ZERO-SHOT SYNTHESIS WITH GROUP-SUPERVISED LEARNING

Yunhao Ge, Sami Abu-El-Haija, Gan Xin, Laurent Itti
University of Southern California
yunhaoge@usc.edu, sami@haija.org, gxin@usc.edu, itti@usc.edu

APPENDIX

A FONTS DATASET

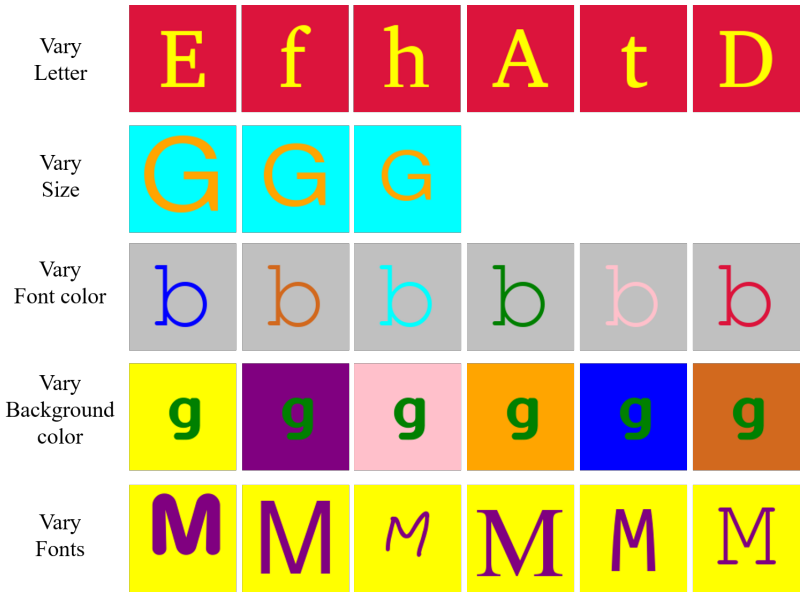

Figure 1: Samples from the Fonts dataset, a new parametric dataset we created by rendering characters under 5 distinct attributes. In each row, we keep all attributes the same but vary one.

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://ilab.usc.edu/datasets/fonts` , which we plan to keep up-to-date with contributions from ourselves and the community.

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

Table 1: 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.

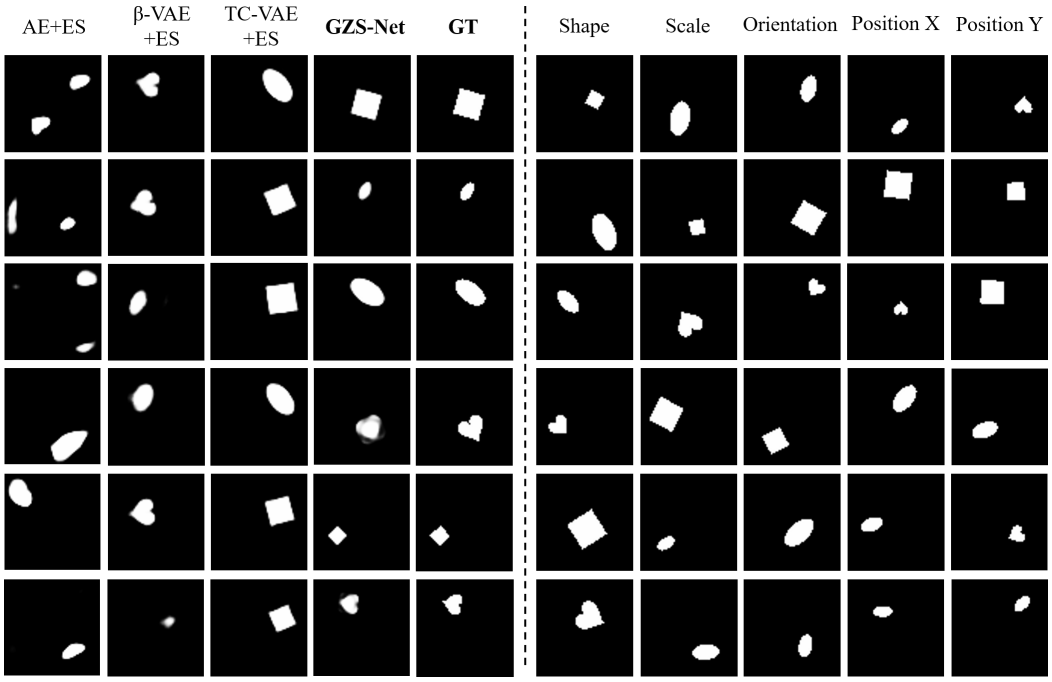

Figure 2: Zero-shot synthesis performance on dSprites. Columns 6-10 are input group images: from each, we want to extract one attribute (title of column). The goal is to combine the attributes to synthesize an new images. Columns 1-4 are synthesized images, respectively using: auto-encoder + Exhaustive 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

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. **??**).

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.

B.3 ELEGANT (Xiao et al., 2018)

We utilize the author's open-sourced code: `https://github.com/Prinsphield/ELEGANT`. For ELEGANT and starGAN (Section B.4), we want to synthesis a target image has same identity as *id provider image*, same background as *background provider image*, and same pose as *pose provider image*. To achieve this, we want to change the background and pose attribute of id image.

Although ELEGANT is strong in making image transformations that are local to relatively-small neighborhoods, however, it does not work well for our datasets, where image-wide transformations are required for meaningful synthesis. This can be confirmed by their model design: their final output is a pixel-wise addition of a residual map, plus the input image. Further, ELEGANT treats all attribute values as binary: **they represent each attribute value in a different** part of the latent space, whereas **our method devotes part of the latent space to represents all values for an attribute**. For investigation, we train dozens of ELEGANT models with different hyperparameters, detailed as:

- For iLab-20M, the pose and background contain a total of 117 attribute values (6 for pose, 111 for background). As such, we tried training it on all attribute values (dividing their latent space among 117 attribute values). We note that this training regime was too slow and the loss values do not seem to change much during training, even with various learning rate choices (listed below).
- To reduce the difficulty of the task for ELEGANT, we ran further experiments restricting attribute variation to **only 17 attribute values** (6 for pose, 11 for background) and this shows more qualitative promise than 117 attributes. This is what we report.
- Fig 3 shows that ELEGANT finds more challenge in changing the pose than in changing the background. We now explain how we generated Columns 3 and 4 of Fig 3 for modifying the background. We modify the latent features for the identity image before decoding. Since the *Identity input image* and the *Background input image* have known but different background values, their background latent features are represented in two different latent spaces. One can swap on one or on both of these latent spaces. Column 3 and 4 of Fig.3 swap only on one latent space. However, in Fig. **??** of the main paper, we swap on both positions. We also show swapping only the pose attribute (across 2 latent spaces) in Column 1 of Fig.3 and swapping both pose and background in Column 2.

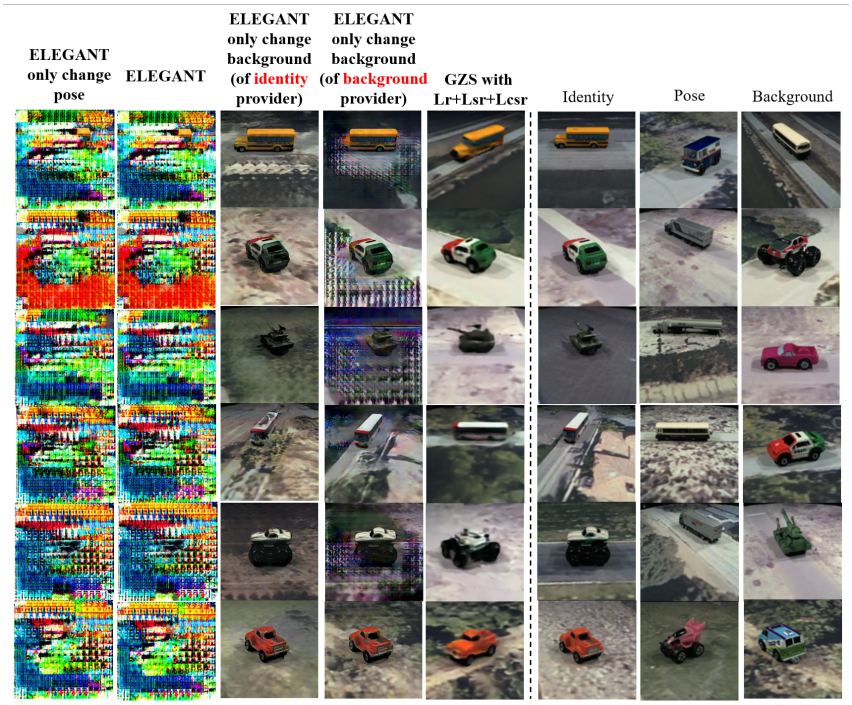

Figure 3: Zero-shot synthesis for ELEGANT, investigating the modification of pose and background attributes on an identity image. Details are in Section B.3.

- To investigate if the model's performance is due to poor convergence of the generator, we qualitatively assess its performance on the **training set**. Fig. 4 shows output of ELEGANT on training samples. We see that the reconstruction (right) of input images (left) shows decent quality, suggesting that the generator network has converged to decently good parameters. Nonetheless, we see artefacts in its outputs when amending attributes, particularly located in pixel locations where a change is required. This shows that the model setup of ELEGANT is aware that these pixel values need to be updated, but the actual change is not coherent across the image.
- For the above, we applied a generous sweep of training hyperparameters, including:
  - **Learning rate**: author's original is 2e-4, we tried several values between 1e-5 and 1e-3, including different rates for generator and discriminator.
  - **Objective term coefficients**: There are multiple loss terms for the generator, adversarial loss and reconstruction loss. We used a grid search method by multiplying the original parameters by a number from [0.2, 0.5, 2, 5] for each of the loss terms and tried several combinations.
  - **The update frequency** of weights on generator (G) and discriminator (D). Since D is easier to learn, we performing k update steps on G for every update step on D. We tried $k = 5, 10, 15, 20, 30, 40, 50$.

  We report ELEGANT results showing best qualitative performance.

Overall, ELEGANT does not work well for holistic image manipulation (though works well for **local image edits**, per experiments by authors (Xiao et al., 2018)).

B.4 STARGAN (CHOI ET AL., 2018)

We utilize the author's open-sourced code: `https://github.com/yunjey/stargan`. Unlike ELEGANT (Xiao et al., 2018) and our method, starGAN only accepts one input image and an edit information: the edit information, is **not extracted from another image** – this is following their method and published code.

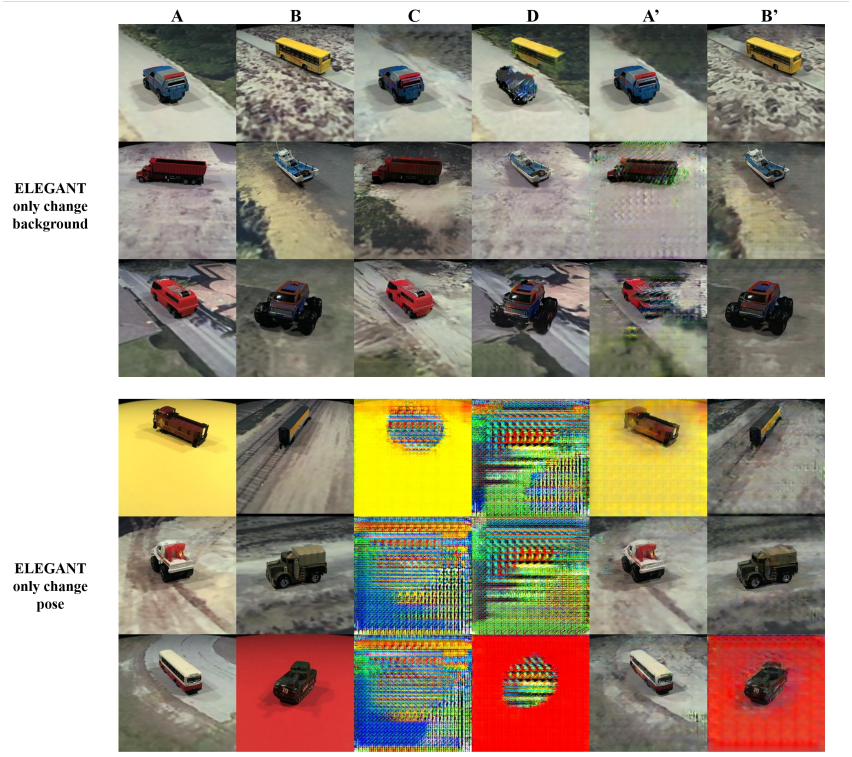

Figure 4: Training performance of ELEGANT. The left 2 columns (A and B) are input image. The following 4 columns are the generated synthesized images: A' and B' are reconstructions of the input (acceptable quality, suggesting convergence of generator), whereas C and D are the result of swapping features before the generator: C (/ D) uses the latent features of A (/ B) except by swapping background with B (/ A). All (C, D, A', B') share the same generator.

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).

REFERENCES

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.

Yunjey Choi, Minje Choi, Munyoung Kim, Jung-Woo Ha, Sunghun Kim, and Jaegul Choo. Stargan: Unified generative adversarial networks for multi-domain image-to-image translation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

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.

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

Taihong Xiao, Jiapeng Hong, and Jinwen Ma. Elegant: Exchanging latent encodings with gan for transferring multiple face attributes. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 172–187, September 2018.