Received 18 September 2023, accepted 4 October 2023, date of publication 17 October 2023, date of current version 27 October 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3325284

RESEARCH ARTICLE

Class-Variational Learning With Capsule Networks for Deep Entity-Subspace Clustering

NIKOLAI A. K. STEUR AND FRIEDHELM SCHWENKER, (Member, IEEE)
Institute of Neural Information Processing, Ulm University, 89081 Ulm, Germany

Corresponding author: Friedhelm Schwenker (friedhelm.schwenker@uni-ulm.de)

The work of Nikolai A. K. Steur was supported by a Landesgraduiertenförderungsgesetz (LGFG) Scholarship through Ulm University. The work of Friedhelm Schwenker was supported by the German Research Foundation (DFG) under Grant SCHW 623/7-1.

ABSTRACT The progression of deep clustering techniques in the recent years emphasizes the need for unsupervised representation learning methods that build lower-dimensional embeddings within expressive latent feature spaces. An important performance factor for such techniques constitutes the representational capacity of the used neural network technology. Although Capsule Networks (CapsNet)s are predestinated for the task of deep clustering through their rich entity representations and inter-layer dynamics related to clustering, CapsNets are to date rarely explored in this context. The main challenge for enabling unsupervised representation learning with CapsNets results from the required differentiation of the output capsules to encompass data-intrinsic classes. This paper proposes a novel end-to-end framework denominated as *Class-Variational Learning (CVL)* which utilizes an asymmetric autoencoder consisting of a CapsNet encoder and a non-capsular decoder network for facilitating entity-subspace clustering. To the best of our knowledge CVL represents the first approach which accomplishes a class-to-capsule specialization of the output capsules without external supervisory signals. As unique characteristic, CVL forms an equivariant latent space with continuous transitions between data-intrinsic classes. This means a crucial gain in the explainability of the constructed inference mechanism, since the class-discriminative equivariant space is linearly navigable and fully accessible by a human actor. Despite our CVL model does currently not lead to competitive accuracies compared to the state-of-the-art deep clustering techniques, CVL opens promising perspectives on the use of CapsNets as the basis for deep clustering which hopefully motivates future research in this field.

INDEX TERMS Autoencoder, capsule networks, class-discriminative equivariance, deep clustering, unsupervised representation learning.

I. INTRODUCTION

For unleashing the potency of neural networks to completely learn from unlabled data, a promising trend called *deep clustering* [1], [2], [3], [4], [5], [6], [7], [8], [9], [10] is emerging in which state-of-the-art neural networks are equipped with clustering techniques or objectives to establish unsupervised representation learning. In fact, Kauffmann et al. [11] recently explored the natural relationship between neural networks and standard clustering algorithms by transforming clustering algorithms fitted to the data into *functionally equivalent* neural networks without the need for retraining, and denominated this proceeding as *neuralization*. In general, deep clustering models learn a parametric, nonlinear mapping from the original data space to a lower-dimensional embedding space which satisfies an implicit or explicit clustering objective (cf. [6], [9], [12]).

A special type of neural networks, namely *Capsule Networks (CapsNet)s* [13], [14], are predestinated to form the basis of deep clustering methods since CapsNets usually manage their inter-layer dynamics as clustering between lower-layer entity embeddings to higher-layer ones [14], which is referred to as *routing-by-agreement* [13], [14]. Moreover, CapsNets offer high representational capacity through the segregation of entity instantiations in the distributed representation of capsules and signifying entity

The associate editor coordinating the review of this manuscript and approving it for publication was Muhammad Sharif.

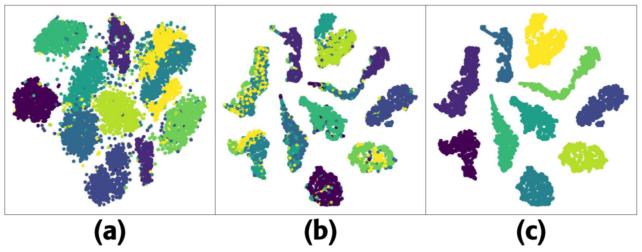

FIGURE 1. 2D-Projection of the test set from MNIST using (a) t-SNE and (b) & (c) t-SNE on the embeddings from a CVL model. Coloring equals (a) & (b) the dataset labels and (c) the cluster assignments from the CVL model.

existence as capsule vector length [13]. Despite this beneficial prerequisites, CapsNets are to date rarely explored for the task of deep clustering in the sense of general-purpose unsupervised representation learning. This situation apparently has its origin in the obstacle to orchestrate an unsupervised training algorithm that leads to a discriminative specialization of capsules in the output layer within a CapsNet. In particular, previous investigation pointed out that naive attempts such as training a CapsNet autoencoder, without selecting a specific output capsule in the encoder (known as *masking* [13]), degrades the beneficial CapsNet properties by collapsing into a non-capsular autoencoder [15]. To cope with this circumstance we structure the task of deep clustering with CapsNets into the following challenges:

1) **Discriminative capsule specialization,**
2) **Preservation of capsular properties,**
3) **Explainability of cluster assignments.**

In this work we address these challenges with a deep clustering method dedicated to the use of a CapsNet encoder, enclosed in the novel training paradigm *Class-Variational Learning (CVL)* which defines an implicit clustering objective. As quick outlook, we design an asymmetric autoencoder architecture with a CapsNet encoder and a non-capsular decoder network to direct an end-to-end training procedure for discriminative representation learning, where the output capsules of the CapsNet encoder form separated entity subspaces to take the role of clusters. Fig. 1 supplies a first impression of the latent space produced by the output capsules of a CVL model for the images from the handwritten digits dataset *MNIST* [16], [17]. For the visualization in the two-dimensional space, the dimensionality reduction method *t-distributed Stochastic Neighbor Embedding (t-SNE)*[1] [19], [20] is applied on the original data points and the latent representations of the CVL model. Although the CVL model does not perfectly match the predefined labels for MNIST, CVL accomplishes the creation of ten well-separated and similarly sized clusters. Furthermore, we will show that a CVL model satisfies the three identified challenges by constructing a class-discriminative equivariant space, which allows the continuous observation and evaluation of latent representations on the entity appearance manifold.

Our contribution is threefold: Firstly, we introduce CVL as novel unsupervised learning procedure to receive lower-dimensional embeddings from the output capsules of a CapsNet encoder. Our framework presents step-by-step strategies to deal with difficulties occurring with the use of CapsNets in an unsupervised learning setting. Specifically, we propose a new sparsity boosting mechanism for balancing the average usage of output capsules which can be adapted to other tasks than CVL. Secondly, we prove our concept by means of descriptive experiments and investigate special properties of our approach. A key result of our experiments constitutes the fusion of equivariant features with the task of class discrimination. Thirdly, CVL contributes to the progress of deep clustering in the field of CapsNets with a relatively simple framework for associating meaningful clusters with the output capsules and preventing non-capsular network degradation. Despite the exclusive use of the MNIST dataset in our experiments, we declare CVL as domain-independent algorithm with an adjustable sampling procedure and reconstruction quality criterion for distinct applications.

The rest of the paper is structured as follows: Section II reviews the literature regarding non-capsular deep clustering approaches and existing perspectives in the unsupervised training of CapsNets, section III introduces the CVL paradigm and builds stepwise its mathematical framework, section IV evaluates CVL's performance on various experiments with MNIST, section V discusses the empirical results of the experiments and provides important implications, section VI summarizes the central findings of this work.

II. RELATED WORK

A. NON-CAPSULAR DEEP CLUSTERING

The most deep clustering approaches [2], [3], [4], [6], [9], [10], [12], [21] utilize an autoencoder [22] architecture in combination with a reconstruction loss [2], [3], [4], [6], [9], [12], [21] to learn rich representations by realizing a bidirectional mapping between the original data space and the latent feature space. These approaches show a variety of autoencoder implementations: Often fully-connected neural networks constitute an autoencoder [4], [12], [21], in other works convolutional variants [2], [3], [6] or self-evolving autoencoders [10] are used. Sometimes net architectures with multiple encoder or decoder components are exploited for increasing training success [3], [6], [9]. In rare cases, decoder components are removed or ignored after a first stage in the training process [12], [21].

Apart from the concrete choice for an architectural design, deep clustering methods meaningfully differ in their optimization strategies, as well. Usually, pretext tasks are conducted for learning an expressive embedding space, for example, through recovering noisy data samples [3], [6], [12], preserving instance-based local neighborhoods [23], predicting image rotation angles [7] or by applying layer-wise pretraining for autoencoders [4], [6], [12], [21]. Other approaches [21], [23] divide the model training into two separated phases of feature learning and clustering. A similar

[1]Implementation used from *scikit-learn* [18].

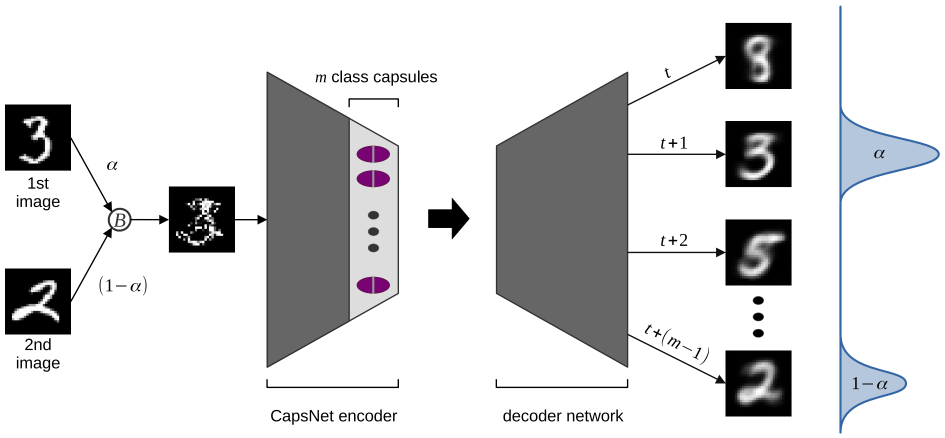

FIGURE 2. Overview of the unsupervised CVL scheme using image data from MNIST. A CapsNet encoder with *m* class capsules gets jointly trained with a non-capsular decoder network for input reconstruction. First, a CV image is generated by sampling pixels from two source images using a Bernoulli distribution with probability α. Each class capsule of the encoder emits a representation for the CV input image. The decoder network produces a reconstruction for each class capsule representation. The training objective comprises the attribution of both source images to the capsules with the minimal-distant reconstructions. In this example, the second ($t+1$) and the last ($t+(m-1)$) reconstruction constitute the most similar ones to both source images. The target signal (in blue) for the class distribution is respectively stated with the given Bernoulli probabilities, positioned at the capsule indices with both minimal-distant reconstructions. Thus, CVL demands from a model to detect and describe intrinsic classes within a data collection.

proceeding represents the alternation between epochs for feature learning and clustering during model training [7]. However, many deep clustering techniques [1], [2], [3], [5], [8], [24] prefer a training strategy in an end-to-end fashion where embeddings are simultaneously learned with clustering objectives.

For sure, a key benefit of neural networks for the use as clustering method results from their architectural flexibility and inherent ability of multi-modal optimization using individual loss functions per task-specific goal. This circumstance allows the reformulation of standard clustering algorithms in terms of neural network training objectives. For instance, some deep clustering methods [1], [6], [21] are derived from insights of spectral clustering, another work [24] formulates a recurrent training framework with an agglomerative clustering objective for deep representation learning, and a further method [8] consists of task-specific sparse-coding components trained on one or multiple clustering objectives. Interestingly, the natural relation between deep clustering and traditional clustering techniques can also be emphasized by regarding deep clustering as Expectation-Maximization (EM) [25] algorithm in which the M-step computes point-to-cluster assignments with fixed network parameters and the E-step corresponds to the adjustment of the network parameters based on the former assignments (cf. [3]). Distinct realizations of both EM-steps can be attributed to several deep clustering approaches [2], [3], [4], [5], [7], [10], [12], [24].

B. UNSUPERVISED LEARNING WITH CAPSNETS

In the field of unsupervised learning with CapsNets autoencoder architectures as well emerge [15], [26]. Rawlinson et al. [15] preserved the specialization of latent capsules within an asymmetric autoencoder composed of a CapsNet encoder and a non-capsular decoder network, after the design from Sabour et al. [13], by implementing a rank-based sparse masking depending on individual capsule activity. Since Rawlinson et al.'s [15] training algorithm applies a masking on the joined vector of all output capsules in the encoder to activate a small fraction of capsules per sample, each reconstruction is still controlled by multiple output capsules. Although Rawlinson et al.'s [15] approach was not intended for deep clustering, their empirical results serve as theoretical foundation for successfully training CapsNets in an unsupervised learning setting. Also not intended for the task of clustering, Kosiorek et al. [26] proposed an extensive framework using a set transformer architecture for input encoding and unsupervised stacked capsule autoencoders to model geometric relationships between visual objects and their parts, resulting in impressive accuracies when clustering the learned latent space. However, replacing the set transformer with a standard multi-layer perceptron drastically degrades clustering accuracies on the tested image datasets [26]. With the aim of clustering sound recordings, Lin et al. [27] designed an asymmetric convolutional autoencoder, consisting of a CapsNet encoder and a non-capsular decoder network, which is jointly trained using a reconstruction loss and a *clustering layer* [12] for the embedding space. In particular, Lin et al. [27] concatenate the vector lengths of the output capsules from the CapsNet encoder to constitute deep embeddings for the clustering.

In coherence to previous attempts, our CVL model constitutes an asymmetric autoencoder architecture with a CapsNet encoder and a non-capsular decoder network (cf. Fig. 2), but CVL's training algorithm exploits the full capacity of capsules to represent entity subspaces as separated clusters

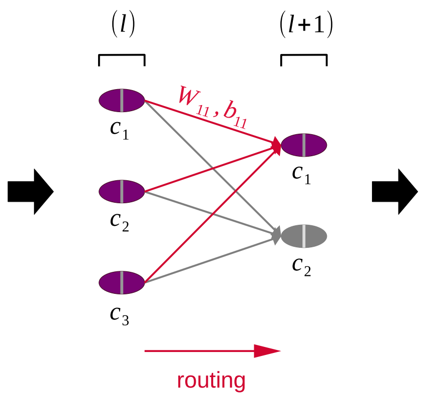

FIGURE 3. Schematic communication between two fully-connected capsule layers with nonlinear routing. The instantiation parameters of each capsule $\mathbf{c}_i$ from the $(l+1)$-layer are individually computed as aggregation over all capsules from the previous layer.

through the reconstruction of input samples with single capsules. This leads to a specialization of capsules in detecting and representing discriminative intrinsic classes.

III. CLASS-VARIATIONAL LEARNING

The aim of CVL is the unsupervised identification of inherent data characteristics that allow the effective discrimination between *different kinds* of data instances. In unsupervised learning data instances are associated with clusters, which represent groups of elements with similar properties, whereas in supervised learning data instances are assigned to classes according to predefined criteria in the form of a labeled dataset. Although CVL is by definition originated in the unsupervised domain, using the term *clusters* instead of *classes* would be misleading due to the fact that humans expect classes, means diverging object types, in unknown data collections that are commonly verbalizable. More precisely, CVL makes the strong assumption that every data point $\mathbf{x}$ within an arbitrary dataset X exclusively belongs to an intrinsic class $\mathcal{C}_i$:

$$\forall \mathbf{x} \in X,\ \exists! \mathcal{C}_i \in \{\mathcal{C}_1, .., \mathcal{C}_m\} : \mathbf{x} \in \mathcal{C}_i. \tag{1}$$

This proceeding reformulates a clustering operation as single-label classification problem for realizing unsupervised representation learning. CVL was inspired by the learning principle *time-contrastive learning* [28] which exploits nonstationarities in the temporal structure of time series data to extract latent variables per specified time segment, based on the interpretation of segments as classes. Contrary to this, CVL makes no further assumptions about data object properties at all, instead, it targets the observable variances in the data generation process to organize instances into categories.

A. MODEL OVERVIEW

Fig. 2 introduces the generic CVL model with an exemplary training step for image data from MNIST. First, a Class-Variational (CV) input image is generated by sampling each pixel value from a Bernoulli distribution with probability α for the first source image and with probability $(1-\alpha)$ for the second one. The CapsNet encodes the CV sample into vector repesentations encapsulated in m output capsules, referred to as *class capsules* (cf. [14]). In the next step, the non-capsular decoder network sequentially computes a reconstruction for each class capsule. The CVL model is jointly trained by: 1) Calculating the mean squared error for each reconstructed image to both source images. 2) Determining and backpropagating the minimal reconstruction errors for both source images with loss weights α and $(1-\alpha)$, respectively. 3) Entangling the vector lengths of the class capsules with their reconstruction ability in relation to the generation process for CV images with Bernoulli sampling.

In addition, CVL conducts certain regularizations to promote clustering quality. The subsequent sections further elucidate the CVL process by covering each training step in detail and building up a comprehensive mathematical framework for the abstract definition of CVL. Note that the mathematical framework uses its own numbering for logical separation between model components and diverse training objectives.

B. MATHEMATICAL FRAMEWORK

1) Class-Variational Sampling (CVS)

In general, CVS can be defined as the operation of randomly combining at least two samples from a training corpus using a known discrete probability distribution. The combine operation is here represented as the exclusive feature value selection per feature dimension. Thus, CVS can be interpreted as the cartesian product of the original dataset $X \in \mathbb{R}^{n \times d}$ mapped to a CV dataset $\tilde{X} \in \mathbb{R}^{n^* \times d}$ with probabilistically mixed feature values. The element-wise process of generating a CV sample $\tilde{\mathbf{x}}$ at time step t can be described with

$$\tilde{\mathbf{x}}(t) \leftarrow \forall i \in \{1, .., d\} : \tilde{x}_i \sim M(x_i^{(1)}, x_i^{(2)}, .., x_i^{(k)}; \boldsymbol{\alpha}),$$
$$\mathbf{x}^{(j)} \sim P_X \tag{2}$$

where P_X equals the prior class probability distribution for drawing a sample $\mathbf{x}^{(j)}$ from the original dataset X, $\tilde{x}_i$ means the i-th vector element of the received CV sample and M refers to a Multinoulli distribution (also known as *categorical distribution*) parameterized with the probability vector $\boldsymbol{\alpha}$. Using the Multinoulli distribution allows the generalized definition of CVS (including the Bernoulli sampling in Fig. 2). The probability density function

$$p_M(i; \boldsymbol{\alpha}) = \alpha_i, \qquad \sum_{i=1}^{k} \alpha_i = 1, \tag{3}$$

of the Multinoulli distribution assigns to each original sample $\mathbf{x}_i$ for the element-wise choice of feature values the probability α_i. Furthermore, the α-values establish relevance scores for reconstructions produced by the CVL model. The combination of source samples with the same entity type is consciously not prohibited since we fully leave the discovery of data-intrinsic classes to the CapsNet encoder.

2) CAPSNET ENCODER & DECODER NETWORK

For the feature encoder in our deep CVL model, we use a CapsNet with vector-based capsules $\mathbf{c}_i$ (initially introduced by Hinton et al. [29]) for each network layer l with distinct number of vector elements per layer (i.e. $\mathbf{c}_i^{(l)} \in \mathbb{R}^{v^{(l)}}$). The existence probability for an entity observed by a capsule is given as the length of the capsule vector [13], and denominated as *capsule activation* (cf. [14], [30]). The dynamic behavior between two consecutive capsule layers can be mathematically expressed as [13], [14], [30], and [31]:

$$\mathbf{c}_i^{(l+1)} = \text{route}(\forall \mathbf{c}_j^{(l)} : W_{ij}\mathbf{c}_j^{(l)} \oplus b_{ij}), \tag{4}$$

$$\text{route} : \mathbb{R}^{m^{(l)} \times v^{(l)}} \longmapsto \mathbb{R}^{v^{(l+1)}}, \tag{5}$$

where the instantiation parameters of a capsule $\mathbf{c}_i^{(l+1)}$ from the subsequent layer are computed as aggregation over the transformed entity representations of all $m^{(l)}$ capsules $\mathbf{c}_j^{(l)}$ from the previous layer, processed by a nonlinear routing procedure. Routing procedures in the context of CapsNets are usually implemented as iterative routing-by-agreement process for entity harmonization between adjacent layers [13], [14]. Similar to Sabour et al.'s [13] *dynamic routing* – but more general, we restrict the route function to output capsule vectors with an activation in the range of $[0, 1]$. Each connection between a lower-layer capsule $\mathbf{c}_j^{(l)}$ and a higher-layer capsule $\mathbf{c}_i^{(l+1)}$ owns an individual transformation matrix W_{ij} and a bias b_{ij}. The bias gets element-wise added to the intermediate vector $W_{ij}\mathbf{c}_j^{(l)}$. The capsule dimensionality can arbitrarily vary per layer using specific-shaped transformation matrices and if necessary applying additional reshaping operations [30]. Fig. 3 visualizes the interaction between capsules from two neighbored layers. It is important to note that each higher-layer capsule has another data-characteristic perspective on the entity representations from the previous layer through the projection onto an exclusive feature subspace for each lower-layer entity (cf. [14]). The CapsNet Encoder (CEnc) corresponds to a mapping function of a data point $\mathbf{x}$ from the original dataset $X \in \mathbb{R}^{n \times d}$ to a matrix C with row-wise m class capsules $\mathbf{c} \in \mathbb{R}^v$:

$$\text{CEnc} : \mathbf{x} \in \mathbb{R}^d \longmapsto C \in \mathbb{R}^{m \times v}. \tag{6}$$

Note that layer indices are here omitted for simplicity. The Decoder (Dec) conducts the reverse mapping to (6) as

$$\text{Dec} : \mathbf{c} \in \mathbb{R}^v \longmapsto \mathbf{r} \in \mathbb{R}^d \tag{7}$$

which projects an incoming capsule vector, one at a time, onto the original feature space of the dataset X. We denominate the decoding scheme as *reconstructing* or *recovering* original data samples in the view of class-specific capsules. CVL's asymmetric autoencoder can be seen as multiplicative more efficient than a classical autoencoder [22] due to the reuse of lower-level features for distinct entity aggregation distributed over class capsules. Such an asymmetric autoencoder structure was primarily constructed by Sabour et al. [13] to establish entity visualization for an image classification task.

3) CLASS-VARIATIONAL RECONSTRUCTION LOSS

The CV reconstruction loss corresponds to the composition loss for recovering each source input of a CV sample weighted by the probability density function of the sampling distribution. A conventional measure for the dissimilarity between multi-dimensional data points represents the Mean Squared Error (MSE). Hence, the minimal-distant reconstructions for each of the k source inputs are determined by

$$\forall j \in \{1, .., k\} : \mathbf{r}_i^{(j)} = \underset{\mathbf{r} \in \mathcal{R}(\tilde{\mathbf{x}}_i)}{\arg\min} ||\mathbf{r} - \mathbf{x}_i^{(j)}||_2^2 \tag{8}$$

where the set $\mathcal{R}(\tilde{\mathbf{x}}_i)$ contains the reconstructions of all m class capsules for the current CV sample $\tilde{\mathbf{x}}_i$, and the resulting vector $\mathbf{r}_i^{(j)}$ represents the minimal-distant reconstruction with respect to the j-th source sample $\mathbf{x}_i^{(j)}$ used for generating the i-th CV sample. The expected reconstruction loss $\mathcal{L}_{R_j}$ for the j-th source sample over a batch $\tilde{X}_B$ from the CV dataset (i.e. $\tilde{X}_B \subseteq \tilde{X}$) can be analogously formulated as

$$\mathcal{L}_{R_j} = \mathbb{E}_{\tilde{X}_B}\left[||\mathbf{r}_i^{(j)} - \mathbf{x}_i^{(j)}||_2^2\right]. \tag{9}$$

Since (3) ensures a valid probability distribution for selecting feature values from the k source components $\mathbf{x}_i^{(j)}$, this probability distribution reflects the assumable ratio of class-specific properties within a CV sample. In the general case, this leads to a total reconstruction loss of

$$\mathcal{L}_R = \lambda_R \mathbb{E}_{\sim M}\left[\mathcal{L}_{R_j}\right] = \lambda_R \sum_{j=1}^{k} \alpha_j \mathcal{L}_{R_j} \tag{10}$$

where the hyperparameter λ_R is introduced for potential loss balancing. The training goal of recovering each source input from a CV sample should enforce the model to build expressive representations as well as support the discrimination between class-specific characteristics. Moreover, the risk of finding just trivial solutions (cf. [4], [5], [7], [10], [23]) by learning one universal-class capsule or replicating entity representations over class capsules mitigates with the CV scheme, because reconstruction quality and model efficiency should improve by spreading dissimilar entity representations over the available class capsules. Subfigure **(a)** in Fig. 4 gives a visual explanation for the expected effects of using the CV reconstruction loss on the latent representation space. The effects comprise the building of an expressive latent feature space with similar samples gathered in clusters $\mathcal{C}_i$ represented as class capsules.

4) CAPSULE ACTIVATION LOSS

For providing an entanglement between entity representations and capsule activations, a *soft-classification* criterion is proposed. The soft-classification criterion comprises the creation of pseudo labels with the aid of the used sampling distribution for CV data generation and the indices of the capsules which produce the minimal-distant reconstructions for the k source inputs. Fig. 5 visualizes the creation of a target distribution for a CV sample with k source components

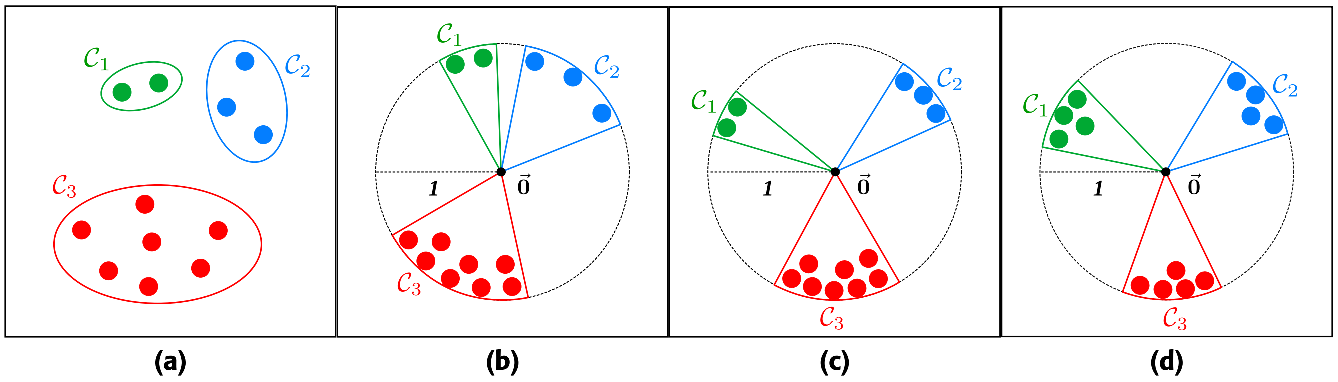

FIGURE 4. Visual explanation for the intentional effects on CVL's clustering behavior in the latent feature space based on the composition loss and the regularizations proposed in the mathematical framework. Beginning with (a) the reconstruction loss and successively adding (b) the activation loss, (c) the clustering and similarity loss, and (d) the rank-based sparsity boosting.

and m class capsules. For each of the k source inputs the capsule with the minimal-distant reconstruction is identified and associated with the probability α_i of the corresponding source input. Finally, the k pseudo labels, represented as one-hot vectors multiplied with the corresponding α-values, get added to constitute a target distribution for a CV sample. Similar to other deep clustering approaches [12], [23], CVL steers model optimization by means of high-confidence predictions, but specifically defined over reconstruction quality. Note that a single capsule can achieve the best recovery for an arbitrary number of source components. This concession is obligatory because in the general case we have no *a priori* knowledge about the inherent class memberships within an unknown data collection. In particular, we want to prevent human-based biases [23], [32] in representation learning by omitting external intervention. The logical process for the generation of a target distribution can be mathematically described with the loss function

$$\begin{aligned}\mathcal{L}_A &= -\lambda_A \mathbb{E}_{\tilde{X}_B} \mathbb{E}_{\sim p_{\hat{Y}}} [\log\ \mathrm{softmax}(\mathbf{a})] \\ &= -\lambda_A \mathbb{E}_{\tilde{X}_B} \sum_{c=1}^{m} p_{\hat{Y}}(c|\tilde{\mathbf{x}}_i) \log\ \mathrm{softmax}(\mathbf{a})_c, \end{aligned} \tag{11}$$

$$p_{\hat{Y}}(c|\tilde{\mathbf{x}}_i) = \sum_{j=1}^{k} \mathrm{attr}(c, \mathbf{r}_i^{(j)}), \tag{12}$$

$$\mathrm{attr}(c, \mathbf{r}_i^{(j)}) = \begin{cases} p_M(j) & \text{if index}(\mathbf{r}_i^{(j)}) = c \\ 0 & \text{otherwise} \end{cases}, \tag{13}$$

where $\mathcal{L}_A$ simply calculates the cross entropy between the estimated target distribution $p_{\hat{Y}}$ and the current capsule activations $\mathbf{a}$, which are represented as the second norm of the capsule vectors. Note that the entropy of $p_{\hat{Y}}$ behaves nonstationary over different training iterations. In (11) the application of the softmax function guarantees a valid probability distribution over the capsule activations with *independent* logits in the value range of $[0, 1]$. The help function $\mathrm{attr}(c, \mathbf{r}_i^{(j)})$ attributes the c-th capsule a target probability depending on the present minimal-distant reconstructions, which equals the row-wise summation for a specific element in Fig. 5. The entanglement of entity representations with

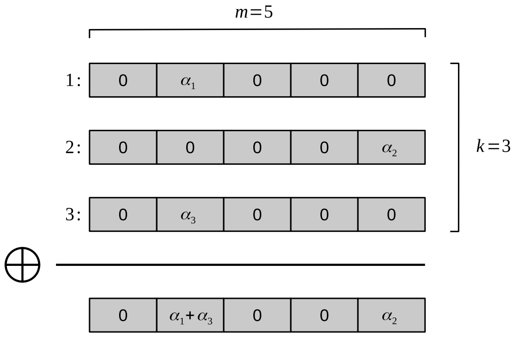

FIGURE 5. Logical process of determining the target distribution for capsule activations referring to a CV sample consisting of 3 source inputs and a CapsNet encoder with 5 class capsules. Attribution of probability values α_i is based on class capsule reconstruction quality.

capsule activations especially supports biological plausibility since a low activation of an entity detector implies incoherent instantiation parameter constellation (cf. [13]). As illustrated in subfigure **(b)** of Fig. 4, the activation loss pushes reliable capsule instantiation vectors in the direction of the surface of a unit hypersphere whereas inconsistent instantiation vectors are pulled towards the origin of the latent feature space, forming sections in the unit hypersphere belonging to specific capsules. Despite not explicitly prohibited, it is unlikely that a single capsule produces multiple sections with significant margin because the reconstruction loss forces the creation of an equivariant latent feature space in relation to the original sample space. Organizing reliable latent representations on the surface of a unit hypersphere is rather conventional in the case of CapsNets, induced by the normalization of capsule vector magnitudes to be in range of $[0, 1]$ and their interpretation as class probabilities. Interestingly, Shiran et al.'s [7] non-capsular deep clustering approach as well manages clusters on the surface of a unit hypersphere, but with the distinction that each embedding constitutes a unit vector.

5) INTER-SUBSPACE CLUSTERING REGULARIZER

An essential property of CVL constitutes the instantiation of an incoming sample as member of each cluster, represented as class capsule vectors. The quality of cluster-specific instantiation, which can be quantified using the individual reconstruction losses, heavily depends on the class relevance of sample features and the degree of class-capsule specialization. For instance, in the case of MNIST potentially shared features between digits comprise line thickness, digit skewness and digit size, whereas the existence and position of loops characterize distinct digits. We identify the *compactness* of entity representations per cluster as quality attribute for class-capsule differentiation. In terms of clustering, the compactness objective corresponds to the reduction of within-cluster variances. Regarding vector geometry, compactness depicts the average angle margin of instantiation vectors per capsule. To support the CVL model in forming compact clusters, we modify an implicit clustering objective originated from previous deep clustering approaches [2], [12]:

$$\mathcal{L}_C = \lambda_C \sum_{i=1}^{|\tilde{X}_B|} \sum_{c=1}^{m} p_{ic} \log \frac{p_{ic}}{q_{ic}}, \tag{14}$$

$$q_{ij} = \frac{(1 + d(\mathbf{c}_j(\tilde{\mathbf{x}}_i), \boldsymbol{\mu}_j))^{-1}}{\sum_{c=1}^{m} (1 + d(\mathbf{c}_c(\tilde{\mathbf{x}}_i), \boldsymbol{\mu}_c))^{-1}}, \tag{15}$$

$$p_{ij} = \frac{f_{ij}}{\sum_{c=1}^{m} f_{ic}}, \quad f_{ij} = \frac{q_{ij}^2}{\sum_{i=1}^{|\tilde{X}_B|} q_{ij}}. \tag{16}$$

The clustering loss in (14) minimizes the Kullback-Leibler divergence $D_{KL}(P||Q)$ between the soft-assignment distribution Q and the pseudo-target distribution P. The soft-assignment distribution is computed over the distance between a capsule vector $\mathbf{c}_j$ for the CV sample $\tilde{\mathbf{x}}_i$ and its cluster prototype $\boldsymbol{\mu}_j$ normalized with the heavy-tailed Student's t-distribution for modeling long-range distances. During our experiments we found that using the batch-wise prototype for the i-th capsule

$$\boldsymbol{\mu}_i = \mathbb{E}_{a_i(\tilde{\mathbf{x}})}[\mathbf{c}_i(\tilde{\mathbf{x}})], \quad \tilde{\mathbf{x}} \in \tilde{X}_B, \tag{17}$$

promotes the learning of differentiated entity representations distributed over the available class capsules. For establishing the distance function in (15) we adopt the cosine similarity

$$s(\mathbf{c}_i, \mathbf{c}_j) = \frac{\langle \mathbf{c}_i, \mathbf{c}_j \rangle}{||\mathbf{c}_i||_2 \cdot ||\mathbf{c}_j||_2} = \frac{\langle \mathbf{c}_i, \mathbf{c}_j \rangle}{a_i a_j} \in [-1, 1] \tag{18}$$

where a value of zero denotes orthogonality between two vectors and an absolute value of one signals collinearity with same or reversed vector orientations. We define the distance measure as negated cosine similarity shifted and scaled to values between zero and one, i.e.

$$d(\mathbf{c}_i, \mathbf{c}_j) = \frac{1 - s(\mathbf{c}_i, \mathbf{c}_j)}{2} \in [0, 1]. \tag{19}$$

The pseudo-target distribution in (16) reinforces the model's own high-confidence predictions [12]. An important adjustment of the implicit clustering objective compared to the previous approaches [2], [12] constitutes the batch-wise estimation of cluster prototypes instead of using trainable weights. Furthermore, the modified clustering objective interacts between the latent subspaces spanned by each class capsule. As displayed in subfigure **(c)** in Fig. 4, the designed clustering regularization loss aims to tighten the sections within the unit hypersphere corresponding to capsules' entity subspaces.

6) CAPSULE SIMILARITY REGULARIZER

Besides the gain of cluster compactness, another essential property for improving clustering quality corresponds to the increasing of between-cluster variances. For this purpose, we apply the cosine similarity in (18) on all possible class capsule pairs as regularization loss

$$\mathcal{L}_S = \lambda_S \sum_{i \neq j} s(\boldsymbol{\mu}_i, \boldsymbol{\mu}_j). \tag{20}$$

For computational simplicity, we calculate in our implementation the similarity matrix between all capsule pairs and determine the arithmetic mean over the matrix elements. Potential loss scaling issues can be compensated with the balancing factor λ_S. Regularizing capsule similarity mitigates co-adaption and breaks network symmetry. Specifically, the degenerative behavior of embracing different entity types in a few class-universal capsules should become inefficient for model optimization. The visual explanation in subfigure **(c)** of Fig. 4 emphasizes the maximization of the margin between clusters as the effect of the capsule similarity loss.

7) RANK-BASED SPARSITY BOOSTING

An early work [15] about unsupervised learning with CapsNets provided insights into the importance of sparsity constraints to retain the beneficial properties of CapsNets. More precisely, CapsNets without an appropriate restriction on individual capsule usage, e.g. based on a supervised task, harbor the risk to collapse into non-capsular networks [15]. In fact, the directed optimization using classification labels constrains an activation distribution on the class capsules pre-defined by the amount of samples per class within the training set. A logical choice for clustering, without prior knowledge about the considered data, represents a uniform activation distribution constraint on the class capsules. Inspired by [15] we propose a rank-based boosting mechanism to influence the average class capsule usage during model training. Specifically, we multiply the boost coefficient

$$g_i = 1 - \text{softmax}(\mathbf{b})_i + \epsilon \tag{21}$$

with the MSE in (8) for the i-th capsule, where $\mathbf{b}$ corresponds to the boost logits for all class capsules and the summand ϵ equals a small constant which prevents zero-coefficients. Therefore, the selection of the *winning* capsule for producing the best reconstruction for a certain source input is influenced by all boost logits. Algorithm 1 summarizes the proceeding for adjusting one boost logit b_i per training batch. The passed arguments consist of the previous boost logit value b_i, the average capsule usage represented as the mean rank $\bar{r}_i$, the

Algorithm 1 Batch-Wise Update for the Rank-Based Boost Logit b_i of the i-Th Class Capsule

```
BOOST-LOGIT($b_i$; $\bar{r}_i$; $m$; $\beta = 0.05$)

# target mean-rank
$t_i \leftarrow 1/m$

# distance-proportional logit adjustment
$\delta_i \leftarrow t_i - \bar{r}_i$
$b_i \leftarrow b_i + \mathrm{sign}(\delta_i) \cdot \max(|\delta_i| - \beta t_i, 0)$

# restrict value range to $[0, 1]$
$b_i \leftarrow \max(b_i, 0)$
$b_i \leftarrow \min(b_i, 1)$
return $b_i$
```

number of class capsules m and a fixed deviation tolerance value β. The mean rank $\bar{r}_i$ for the i-th class capsule over the CV batch $\tilde{X}_B$ is calculated as

$$\bar{r}_i = \mathbb{E}_{\tilde{X}_B}\left[\sum_{j=1}^{k} \alpha_j \cdot \mathrm{wins}(\mathbf{R}_j, i)\right], \quad \mathbf{R} \in \mathbb{R}^{k \times m \times d}, \tag{22}$$

where the function wins($\mathbf{R}_j$, i) counts the number of times the i-th capsule produces the lowest (boosted) reconstruction error for the j-th source sample, and the tensor $\mathbf{R}$ contains the reconstructions of the m class capsules for all k sources. In Algorithm 1 the target mean-rank t_i reflects the assumed uniform prior for the classes in an unknown data collection. The boost logit b_i gets proportionally adjusted to the distance between the target rank t_i and the current rank $\bar{r}_i$ if the absolute distance exceeds the tolerance value βt_i. Finally, boost logits are bounded to be in range of $[0, 1]$ which prohibits negative values as well as exploding logits. We set the deviation tolerance factor β to 5% of the target rank for preventing leakage modifications. The rank-based boosting causes a balancing effect on the samples assigned to the class capsules which results in a sparsity constraint on average capsule usage during the training of the CVL model. The balancing effect on cluster sizes is depicted in subfigure **(d)** from Fig. 4.

IV. EXPERIMENTS

A. MNIST DATASET

The *MNIST* [16], [17] classification dataset comprises a collection of grayscale images from handwritten digits in a resolution of 28×28 pixels. The MNIST image collection is approximately balanced by having circa $7K$ samples per class. MNIST consists of a training set with $60K$ samples and a test set with $10K$ samples. The dataset version provided in *TensorFlow Datasets* [33] is utilized. We normalize the pixel intensities from the image data to be in range of $[0, 1]$. The CV dataset $\tilde{X}$ is generated using the training set X_{Train} of MNIST with preservation of the original set size.

B. GLOBAL SETUP

In our experiments we use the original *dynamic routing* [13] procedure as routing-by-agreement implementation between capsule layers from the CapsNet encoder. Dynamic routing applies a nonlinear *squash* [13] function which scales the length of the resulting capsule vector to be in range of $[0, 1]$ while preserving vector orientation. This allows the direct interpretation of each class capsule activation as probability value for class recognition. Since the proposed mathematical framework estimates cluster prototypes and mean ranks over a training batch, the batch size must be chosen sufficiently large to produce stable statistics. We found a training batch size of $|\tilde{X}_B| = 128$ to be appropriate for the MNIST dataset with its ten digit classes. In addition, we equally weight all partial loss functions, i.e. $\lambda_i = 1.0$. Note that the introduced scaling factors λ_i can generally prioritize clustering objectives, which can be helpful if clustering task conditions vary such as the number of class capsules. Each boost logit b_i for the rank-based sparsity boosting is initialized with a uniform distribution which samples values between zero and one. The implementation of this paper utilizes the numerical and machine learning programming library *TensorFlow* [34]. In particular, we use the *Adam* [35] optimizer with an exponentially decaying learning rate, starting with $\eta = 0.001$ and reducing with a decay rate of 0.95 after two epochs without significant improvement with respect to the total loss. In each analysis, CVL is applied for five runs if not otherwise stated. Each run conducts 50 epochs. Since the CVL procedure behaves non-deterministic over different runs due to the probabilistic nature of CVS and random model weight initialization, we expect slight variation in the experimental results with the same CVL and CVS configuration.

C. CVL-MODEL ARCHITECTURE

Fig. 6 displays the designed architecture for the CapsNet encoder and the non-capsular decoder network in the CVL model for MNIST. Both sub-networks are straightforwardly created without extensive hyperparameter testing or tuning. The CapsNet encoder and decoder network are composed of two-dimensional convolutional layers (Conv2D)s and fully-connected layers. The respective kernel sizes, number of filters and stride sizes per convolutional layer can be deduced from the network visualization. All convolutional layers use same padding between layers and the Rectified Linear Unit (ReLU) [36] activation function as nonlinearity. In the CapsNet encoder an ordinary convolutional layer acts as introducing capsule layer through the rearrangement of its scalar-output neurons into distributed entity representations. The initial capsule layer is usually denoted as *Primary Capsules (PrimaryCaps)* [13] and serves as entry point for further capsule layers. Building up PrimaryCaps with the aid of a convolutional layer with scalar-output neurons is a common strategy to organize a transition from a regular neural network to a CapsNet [13], [14], [30].

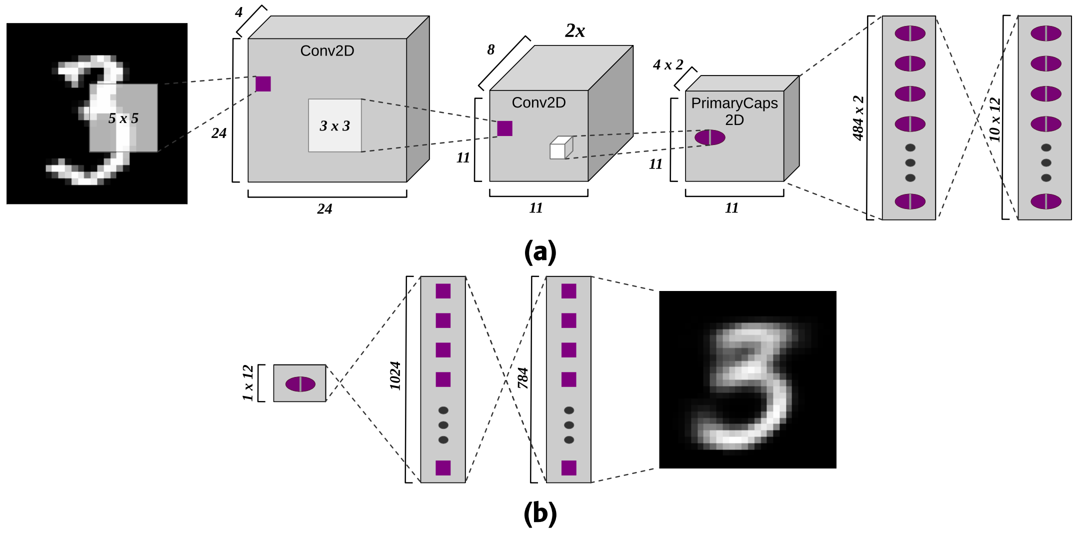

FIGURE 6. **Network architectures of (a) the CapsNet encoder and (b) the decoder network for all experiments with the MNIST dataset.**

Convolutional PrimaryCaps layers are specified by the number of *capsule types* [13], [14], which are equivalent to filters in ordinary convolutional layers, and the defined capsule dimensionality. Capsule dimensionalities are stated as multiplicator supplement. The decoder network receives as input one class capsule vector per processing step in order to map an entity embedding to the original data space. The decoder network solely consists of two fully-connected layers, where the first layer applies ReLU and the second one the sigmoid function as nonlinearity which guarantees features in the same value range as the preprocessed input data. Our decoder network is oriented to Sabour et al.'s [13] reconstruction network for MNIST, but we omit one layer and slightly reduce the dimensionality of the class capsule vectors from 16 to 12. In our experiments we will show that this reduced network architecture is sufficient for the clustering task with MNIST. Previous deep clustering approaches [2], [12] successfully clustered the MNIST data in a latent feature spaces with 10 dimensions. Hence, the chosen dimensionality of 12 should equip each class capsule with enough representational capacity to potentially capture the whole latent feature space. This eliminates the possibility of introducing an hidden architectural regularization on the class capsule usage.

D. CLUSTERING PERFORMANCE

In our first experiment, we holistically investigate the performance properties resulting from a CVL model trained on MNIST. For this purpose, we generate a CV dataset using CVS based on a Multinoulli distribution parameterized with $\boldsymbol{\alpha} = (0.7 \quad 0.3)^T$. Fig. 7 illustrates the gain of *clustering accuracy (ACC)* [37], which calculates regular accuracy on the optimal mapping between clustering predictions and the classification labels, on the full test set from MNIST after each training epoch. In the first plot predictions are determined via hard-assignment cluster memberships using the highest class capsule activation, whereas the second plot fulfills cluster assignments with respect to the minimal-distant reconstruction. The best model corresponds to the model with the lowest total loss after completing the 50 training epochs. Evidently, the lowest total loss is consistent with the highest clustering accuracy. However, we observed that the total losses of all models only minimally differ, although the clustering accuracies show stronger dispersion. Thus, in concrete applications the decision criteria for the best model may vary. Both clustering accuracy progressions are coherent after a few epochs, proving the successful entanglement of capsule activation with reconstruction ability. The resulting accuracy value of 63.83% for activation-directed and 62.93% for reconstruction-directed predictions with respect to the best model even demonstrate a slight dominance for cluster assignments based on class capsule activation. This observation is also supported by the respective mean accuracy values of 58.97% and 57.03% over all runs. Fig. 8 provides evidence about the effectiveness of the rank-based sparsity boosting on balancing mean class capsule usage over the training process. The boost logits in subfigure **(a)** continuously regulate individual reconstruction ability per class capsule in order to satisfy the expectation of an equal capsule usage on average, as shown in subfigure **(b)**. More precisely, in the case of the ten classes from the MNIST dataset the expectation means a 10% usage of each class capsule over the entire training set. Similar to [15] the rank-based sparsity boosting is only applied during CV training with reconstruction ability as the class-discriminative criterion. In the subsequent sections we further investigate the clustering performance properties using the identified best model.

1) CLASS-ACTIVATION HISTOGRAMS

After training the CVL model from Fig. 6 on the CV dataset from MNIST, we obtain distributions of class-capsule

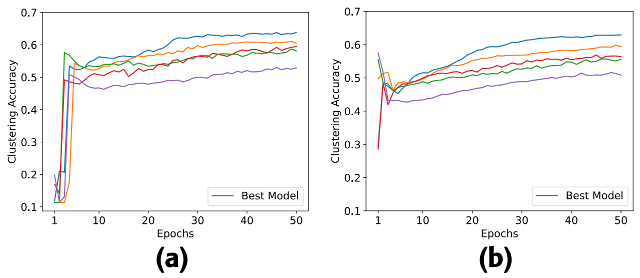

FIGURE 7. Gain of clustering accuracy on the test set from MNIST during model training. Predictions are obtained by (a) the highest capsule activation and (b) the minimal-distant reconstruction. The best model refers to the model with the lowest total loss after 50 epochs.

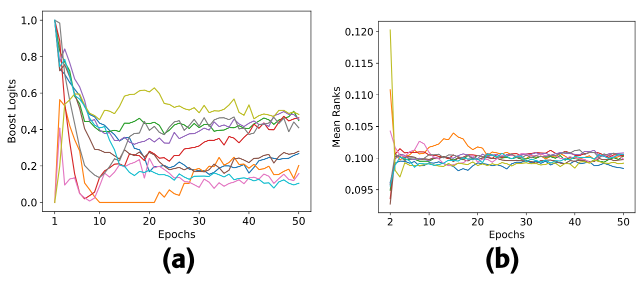

FIGURE 8. (a) Progression of the boost logits of all class capsules during model training. (b) Corresponding mean rank per class capsule tracked as moving statistics to remove temporary fluctuations. The plot starts with the second epoch to focus on the mean ranks development after the first deviations caused by the random intialization of the boost logits.

activations that reveal data-intrinsic similarities and differences between instances from the human-defined categories. Fig. 9 illustrates histrograms about the average activation for each class capsule per predefined digit class. The capsule activations are received from the application of the CapsNet encoder on the test set of MNIST. Note that the capsule activations per sample are not constrained to constitute a valid probability distribution to obviate deformation effects on distribution shapes. One can recognize the two subsets $\{3, 5, 8\}$ and $\{4, 7, 9\}$ with similar appearance in their distribution characteristics originated from the activation of the same class capsules. This circumstance implies an inherent proximity between the digits per subset and their constructive features from a data-oriented perspective. Furthermore, this observation is congruent with human expectation because digits in the same subset share many visual features (e.g. circle-like structures, proportions or distinct vertical lines). In fact, the crucial distinctive feature between the digits 3 and 5 emerges as the position of one vertical line in the upper part. The digits 3 and 8 have a special visual relation, too, roughly speaking closing both open half-circles of a 3 results in an 8. Analogously, the digits 4, 7 and 9 are grounded on significantly related features, of course, depending on the concrete writing style of a person. An important observation is that for almost each digit a specific class capsule dominates with highest activation on average. Evidently, the data-intrinsic classes, or *clusters* speaking in terms of unsupervised learning, identified by CVL are mainly consistent with the human-defined categories. This behavior is natural since humans must be able to efficiently discriminate between digits, as well. Despite the interpretation of hard cluster assignments based on highest capsule activations, soft assignments can be obtained by normalizing the activation distribution. Thus, each average capsule activation quantifies the fraction a digit belongs to a class capsule from a *global* view on the data. In opposite to this, a *local* view can be examined by propagating a single data sample through the trained CapsNet encoder.

2) CLASS-CAPSULE PROTOTYPES

Class-activation histograms concentrate on the reasoning about data-intrinsic relationships between predefined classes from labeled datasets. However, clustering tasks rarely offer any predefined class labels. For this reason we want to inspect in this section the entity clusters represented by the class capsules via their prototypes. We state that from the output of the CapsNet encoder for a given sample $\mathbf{x}$ the entity representation $\mathbf{c}_i$ and the activation a_i for the i-th capsule can be extracted, i.e.

$$\mathbf{c}_i(\mathbf{x}), a_i(\mathbf{x}) \leftarrow \text{CEnc}(\mathbf{x})_i. \tag{23}$$

Based on this simplified notation the formula for the hard-assignment prototype $\boldsymbol{\mu}_i^{(H)}$ of the i-th capsule reduces to

$$\boldsymbol{\mu}_i^{(H)} = \mathbb{E}_{\mathcal{C}_i}\left[\mathbf{c}_i(\mathbf{x})\right], \tag{24}$$

$$\mathcal{C}_i = \{\mathbf{x} \mid \mathbf{x} \in X_B \subseteq X,\ i = \underset{j\in\{0,..,m-1\}}{\arg\max}\ a_j(\mathbf{x})\}, \tag{25}$$

where X_B equals a representative subset of the original dataset and the cluster $\mathcal{C}_i$ comprises all data points $\mathbf{x}$ for which the i-th capsule induces highest activation. On the other hand, soft-assignment prototypes $\boldsymbol{\mu}_i^{(S)}$ are the same way defined as in (17) but computed over a subset X_B from the original data:

$$\boldsymbol{\mu}_i^{(S)} = \mathbb{E}_{a_i(\mathbf{x})}\left[\mathbf{c}_i(\mathbf{x})\right], \quad \mathbf{x} \in X_B \subseteq X. \tag{26}$$

To be consistent with the class-activation histograms from our previous considerations, all prototypes are determined using the test set of MNIST (i.e. $X_B = X_{Test}$). Therefore, the prototypes can be immediately evaluated in the context to the class-activation histograms from Fig. 9. Since (24) and (26) produce prototypes as weighted latent entity representations, the prototypes must be subsequently processed by the associated decoder to ensure visual interpretability. The hard and soft-assignment prototypes for MNIST are respectively visualized in the subfigures **(a)** and **(b)** from Fig. 10. The hard-assignment prototypes confirm our conjecture that clusters, represented as class capsules, are particularly specialized on single digits. This situation especially holds for the digits 0 to 3 and 6 to 8 which are clearly recognizable in the reconstructed prototypes. Interestingly, the digits 4 and 9 seem to be merged into the first two prototypes which are rather distinguished by instance skewness than class-discriminative properties from a human perspective. This observation agrees with the highly-activated first two

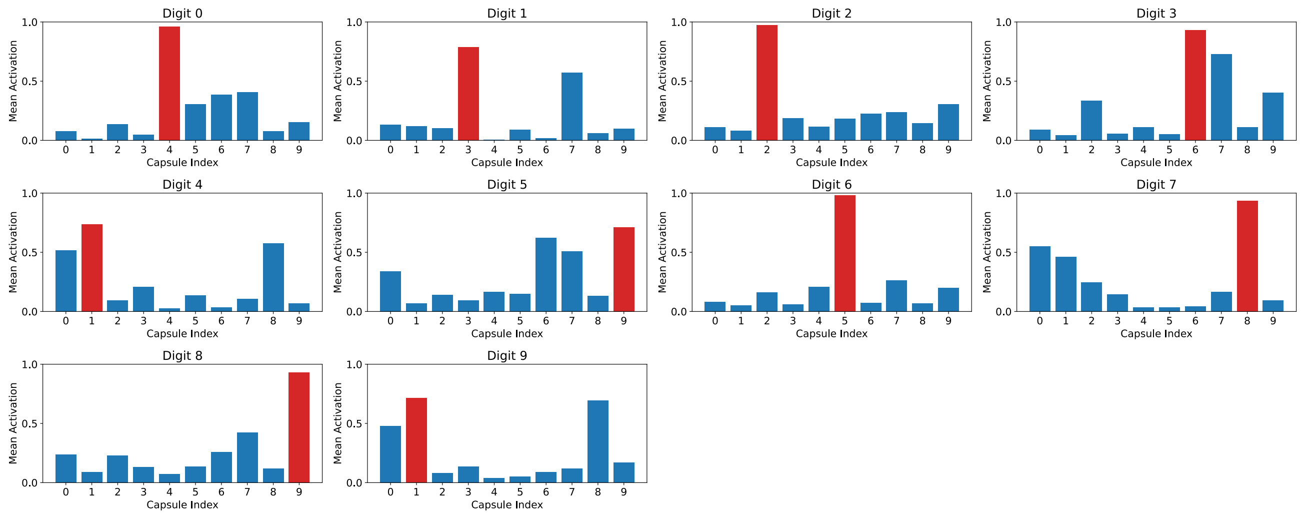
FIGURE 9. Mean capsule activation for each predefined class in the test set from MNIST. The highest-reached average activation is highlighted in red color per subplot. The class capsules tend to specialize on distinct digits. Between-class correlations can be detected for the digit subsets {3, 5, 8} and {4, 7, 9}.

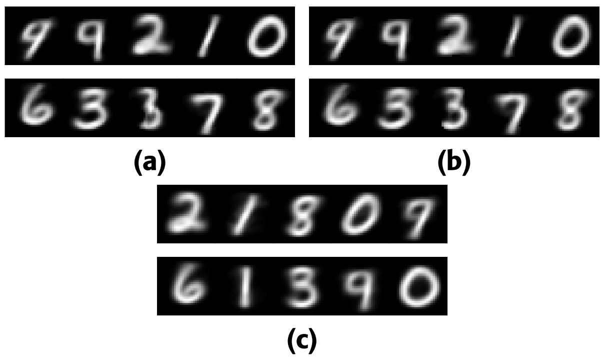
FIGURE 10. Decoded prototypes of the class capsules with (a) hard and (b) soft assignments, computed over the encodings of all images from the test set of MNIST. (c) Hard-assignment prototypes over the test set of MNIST based on the clustering result from k-means.

capsules in the corresponding class-activation histograms. Moreover, the third hard-assignment prototype in the bottom row encapsulates a mixture of instances from multiple digits presumably caused by dominant common features like digit squeeze. Finally, the soft-assignment prototypes display solely marginal uncertainties in the class-specific entity representations.

3) CLASS-DISCRIMINATIVE EQUIVARIANCE

One main objective in the training of CapsNets represents the creation of a latent feature space which satisfies the equivariance property. Equivariance describes the continuous modeling of data-centered variations in distributed entity representations (cf. [15]). A capsule forming an equivariant instantiation parameter vector, which locally captures the variations associated with its observed entity type, permits invariant entity recognition to these variations [29]. Obviously, the kind of equivariant features strongly depends on the domain of the considered dataset. In the image domain, for instance, equivariant features appear among others as viewpoints, lightning conditions, localization, scaling and deformation [13], [14], [29], in contrast textual entities may vary in sentiment occurrences and intensities or specific grammar structures (despite a lack of intuitive explainability) [30]. In each case equivariant modeling requires generative capabilities which generally emerge as interpolative and extrapolative function approximation of intrinsic concepts to encompass punctual variations in the regarded data. As an example, imagine a generative model that learns a continuous translation of a visual object from the left to the right image border based on several discrete observations with occurring differences in object location. Thus, equivariance can be understood as superior quality of a model to *explain* natural variations as latent concepts originated from the data generation process.

The appearance of equivariance is also determined by task-specific conditions such as the design of the learning procedure. In particular, the learning procedure steers the model's attention on shaping equivariant features with high relevance for accomplishing the defined task. The choice of equivariant features can be restricted with supervisory signals or completely leaved to model optimization as frequently observed with traditional autoencoders. A special characteristic of CVL constitutes the creation of a class-discriminative equivariant space in an unsupervised learning setting. Fig. 11 visualizes the model's generated continuous transitions from each class capsule prototype to all other prototypes. The prototypes correspond to the soft-assignment prototypes in (26) which are illustrated in subfigure **(b)** of Fig. 10. Each row in Fig. 11 displays a smooth transformation from the first prototype into the second one by means of ten discrete intermediate representations. In more detail, the transformation is directed by the difference vector between both latent prototypes which is divided into ten equally-sized steps. At each step the decoder component computes the reconstruction of the intermediate latent representation.

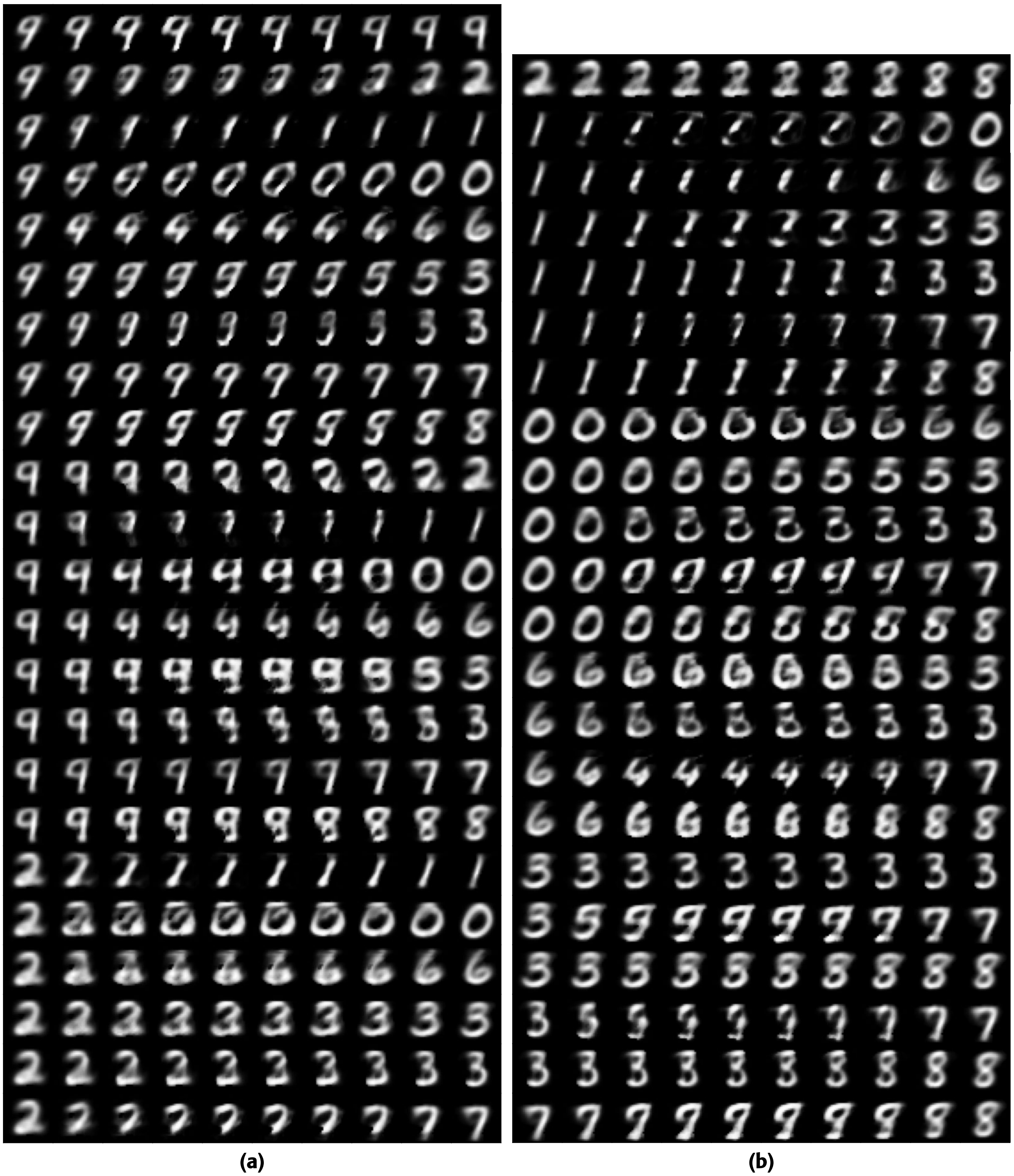

FIGURE 11. The CVL paradigm with its reconstruction-centered scheme enforces the learning of an equivariant latent space with generative continuous transitions between data-intrinsic classes. (a) & (b) display in each row a transition between two cluster prototypes from the CVL model trained on MNIST.

We observe that the regarded model produces equivariant features with the aim to improve class discrimination as implicitly demanded by the CVL paradigm. In that sense the CVL model interprets natural variances in the data in terms of class discrimination as well as derives continuous class transitions without direct evidence given by the data. This

TABLE 1. Quantitative comparison between the clusterings obtained from CVL, k-means and the classification labels.

	ARI	H	C	V-M	ACC	VRC	DBI
CVL	**<u>0.51</u>**	**<u>0.62</u>**	**<u>0.61</u>**	**<u>0.61</u>**	**<u>63.83%</u>**	336	3.10
k-means	0.37	0.50	0.49	0.50	51.74%	**<u>391</u>**	**<u>2.86</u>**
Labels			–			311	3.81

circumstance reflects the both characteristics of interpolative and extrapolative function approximation, respectively. For instance, one can characterize stroke thickness (e.g. in **(a)** the 7-th row from the top), digit skewness (e.g. in **(a)** the 6-th row from the bottom), digit size (e.g. in **(b)** the 8-th row from the bottom) etc. as interpolation between observed samples in the dataset. In opposite to this, extrapolation appears in the form of purposive deformation between the recognized classes (e.g. in **(a)** the 5-th row from the bottom or in **(b)** the 2-nd row from the top).

Although Sabour et al. [13] previously reported similar equivariant features (e.g. stroke thickness and digit skewness) in the context of a supervised learning task with CapsNets, they primarily used the reconstruction loss to support entity representation learning as addition to the classification loss. This means a crucial difference to our CVL approach which enforces the learning of equivariant features for the purpose of effective class discrimination quantified via reconstruction ability. Note that in CVL each class capsule constructs its own locally equivariant subspace associated with its observed entity type and all these subspaces are joined in the superior latent feature space of the decoder network. In consequence of this, we can linearly navigate through this continuous superior space to manually inspect the model's inference mechanism, as emphasized in Fig. 11.

E. REFERENCE CLUSTERING WITH SIMPLE K-MEANS

After investigating the special characteristics of CVL, we further examine in this section its clustering performance with diverse clustering validity measures. For better interpretability we also provide a reference clustering with the prominent and non-neural clustering algorithm *k-means*[1] [38], [39] which can be formulated as:

$$\boldsymbol{\mu}_i^{(K)} = \mathbb{E}_{\mathcal{C}_i^{(K)}}[\mathbf{x}], \tag{27}$$

$$\mathcal{C}_i^{(K)} = \{\mathbf{x} \mid \mathbf{x} \in X_B \subseteq X,\ i = \underset{j \in \{0,..,m-1\}}{\arg\min} \ ||\boldsymbol{\mu}_j^{(K)} - \mathbf{x}||_2^2\}, \tag{28}$$

where the prototype $\boldsymbol{\mu}_i^{(K)}$ for the i-th cluster is defined as the arithmetic mean over the associated data points, a cluster $\mathcal{C}_i^{(K)}$ consists of all data points nearest to its prototype $\boldsymbol{\mu}_i^{(K)}$ and m corresponds to the predefined number of clusters. Evidently, using the squared Euclidean distance as measure for point-to-cluster similarity leads to spherical clusters. The selected hyperparameters for k-means comprise: 1) The number of clusters m equals the number of classes from MNIST. 2) The initial prototypes correspond to random points from the regarded data. 3) k-means is conducted for 100 runs to receive the most frequently occurring clustering result. 4) Maximal 100 iterations are performed per run.

1) QUALITATIVE EVALUATION

For ensuring comparability, the k-means algorithm processes the training set from MNIST (i.e. $X_B = X_{Train}$) to estimate cluster prototypes. After the determination of the cluster prototypes, each data point in the test set gets assigned to its nearest cluster prototype, according to (28). Finally, we calculate the arithmetic mean of the data points within each cluster in order to receive cluster representatives. The cluster representatives for the test set from MNIST are displayed in subfigure **(c)** of Fig. 10. In comparison to the decoded hard and soft-assignment prototypes of CVL in the subfigures **(a)** and **(b)** we can recognize several prototypes with nearly identical appearance. However, in contrast to CVL k-means distinguishes instances from the digits 0 and 1 by their skewness. This behavior can be seen as an indicator for k-means' missing ability to grasp constructive features of the same instrinsic class.

2) QUANTITATIVE EVALUATION

The quantification of clustering validity is established with the measures *Adjusted Rand Index (ARI)* [40], *Homogenity (H)* [41], *Completeness (C)* [41], *V-Measure (V-M)* [41], *Variance Ratio Criterion (VRC)* [42] and *Davies-Bouldin Index (DBI)* [43].[1] The first four evaluate clustering quality in reference to the labels of MNIST as ground truth whereas the latter two quantify clustering quality based on criteria for cluster structure. Additionally, we state the clustering accuracy using the classification labels from MNIST. The ARI extends the *Rand Index (RI)* [44] by *correcting* its clustering validity measure against random choices for point-to-cluster assignments, which results in values in the range of $[-1, 1]$ with higher positive values for stronger concordance between two partitions [40]. The RI and, consistently, the ARI define the similarity between two clusterings as their agreement on pairwise point-to-cluster memberships [44]. Homogenity describes the degree of certainty of class members within clusters whereas completeness captures the compactness of class members over clusters [41]. The V-M summarizes homogenity and completeness as (equally) weighted harmonic mean [41]. The VRC describes the quality of a partition as the ratio over the average of between-cluster and within-cluster dispersion [42]. The DBI measures the mean similarity over each cluster to its minimal-distant cluster [43].

Table 1 summarizes the results of the quantitative analysis. The best reached values for each validity measure are highlighted in bold and underlined. The results with respect to the classification labels show the expected superiority of CVL for identifying and modeling data-intrinsic classes. In particular, the poor values for VRC and DBI resulting from CVL and the classification labels reveal the necessity

TABLE 2. Effects of diverse Multinoulli distributions in CVL on the resulting clustering accuracy.

Multinoulli Distribution α^T	Clustering Accuracy (in %) by Activation		Reconstruction	
	Best	AVG	Best	AVG
$(0.9 \quad 0.1)$	**<u>72.58</u>**	57.94 ± 9.0	**<u>71.12</u>**	56.84 ± 8.4
$(0.7 \quad 0.3)^*$	63.83	**<u>58.97</u>** ± 3.6	62.93	**<u>57.03</u>** ± 4.0
$(0.5 \quad 0.5)$	54.24	46.09 ± 5.3	53.43	46.47 ± 5.3
$(0.5 \quad 0.3 \quad 0.2)$	50.96	49.92 ± 3.9	53.08	49.75 ± 4.6
$(0.33)^{1\times3}$	11.35	27.95 ± 14.0	13.24	32.42 ± 11.5

of applying the clustering operation within or by forming an advantageous latent space. The stated results for clustering performance can be used for comparative considerations in future work.

F. SAMPLING FROM DIFFERENT DISTRIBUTIONS

In our last experiment we investigate the implications on CVL's performance using various Multinoulli distributions. The number k of source inputs as well as their contribution to a CV sample determine the severity for the task of class discrimination. We vary both of these factors for the CVL model in Fig. 6. As stated in the global setup each configuration is conducted for five runs. Table 2 summarizes the achieved best and average clustering accuracies (with standard deviations) for the tested Multinoulli distributions. Furthermore, the clustering accuracies are divided into predictions received by the highest capsule activation and the minimal-distant reconstruction. The best accuracy per column is emphasized in bold and underlined. The configuration marked with the asterisk reuses the models obtained from our former experiments. Again, the best model per configuration is chosen by the smallest total loss during training.

The resulting clustering accuracies allow several observations. Firstly, the results for both activation-based and reconstruction-based predictions are mostly on par with a slight advance of activation-based predictions for the Multinoulli distributions with two components. This situation demonstrates the effectiveness of the proposed capsule activation loss on replacing the reconstruction ability as class-discrimination criterion during training with capsule activation during inference. In addition, the models using Multinoulli distributions with two components appear superior to the configurations with three components. Specifically, the uniform distribution in the last row leads to models with performance near chance. It is likely that three source components produce CV samples with a destructive degree of interferences. This problem is even intensified if each source input equally contributes to the CV samples. In fact, the preference of a single source input expressed with a high probability value for feature selection seems to be beneficial for model optimization. This circumstance can be validated with the configurations $\alpha^T = (0.9 \quad 0.1)$ and $\alpha^T = (0.7 \quad 0.3)$ where the dominance of the first source component leads to higher clustering accuracies compared to the Multinoulli distribution parameterized with $\alpha^T = (0.5 \quad 0.5)$. We further examined the reasons for this performance discrepancy by manually inspecting the class-activation histograms of the best model per configuration. We found that the models below the second row successively fail to distribute entity representations over the available class capsules. These observations imply that keeping CV sample distortion adequate, by using only two source components as well as preferring one source, improves the specialization of class capsules.

Although the first configuration delivers the best model over all runs according to its clustering accuracy, the second configuration reaches the best mean results with a smaller standard deviation. By exploring the best model of the first configuration, we found that its CVL model leaves the cluster of one class capsule almost empty but the encoder learned as default mapping a representative of digit 5, resulting in the highest overall clustering accuracy. Since clustering tasks in general do not provide any classification labels, we would typically reject clusterings with empty clusters at first. For that reason we denominate the use of the Multinoulli distribution with $\alpha^T = (0.7 \quad 0.3)$ as preferable default configuration in terms of average model robustness. Note that this insight differentiates a CVL model from a regular denoising autoencoder because the class-discriminative features of two data instances constitute a CV sample, instead of just adding some noise as in the first configuration.

V. DISCUSSION

1) HIGH-CONFIDENCE PREDICTIONS

A special characteristic of CVL, which it shares with some other deep clustering techniques [5], [12], [23], depicts the strategy to steer parameter optimization by the models' own high-confidence predictions. Specifically, Xie et al. [12] stated that learning objectives guided by a model's high-confidence predictions can be categorized as *self-training* [45], where approving model confidence should lower model uncertainty. This adaptable target definition could possibly be helpful for overcoming local optima during model training, by iteratively changing the perspective on the considered data [45].

2) RELATION TO MIXUP

The data augmentation method *mixup* [46] follows the optimization principle *Vicinal Risk Minimization (VRM)* [47], instead of the commonly used *Empirical Risk Minimization* [48], by using convex combinations of random pairs of samples and their respective labels in the form of one-hot vectors for establishing linear interpolations between samples on the latent decision surface. Recent works [49], [50] provide theoretical foundations about mixup's data dependency as well as its effects on model robustness and generalization ability. Besides, there exist several variants of mixup such as *CutMix* [51] which randomly inserts a sample patch to another sample, or *Puzzle Mix* [52] and

Co-Mixup [53] which additionally exploit saliency information for enhancing their sample mixture procedures.

Interestingly, the experimental investigation of CVL implies a linear-interpolative behavior between original samples in the learned representation space (cf. Fig. 11) as intended in the classic mixup method, though, CVL is originated from an entirely nonlinear process. In fact, CVL can also be characterized as VRM algorithm by structuring its nonlinear process into three phases: 1) CVS conducts a nonlinear mapping of the cartesian product of at least two original samples $\mathbf{x}_i$ to constitute a CV sample $\tilde{\mathbf{x}}$ (i.e. $\mathbb{R}^d \times \cdots \times \mathbb{R}^d \longmapsto \mathbb{R}^d$) as probabilistic value selection per feature dimension. Hence, CVS estimates the underlying true distribution $P(\mathbf{x})$ with the vicinal distribution

$$P_v(\tilde{\mathbf{x}}) = \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{k}\mathcal{D}_v(\tilde{\mathbf{x}}|\mathbf{x}_i^{(j)}) \tag{29}$$

where $\mathcal{D}_v$ describes the probability distribution of assigning the CV sample $\tilde{\mathbf{x}}$ to the source input $\mathbf{x}_i^{(j)}$. 2) Class capsules learn to reverse the CVS procedure by segregating CV samples into their source inputs and predicting the missing parts. Since each class capsule takes the responsibility for a special object class, the prediction distribution $P_p(\mathbf{x}, y)$ containing reconstructions and cluster memberships is implicitly derived from the vicinal distribution $P_v(\tilde{\mathbf{x}})$. 3) Finally, the prediction distribution $P_p(\mathbf{x}, y)$ gets punctually evaluated on the empirical distribution $P_e(\mathbf{x}, \hat{y})$ with self-supervised pseudo labels $\hat{y}_i$ entangled with reconstruction quality. This whole process is supported by the auxiliary regularizations proposed in the mathematical framework of CVL which further restrict the function solution space for optimization.

In opposite to mixup, CVS ignores structural dependence between feature dimensions by rather forming a component-wise contrast than interpolating between original samples. This means that while mixup directly provides examples of linear interpolations, CVS forces a model to predict the representational vicinity based on fragmented variations between samples from arbitrary classes, by finding a suitable function set to extrapolate and interpolate around discrete observations of the empirical data distribution. Therefore, we speculate that CVS integrated in CVL possibly represents an even more general data augmentation method which may enclose the function set of linear interpolations from mixup. Note that the mixup variants CutMix, Puzzle Mix and Co-Mixup also rather create a contrast than interpolating between training samples, but these variants particularly retain contiguous structural information. Since CVL does not require predefined labels compared to mixup and its prominent variants, it also resolves potential dependencies on classification datasets. Nevertheless, a systematic comparison between CVS, CVL, mixup and its variants is desirable in future work to provide theoretical and empirical analyses for improving the understanding of their latent relationships.

TABLE 3. Chronological evolution of clustering accuracies for non-capsular deep clustering methods on the test set from MNIST.

Reference	Year	Method	ACC
Wang *et al.* [8]	2015	TAGnet-MML	69.22%
Xie *et al.* [12]	2016	DEC	84.30%
Yang *et al.* [4]	2017	DCN	83.00%
Guo *et al.* [2]	2017	DCEC	85.29%
Dizaji *et al.* [3]	2017	DEPICT	96.30%
Yang *et al.* [6]	2019	Dual Autoencoder + Deep Spectral Clustering	98.00%
Shiran *et al.* [7]	2020	MMDC	99.10%

3) MODEL ROBUSTNESS

CVL demonstrated stable performance with relatively small variance over five runs using the Multinoulli distribution with $\boldsymbol{\alpha}^T = (0.7 \quad 0.3)$. In addition, the loss balancing hyperparameters λ_i were explicitly not tuned on satisfying the requirements on the MNIST dataset to obtain CVL's plain clustering performance. But the possibility for prioritizing the individual loss functions means a gain in flexibility for clustering with CVL, in general. The rank-based sparsity boosting increases model robustness in terms of orchestrating a fair class capsule usage during training without introducing a hard constraint. In particular, the sparsity boosting breaks network symmetry with its random initialization and causes a stabilizing effect on class capsule specialization. The key argument for the robustness of CVL models results from the class-discriminative equivariance property which ensures the creation of an expressive latent feature space with a focus on the transformation between data-intrinsic classes. This generative behavior allows a continuous evaluation of internal representations and accounts for an enhanced inference mechanism for determining cluster assignments. Despite CVL needs a predefined number of clusters, sufficient prior domain knowledge is often available to choose an appropriate amount of clusters [23]. Contrary to previous deep clustering approaches [3], [6], [7], [12], [23] that explicitly augment data samples to strengthen the robustness of representation learning, CVL owns built-in data augmentation within CVS by randomly drawing features from a selection of samples. Finally, the concrete implementation of CVS and the reconstruction criterion for class discrimination are easily adaptable to other application domains.

4) EXPLAINABILITY

As known from other clustering techniques, CVL's clustering results can be investigated by means of hard and soft-assignment prototypes. Since CVL constitutes a neuralized clustering method, prototypes are calculated as weighted mean over the latent representations of their cluster members. The generative nature of CVL models ensures a continuous mapping from latent representations into the original data space using the decoder network. From a human perspective the explainability of decoded cluster prototypes is only restricted to the interpretability of the original data space.

In cases of labeled datasets, inspecting cluster prototypes is a complementary approach to the informative class-activation histograms resulting from the use of CVL. The essential gain in explainability using CVL as clustering algorithm represents the class-discriminative equivariance property which explains interpolative and extrapolative relationships in the unsupervised model with human-observable, continuous transitions between data-intrinsic classes. The generative equivariant space is fully accessible by linearly navigating over the latent appearance manifold. Moreover, an arbitrary latent representation can be mapped into the original space by the decoder and propagated through the CapsNet encoder to receive cluster membership predictions as class capsule activations.

5) CLUSTERING ACCURACY

Despite the convincing qualitative results of CVL's clustering with its class-discriminative equivariant latent space, the reached clustering accuracy is currently not competitive to state-of-the-art methods for non-capsular deep clustering, as illustrated in Table 3. Additionally, Table 3 conveys an impression about the chronological gain in clustering accuracies until the deep clustering task on the MNIST dataset is approximately solved. In fact, the various loss functions in the mathematical framework of CVL emphasize the needed effort to accomplish a class-discriminative specialization of output capsules in an unsupervised learning setting. In that sense our work represents a first attempt to open the promising CapsNet architecture with its full capacity for the task of deep clustering. We hope that our CVL approach motivates future research to further improve clustering accuracies on diverse datasets with distinct application domains.

VI. CONCLUSION

This paper introduced the novel learning paradigm CVL for unsupervised representation learning in the form of deep clustering using CapsNets. A CVL model corresponds to an asymmetric autoencoder, comprising a CapsNet encoder and a non-capsular decoder network, to detect and represent data-intrinsic classes through the output capsules of the encoder. The conducted experiments demonstrate CVL's performance on the MNIST dataset from various perspectives. As special property, a CVL model accesses its generative ability to construct an equivariant latent feature space with continuous transitions between data-intrinsic classes. This behavior reveals the potential of CapsNets for deep clustering to create a robust and explainable inference mechanism. Future research efforts are desirable to strengthen the theory about the requirements for securing CapsNet stability as well as approaches to increase the resulting clustering accuracy.

ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewer 1 for the helpful comment about including a consideration of the data augmentation method mixup in relation to their proposed learning paradigm.

REFERENCES

[1] J. R. Hershey, Z. Chen, J. Le Roux, and S. Watanabe, "Deep clustering: Discriminative embeddings for segmentation and separation," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Mar. 2016, pp. 31–35.

[2] X. Guo, X. Liu, E. Zhu, and J. Yin, "Deep clustering with convolutional autoencoders," in *Proc. Int. Conf. Neural Inf. Process. (ICONIP)*, 2017, pp. 373–382.

[3] K. G. Dizaji, A. Herandi, C. Deng, W. Cai, and H. Huang, "Deep clustering via joint convolutional autoencoder embedding and relative entropy minimization," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 5747–5756.

[4] B. Yang, X. Fu, N. D. Sidiropoulos, and M. Hong, "Towards K-means-friendly spaces: Simultaneous deep learning and clustering," in *Proc. 34th Int. Conf. Mach. Learn. (ICML)*, vol. 70, 2017, pp. 3861–3870.

[5] M. Caron, P. Bojanowski, A. Joulin, and M. Douze, "Deep clustering for unsupervised learning of visual features," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 139–156.

[6] X. Yang, C. Deng, F. Zheng, J. Yan, and W. Liu, "Deep spectral clustering using dual autoencoder network," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 4061–4070.

[7] G. Shiran and D. Weinshall, "Multi-modal deep clustering: Unsupervised partitioning of images," in *Proc. 25th Int. Conf. Pattern Recognit. (ICPR)*, Jan. 2020, pp. 4728–4735.

[8] Z. Wang, S. Chang, J. Zhou, M. Wang, and T. S. Huang, "Learning a task-specific deep architecture for clustering," 2015, *arXiv:1509.00151v3*.

[9] X. Yang, C. Deng, K. Wei, J. Yan, and W. Liu, "Adversarial learning for robust deep clustering," in *Proc. 34th Conf. Neural Inf. Process. Syst. (NeurIPS)*, 2020, pp. 1–11.

[10] A. Ashfahani and M. Pratama, "Unsupervised continual learning in streaming environments," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Apr. 13, 2022, doi: 10.1109/TNNLS.2022.3163362.

[11] J. Kauffmann, M. Esders, L. Ruff, G. Montavon, W. Samek, and K.-R. Müller, "From clustering to cluster explanations via neural networks," 2021, *arXiv:1906.07633v2*.

[12] J. Xie, R. Girshick, and A. Farhadi, "Unsupervised deep embedding for clustering analysis," in *Proc. 33rd Int. Conf. Mach. Learn. (ICML)*, 2016, pp. 1–10.

[13] S. Sabour, N. Frosst, and G. E. Hinton, "Dynamic routing between capsules," in *Proc. 31st Conf. Neural Inf. Process. Syst. (NIPS)*, 2017, pp. 1–11.

[14] G. Hinton, S. Sabour, and N. Frosst, "Matrix capsules with em routing," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2018, pp. 1–15.

[15] D. Rawlinson, A. Ahmed, and G. Kowadlo, "Sparse unsupervised capsules generalize better," 2018, *arXiv:1804.06094*.

[16] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[17] Y. LeCun, C. Cortes, and C. J. C. Burges. (2010). *The MNIST Database of Handwritten Digits*. [Online]. Available: http://yann.lecun.com/exdb/mnist

[18] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, and J. Vanderplas, "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, no. 85, pp. 2825–2830, 2011.

[19] G. Hinton and S. Roweis, "Stochastic neighbor embedding," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, 2002, pp. 1–8.

[20] L. van der Maaten and G. Hinton, "Visualizing data using t-SNE," *J. Mach. Learn. Res.*, vol. 9, pp. 2579–2605, Nov. 2008.

[21] F. Tian, B. Gao, Q. Cui, E. Chen, and T.-Y. Liu, "Learning deep representations for graph clustering," in *Proc. 28th AAAI Conf. Artif. Intell.*, 2014, vol. 28, no. 1, pp. 1293–1299.

[22] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, Jul. 2006.

[23] W. Van Gansbeke, S. Vandenhende, S. Georgoulis, M. Proesmans, and L. Van Gool, "SCAN: Learning to classify images without labels," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2020, pp. 268–285.

[24] J. Yang, D. Parikh, and D. Batra, "Joint unsupervised learning of deep representations and image clusters," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 5147–5156.

[25] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *J. Roy. Stat. Soc., Ser. B*, vol. 39, no. 1, pp. 1–22, Sep. 1977.

[26] A. R. Kosiorek, S. Sabour, Y. W. Teh, and G. E. Hinton, "Stacked capsule autoencoders," in *Proc. 33rd Conf. Neural Inf. Process. Syst. (NeurIPS)*, 2019, pp. 1–11.

[27] Z. Lin, Y. Li, Z. Huang, W. Zhang, Y. Tan, Y. Chen, and Q. He, "Domestic activities clustering from audio recordings using convolutional capsule autoencoder network," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Jun. 2021, pp. 835–839.

[28] A. Hyvärinen and H. Morioka, "Unsupervised feature extraction by time-contrastive learning and nonlinear ICA," in *Proc. 30th Conf. Neural Inf. Process. Syst. (NIPS)*, 2016, pp. 1–9.

[29] G. E. Hinton, A. Krizhevsky, and S. D. Wang, "Transforming auto-encoders," in *Proc. Int. Conf. Artif. Neural Netw. (ICANN)*, 2011, pp. 44–51.

[30] N. A. K. Steur and F. Schwenker, "Next-generation neural networks: Capsule networks with routing-by-agreement for text classification," *IEEE Access*, vol. 9, pp. 125269–125299, 2021.

[31] W. Zhao, J. Ye, M. Yang, Z. Lei, S. Zhang, and Z. Zhao, "Investigating capsule networks with dynamic routing for text classification," in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, 2018, pp. 3110–3119.

[32] I. Misra, C. L. Zitnick, M. Mitchell, and R. Girshick, "Seeing through the human reporting bias: Visual classifiers from noisy human-centric labels," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 2930–2939.

[33] *TensorFlow Datasets: A Collection of Ready-to-Use Datasets*. Accessed: Jun. 21, 2023. [Online]. Available: https://www.tensorflow.org/datasets

[34] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, and S. Ghemawat, "TensorFlow: Large-scale machine learning on heterogeneous distributed systems," 2016, *arXiv:1603.04467v2*.

[35] D. P. Kingma and J. Lei Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2015, pp. 1–15.

[36] V. Nair and G. E. Hinton, "Rectified linear units improve restricted Boltzmann machines," in *Proc. 27th Int. Conf. Mach. Learn. (ICML)*, 2010, pp. 1–8.

[37] Y. Yang, D. Xu, F. Nie, S. Yan, and Y. Zhuang, "Image clustering using local discriminant models and global integration," *IEEE Trans. Image Process.*, vol. 19, no. 10, pp. 2761–2773, Oct. 2010.

[38] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proc. 5th Berkeley Symp. Math. Statist. Probab.*, 1967, vol. 5, no. 1, pp. 281–297.

[39] S. P. Lloyd, "Least squares quantization in PCM," *IEEE Trans. Inf. Theory*, vol. IT-28, no. 2, pp. 129–137, Mar. 1982.

[40] L. Hubert and P. Arabie, "Comparing partitions," *J. Classification*, vol. 2, no. 1, pp. 193–218, Dec. 1985.

[41] A. Rosenberg and J. Hirschberg, "V-measure: A conditional entropy-based external cluster evaluation measure," in *Proc. Joint Conf. Empirical Methods Natural Lang. Process. Comput. Natural Lang. Learn. (EMNLP-CoNLL)*, 2007, pp. 410–420.

[42] T. Calinski and J. Harabasz, "A dendrite method for cluster analysis," *Commun. Statist.*, vol. 3, no. 1, pp. 1–27, 1974.

[43] D. L. Davies and D. W. Bouldin, "A cluster separation measure," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-1, no. 2, pp. 224–227, Apr. 1979.

[44] W. M. Rand, "Objective criteria for the evaluation of clustering methods," *J. Amer. Stat. Assoc.*, vol. 66, no. 336, pp. 846–850, Dec. 1971.

[45] K. Nigam and R. Ghani, "Analyzing the effectiveness and applicability of co-training," in *Proc. 9th Int. Conf. Inf. Knowl. Manag. (CIKM)*, Nov. 2000, pp. 86–93.

[46] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "mixup: Beyond empirical risk minimization," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2018, pp. 1–13.

[47] O. Chapelle, J. Weston, L. Bottou, and V. Vapnik, "Vicinal risk minimization," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, vol. 13, 2000, pp. 1–7.

[48] V. N. Vapnik, "An overview of statistical learning theory," *IEEE Trans. Neural Netw.*, vol. 10, no. 5, pp. 988–999, Sep. 1999.

[49] L. Zhang, Z. Deng, K. Kawaguchi, A. Ghorbani, and J. Zou, "How does mixup help with robustness and generalization?" in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2021, pp. 1–12.

[50] M. Chidambaram, X. Wang, Y. Hu, C. Wu, and R. Ge, "Towards understanding the data dependency of mixup-style training," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2022, pp. 1–12.

[51] S. Yun, D. Han, S. J. Oh, S. Chun, J. Choe, and Y. Yoo, "CutMix: Regularization strategy to train strong classifiers with localizable features," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 6022–6031.

[52] J.-H. Kim, W. Choo, and H. O. Song, "Puzzle mix: Exploiting saliency and local statistics for optimal mixup," in *Proc. 37th Int. Conf. Mach. Learn. (ICML)*, 2020, pp. 5275–5285.

[53] J.-H. Kim, W. Choo, H. Jeong, and H. O. Song, "Co-mixup: Saliency guided joint mixup with supermodular diversity," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2021, pp. 1–12.

NIKOLAI A. K. STEUR received the B.Sc. degree in computer science from Baden-Württemberg Cooperative State University, Mosbach, Germany, in 2018, and the M.Sc. degree with distinction in computer science from Ulm University, Germany, in 2021, where he is currently a doctoral student with the Institute of Neural Information Processing.

His research interests include advanced theory and methods for data science, machine learning, and artificial intelligence. In particular, he is interested in the state-of-the-art development of artificial neural networks in various application domains.

Mr. Steur was awarded for the Best Presentation from the Conference Committee at ICACTE, in 2018.

FRIEDHELM SCHWENKER (Member, IEEE) received the Diploma and Ph.D. degrees from Osnabrück University.

He is currently a Professor in computer science with the Institute of Neural Information Processing, Ulm University. He has (co-)edited more than 20 special issues and workshop proceedings published in international journals and publishing companies, and published more than 250 papers at international conferences and journals. His research interests include artificial neural networks, machine learning, statistical learning theory, data mining, pattern recognition, information fusion, and affective computing.

Dr. Schwenker served as the (Co-)Chair for the IAPR TC3 on Neural Networks and Computational Intelligence. From 2016 to 2020, he was the chair. He founded the IAPR TC9 on Pattern Recognition in human–computer interaction.