ZERO-SHOT SYNTHESIS WITH GROUP-SUPERVISED LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Visual cognition of primates is superior to that of artificial neural networks in its ability to "envision" a visual object, even a newly-introduced one, in different *attributes* including pose, position, color, texture, etc. To aid neural networks to envision objects with different attributes, we propose a family of objective functions, expressed on *groups of examples*, as a novel learning framework that we term Group-Supervised Learning (GSL). GSL allows us to decompose inputs into a disentangled representation with swappable components, that can be recombined to synthesize new samples. For instance, images of *red boats* & *blue cars* can be decomposed and recombined to synthesize novel images of *red cars*. We propose an implementation based on auto-encoder, termed group-supervised zero-shot synthesis network (GZS-Net) trained with our learning framework, that can produce a high-quality *red car* even if no such example is witnessed during training. We test our model and learning framework on existing benchmarks, in addition to a new dataset that we open-source. We qualitatively and quantitatively demonstrate that GZS-Net trained with GSL outperforms state-of-the-art methods.

1 INTRODUCTION

Primates perform well at generalization tasks. If presented with a single visual instance of an object, they often immediately can generalize and envision the object in different attributes, e.g., in different 3D pose (Logothetis et al., 1995). Primates can readily do so, as their previous knowledge allows them to be cognizant of attributes. Machines, by contrast, are most-commonly trained on sample features (e.g., pixels), not taking into consideration attributes that gave rise to those features.

To aid machine cognition of visual object attributes, a class of algorithms focuses on *learning disentangled representations* (Kingma & Welling, 2014; Higgins et al., 2017; Burgess et al., 2018; Kim & Mnih, 2018; Chen et al., 2018), which map visual samples onto a latent space that separates the information belonging to different attributes. These methods show disentanglement by interpolating between attribute values (e.g., interpolate pose, etc). However, these methods usually process one sample at a time, rather than contrasting or reasoning about a group of samples. We posit that semantic links across samples could lead to better learning.

We are motivated by the visual generalization of primates. We seek a method that can synthesize realistic images for arbitrary queries (e.g., a particular car, in a given pose, on a given background), which we refer to as *controlled synthesis*. We design a method that enforces semantic consistency of attributes, facilitating controlled synthesis by leveraging semantic links between samples. Our method maps samples onto a disentangled latent representation space that (i) consists of subspaces, each encoding one attribute (e.g., identity, pose, ...), and, (ii) is such that two visual samples that share an attribute value (e.g., both have identity "car") have identical latent values in the shared attribute subspace (identity), even if other attribute values (e.g., pose) differ. To achieve this, we propose a general learning framework: *Group Supervised Learning* (GSL, Sec. 3), which provides a learner (e.g., neural network) with groups of semantically-related training examples, represented as *multigraph*. Given a query of attributes, GSL proposes groups of training examples with attribute combinations that are useful for synthesizing a test example satisfying the query (Fig. 1). This endows the network with an envisioning capability. In addition to applications in graphics, controlled synthesis can also augment training sets for better generalization on machine learning tasks (Sec. 5.4). As an instantiation of GSL, we propose an encoder-decoder network for zero-shot synthesis: *Group-*

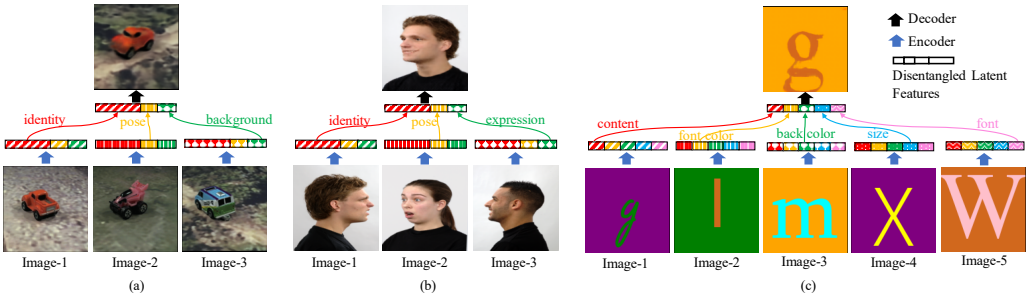


Figure 1: Zero-shot synthesis performance of our method. (a), (b), and (c) are from datasets, respectively, iLab-20M, RaFD, and Fonts. Bottom: training images (attributes are known). Top: Test image (attributes are a *query*). Train images go through an encoder, their latent features get combined, passed into decoder, to synthesize the requested image. Sec. 4.1 shows how we disentangle the latent space, with explicit latent feature swap during training.

Supervised Zero-Shot Synthesis Network (GZS-Net, Sec. 4). While learning (Sec. 4.2), it repeatedly draws a group of semantically-related examples, as informed by a multigraph created by GSL: It encodes group examples, to obtain latent vectors, then *swaps* entries for one or more attributes in the latent space *across examples*, through multigraph edges, then decodes into an example within the group (Sec. 4.1).

Our contributions are: (i) We propose Group-Supervised Learning (GSL), explain how it casts its admissible datasets into a multigraph, and show how it can be used to express learning from semantically-related groups and to synthesize samples with controllable attributes; (ii) We show one instantiation of GSL: Group-supervised Zero-shot Synthesis Network (GZS-Net), trained on groups of examples and reconstruction objectives; (iii) We demonstrate that GZS-Net trained with GSL outperforms state-of-the-art alternatives for controllable image synthesis on existing datasets; (iv) We provide a new dataset, Fonts[1], with its generating code. It contains 1.56 million images and their attributes. Its simplicity allows rapid idea prototyping for learning disentangled representations.

2 Related Work

We review research areas, that share similarities with our work, to position our contribution.

Self-Supervised Learning (e.g., Gidaris et al. (2018)) admits a dataset containing features of training samples (e.g., upright images) and maps it onto an auxiliary task (e.g., rotated images): dataset examples are drawn and a random transformation (e.g., rotate 90°) is applied to each. The task could be to predict the transformation (e.g., =90°) from the transformed features (e.g., rotated image). Our approach is similar, in that it also creates auxiliary tasks, however, the tasks we create involve semantically-related *group of examples*, rather than from one example at a time.

Disentangled Representation Learning are methods that infer *latent factors* given example visible features, under a generative assumption that each latent factor is responsible for generating one semantic attribute (e.g. color). Following Variational Autoencoders (VAEs, Kingma & Welling, 2014), a class of models (including, Higgins et al., 2017; Chen et al., 2018) achieve disentanglement *implicitly*, by incorporating into the objective, a distance measure e.g. KL-divergence, encouraging the latent factors to be statistically-independent. While these methods can disentangle the factors *without* knowing them beforehand, unfortunately, they are unable to generate novel combinations not witnessed during training (e.g., generating images of *red car*, without any in training). On the other hand, our method *requires* knowing the semantic relationships between samples (e.g., which objects are of same identity and/or color), but can then synthesize novel combinations (e.g., by stitching latent features of "any car" plus "any red object").

Conditional synthesis methods can synthesize a sample (e.g., image), using information external to the synthesized modalities, e.g., natural language sentence Zhang et al. (2017); Hong et al. (2018) or class label Mirza & Osindero (2014); Tran et al. (2017). Ours differ, in that our "external information"

[1] `http://anonymous/url/for/blind/review`

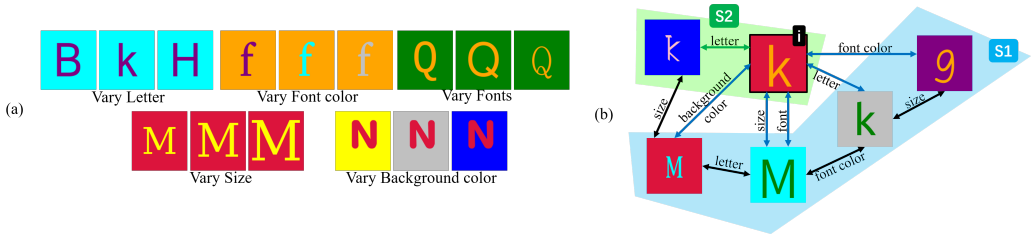


Figure 2: (a) Samples from our proposed Fonts dataset, shown in groups. In each group, we vary one attribute but keep others the same. (b) (Sub-)multigraph of our Fonts dataset. Each edge connect two examples sharing an attribute. Sets S_1 and S_2 *cover* sample i.

takes the form of semantic relationships between samples. Regardless, these methods are based on Generative Adversarial Networks (Goodfellow et al., 2014), which are known for their unstable learning including mode-collapse, and many efforts are devoted to alleviating these instabilities. Our learning framework, on the other-hand, allows expressing much simpler architectures, such as feed-forward auto-encoders trained with only reconstuction objectives. We find that the Motion Re-targeting (Yang et al., 2020) is similar to ours, however, it is domain-specific and requires hand-engineering to detect and track human body parts. On the other hand, we design and apply our method on different tasks (including people faces, vehicles, fonts; see Fig. 1).

Zero-shot learning also consumes side-information. For instance, models of Lampert (2009); Atzmon & Chechik (2018) learn from object attributes, like our method. However, (i) these models are supervised to accurately predict attributes, (ii) they train and infer one example at a time, and (iii) they are concerned with classifying unseen objects. We differ in that (i) no learning gradients (supervision signal) are derived from the attributes, as (ii) these attributes are used to group the examples (based on shared attribute values), and (iii) we are concerned with generation rather than classification: we want to synthesize an object in previously-unseen attribute combinations.

Graph Neural Networks (GNNs) (Scarselli et al., 2009) are a class of models described on graph structured data. This is applicable to our method, as we propose to create a multigraph connecting training samples. In fact, our method can be described as a GNN, with *message passing* functions (Gilmer et al., 2017) that are aware of the latent space partitioning per attribute (explained in Sec. 4). Nonetheless, for self-containment, we introduce our method in the absence of the GNN framework.

3 Group-Supervised Learning

3.1 Datasets admissible by GSL

Formally, a dataset admissible by GSL containing n samples $\mathcal{D} = \{x^{(i)}\}_{i=1}^n$ where each example is accompanied with m attributes $\mathcal{D}_a = \{(a_1^{(i)}, a_2^{(i)}, \dots a_m^{(i)})\}_{i=1}^n$. Each attribute value is a member of a countable set: $a_j \in \mathcal{A}_j$. For instance, pertaining to visual scenes, $\mathcal{A}_1$ can denote foreground-colors $\mathcal{A}_1 = \{\text{red}, \text{yellow}, \dots\}$, $\mathcal{A}_2$ could denote background colors, $\mathcal{A}_3$ could correspond to foreground identity, $\mathcal{A}_4$ to (quantized) orientation. Such datasets have appeared in literature, e.g. in Borji et al. (2016); Matthey et al. (2017); Langner et al. (2010); Lai et al. (2011).

3.2 Auxiliary tasks via Multigraphs

Given a dataset of n samples and their attributes, we define a multigraph M with node set $[1..n]$. Two nodes, $i, k \in [1..n]$ with $i \neq k$ are connected with edge labels $M(i,k) \subseteq [1..m]$ as:

$$M(i,k) = \left\{ j \,\middle|\, a_j^{(i)} = a_j^{(k)}; j \in [1..m] \right\}.$$

In particular, M defines a multigraph, with $|M(i,k)|$ denoting the number of edges connecting nodes i and k, which is equals the number of their shared attributes. Fig. 2 depicts a (sub-)multigraph for the Fonts dataset (Sec. A).

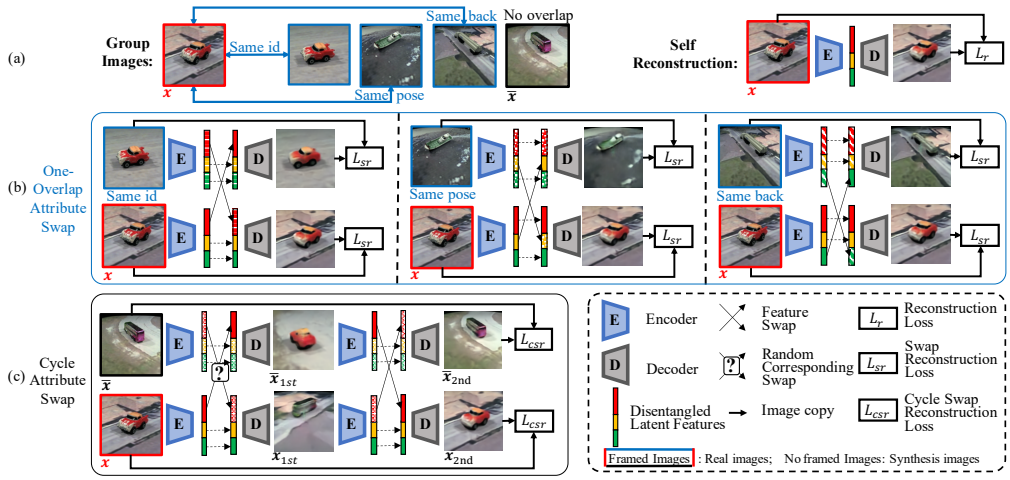


Figure 3: Architecture of GZS-Net, consisting of an encoder E: maps sample onto latent vector, and a decoder D: maps latent vector onto sample. The latent space is pre-partitioned among the attribute classes (3 shown: *identity*, *pose*, *background*). (a, left) considered examples: a center image (x, red border) and 3 images sharing one attribute with it, as well as a *no overlap* image sharing no attributes ($\bar{x}$, black border). (a, right) standard reconstruction loss, applied for all images. (b) **One-overlap attribute swap:** Two images with identical values for one attribute should be reconstructed into nearly the original images when the latent representations for that attribute are swapped ("no-op" swap; left: identity; middle: pose; right: background). (c) **Cycle swap**: given any example pair, we randomly pick an attribute class j. We encode both images, swap representations of j, decode, re-encode, swap on j again (to reverse the first swap), and decode to recover the inputs. This unsupervised cycle enforces that double-swap on j does not destroy information for other attributes.

Definition 1 COVER(S, i): *Given node set $S \subseteq [1..|\mathcal{D}_g|]$ and node $i \in [1..|\mathcal{D}_g|]$ we say set S covers node i if every attribute value of i is in at least one member of S. Formally:*

$$\text{COVER}(S, i) \iff [1..m] = \bigcup_{k \in S} M(i, k). \tag{1}$$

When COVER(S, i) holds, there are two mutually-exclusive cases: either $i \in S$, or $i \notin S$, respectively shaded as green and blue in Fig. 2 (b). The first case trivially holds even for small S, e.g. COVER$(\{i\}, i)$ holds for all i. However, we are interested in non-trivial sets where $|S| > 1$, as sets with $|S| = 1$ would cast our proposed network (Section 3) to a standard Auto-Encoder. The second case is crucial for zero-shot synthesis. Suppose the (image) features of node i (in Fig. 2 (b)) are not given, we can search for S_1, under the assumption that if COVER(S_1, i) holds, then S_1 contains sufficient information to synthesize i's features as they are not given ($i \notin S_1$).

Until this point, we made no assumptions how the pairs (S, i) are extracted (mined) from the multigraph s.t. COVER(S, i) holds. In the sequel, we restrict ourselves with $|S| = 2$ and $i \in S$. We find that this particular specialization of GSL is easy to program, and we leave-out analyzing the impact of mining different kinds of cover sets for future work.

4 GROUP-SUPERVISED ZERO-SHOT SYNTHESIS NETWORK

As an instantiation of GSL, we propose group-supervised zero-shot synthesis network (GZS-Net, Fig. 3). GZS-Net explicitly learns a disentangled representation by partitioning the latent space among the attribute classes. It *swaps attributes* before feature reconstruction, according to multigraph edges M, as we explain next. This disentanglement enables controlled synthesis by combining portions of latent space from various examples, to synthesize examples with unseen attribute combinations.

4.1 DISENTANGLEMENT BY EXPLICIT SWAPPING

As illustrated in Fig. 3, GZS-Net consists of an encoder $E : \mathcal{X} \to \mathbb{R}^d$, mapping an image sample $x^{(i)} \in \mathcal{X}$ to a latent vector $z^{(i)} = E(x^{(i)})$ and a decoder $D : \mathbb{R}^d \to \mathcal{X}$ that maps a latent vector, which

may be composed by combining the encodings of **one or more** images, back onto the image space. The key ingredient for training GZS-Net is *swapping attributes* by swapping corresponding entries in the latent space. Before any training, as we construct the network, we partition the latent space among the the m attribute classes. Let row-vector $z^{(1)} = [g_1^{(1)}, g_2^{(1)}, \ldots, g_m^{(1)}]$ be the concatenation of m row vectors $\{g_j^{(1)} \in \mathbb{R}^{d_j}\}_{j=1}^m$ where $d = \sum_{j=1}^m d_j$ and the values of $\{d_j\}_{j=1}^m$ are hyperparameters. To simplify the following notation, we define an operation **swap** : $\mathbb{R}^d \times \mathbb{R}^d \times [1..m] \to \mathbb{R}^d \times \mathbb{R}^d$, which accepts two latent vectors (e.g., $z^{(1)}$ and $z^{(2)}$) and an attribute (e.g., 2) and returns the input vectors except that the latent features corresponding to the attribute are swapped. E.g.,

$$\begin{aligned}\textbf{swap}(z^{(1)}, z^{(2)}, 2) &= \textbf{swap}([g_1^{(1)}, g_2^{(1)}, g_3^{(1)}, \ldots, g_m^{(1)}], [g_1^{(2)}, g_2^{(2)}, g_3^{(2)}, \ldots, g_m^{(2)}], 2) \\ &= [g_1^{(1)}, g_2^{(2)}, g_3^{(1)}, \ldots, g_m^{(1)}], [g_1^{(2)}, g_2^{(1)}, g_3^{(2)}, \ldots, g_m^{(2)}]\end{aligned}$$

One-Overlap Attribute Swap. To encourage disentanglement in the latent representation of attributes, we consider group S and example x s.t. $\textsc{Cover}(S, x)$ holds, and for all $x^o \in S, x \neq x^o$, the pair (x^o, x) share **exactly one** attribute value ($|M(x^o, x)| = 1$). Encoding those pairs, swapping the latent representation of the attribute, and decoding should then be a no-op if the swap did not affect other attributes (Fig. 3b). Specifically, we would like for a pair of examples, x (red border in Fig. 3b) and x^o (blue border) sharing only attribute j (e.g., identity)[2], with $z = E(x)$ and $z^o = E(x^o)$, be s.t.

$$D\left(z_s\right) \approx x \quad \text{and} \quad D\left(z_s^{(o)}\right) \approx x^{(o)}; \;\; \text{with} \;\; z_s, z_s^{(o)} = \textbf{swap}(z, z^o, j). \tag{2}$$

If, for each attribute, sufficient sample pairs share only that attribute, and Eq. 2 holds for all with zero residual loss, then disentanglement is achieved for that attribute (on the training set).

Cycle Attribute Swap. This operates on all example pairs, regardless of whether they share an attribute or not. Given two examples and their corresponding latent vectors, if we swap latent information corresponding to *any attribute*, we should end up with a sensible decoding. However, we may not have ground-truth supervision samples for swapping all attributes of all pairs. For instance, when swapping the *color attribute* between pair *orange truck* and *white airplane*, we would like to learn from this pair, even without any *orange airplanes* in the dataset. To train from any pair, we are motivated to follow a recipe similar to CycleGAN (Zhu et al., 2017). As shown in Fig. 3c, given two examples x and $\bar{x}$: (i) sample an attribute $j \sim \mathcal{U}[1..m]$; (ii) encode both examples, $z = E(x)$ and $\bar{z} = E(\bar{x})$; (iii) swap features corresponding to attribute j with $z_s, \bar{z}_s = \textbf{swap}(z, \bar{z}, j)$; (iv) decode, $\widehat{x} = D(z_s)$ and $\widehat{\bar{x}} = D(\bar{z}_s)$; (v) on a second round (hence, cycle), encode again as $\widehat{z} = E(\widehat{x})$ and $\widehat{\bar{z}} = E(\widehat{\bar{x}})$; (vi) another swap, which should reverse the first swap, $\widehat{z}_s, \widehat{\bar{z}}_s = \textbf{swap}(\widehat{z}, \widehat{\bar{z}}, j)$; (vii) finally, one last decoding which should approximately recover the original input pair, such that:

$$D\left(\widehat{z}_s\right) \approx x \quad \text{and} \quad D\left(\widehat{\bar{z}}_s\right) \approx \bar{x}; \tag{3}$$

If, after the two encode-swap-decode, we are able to recover the input images, regardless of which attribute we sample, this implies that swapping one attribute does not destroy latent information for other attributes. As shown in Sec. 5, this can be seen as a data augmentation, growing the effective training set size by adding all possible new attribute combinations not already in the training set.

4.2 Training and Optimization

Algorithm 1 lists our sampling strategy and calculates loss terms, which we combine into a total loss

$$\mathcal{L}(E, D; \mathcal{D}, M) = L_{\text{r}} + \lambda_{\text{sr}} L_{\text{sr}} + \lambda_{\text{csr}} L_{\text{csr}}, \tag{4}$$

where scalars $\lambda_{\text{sr}}, \lambda_{\text{csr}} > 0$ control the relative importance of the loss terms. The loss $\mathcal{L}$ can be minimized w.r.t. parameters of encoder (E) and decoder (D) via gradient descent.

5 Experiments

We qualitatively evaluate our method on zero-shot synthesis tasks, and on its ability to learn disentangled representations, on existing datasets (Sec. 5.2), as well as a novel dataset we contribute (Sec. A). Further, we quantitatively (i) evaluate disentanglement by calculating a model-based confusion matrix between attributes: can the latent features of one attribute predict other attributes (Sec. 5.3)?; and (ii)

[2]It holds that $\textsc{Cover}(\{x, x^o\}, x)$ and $\textsc{Cover}(\{x, x^o\}, x^o)$

Algorithm 1: Training Regime; for sampling data and calculating loss terms

Input: Dataset $\mathcal{D}$ and Multigraph M
Output: $L_{\text{r}}, L_{\text{sr}}, L_{\text{csr}}$

Sample $x \in \mathcal{D}, S \subset \mathcal{D}$ such that $\textsc{Cover}(S, x)$ and $|S| = m$ and $\forall k \in S, |M(x,k)| = 1$
for $x^{(o)} \in S$ **do**
  $z \leftarrow E(x);\ z^{(o)} \leftarrow E(x^{(o)});\ \left(z_s, z_s^{(o)}\right) \leftarrow \textbf{swap}(z, z^{(o)}, j)$
  $L_{\text{sr}} \leftarrow L_{\text{sr}} + ||D(z_s) - x||_{l_1} + \left|\left|D\left(z_s^{(o)}\right) - x^{(o)}\right|\right|_{l_1}$ # Swap reconstruction loss
$\bar{x} \sim \mathcal{D}$ and $j \sim \mathcal{U}[1..m]$ # Sample for Cycle swap
$z \leftarrow E(x);\ \bar{z} \leftarrow E(\bar{x});\ (z_s, \bar{z}_s) \leftarrow \textbf{swap}(z, \bar{z}, j);\ \widehat{x} \leftarrow D(z_s);\ \widehat{\bar{x}} \leftarrow D(\bar{z}_s)$
$\widehat{z} \leftarrow E(\widehat{x});\ \widehat{\bar{z}} \leftarrow E(\widehat{\bar{x}});\ (\widehat{z}_s, \widehat{\bar{z}}_s) \leftarrow \textbf{swap}(\widehat{z}, \widehat{\bar{z}}, j)$
$L_{\text{csr}} \leftarrow ||D(\widehat{z}_s) - x||_{l_1} + \left|\left|D\left(\widehat{\bar{z}}_s\right) - \bar{x}\right|\right|_{l_1}$ # Cycle reconstruction loss
$L_{\text{r}} \leftarrow ||D(E(x)) - x||_{l_1}$ # Standard reconstruction loss

shows a case that our zero-shot synthesised images can augment and boost training of a visual object recognition classifier (Sec. 5.4).

GZS-Net architecture. For all experiments, the encoder E is composed of two convolutional layers with stride 2, followed by 3 residual blocks, followed by a convolutional layer with stride 2, followed by reshaping the response map to a vector, and finally two fully-connected layers to output 100-dim vector as latent feature. The decoder D mirrors the encoder, and is composed of two fully-connected layers, followed by reshape into cuboid, followed by de-conv layer with stride 2, followed by 3 residual blocks, then finally two de-conv layers with stride 2, to output a synthesized image.

5.1 Fonts Dataset & Zero-shot synthesis Performance

Design Choices. Fonts is a computer-generated image datasets. Each image is of an alphabet letter and is accompanied with its generating attributes: Letters (52 choices, of lower- and upper-case English alphabet); size (3 choices); font colors (10); background colors (10); fonts (100); giving a total of 1.56 million images, each with size (128×128) pixels. We propose this dataset to allow fast testing and idea iteration on zero-shot synthesis and disentangled representation learning. Samples from the dataset are shown in Fig. 2. Details and source code are in the Appendix. We partition the 100-d latents equally among the 5 attributes. We use a train:test split of 75:25.

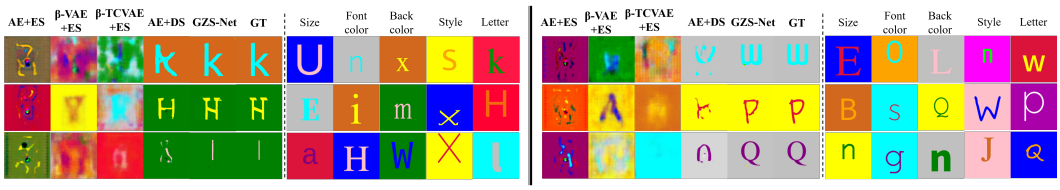


Figure 4: Zero-shot synthesis performance compare on Fonts. 7-11 and 18-22 columns are input group images and we want to combine the specific attribute of them to synthesize an new images. 1-5 and 12-16 columns are synthesized images use auto-encoder + Exhaustive Swap (AE+ES), β-VAE + Exhaustive Swap (β-VAE+ES), β-TCVAE + Exhaustive Swap (β-TCVAE+ES), auto-encoder + Directly Supervision (AE+DS) and GZS-Net respectively. 6 and 17 columns are ground truth (GT)

Baselines. We train four baselines:

- The first three are a standard Autoencoder, a β-VAE (Higgins et al., 2017), and β-TCVAE (Chen et al., 2018). β-VAE and β-TCVAE show reasonable disentanglement on the dSprites dataset (Matthey et al., 2017). Yet, they do not make explicit the assignment between latent variables and attributes, which would have been useful for precisely controlling the attributes (e.g. color, orientation) of synthesized images. Therefore, for these methods, we designed a best-effort approach by *exhaustively* searching for the assignments. Once assignments are known, swapping attributes between images might become possible with these VAEs, and hopefully enabling for

controllable-synthesis. We denote these three baselines with this **E**xhaustive **S**earch, using suffix **+ES** (Fig. 4). Details on Exhaustive Search are in the Appendix.

- The fourth baseline is an auto-encoder where its latent space is partitioned and each partition receives supervision from one attribute. Further details are in the Appendix.

5.2 Zero-shot synthesis on iLab-20M and RaFD

iLab-20M (Borji et al., 2016): an attributed dataset collected by shooting images of toy vehicles placed on a turntable using 11 cameras at different viewing points. There are 3 attribute classes: vehicle identity: 15 categories, each having 25-160 instances; pose; and backgrounds: over 14 for each identity: projecting vehicles in relevant contexts. Further details are in the Appendix. We partition the 100D latent space among attributes as: 60 for identity, 20 for pose, and 20 for background. iLab-20M has limited attribute combinations (identity shows only in relevant background; e.g., cars on roads but not in deserts), GZS-Net can disentangle these three attributes and reconstruct novel combinations (e.g., cars on desert backgrounds; see Fig. 5 for more examples).

Similar to 5.1 with Fonts, we also implement the baseline model of auto-encoder + Direct Supervision (AE+DS) with iLab-20M and obtain similar results. Using discriminative models, the classification tasks focus on the most discriminative information, e.g., to distinguish a car vs. a $semi$ can easily be achieved by size or length, while using only size or length information can hardly synthesize photo-realistic letters.

RaFD (Radboud Faces Database, Langner et al., 2010): contains pictures of 67 models displaying 8 emotional expressions taken by 5 different camera angles simultaneously. There are 3 attributes: identity, camera position (pose), and expression. We partition the 100D latent space among the attributes as 60 for identity, 20 for pose, and 20 for expression. We use a 80:20 split for train:test, and use GZS-Net to synthesize images with novel combination of attributes (Fig. 6). The synthesized images can capture the corresponding attributes well, especially for pose and identity.

5.3 Quantifying Disentanglement through attribute co-prediction

We want to obtain quantitative measures on disentanglement. Our network did **not** receive attribute information through supervision, but rather, through swapping. Nonetheless, can the latent features, assigned to an attribute class, predict the attribute value? Can it also predict values for other attributes? Under *perfect* disentanglement, we should answer *always* for the first and *never* for the second. We apply the analysis over our model trained on the Fonts dataset. We take the Test examples from Font, and split them 80:20 for train_{DR}:test_{DR}. For each attribute pair $j \in [1..m]$, we train an m-way classifier (3 layer MLP) from g_j of train_{DR} to the m attributes, then obtain the accuracy of each attribute by testing with g_j of train_{DR}. Table 1 compares how well features of each attribute (row) can predict an attribute value (column): perfect should be as close as possible to Identity matrix, with off-diagonal entries close to random (i.e., 1 / column header value).

5.4 GZS-Net Boost Object Recognition

GZS-Net as a generative model can boost image classification tasks by synthesizing zero-shot images. Two different training datasets (Fig. 7a) are tailored from iLab-20M, pose and background unbalanced

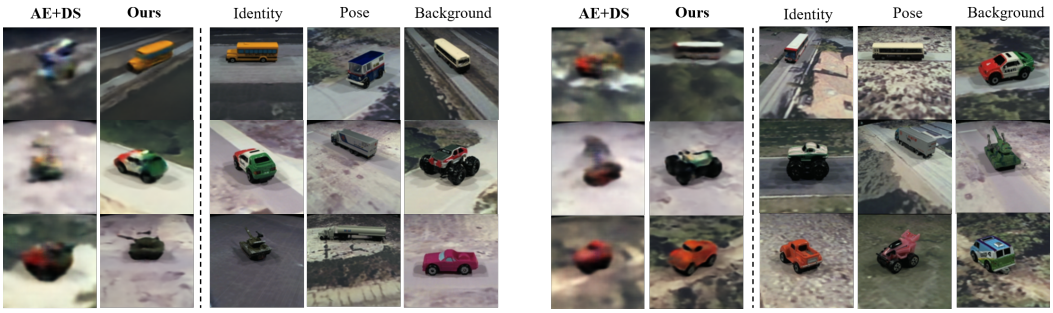


Figure 5: GZS-Net zero-shot synthesis performance on ilab-20M. 1-2 and 6-7 columns are synthesized images using auto-encoder + Directly Supervision (AE+DS) and GZS-Net respectively, Other columns are the input images which provide specific attributes to compose the zero-shot image.

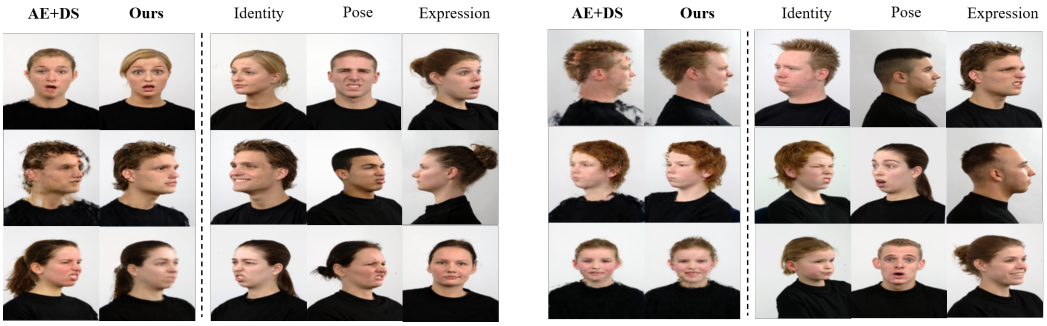


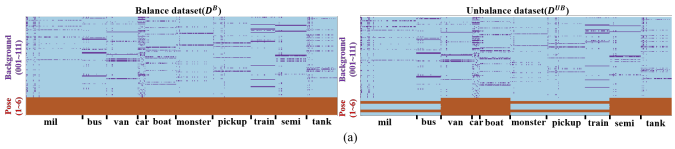
Figure 6: GZS-Net zero-shot synthesis performance on RaFD. 1-2 and 6-7 columns are the synthesized novel images using auto-encoder + Directly Supervision (AE+DS) and GZS-Net respectively. Remaining columns are training set images with their attributes provide.

Table 1: Disentangled representation analysis

	GZS-Net					Auto-encoder					Auto-encoder + Directly Supervise				
	Content (52)	Size (3)	Font color (10)	Back color (10)	style (100)	Content (52)	Size (3)	Font color (10)	Back color (10)	style (100)	Content (52)	Size (3)	Font color (10)	Back color (10)	style (100)
Content	**0.99**	0.92	0.11	0.13	0.30	**0.48**	0.60	0.71	0.92	0.06	**0.99**	0.72	0.22	0.20	0.25
Size	0.78	**1.0**	0.11	0.15	0.36	0.45	**0.61**	0.77	0.96	0.07	0.54	**1.0**	0.19	0.23	0.25
Font color	0.70	0.88	**1.0**	0.16	0.23	0.48	0.60	**0.67**	0.95	0.06	0.19	0.64	**1.0**	0.66	0.20
Back color	0.53	0.78	0.21	**1.0**	0.15	0.53	0.63	0.64	**0.93**	0.08	0.32	0.65	0.29	**1.0**	0.25
style	0.70	0.93	0.12	0.12	**0.63**	0.49	0.60	0.70	0.94	**0.06**	0.38	0.29	0.20	0.20	**0.65**

datasets ($\mathcal{D}^{\text{UB}}$) (half classes with 6 poses per object instance, other half with only 2 poses; as we cut poses, some backgrounds are also eliminated), as well as pose and background balanced dataset ($\mathcal{D}^{\text{B}}$) (all classes with all 6 poses per object instance).

We use GZS-Net to synthesize the missing images of $\mathcal{D}^{\text{UB}}$ and synthesize a new (augmented) balanced dataset $\mathcal{D}^{\text{B-s}}$. We alternatively use common data augmentation methods (random crop, horizontal flip, scale resize, etc) to augment the $\mathcal{D}^{\text{UB}}$ dataset to the same number of images as $\mathcal{D}^{\text{B-s}}$, called $\mathcal{D}^{\text{UB-a}}$. We show object recognition performance on the test set using these four datasets respectively. Comparing $\mathcal{D}^{\text{B-s}}$ with $\mathcal{D}^{\text{UB}}$ shows $\sim 7\%$ points improvements on classification performance, due to augmentation with synthesized images for missing poses in the training set, reaching the level of when all real poses are available ($\mathcal{D}^{\text{B}}$). Our synthesized poses outperform traditional data augmentation ($\mathcal{D}^{\text{UB-a}}$)



Dataset	$\mathcal{D}^{\text{UB}}$	$\mathcal{D}^{\text{B}}$	$\mathcal{D}^{\text{B-s}}$	$\mathcal{D}^{\text{UB-a}}$
Source	real	real	GZS-Net synthesized	Traditional augmented
Size	25149	37417	37395	37395
Accuracy	56.5	64.4	**63.5**	56.6

(b)

Figure 7: (a) Dataset details for training object recognition task, where the x-axis represents different identities (1004) and the y-axis represents the backgrounds (111) and poses (6) each purple and brown pixel means our dataset covers the specific combination of attributes. (b) object recognition accuracy (%) on 37469 test examples, after training on (augmented) datasets.

6 Conclusion

We propose a new learning framework, Group Supervised Learning (GSL), which admits datasets of example features and their attributes. It provides a learner groups of semantically-related samples, which we show is powerful for zero-shot synthesis. In particular, our Group-supervised Zero-Shot synthesis network (GZS-Net) is capable of training on groups of examples, and can learn disentangled representations by explicitly swapping latent features across training examples, along edges suggested by GSL. We show that, to synthesize samples given a query with custom attributes, it is sufficient to find one example per requested attribute and to combine them in the latent space. We hope that researchers find our learning framework useful and extend it for their applications.

REFERENCES

Yuval Atzmon and Gal Chechik. Probabilistic and-or attribute grouping for zero-shot learning. In *Uncertainty in Artificial Intelligence*, 2018.

A. Borji, S. Izadi, and L. Itti. ilab-20m: A large-scale controlled object dataset to investigate deep learning. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

Christopher P Burgess, Irina Higgins, Arka Pal, Loic Matthey, Nick Watters, Guillaume Desjardins, and Alexander Lerchner. Understanding disentangling in $beta$-vae. *arXiv preprint arXiv:1804.03599*, 2018.

Ricky T. Q. Chen, Xuechen Li, Roger B Grosse, and David K Duvenaud. Isolating sources of disentanglement in variational autoencoders. In *Advances in Neural Information Processing Systems*, 2018.

Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations. In *International Conference on Learning Representations*, 2018.

Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. In *ICML*, 2017.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. In *Neural Information Processing Systems*, 2014.

I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner. β-vae: Learning basic visual concepts with a constrained variational framework. In *International Conference on Learning Representations*, 2017.

Seunghoon Hong, Dingdong Yang, Jongwook Choi, and Honglak Lee. Inferring semantic layout for hierarchical text-to-image synthesis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7986–7994, 2018.

Hyunjik Kim and Andriy Mnih. Disentangling by factorising. *arXiv preprint arXiv:1802.05983*, 2018.

Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *International Conference on Learning Representations*, 2014.

Kevin Lai, Liefeng Bo, Xiaofeng Ren, and Dieter Fox. A large-scale hierarchical multi-view rgb-d object dataset. In *2011 IEEE international conference on robotics and automation*, pp. 1817–1824. IEEE, 2011.

C. H. Lampert. Learning to detect unseen object classes by between-class attribute transfer. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

Oliver Langner, Ron Dotsch, Gijsbert Bijlstra, Daniel HJ Wigboldus, Skyler T Hawk, and AD Van Knippenberg. Presentation and validation of the radboud faces database. *Cognition and emotion*, 24(8):1377–1388, 2010.

Nikos K. Logothetis, Jon Pauls, and Tomaso Poggiot. Shape representation in the inferior temporal cortex of monkeys. In *Current Biology*, 1995.

Loic Matthey, Irina Higgins, Demis Hassabis, and Alexander Lerchner. dsprites: Disentanglement testing sprites dataset. https://github.com/deepmind/dsprites-dataset/, 2017.

Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2009.

Luan Tran, Xi Yin, and Xiaoming Liu. Disentangled representation learning gan for pose-invariant face recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1415–1424, 2017.

Zhuoqian Yang, Wentao Zhu, Wayne Wu, Chen Qian, Qiang Zhou, Bolei Zhou, and Chen Change Loy. Transmomo: Invariance-driven unsupervised video motion retargeting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5306–5315, 2020.

Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaogang Wang, Xiaolei Huang, and Dimitris N Metaxas. Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pp. 5907–5915, 2017.

Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *International Conference on Computer Vision (ICCV'17)*, 2017.

APPENDIX

A FONTS DATASET

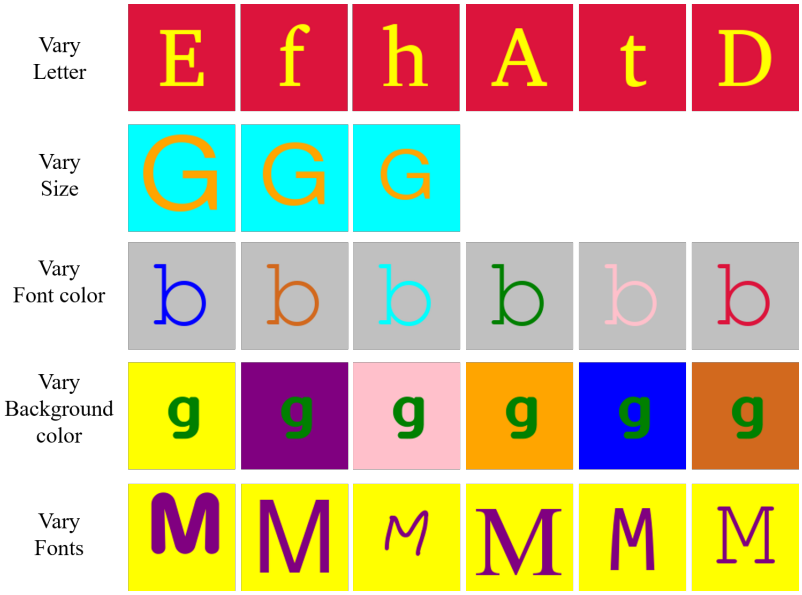


Figure 8: Samples from the Fonts dataset, a new parametric dataset we created by rendering characters under 5 distinct attributes. In each row, we keep all attributes the same but vary one.

Fonts is a computer-generated RGB image datasets. Each image, with 128×128 pixels, contains an alphabet letter rendered using 5 independent generating attributes: letter identity, size, font color, background color and font. Fig.1 shows some samples: in each row, we keep all attributes values the same but vary one attribute value. Attribute details are shown in Table 1. The dataset contains all possible combinations of these attributes, totaling to 1560000 images. Generating attributes for all images are contained within the dataset. Our primary motive for creating the Fonts dataset, is that it allows fast testing and idea iteration, on disentangled representation learning and zero-shot synthesis.

You can download the dataset and its generating code from: `http://anonymous/url/for/blind/review` , which we plan to keep up-to-date with contributions from ourselves and the community.

Table 2: Attributes generating the Fonts dataset

Attribute	Number of Attribute Values	Attribute Value Details
Letter	52	Uppercase Letters (A-Z) Lowercase Letters (a-z)
Size	3	Small, Medium, Large (80, 100, 120 pixel height respectively)
Font color	10	Red, Orange, Yellow, Green, Cyan Blue, Purple, Pink, Chocolate, Silver
Background color	10	Red, Orange, Yellow, Green, Cyan Blue, Purple, Pink, Chocolate, Silver
Font	100	Ubuntu system fonts e.g. aakar, chilanka, sarai, etc.

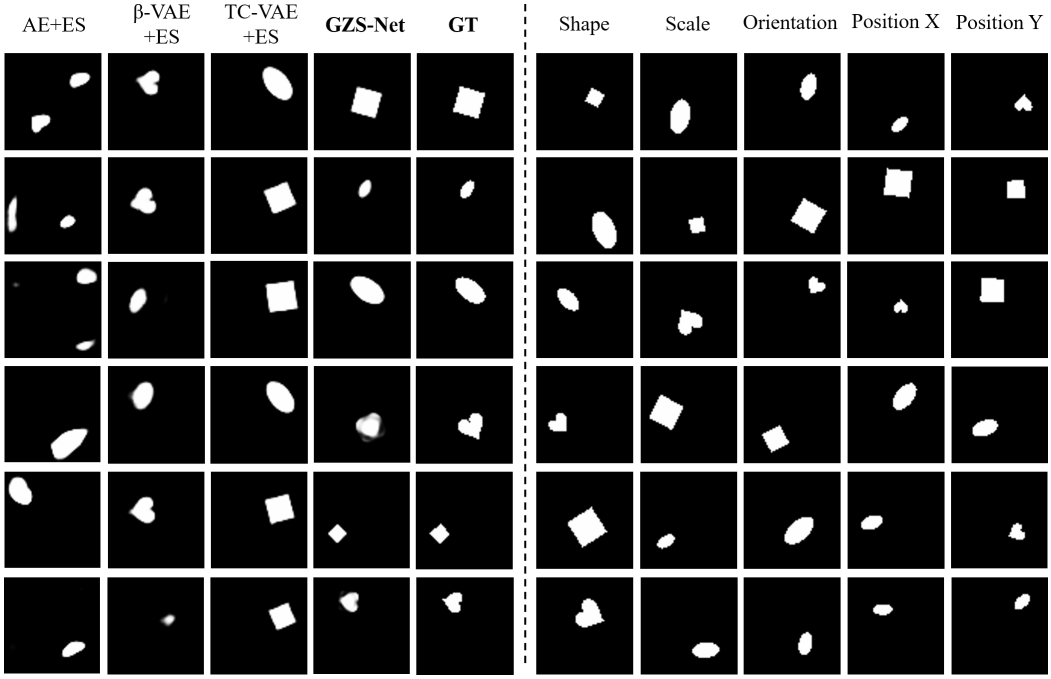


Figure 9: Zero-shot synthesis performance on dSprites. Columns 6-10 are input group images: from each, we want to extract one attribute (title of column). The goal is to combine the attributes to synthesize an new images. Columns 1-4 are synthesized images, respectively using: auto-encoder + Exhaustive Search (AE+ES), β-VAE + Exhaustive Search (β-VAE+ES), TC-VAE + Exhaustive Search (TC-VAE+ES) and GZS-Net respectively. The 5th column are ground truth (GT), which none of the methods saw during training or synthesis

B BASELINES

B.1 EXHAUSTIVE SEARCH (ES) AFTER TRAINING AUTO-ENCODER BASED METHODS

After training the baselines: standard Autoencoder, a β-VAE (Higgins et al., 2017), and TC-VAE (Chen et al., 2018). We want to search for the assignment between latent variables and attributes, as these VAEs do not make explicit the assignment. This knowing the assignment should hypothetically allow us to trade attributes between two images by swapping feature values belonging to the attribute we desire to swap.

To discover the assignment from latent dimension to attribute, we map all n training images through the encoder, giving a 100D vector per training sample $\in \mathbb{R}^{n\times 100}$. We make an 80:20 split on the vectors, obtaining $X_{\text{train}_{\text{ES}}} \in \mathbb{R}^{0.8n\times 100}$ and $X_{\text{test}_{\text{ES}}} \in \mathbb{R}^{0.2n\times 100}$. Then, we randomly sample K different partitionings P of the 100D space evenly among the 5 attributes. For each partitioning $p \in P$, we create 5 classification tasks, one task per attribute, according to p: $\left\{\left(X_{\text{train}_{\text{ES}}}[:, p_j] \in \mathbb{R}^{0.8n\times 20}, X_{\text{test}_{\text{ES}}}[:, p_j] \in \mathbb{R}^{0.2n\times 20}\right)\right\}_{j=1}^{5}$. For each task j, we train a 3-layer MLP to map $X_{\text{train}_{\text{ES}}}[:, p_j]$ to their known attribute values and measure its performance on $X_{\text{test}_{\text{ES}}}[:, p_j]$. Finally, we commit to the partitioning $p \in P$ with highest average performance on the 5 attribute tasks. This p represents our best effort to determine which latent feature dimensions correspond to which attributes. For zero-shot synthesis with baselines, we swap latent dimensions indicated by partitioning p. We denote three baselines with this **E**xhaustive **S**earch, using suffix **+ES** (Fig. 4).

B.2 Direct Supervision (DS) on Auto-encoder latent space

The last baseline (AE+DS) directly uses attribute labels to supervise the latent disentangled representation of the auto-encoder by adding auxiliary classification modules. Specifically, the encoder maps an image sample $x^{(i)}$ to a 100-d latent vector $z^{(i)} = E(x^{(i)})$, equally divided into 5 partitions corresponding to 5 attributes: $z^{(i)} = [g_1^{(i)}, g_2^{(i)}, \ldots, g_5^{(i)}]$. Each attribute partition has a attribute label, $[y_1^{(i)}, y_2^{(i)}, \ldots, y_5^{(i)}]$, which represent the attribute value (e.g. for font color attribute, the label represent different colors: red, green, blue,.etc). We use 5 auxiliary classification modules to predict the corresponding class label given each latent attribute partitions as input. We use Cross Entropy loss as the classification loss and the training goal is to minimize both the reconstruction loss and classification loss.

After training, we have assignment between latent variables and attributes, so we can achieve attribute swapping and controlled synthesis (Fig. 4 (AE+DS)). The inferior synthesis performance demonstrates that: The supervision (classification task) preserves discriminative information that is insufficient for photo-realistic generation. While our GZS-Net uses one attribute swap and cross swap which enforce disentangled information to be sufficient for photo-realistic synthesis.

C Zero-shot synthesis Performance on dSprites dataset

We qualitatively evaluate our method, Group-Supervised Zero-Shot Synthesis Network (GZS-Net), against three baseline methods, on zero-shot synthesis tasks on the dSprites dataset.

C.1 dSprites

dSprites (Matthey et al., 2017) is a dataset of 2D shapes procedurally generated from 6 ground truth independent latent factors. These factors are color, shape, scale, rotation, x- and y-positions of a sprite. All possible combinations of these latents are present exactly once, generating 737280 total images. Latent factor values (Color: white; Shape: square, ellipse, heart; Scale: 6 values linearly spaced in [0.5, 1]; Orientation: 40 values in [0, 2 pi]; Position X: 32 values in [0, 1]; Position Y: 32 values in [0, 1])

C.2 Experiments of Baselines and GZS-Net

We train a 10-dimensional latent space and partition the it equally among the 5 attributes: 2 for shape, 2 for scale, 2 for orientation, 2 for position X, and 2 for position Y. We use a train:test split of 75:25.

We train 3 baselines: a standard Autoencoder, a β-VAE (Higgins et al., 2017), and TC-VAE (Chen et al., 2018). To recover the latent-to-attribute assignment for these baselines, we utilize the *Exhaustive Search* best-effort strategy, described in the main paper: the only difference is that we change the dimension of Z space from 100 to 10. Once assignments are known, we utilize these baseline VAEs by attribute swapping to do controlled synthesis. We denote these baselines using suffix **+ES**.

As is shown in Figure 2, GZS-Net can precisely synthesize zero-shot images with new combinations of attributes, producing images similar to the groud truth. The baselines β-VAE and TC-VAE produce realistic images of good visual quality, however, not satisfying the requested query: therefore, they

cannot do controllable synthesis even when equipped with our best-effort Exhaustive Search to discover the disentanglement. Standard auto-encoders can not synthesis meaningful images when combining latents from different examples, giving images outside the distribution of training samples (e.g. showing multiple sprites per image).