679. PMLR, 2018.

Zhang, J. and Ma, K. Rethinking the augmentation module in contrastive learning: Learning hierarchical augmentation invariance with expanded views. In *Conference on Computer Vision and Pattern Recognition CVPR*, pp. 16650–16659, 2022.

Zhang, S., Guo, B., Dong, A., He, J., Xu, Z., and Chen, S. X. Cautionary tales on air-quality improvement in beijing. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 473(2205):20170457, 2017.

Zhu, R., Zhao, B., Liu, J., Sun, Z., and Chen, C. W. Improving contrastive learning by visualizing feature transformation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 10306–10315, 2021.

Zhu, Y., Min, M. R., Kadav, A., and Graf, H. P. S3VAE: Self-supervised sequential VAE for representation disentanglement and data generation. In *Conference on Computer Vision and Pattern Recognition CVPR*, pp. 6538–6547, 2020.

A. Experimental Setup

A.1. Datasets

Sprites. A dataset introduced by (Reed et al., 2015) that includes animated cartoon characters that have both static and dynamic attributes. The static attributes include variations in skin, tops, pants, and hair color, each of which has six possible options. The dynamic attributes consist of three different types of motion (walking, casting spells, and slashing) that can be performed in three different orientations (left, right, and forward). In total, there are 1296 unique characters that can perform nine different motions. Each sequence in the dataset consists of eight RGB images with a size of 64×64 pixels. In our experiments, we use 9000 samples for training and 2664 samples for testing.

MUG. A Facial expression dataset created by (Aifanti et al., 2010) that includes image sequences of 52 subjects displaying six different facial expressions (anger, fear, disgust, happiness, sadness, and surprise). Each video in the dataset consists of between 50 and 160 frames. In order to create sequences of length 15, as was done in previous work (Bai et al., 2021), we randomly select 15 frames from the original sequences. We then use Haar Cascades face detection to crop the faces and resize them to 64×64 pixels, resulting in sequences $x \in \mathbb{R}^{15\times3\times64\times64}$. The final dataset consists of 3429 samples.

TIMIT. A dataset introduced by (Garofolo et al., 1992) which consists of read speech that is used for acoustic-phonetic research and other speech tasks. It contains 6300 utterances (5.4 hours of audio). There are 10 sentences per speaker, for a total of 630 speakers. The dataset includes adult men and women. For the data pre-processing, we follow the same procedure as in prior work (Yingzhen & Mandt, 2018). We extract spectrogram features (10ms frame shift) from the audio, and we sample segments of 200ms duration (20 frames) from the audio, which are used as independent samples.

Jester. A dataset introduced by (Materzynska et al., 2019). The Jester dataset comprises of 148.092 labeled video segments of more then 1300 unique individuals making 27 simple, predefined hand gestures in front of a laptop camera or webcam. The gestures are labeled whereas the subject is not, and thus the dataset contains only dynamic labels. This dataset is significantly more complex in comparison to MUG since there are variations in the background, light, and pose, and more elements in the image are much bigger. We used five gestures (Pushing Hand Away, Rolling Hand Forward, Shaking Hand, Sliding Two Fingers Left, Sliding Two Fingers Right). We extracted videos with 10 frames where the gap between two frames has been selected by taking the total sequence length divided by 10.

Letters. The Letters dataset (Ibrahim et al., 2019) comprises of English letters and numbers written by 66 individuals, both offline and online handwritten letters. In our setup, we utilized only small English letters (a-z) from the offline subset of the dataset. We created a lexicon of 100 words, each consisting of seven letters, and then we generated word sequences using images of the letters. As an example, a sequence may appear as "science". We excluded subject number '61' due to missing data and generated 100 word sequences using the handwriting of the remaining 65 subjects. Each subject has two samples for each letter, which were randomly selected.

Physionet. The Physionet ICU Dataset (Goldberger et al., 2000) is a medical time series corpus of 12.000 adult patients' stays in the Intensive Care Unit (ICU). The data includes time-dependent measurements such as physiological signals and lab measurements as well as general information about the patients, such as their age, the reason for their ICU admission, and etc. Additionally, the dataset includes labels that indicate in-hospital mortality. For pre-processing we follow (Tonekaboni et al., 2022).

Air Quality. The UCI Beijing Multi-site Air Quality dataset (Zhang et al., 2017) was collected over four years from March 1st, 2013 to February 28th, 2017. It includes hourly measurements of multiple air pollutants from 12 nationally controlled monitoring sites. The meteorological data in each air-quality site are matched with the nearest weather station from the China Meteorological Administration. For our experiments we follow (Tonekaboni et al., 2022) and pre-process the data by dividing it into samples from different stations and of different months of the year.

A.2. Disentanglement Metrics

Accuracy (Acc). This is a metric that measures the ability of a model to preserve fixed features while generating others. For instance, freeze the dynamic features and sample the static features. The metric computed by using a pre-trained classifier (called C or the "judge"). The classifier training has been on the same train set of the model and testing is on the

same test set as of the model. For example, for the MUG dataset, the classifier would output the facial expression and check that it did not change during the static features sampling.

Inception Score (IS). This is a metric for the generator performance. First, we apply the judge on all the generated sequences $x_{1:T}$. Thus, getting $p(y \mid x_{1:T})$ which is the conditional predicted label distribution. Second, we take $p(y)$ which is the marginal predicted label distribution and we calculate the KL-divergence $\mathrm{KL}[p(y \mid x_{1:T}) \parallel p(y)]$. Finally, we compute $\mathrm{IS} = \exp(\mathbb{E}_x \mathrm{KL}[p(y \mid x_{1:T}) \parallel p(y)])$.

Inter-Entropy ($H(y|x)$). This metric reflects the confidence of the classifier C regarding label prediction. Low Intra Entropy means high confidence. We measure it by entering k generated sequences into the classifier and computing $\frac{1}{k}\sum_{i=1}^{k} H(p(y|x^i_{1:T}))$.

Intra-Entroy ($H(y)$). This metric reflects the diversity among the generate sequence. High Intra-Entropy score means high diversity. It is computed by taking the generated sample from the learned prior distribution $p(y)$ and then using the judge output on the predicted labels y.

Equal Error Rate (EER). This metric is used in the TIMIT dataset for speaker verification task evaluation. It measures the value of the false negative rate, or equally, the value of the false positive rate of a model over the speaker verification task. EER is measured when the above rates are equal.

Latent Accuracy (L-Acc). This metric measures the ability of a model to generate meaningful latent features for a downstream classification task. For instance, for the MUG dataset, taking the static latent factor s of a sample x and trying to predict the subject label or the facial expression label. In such case, a meaningful and disentangled model will produce static features that will contain information about the subject label but not on the facial expression label of x. We compute the prediction accuracy by training a Support Vector Machine Classifier for the static and dynamic features. We flatten the dynamic features $d_{1:T}$ into one vector d, that is, assuming $d_i \in \mathbb{R}^k$ and $i = 1, ..., T$. Then the dimension of the flattened vector d is $k \times T$. The exact dimension changes between dataset types. We split the test set data into two parts (80-20) and use its first part to train the different classifiers and the second one to evaluate the prediction accuracy. Finally, we also train a Random Forest Classifier and KNN Classifier to show the robustness of the benchmark to the classifier choice.

Table 5. Image Model Architecture.

Encoder
Input: $64 \times 64 \times 3$ image
Conv2D$(3, 32, 4, 2, 1)$
Conv2D$(32, 64, 4, 2, 1)$
Conv2D$(64, 128, 4, 2, 1)$
Conv2D$(128, 256, 4, 2, 1)$
Conv2D$(256, 128, 4, 2, 1)$
Where each Conv2D layer is followed by BN2D and LeakyReLU activation
Bidi-LSTM$(128, 256)$
$s^{\mu} = \text{Linear}(512, d_s)$, $s^{\log(\sigma)} = \text{Linear}(512, d_s)$
RNN$(512, 256)$
$d^{\mu}_{1:T} = \text{Linear}(256, d_d)$, $d^{\log(\sigma)}_{1:T} = \text{Linear}(256, d_d)$
Decoder
Reparameterize to obtain s and $d_{1:T}$
concat $(s, d_{1:T}) = z$ with d_z dimension size
Conv2DT$(d_z, 256, 4, 1, 0)$
Conv2DT$(256, 128, 4, 1, 0)$
Conv2DT$(128, 64, 4, 1, 0)$
Conv2DT$(64, 32, 4, 1, 0)$
Where each Conv2DT layer is followed by BN2D and LeakyReLU activation
Output: Sigmoid(Conv2DT$(32, 3, 4, 1, 0)$)

A.3. Model Architecture

All the models have been implemented using Pytorch (Paszke et al., 2019). The Conv2D and Conv2DT denote a 2D convolution layer and its transpose, and BN2D is a 2D batch normalization layer.

Image Datasets. Our image model architecture follows (Zhu et al., 2020) implementation. The static latent distribution variables s^{μ}, $s^{\log(\sigma)}$ are parameterized by taking the last hidden state of a bi-directional LSTM and propagating it through linear layers. The dynamic latent distribution variables $d^{\mu}_{1:T}$, $d^{\log(\sigma)}_{1:T}$ are given by propagating the hidden states of the bi-directional LSTM through a RNN and then Linear layers. In Tab. 5 we describe the encoder and the decoder of our model. Sprites, MUG, Letters, and Jesters, share the same architecture. We denote the dimension of $d_{1:T}$ by d_d, and s dimension as d_s. The values are chosen per dataset and reported in Tab. 6.

Audio Datasets. The architecture of the TIMIT dataset model follows (Yingzhen & Mandt, 2018) and was used by the previous methods (Zhu et al., 2020; Bai et al., 2021). The only difference from the image architecture is the removal of the convolutions from the encoder and the replacement of the decoder with two linear layers. The first linear layer input dimension is d_z and its output dimension is 256 followed by LeakyReLU activation. Finally, we feed the second linear layer followed by LeakyReLU activation and its output dimension is 200.

Time Series Datasets. The architecture of the time series dataset is simpler. The encoder is composed of 3 linear layers, $\text{Linear}(10, 32) \rightarrow \text{Linear}(32, 64) \rightarrow \text{Linear}(64, 32)$ with ReLU activations after each linear layer followed by similar architecture from image models (Bidi-LSTM etc.) to model s^{μ}, $s^{\log(\sigma)}$, $d^{\mu}_{1:T}$, $d^{\log(\sigma)}_{1:T}$. The decoder is composed of a Linear layer that projects the latent codes onto a dimension of size 32, followed by tanh activation. Then, the output is propagated through an LSTM with a hidden size of 32. We feed the output of the LSTM to 2 linear layers, each followed by a ReLU activation, $\text{Linear}(32, 64)$ and $\text{Linear}(64, 32)$. Finally, we project the output onto 2 linear layers to produce the mean and covariance from which we sample the final output. This architecture follows (Tonekaboni et al., 2022).

A.4. Hyperparameters

We estimate the following objective function:

$$\begin{aligned}\max_{p,q}\ & \mathbb{E}_{x_{1:T}\sim p_{\mathcal{D}}} \lambda_1 \mathbb{E}_{q(z\,|\,x_{1:T})} \log p(x_{1:T}\,|\,z) \\ & - \lambda_2 \text{KL}[q(s\,|\,x_{1:T})\,\|\,p(s)] - \lambda_3 \text{KL}[q(d_{1:T}\,|\,x_{1:T})\,\|\,p(d_{1:T})] \\ & + \lambda_4 I_q(d_{1:T}; x_{1:T}) + \lambda_4 I_q(s; x_{1:T}) - \lambda_5 I_q(s; d_{1:T})\end{aligned} \tag{9}$$

To control the contribution of each loss component we add λ_1 coefficient to the reconstruction loss, λ_2 for static KL term, λ_3 for the dynamic KL term, and finally λ_4, λ_5 to the contrastive terms. The hyperparameter λ_1 is tuned over $\{1, 2.5, 5, 10\}$, we do not divide the MSE loss by the batch size. λ_2 is tuned over $\{1, 3, 5, 7, 9\}$, and λ_4 and λ_5 are tuned over $\{0.1, 0.5, 1, 2.5, 5\}$ while λ_3 is fixed to 1. We used Adam optimizer (Kingma & Ba, 2014) with the learning rate chosen from $\{0.001, 0.0015, 0.002\}$. The dimensions of the static and dynamic features which were chosen among $\{128, 256\}$ for the static and $\{32, 64\}$ for the dynamic factors. Our optimal hyperparameters for each task and dataset are given in Table. 6. All the tasks were trained for at most 600 epochs.

Table 6. Hyperparameters for all datasets, lr and bsz are abbreviations for learning rate and batch size, respectively.

Dataset	λ_1	λ_2	λ_3	λ_4	λ_5	lr	bsz	d_s	d_d
Sprites	10	5	1	5	1	2e−3	100	256	32
MUG	5	9	1	0.5	2.5	15e−4	16	256	64
Letters	2.5	1	1	5	5	2e−3	64	256	32
Jesters	5	1	1	1	1	1e−3	16	256	64
TIMIT	5	1	1	0.5	1	1e−3	10	256	64
Physionet	2.5	7	1	0.1	2.5	1e−3	10	12	4
Air Quality	2.5	5	1	0.1	2.5	1e−3	10	12	4

Algorithm 1 Static predictive sampling trick

Input: Batch of samples $\{x^j_{1:T}\}^n_{j=1} \sim p_{\mathcal{D}}$

$\mathcal{L}_{\text{iNCE}} \leftarrow 0$
$\{s^j\}^n_{j=1} \sim \tilde{q}(\{s^j\}^n_{j=1} \mid \{x^j_{1:T}\}^n_{j=1})$
for $i = 1$ **to** n **do**
  for $j = 1$ **to** n **do**
    $D_{ij} \leftarrow \text{KL}[\tilde{q}(s^i \mid x^i_{1:T}) \parallel \tilde{q}(s^j \mid x^j_{1:T})]$
  end for
  $\chi_i \leftarrow \text{argsort}(D_i)$
end for
for $i = 1$ **to** n **do**
  $\omega \leftarrow \chi_i[: \lfloor \frac{n}{3} \rfloor], \rho \leftarrow \chi_i[\lfloor \frac{2n}{3} \rfloor :]$
  $S^+ \leftarrow \{\tilde{q}(s^\omega \mid x^\omega_{1:T})\}, S^- \leftarrow \{\tilde{q}(s^\rho \mid x^\rho_{1:T})\}$

  /* Sample Positive Example */
  $\tilde{d}_{1:T} \sim p(d_{1:T}), \tilde{s}^+ \sim S^+$
  $x^+_{1:T} \sim p(x^+_{1:T} \mid \tilde{s}^+, \tilde{d}_{1:T})$
  $s^+ \sim q(s^+ \mid x^+_{1:T})$

  /* Sample Negative Examples */
  $\{\tilde{s}^-\}^{2n}_{j=1} \sim S^-$
  $\{x^-_{1:T}\}^{2n}_{j=1} \sim p(\{x^-_{1:T}\}^{2n}_{j=1} \mid \{\tilde{s}^-\}^{2n}_{j=1}, \{d_{1:T}\}^{2n}_{j=1})$
  $\{s^-\}^{2n}_{j=1} \sim q(\{s^-\}^{2n}_{j=1} \mid \{x^-_{1:T}\}^{2n}_{j=1})$

  $\mathcal{L}_{\text{iNCE}} \leftarrow \mathcal{L}_{\text{iNCE}} + \log \frac{\phi(s,s^+)}{\phi(s,s^+)+\sum^{2n}_{j=1}\phi(s,s^{-,j})}$
end for
return $\mathcal{L}_{\text{iNCE}}/n$

B. More Experiments, Analyses and Information

B.1. Method Implementation and Pseudocode

In what follows, we will explain in detail the implementation of our sampling procedure. In addition, we add a pseudocode in Alg. 1 which describes the process and shows how our framework can be implemented.

1. **Producing static (s) and dynamic ($d_{1:T}$) distributions:** Let $x_{1:T} \sim p_{\mathcal{D}}$. Using the model architecture, elaborated in the previous section, we can compute the mean and log variance vectors that represent the s and $d_{1:T}$ posterior distributions: s^μ, $s^{\log(\sigma)}$, $d^\mu_{1:T}$, $d^{\log(\sigma)}_{1:T}$

2. **Producing Positive and Negative Views:** W.l.o.g we focus on how to produce positive and negative static views. However, it is a similar process for the positive and negative dynamic views. To produce the positive view, we need a vector s^+ that represents a positive view of the static factor (potentially from the same class) and vectors $\tilde{d}_{1:T}$ that represent arbitrary dynamics.

   The vectors $\tilde{d}_{1:T}$ can be simply sampled from the prior distribution $\tilde{d}_{1:T} \sim p(d_{1:T})$. To produce s^+, we compute the pairwise KL divergence distances matrix $D \in \mathbb{R}^{n\times n}$ for the batch as described in the main text in Eq 7. We then sort the row $D_{i:}$ in ascending order, and we sample positive views from the *first* third of distributions denoted by S^+, whereas negative views are sampled from the *last* third denoted by S^-. To increase the variability in the views, our predictive sampling trick generates these examples from *the posterior of the sampled prior*. To achieve this, we sample $\tilde{s}^+ \sim S^+$. Then, the positive instance $x^+_{1:T}$ is defined via $x^+_{1:T} \sim p(x^+_{1:T} \mid \tilde{s}^+, \tilde{d}_{1:T})$. Finally, the positive static view is obtained by sampling the posterior: $s^+ \sim q(s^+ \mid x^+_{1:T})$.

   A similar proccess is performed to obtain negative views. We first sample $2n$ examples from $\{\tilde{s}^-\}^{2n}_{j=1} \sim S^-$ where

Table 7. Disentanglement metrics on Sprites and MUG using only a reparametrization trick (repar. trick) vs. using our predictive sampling trick (ours). Our results are better overall across all metrics.

	Sprites				MUG			
Method	Acc↑	IS↑	$H(y\|x)$↓	$H(y)$↑	Acc↑	IS↑	$H(y\|x)$↓	$H(y)$↑
repar. trick	$100\% \pm 0\%$	$8.865 \pm 9.98\text{e}{-4}$	$0.015 \pm 1.13\text{e}{-4}$	2.197 ± 0	$83.93\% \pm 0.96$	5.495 ± 0.048	$0.092 \pm 7.9\text{e}{-3}$	$1.775 \pm 4.2\text{e}{-3}$
Ours	$100\% \pm 0\%$	$\mathbf{8.942} \pm 3.3\text{e}{-5}$	$\mathbf{0.006} \pm 4\text{e}{-6}$	$\mathbf{2.197 \pm 0}$	$\mathbf{85.71\%} \pm 0.9$	$\mathbf{5.548} \pm 0.039$	$\mathbf{0.066} \pm 4\text{e}{-3}$	$\mathbf{1.779} \pm 6\text{e}{-3}$

S^- obtained from D. Next, the negative instances $\{x^-_{1:T}\}_{j=1}^{2n}$ are defined via

$$\{x^-_{1:T}\}_{j=1}^{2n} \sim p(\{x^-_{1:T}\}_{j=1}^{2n} \mid \{\tilde{s}^-\}_{j=1}^{2n}, \{d_{1:T}\}_{j=1}^{2n}) \ .$$

Finally, the negative static views are obtained by sampling the posterior

$$\{s^-\}_{j=1}^{2n} \sim q(\{s^-\}_{j=1}^{2n} \mid \{x^-_{1:T}\}_{j=1}^{2n}) \ .$$

The computational complexity of the D_{KL} matrix is an important aspect of this stage. In practice, we never construct this matrix. Instead, we exploit the parallel capabilities of PyTorch. Notice that this computation is parallel on the level of the cell, as each matrix cell is independent of the others. Thus, PyTorch can utilize its full parallelization capabilities on this task. In particular, since the computation per cell is constant in time (and memory), the entire computation of D_{KL} can be made in constant time, if every cell is calculated by a separate compute node. Therefore, the first for loop in Alg. 1 utilizes the full parallel compute capabilities of PyTorch. Similarly, the second for loop in Alg. 1 is re-phrased in a tensorial form such that the for loop is avoided completely.

3. **Calculating the Contrastive Loss:** We can compute the contrastive loss ($\mathcal{L}_{\text{iNCE}}$) given the positive s^+ and the negatives $\{s^-\}_{j=1}^{2n}$ samples.

B.2. A Thirds Similarity Distributions Experiment

Throughout our studies, we observed that taking the negatives from the last third, obtained the best results as seen in the ablation study in Sec. 5.7. One possible explanation is that the last third consists of fewer positive points, i.e., samples we consider to be negative but are in fact positive. Since our negative sampling process is random, it might be that using negative samples from the last third avoids positive samples more often than taking such samples from the middle third, which yields better overall results in practice. To strengthen this hypothesis, we calculated the distribution of similarity $\phi(u, v^j)$ for each third. We used our trained model to get the latent space vectors and calculated their similarity scores. We conduct the experiment both for the static and dynamic latent vectors. The results are very similar, thus we decided to show here the dynamic vector similarity distribution. We show the similarity histogram of the various thirds in Fig. 4. The histogram shows an intuitive ordering of the thirds in the sense that the first third yields the most similar samples, and the last third yields the most dissimilar samples. Thus, we believe that the governing factor which made the last third to be better is that wrong samples are probably sampled less often in comparison to the middle third. In addition, notice that the last third does contain several examples with similarity $\phi(u, v^j)$ higher than 0.5, and these samples may be hard negative examples.

B.3. The Predictive Sampling Trick vs a Reparametrization Trick

To further analyze the contribution of the predictive sampling trick, we re-trained two neural networks with the same hyperparameters on Sprites and MUG without the predictive sampling trick, and instead, we used a simple reparametrization trick. We report the results of the comparison between the two in Tab. 7. One can observe that the reparametrization trick models' results are inferior in comparison to the results of the predictive sampling trick results. For instance, while the accuracy metric for Sprites is saturated, a reduction in the other metrics is noticeable, e.g. 8.942, using our method vs. 8.865 using the reparametrization trick on the IS metric. The difference is even more noticeable on the MUG dataset, where the reparametrization trick suffers an almost two percent loss in accuracy. These results further motivate and reinforce our choice and design of the predictive sampling trick.

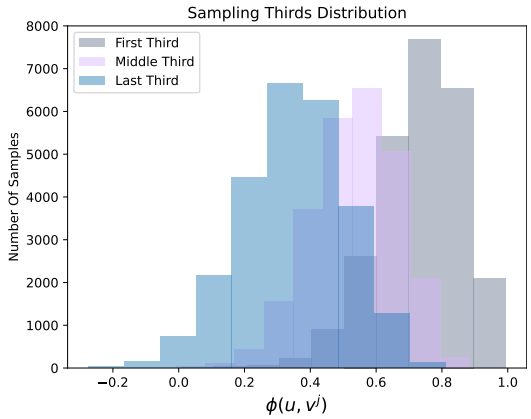

Figure 4. We compute the similarity scores $\phi(u, v^j)$ of the current sample u with respect to samples positioned in the first, middle, and last thirds. The scores are ordered sequentially, i.e., the first third attains the most similar samples, whereas the last third includes the most dissimilar examples.

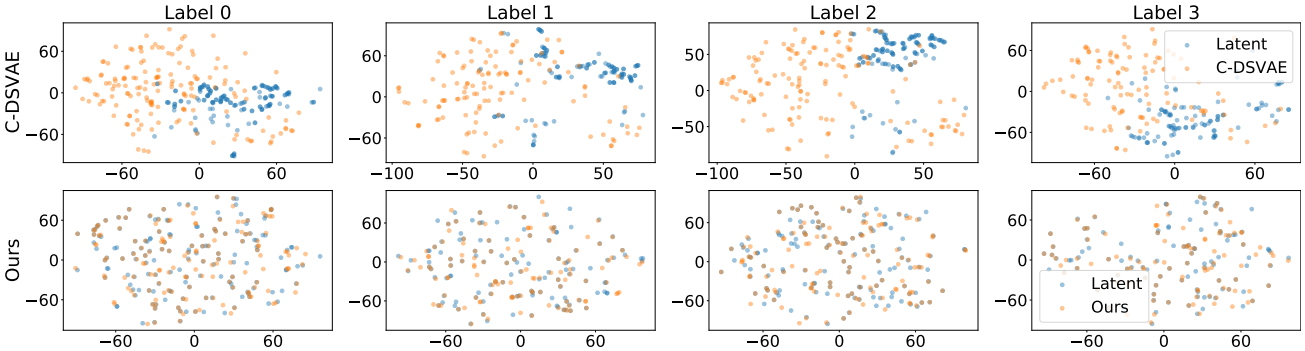

Figure 5. We plot the t-SNE embedding of the dynamic features of the inputs and their positive samples as computed by C-DSVAE (top) and our approach (bottom). In this setting, the blue and orange embeddings should be as close as possible.

B.4. Qualitative Evaluations

Here, we propose an additional qualitative evaluation of our contrastive estimation using the MUG dataset. Specifically, we evaluated the positive and negative samples on two trained models, C-DSVAE and ours. We collect the dynamic latent representations $d^i_{1:T}$ for every sample i in the test set, and we compute the mean value $d = \frac{1}{T}\sum_{j=1}^{T} d_j$, where the index i is omitted for brevity. For each of those samples, we extract a subset of their positive d^+ and negative d^- samples. To visualize these latent features, we project the original representation d and the new samples d^+ and d^- using t-SNE (Van der Maaten & Hinton, 2008). We anticipate that the pair (d, d^+) will be close in latent space as contrastive learning attracts positive data points closer. In contrast, contrastive learning repels negative samples, and thus (d, d^-) should be far from each other. We present the results in 5 and 6. For the positive samples, our method shows an impressive similarity between d and d^+. In comparison, C-DSVAE presents a much bigger distance between the d and d^+. On the negative samples, our method shows good discrimination between d and d^-, and in addition, our samples are much more concentrated. In comparison, discriminating between the input and negative samples in C-DSVAE is more challenging. We obtained similar results for the static setting, i.e., when we studied the t-SNE embeddings of s and its positive s^+ and negative s^- samples.

B.5. Additional Information on the TIMIT Speaker Verification Task

Here, we discuss more on the Audio experiment with TIMIT described in Sec. 5.4. First, the TIMIT test dataset contains eight different sentences for each speaker with 24 unique speakers. In total there are 192 audios. For all those audios, we extract their s and $d_{1:T}$ latent representations. Then, to prepare a single vector representation we calculate the identity representation vector by the same procedure described in (Yingzhen & Mandt, 2018). Last, the EER is being calculated

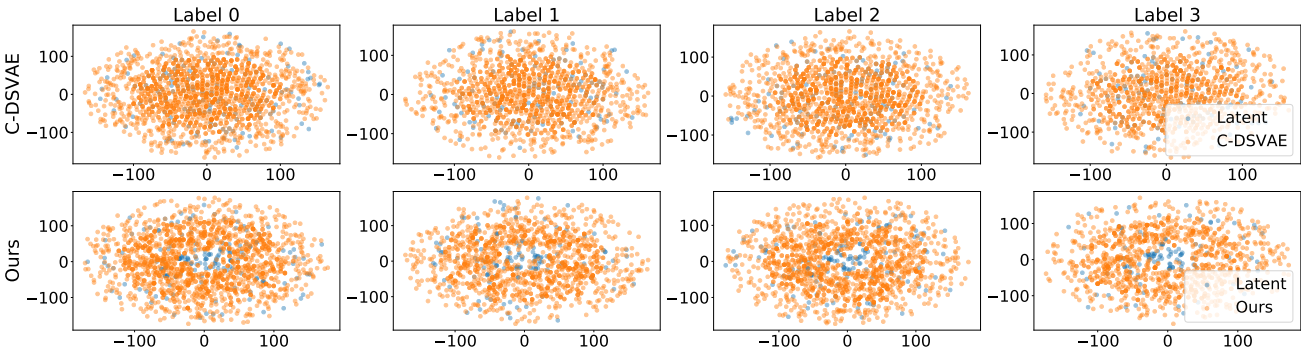

Figure 6. We plot the t-SNE embedding of the dynamic features of the inputs and their negative samples as computed by C-DSVAE (top) and our approach (bottom). In this setting, the blue and orange embeddings should be as separate as possible.

separately for all the combinations of 192 vectors. In total, there are 18 336 pairs. We repeat this process independently once for the s features and once for the $d_{1:T}$ features.

B.6. Standard Deviation Measures for Tab. 1

Here, we report the standard deviation measures related to Tab. 1 in the main text. Notice that the audio experiment is deterministic due to the ERR metric definition, and thus it does not have standard deviation measures. The extended results are provided in Tab. 8. These results indicate that not only our method achieves superior results in comparison to SOTA approaches, but also, it is well within the statistical significance regime, given the standard deviation measures.

B.7. Data Generation

We qualitatively evaluate our model's ability to generate static and dynamics features. Specifically, let $x_{1:t} \sim p_D$ denote a sample from the data with its static and dynamic factors latent representations $(s, d_{1:T})$ given by $s \sim q(s \,|\, x_{1:T})$ and $d_{1:T} \sim q(d_{1:T} \,|\, x_{1:T})$. We generate new static features by sampling from the static prior distribution $p(s)$, namely, $\tilde{s} \sim p(s)$ and fixing the dynamics $d_{1:T}$. Then, we concatenate $(\tilde{s}, d_{1:T})$, and we generate a new sample $\tilde{x}_{1:T} \sim p(\tilde{x}_{1:T} \,|\, \tilde{s}, d_{1:T})$. Finally, we perform a similar process in order to generate the dynamics, where we sample from the dynamic prior distribution and the static features are fixed. The results of static and dynamic features' generation for the Sprites and MUG datasets are given in Fig. 7, Fig. 8, Fig. 9, and Fig. 10. The left column in each figure contains the original samples and the right column contains the generated samples. If the model disentangles the features well and has high generation performance, then, the fixed features should be preserved perfectly and the generated features should be random (independent of the original class).

B.8. Swaps

In this section we perform another qualitative experiment. Specifically, given source and target samples, $x^{\text{src}}_{1:T}, x^{\text{tgt}}_{1:T} \sim p_D$, we swap the static and dynamic features between the source and the target. In practice, we feed the encoder with the samples

Table 8. We augment Tab. 1 for the Sprites and MUG datasets with standard deviation measures.

	Sprites				MUG			
Method	Acc↑	IS↑	$H(y\|x)$↓	$H(y)$↑	Acc↑	IS↑	$H(y\|x)$↓	$H(y)$↑
MoCoGAN	92.89%	8.461	0.090	2.192	63.12%	4.332	0.183	1.721
DSVAE	90.73%	8.384	0.072	2.192	54.29%	3.608	0.374	1.657
R-WAE	98.98%	8.516	0.055	**2.197**	71.25%	5.149	0.131	1.771
S3VAE	99.49%	8.637	0.041	**2.197**	70.51%	5.136	0.135	1.760
SKD	**100%**	**8.999**	**1.6e−7**	**2.197**	77.45%	**5.569**	**0.052**	1.769
C-DSVAE	99.99%	8.871	0.014	**2.197**	81.16%	5.341	0.092	1.775
Ours	**100% ± 0%**	8.942 ± 3.3e−5	0.006 ± 4e−6	**2.197 ± 0**	**85.71%** ± 0.9	**5.548** ± 0.039	0.066 ± 4e−3	**1.779** ± 6e−3

Table 9. Downstream classification task on latent static and dynamic features **using SVC**.

Dataset	Method	Static features			Dynamic features		
		Static L-Acc ↑	Dynamic L-Acc ↓	Gap ↑	Static L-Acc ↓	Dynamic L-Acc ↑	Gap ↑
MUG	random	1.92%	16.66%	-	1.92%	16.66%	-
	C-DSVAE	**98.75% ± 1%**	76.25% ± 2.9%	22.25%	26.25% ± 3.8%	82.50% ± 2.5%	56.25%
	Ours	98.12% ± 0.9%	**68.75% ± 2.6%**	**29.37%**	**10.00% ± 3.4%**	**85.62% ± 2.4%**	**75.62%**
Letters	random	1.65%	3.84%	-	1.65%	3.84%	-
	C-DSVAE	95.47% ± 0.5%	13.0% ± 0.6%	82.47%	**2.79% ± 0.5%**	66.35% ± 1%	63.56%
	Ours	**100% ± 0%**	**12.16% ± 0.6%**	**87.84%**	3.06% ± 0.3%	**69.75% ± 1.4%**	**66.69%**

Table 10. Downstream classification task on latent static and dynamic features **using Random Forest Classifier**.

Dataset	Method	Static features			Dynamic features		
		Static L-Acc ↑	Dynamic L-Acc ↓	Gap ↑	Static L-Acc ↓	Dynamic L-Acc ↑	Gap ↑
MUG	random	1.92%	16.66%	-	1.92%	16.66%	-
	C-DSVAE	**98.75% ± 0.7%**	72.75% ± 2.8%	26%	64% ± 4.1%	83.18% ± 1.7%	19.18%
	Ours	98.06% ± 1.1%	**68.50% ± 2.7%**	**29.56%**	**38.93% ± 4.6%**	**86.75% ± 2.5%**	**47.82%**
Letters	random	1.65%	3.84%	-	1.65%	3.84%	-
	C-DSVAE	100% ± 0%	5.6% ± 0.6%	94.4%	3.66% ± 0.3%	75.44% ± 1%	71.78%
	Ours	100% ± 0%	5.6% ± 0.5%	94.4%	**2.92% ± 0.3%**	**77.63% ± 1%**	**74.71%**

to extract their static and dynamic latent representation, $(s^{\text{src}}, d^{\text{src}}_{1:T})$ s.t $s^{\text{src}} \sim q(s^{\text{src}} \mid x^{\text{src}}_{1:T})$ and $d^{\text{src}}_{1:T} \sim q(d^{\text{src}}_{1:T} \mid x^{\text{src}}_{1:T})$ for the source and $s^{\text{tgt}} \sim q(s^{\text{tgt}} \mid x_{1:T})^{\text{tgt}}$ and $d^{\text{tgt}}_{1:T} \sim q(d^{\text{tgt}}_{1:T} \mid x^{\text{tgt}}_{1:T})$ for the target. Then, we generate swapped samples by feeding the decoder, $\tilde{x}^{\text{src}}_{1:T} \sim p(\tilde{x}^{\text{src}}_{1:T} \mid \tilde{s}^{\text{tgt}}, d^{\text{src}}_{1:T})$ and $\tilde{x}^{\text{tgt}}_{1:T} \sim p(\tilde{x}^{\text{tgt}}_{1:T} \mid \tilde{s}^{\text{src}}, d^{\text{tgt}}_{1:T})$. If the representation is well disentangled, $\tilde{x}^{\text{src}}_{1:T}$ should preserve its original dynamics but have the target's static features and vice versa for $\tilde{x}^{\text{tgt}}_{1:T}$. Fig. 11 and Fig. 12 show four separate examples of Sprites and MUG where the length of the MUG sequences are shorten to $T = 10$ for clarity. The first row of each pair shows the original samples $x^{\text{src}}_{1:T}, x^{\text{tgt}}_{1:T} \sim p_D$. The row below shows the swapping results. Namely, rows $1, 3, 5, 7$ represent the original samples and rows $2, 4, 6, 8$ represent swapped samples.

B.9. Latent Classification Experiments with Different Classifiers and Standard Deviation

Here, we elaborate more on the experiment we reported in Sec. 5.5. We extracted the static s and dynamic $d_{1:T}$ features using a trained model and trained four different classifiers. All trained classifiers are Support Vector Machines. We used the default Support Vector Classifier (SVC) of the sklearn package without changing any hyperparameter. To strengthen the statistical significance of Tab. 3 from the main text, we conduct the same experiment with different classifiers and report their results in Tab. 9 (SVC), Tab. 10 (Random Forest Classifier), and Tab. 11 (KNN). These tables show that our results are robust to the choice of the classifier. We repeated the experiments per classifier for 10 times with different seeds for data splitting and report their means and standard deviations. We used the default sklearn Random Forest Classifier and KNN. We used the sklearn default hyperparameters and conducted the same experiment procedure exactly. In the Jesters dataset, we do the exact same process just without the static features and their classifiers since the subjects in this data are not labeled. In the Letters dataset, there is one difference. For each d_i , $i = 1, ..., T$ in $d_{1:T}$ we try to predict its corresponding letter label in the sequence instead of trying to predict the whole word. Briefly, our model maintains its superior performance in comparison to C-DSVAE (Bai et al., 2021) with respect to the gap metric among all classifiers.

B.10. Robustness with Respect to the Seed Choice

In our work, we based our evaluation section with respect to existing state-of-the-art models and their evaluation protocol and benchmark datasets. Following these approaches, the sensitivity to hyperparameters and randomness is typically not considered. Nevertheless, we re-trained our model on five different seeds in total to test its robustness with respect to the particular choice of seed. We report the results in Tab. 12. These results indicate that our method is statistically significant with respect to previous SOTA approaches.

Table 11. Downstream classification task on latent static and dynamic features **using KNN**.

Dataset	Method	Static features			Dynamic features		
		Static L-Acc ↑	Dynamic L-Acc ↓	Gap ↑	Static L-Acc ↓	Dynamic L-Acc ↑	Gap ↑
MUG	random	1.92%	16.66%	-	1.92%	16.66%	-
	C-DSVAE	98.31% ± 1.1%	50.31% ± 3.6%	48%	26.18% ± 3.4%	**83.25% ± 2.1%**	57.07%
	Ours	**99.31% ± 0.9%**	**49.25% ± 4.5%**	**50.06%**	**19.31% ± 2.8%**	82.50% ± 1.5%	**63.19%**
Letters	random	1.65%	3.84%	-	1.65%	3.84%	-
	C-DSVAE	100% ± 0%	5.8% ± 0.5%	94.2%	**3.41% ± 0.3%**	76.61% ± 0.9%	73.2%
	Ours	100% ± 0%	5.8% ± 0.6%	94.2%	3.54% ± 0.3%	**78.33% ± 0.8%**	**74.92%**

Table 12. We augment Tab. 1 for the MUG dataset with the mean and standard deviation measures using of five models trained with different seed values.

Method	Sprites				MUG			
	Acc↑	IS↑	$H(y\|x)$↓	$H(y)$↑	Acc↑	IS↑	$H(y\|x)$↓	$H(y)$↑
MoCoGAN	92.89%	8.461	0.090	2.192	63.12%	4.332	0.183	1.721
DSVAE	90.73%	8.384	0.072	2.192	54.29%	3.608	0.374	1.657
R-WAE	98.98%	8.516	0.055	**2.197**	71.25%	5.149	0.131	1.771
S3VAE	99.49%	8.637	0.041	**2.197**	70.51%	5.136	0.135	1.760
SKD	**100%**	**8.999**	**1.6e−7**	**2.197**	77.45%	**5.569**	**0.052**	1.769
C-DSVAE	99.99%	8.871	0.014	**2.197**	81.16%	5.341	0.092	1.775
Ours	**100% ± 0%**	8.942 ± 3.3e−5	0.006 ± 4e−6	**2.197 ± 0**	**85.06% ± 1.06**	5.517 ± 0.034	0.073 ± 4e−3	**1.782** ± 3e−3

Figure 7. Content generation results in Sprites dataset. See the text for additional details.

Figure 8. Dynamics generation results in Sprites dataset. See the text for additional details.

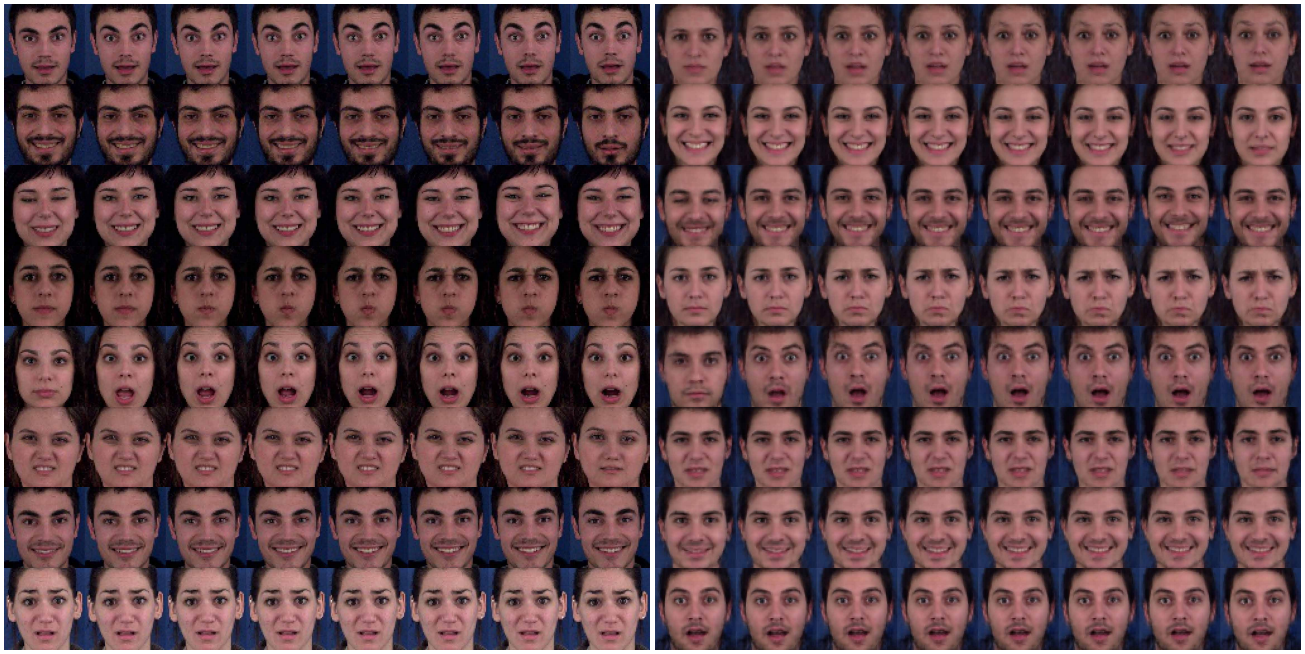

Figure 9. Content generation results in MUG dataset. See the text for additional details.

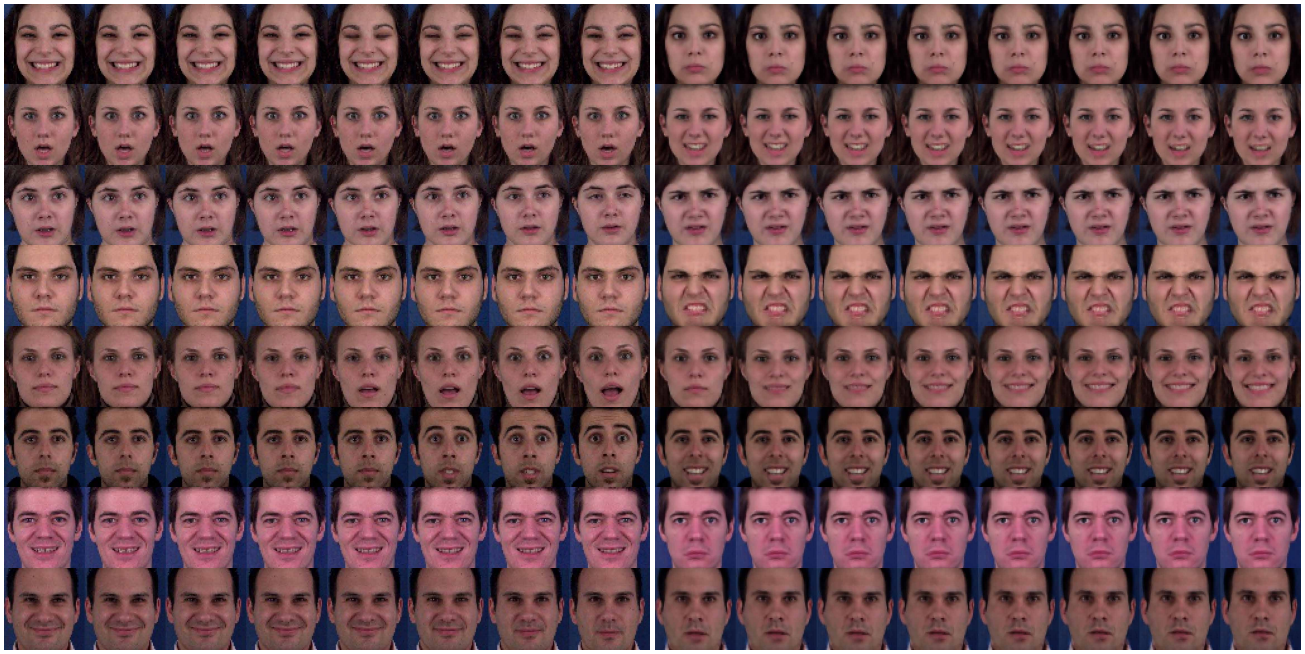

Figure 10. Dynamics generation results in MUG dataset. See the text for additional details.

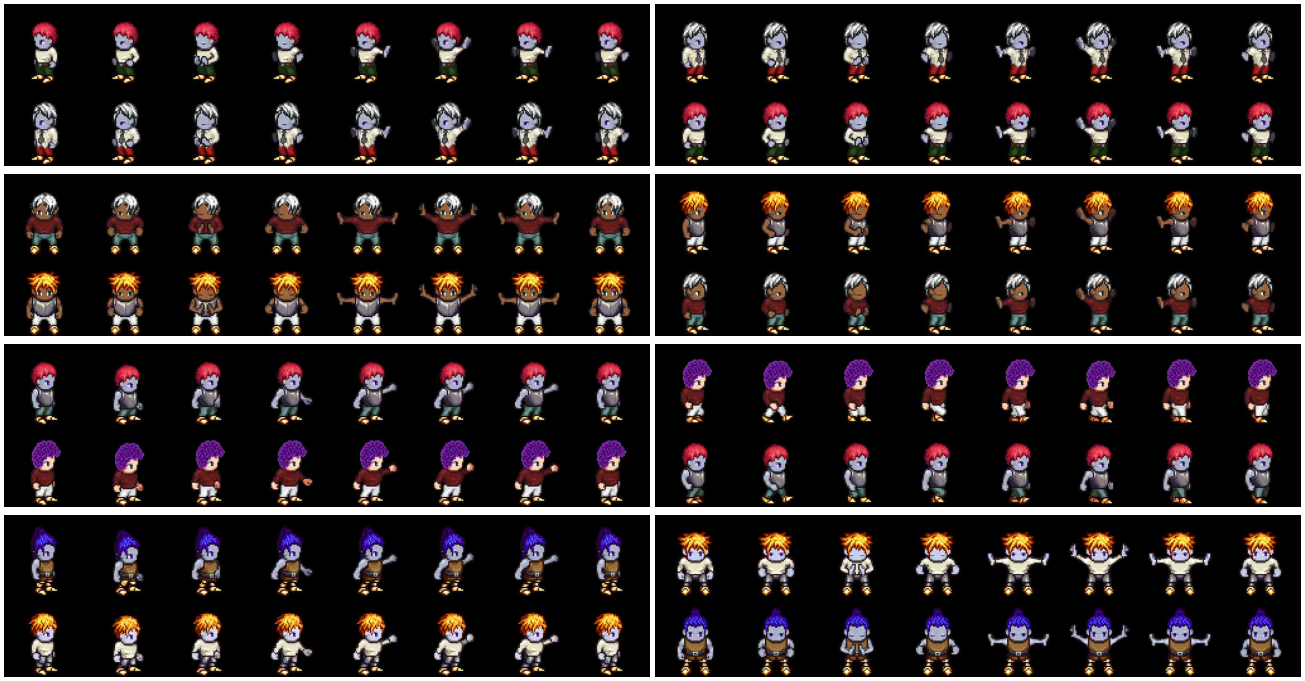

Figure 11. Swapping results in Sprites dataset. See the text for additional details.

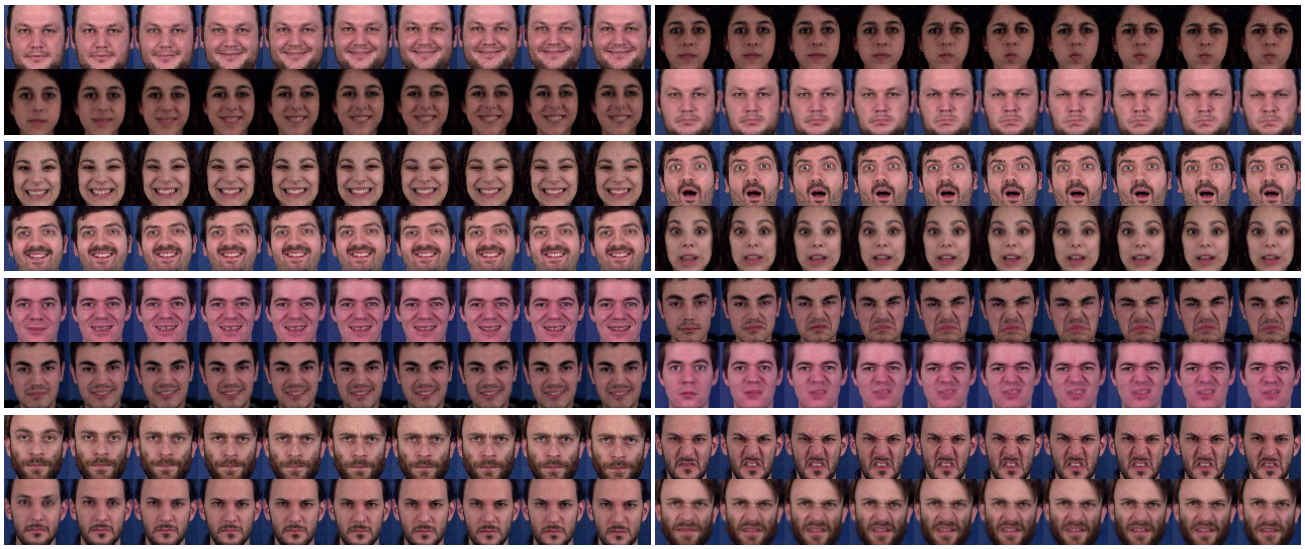

Figure 12. Swapping results in MUG dataset. See the text for additional details.