MixNMatch: Multifactor Disentanglement and Encoding for Conditional Image Generation

Yuheng Li Krishna Kumar Singh Utkarsh Ojha Yong Jae Lee
University of California, Davis

Abstract

We present MixNMatch, a conditional generative model that learns to disentangle and encode background, object pose, shape, and texture from real images with minimal supervision, for mix-and-match image generation. We build upon FineGAN, an unconditional generative model, to learn the desired disentanglement and image generator, and leverage adversarial joint image-code distribution matching to learn the latent factor encoders. MixNMatch requires bounding boxes during training to model background, but requires no other supervision. Through extensive experiments, we demonstrate MixNMatch's ability to accurately disentangle, encode, and combine multiple factors for mix-and-match image generation, including sketch2color, cartoon2img, and img2gif applications. Our code/models/demo can be found at https://github.com/Yuheng-Li/MixNMatch

1. Introduction

Consider the real image of the yellow bird in Figure 1 in the 1st column. What would the bird look like in a different background, say that of the duck? How about in a different texture, perhaps that of the rainbow textured bird in the 2nd column? What if we wanted to keep its texture, but change its shape to that of the rainbow bird, and background and pose to that of the duck, as in the 3rd column? How about sampling shape, pose, texture, and background from four different reference images and combining them to create an entirely new image (last column)?

Problem. While research in conditional image generation has made tremendous progress [17, 49, 30], no existing work can simultaneously disentangle *background*, *object pose*, *object shape*, and *object texture* with minimal supervision, so that these factors can be combined from *multiple real images* for fine-grained controllable image generation. Learning disentangled representations with minimal supervision is an extremely challenging problem, since the underlying factors that give rise to the data are often highly correlated and intertwined. Work that disentangle

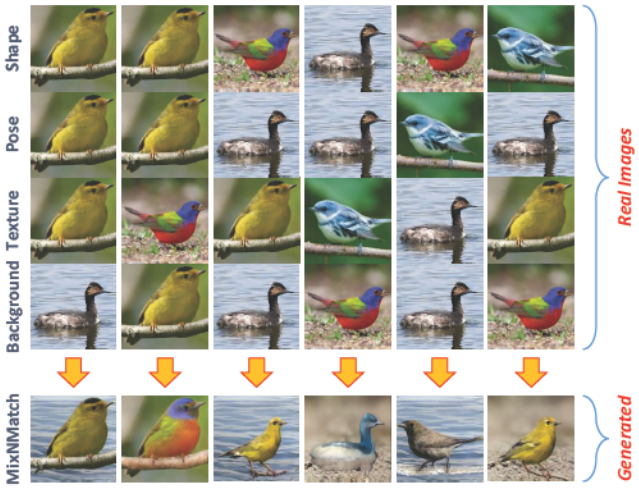


Figure 1: **Conditional mix-and-match image generation.** Our model, *MixNMatch*, can disentangle and encode up to four factors—background, object pose, shape, and texture—from real reference images, and can arbitrarily combine them to generate new images. The only supervision used to train our model is bounding box annotations to model background.

two such factors, by taking as input two reference images e.g., one for appearance and the other for pose, do exist [16, 18, 23, 26, 40]. But they cannot disentangle other factors such as foreground vs. background appearance or pose vs. shape. Since only two factors can be controlled, these approaches cannot arbitrarily change, for example, the object's background, shape, and texture, while keeping its pose the same. Others require strong supervision in the form of keypoint/pose/mask annotations [31, 1, 27, 9], which limits their scalability, and still fall short of disentangling all of the four factors outlined above.

Our proposed conditional generative model, *MixNMatch*, aims to fill this void. MixNMatch learns to disentangle and encode background, object pose, shape, and texture latent factors from real images, and importantly, does so with minimal human supervision. This allows, for example, each factor to be extracted from a different real im-

age, and then combined together for mix-and-match image generation; see Fig. 1. During training, MixNMatch only requires a loose bounding box around the object to model background, but requires no other supervision for modeling the object's pose, shape, and texture.

Main idea. Our goal of mix-and-match image generation i.e., generating a single synthetic image that combines different factors from multiple real reference images, requires a framework that can simultaneously learn (1) an encoder that encodes latent factors from real images into a *disentangled* latent code space, and (2) a generator that takes latent factors from the disentangled code space for image generation. To learn the generator and the disentangled code space, we build upon FineGAN [36], a generative model that learns to hierarchically disentangle background, object pose, shape, and texture with minimal supervision using information theory. However, FineGAN is conditioned only on *sampled latent codes*, and cannot be directly conditioned on real images for image generation. We therefore need a way to extract latent codes that control background, object pose, shape, and texture from *real images*, while preserving FineGAN's hierarchical disentanglement properties. As we show in the experiments, a naive extension of FineGAN in which an encoder is trained to map a fake image into the codes that generated it is insufficient to achieve disentanglement in real images due to the domain gap between real and fake images.

To simultaneously achieve the above dual goals, we instead perform adversarial learning, whereby the joint distribution of real images and their extracted latent codes from the encoder, and the joint distribution of sampled latent codes and corresponding generated images from the generator, are learned to be indistinguishable, similar to ALI [8] and BiGAN [6]. By enforcing matching joint image-code distributions, the encoder learns to produce latent codes that match the distribution of sampled codes with the desired disentanglement properties, while the generator learns to produce realistic images. To further encode a reference image's shape and pose factors with high fidelity, we augment MixNMatch with a *feature mode* in which higher dimensional features of the image that preserve pixel-level structure (rather than low dimensional codes) are mapped to the learned disentangled feature space.

Contributions. (1) We introduce MixNMatch, a conditional generative model that learns to disentangle and encode background, object pose, shape, and texture factors from real images with minimal human supervision. This gives MixNMatch fine-grained control in image generation, where each factor can be uniquely controlled. MixNMatch can take as input either real reference images, sampled latent codes, or a mix of both. (2) Through various qualitative and quantitative evaluations, we demonstrate MixNMatch's ability to accurately disentangle, encode, and combine multiple factors for mix-and-match image generation. Furthermore, we show that MixNMatch's learned disentangled representation leads to state-of-the-art fine-grained object category clustering results of real images. (3) We demonstrate a number of interesting applications of MixNMatch including sketch2color, cartoon2img, and img2gif.

2. Related work

Conditional image generation has various forms, including models conditioned on a class label [29, 28, 3] or text input [33, 48, 42, 47]. A lot of work focuses on image-to-image translation, where an image from one domain is mapped onto another domain e.g., [17, 49, 30]. However, these methods typically lack the ability to explicitly disentangle the factors of variation in the data. Those that do learn disentangled representations focus on specific categories like faces/humans [37, 31, 2, 32, 1, 27] or require clearly defined domains (e.g., pose vs. identity or style/attribute vs. content) [18, 16, 23, 11, 25, 40]. In contrast, MixNMatch is not specific to any object category, and does not require clearly defined domains as it disentangles multiple factors of variation within a single domain (e.g., natural images of birds). Moreover, unlike most unsupervised methods which can disentangle only two factors like shape and appearance [24, 35, 26], MixNMatch can disentangle four (background, object shape, pose, and texture).

Disentangled representation learning aims to disentangle the underlying factors that give rise to real world data [4, 44, 41, 24, 35, 38, 15, 19, 26]. Most unsupervised methods are limited to disentangling at most two factors like shape and texture [24, 35]. Others require strong supervision in the form of edge/keypoint/mask annotations or detectors [31, 1, 27, 9], or rely on video to automatically acquire identity labels [5, 18, 40]. Our most related work is FineGAN [36], which leverages information theory [4] to disentangle background, object pose, shape, and texture with minimal supervision. However, it is conditioned only on sampled latent codes, and thus cannot perform image translation. We build upon this work to enable conditioning on real images. Importantly, we show that a naive extension is insufficient to achieve disentanglement in real images. We also improve the quality of our model's image generations to preserve instance specific details from the reference images. Since MixNMatch is directly conditioned on images, its learned representation leads to better disentanglement and fine-grained clustering of real images.

3. Approach

Let $\mathcal{I} = \{x_1, \ldots, x_N\}$ be an unlabeled image collection of a single object category (e.g., birds). Our goal is to learn a conditional generative model, MixNMatch, which simul-

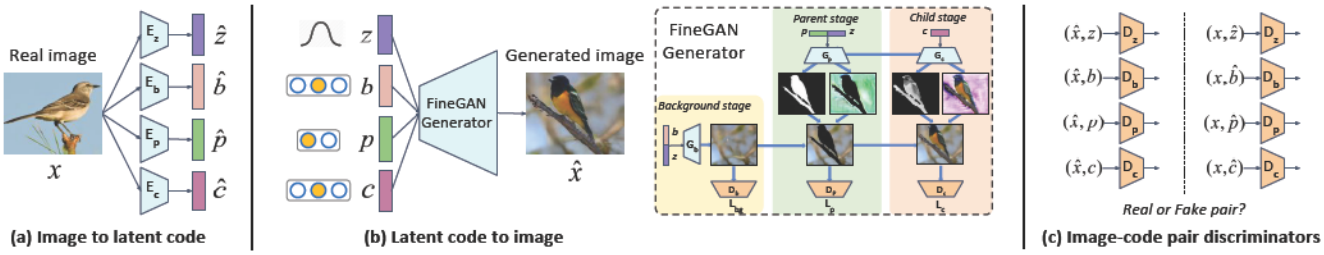


Figure 2: **MixNMatch architecture.** (a) Four different encoders, one for each factor, take a real image as input to predict the codes. (b) Four different latent codes are sampled and fed into the FineGAN generator to hierarchically generate images. (c) Four image-code pair discriminators optimize the encoders and generator, to match their joint image-code distributions.

taneously learns to (1) encode background, object pose, shape, and texture factors associated with images in $\mathcal{I}$ into a disentangled latent code space (i.e., where each factor is uniquely controlled by a code), and (2) generate high quality images matching the true data distribution $P_{data}(x)$ by combining latent factors from the disentangled code space.

We first briefly review FineGAN [36], from which we base our generator. We then explain how to train our model to disentangle and encode background, object pose, shape, and texture from real images, so that it can combine different factors from different real reference images for mix-and-match image generation. Lastly, we introduce how to augment our model to preserve object shape and pose information from a reference image with high fidelity (i.e., at the pixel-level).

3.1. Background: FineGAN

FineGAN [36] takes as input four randomly sampled latent codes (z, b, c, p) to hierarchically generate an image in three stages (see Fig. 2 (b) right): (1) a background stage where the model only generates the background, conditioned on latent one-hot background code b; (2) a parent stage where the model generates the object's shape and pose, conditioned on latent one-hot parent code p as well as continuous code z, and stitches it to the existing background image; and (3) a child stage where the model fills in the object's texture, conditioned on latent one-hot child code c. In both the parent and child stages, FineGAN automatically generates masks (without any mask supervision) to capture the appropriate shape and texture details.

To disentangle the background, it relies on object bounding boxes (e.g., acquired through an object detector). To disentangle the remaining factors of variation without any supervision, FineGAN uses information theory [4], and imposes constraints on the relationships between the latent codes. Specifically, during training, FineGAN (1) constrains the sampled child codes into disjoint groups so that each group shares the same unique parent code, and (2) enforces the sampled background and child codes for each generated image to be the same. The first constraint models the fact that some object instances from the same category share a common shape even if they have different textures (e.g., different duck species with different texture details share the same duck shape), and the second constraint models the fact that background is often correlated with specific object types (e.g., ducks typically have water as background). If we do not follow these constraints, then the generator could generate e.g. a duck on a tree (background code b not equal to texture code c) or e.g. a seagull with red texture (texture code c not tied to a specific shape code p). Then the discriminator would easily classify these images as fake, as they rarely exist in real images. As a result, the desired disentanglement will not be learned. It is also important to note that the parent code p controls viewpoint/pose invariant *3D shape* of an object (e.g., duck vs. seagull shape) as the number of unique p codes is typically set to be much smaller (e.g., 20) than the amount of 2D shape variations in the data, and this in turn forces the continuous code z to control viewpoint/pose. Critically, these factors emerge as a property of the data and the model, and not through any supervision.

FineGAN is trained with three losses, one for each stage, which use either adversarial training [12] to make the generated image look real and/or mutual information maximization [4] between the latent code and corresponding image so that each code gains control over the respective factor (background, pose, shape, color). We simply denote its full loss as:

$$\mathcal{L}_{finegan} = \mathcal{L}_b + \mathcal{L}_p + \mathcal{L}_c, \tag{1}$$

where $\mathcal{L}_b$, $\mathcal{L}_p$, and $\mathcal{L}_c$ denote the losses in the background, parent, and child stages. For more details on these losses and the FineGAN architecture, please refer to [36].

3.2. Paired image-code distribution matching

Although FineGAN can disentangle multiple factors to generate realistic images, it is conditioned on sampled latent codes, and cannot be conditioned on real images. A naive post-processing extension in which encoders that learn to map fake images to the codes that generated them is insufficient to achieve disentanglement in real images due to the domain gap between real and fake images [36], as we show

in our experiments.

Thus, to encode disentangled representations from *real images* for conditional mix-and-match image generation, we need to extract the vector z (controlling object pose), b (controlling background), p (controlling object shape), and c (controlling object texture) codes from *real images*, while preserving the hierarchical disentanglement properties of FineGAN. For this, we propose to train four encoders, each of which predict the z, b, p, c codes from real images. Since FineGAN has the ability to disentangle factors and generate images given latent codes, we naturally resort to using it as our generator, by keeping all the losses (i.e., $L_{finegan}$) to help the encoders learn the desired disentanglement.

Specifically, for each real training image x, we use the corresponding encoders to extract its z, b, p, c codes. However, we cannot simply input these codes to the generator to reconstruct the image, as the model would take a shortcut and degenerate into a simple autoencoder that does not preserve FineGAN's disentanglement properties (factorization into background, pose, shape, texture), as we show in our experiments. We therefore leverage ideas from ALI [8] and BiGAN [6, 7] to help the encoders learn the *inverse mapping*; i.e., a projection from real images into the code space, in a way that maintains the desired disentanglement properties.

The key idea is to perform adversarial learning [12, 6, 8], so that the paired image-code distribution produced by the encoder ($x \sim P_{data}$, $\hat{y} \sim E(x)$) and the paired image-code distribution produced by the generator ($\hat{x} \sim G(y)$, $y \sim P_{code}$) are matched. Here E is the encoder, G is the FineGAN generator, and y is a placeholder for the latent codes z, b, p, c. P_{data} is the data (real image) distribution and P_{code} is the latent code distribution.[1] Formally, the input to the discriminator D is an *image-code pair*. When training D, we set the paired real image x and code $\hat{y}$ extracted from the encoder E to be real, and the paired sampled input code y and generated image $\hat{x}$ from the generator G to be fake. Conversely, when training G and E, we try to fool D so that the paired distributions $P_{(data,E(x))}$ and $P_{(G(y),code)}$ are indistinguishable, via a paired adversarial loss:

$$\mathcal{L}_{bi\_adv} = \min_{G,E} \max_{D} \mathbb{E}_{x \sim P_{data}} \mathbb{E}_{\hat{y} \sim E(x)} [\log D(x, \hat{y})] + \mathbb{E}_{y \sim P_{code}} \mathbb{E}_{\hat{x} \sim G(y)} [\log(1 - D(\hat{x}, y))]. \quad (2)$$

This loss will simultaneously enforce the (1) generated images $\hat{x} \sim G(y)$ to look real, and (2) extracted real image codes $\hat{y} \sim E(x)$ to capture the desired factors (i.e., pose, background, shape, appearance). Fig. 2 (a-c) show our encoders, generator, and discriminators.

[1] Following FineGAN [36]: a continuous noise vector $z \sim \mathcal{N}(0, 1)$; a categorical background code $b \sim \text{Cat}(K = N_b, p = 1/N_b)$; a categorical parent code $p \sim \text{Cat}(K = N_p, p = 1/N_p)$; and a categorical child code $c \sim \text{Cat}(K = N_c, p = 1/N_c)$. N_b, N_p, N_c are the number of background, parent, and child categories and are set as hyperparameters.

3.3. Relaxing the latent code constraints

There is an important issue that we must address to ensure disentanglement in the extracted codes. FineGAN imposes strict code relationship constraints, which are key to inducing the desired disentanglement in an unsupervised way, but which can be difficult to realize in *all* real images.

Specifically, recall from Sec. 3.1 that these constraints impose a group of child codes to share the same unique parent code, and the background and child codes to always be the same. However, for any arbitrary real image, these strict relationships may not hold (e.g., a flying bird can have multiple different backgrounds in real images), and would thus be difficult to enforce in its extracted codes. In this case, the discriminator would easily be able to tell whether the image-code pair is real or fake (based on the code relationships), which will cause issues with learning. Moreover, it would also confuse the background b and texture c encoders since the background and child latent codes are always sampled to be the same ($b = c$); i.e., the two encoders will essentially become identical (as they are always being asked to predict the same output as each other) and won't be able to distinguish between background and object texture.

We address this issue in two ways. First, we train four separate discriminators, one for each code type. This prevents any discriminator from seeing the other codes, and thus cannot discriminate based on the relationships between the codes. Second, when training the encoders, we also provide as input *fake* images that are generated with randomly sampled codes with the code constraints removed. In these images, any foreground texture can be coupled with any arbitrary background ($c \neq b$) and any arbitrary shape (c not tied to a particular p). Specifically, we train the encoders E to predict back the sampled codes y that were used to generate the corresponding fake image:

$$\mathcal{L}_{code\_pred} = CE(E(G(y)), y), \quad (3)$$

where $CE(\cdot)$ denotes cross-entropy loss, and y is a placeholder for the latent codes b, p, c. (For continuous z, we use L1 loss.) This loss helps to guide each encoder, and in particular the b and c encoders, to learn the corresponding factor. Note that the above loss is used only to update the encoders E (and not the generator G), as these fake images can have feature combinations that generally do not exist in the real data distribution (e.g., a duck on top of a tree).

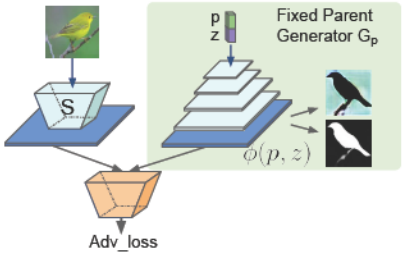
3.4. Feature mode for exact shape and pose

Thus far, MixNMatch's encoders can take in up to four different real images and encode them into b, z, p, c codes which model the background, object pose, shape, and texture, respectively. These codes can then be used by MixNMatch's generator to generate realistic images, which combine the four factors from the respective reference images.

Figure 3: **Comparison between code mode & feature mode.** Rows 1-3 are real reference images, in which we extract background b, texture c, and shape+pose p & z, respectively. Rows 4-5 are MixNMatch's feature mode (which accurately preserves original shape information) and code mode (which preserves shape information at a semantic level) generations.

We denote this setting as MixNMatch's *code mode*. While the generated images already capture the factors with high accuracy (see Fig. 3, "code mode"), certain image translation applications may require exact *pixel-level* shape and pose alignment between a reference image and the output.

The main reason that MixNMatch in code mode cannot preserve exact pixel-level shape and pose details of a reference image is because the capacity of the latent p code space, which is responsible for capturing shape, is too small to model per-instance pixel-level details (typically, tens in dimension). The reason it must be small is because it must (roughly) match the number of unique 3D shape variations in the data (e.g., duck shape, sparrow shape, seagull shape, etc.). In this section, we introduce MixNMatch's *feature mode* to address this. Rather than encode a reference image into a low-dimensional shape code, the key idea is to directly learn a mapping from the image to a higher-dimensional feature space that preserves the reference image's *spatially-aligned* shape and pose (pixel-level) details.



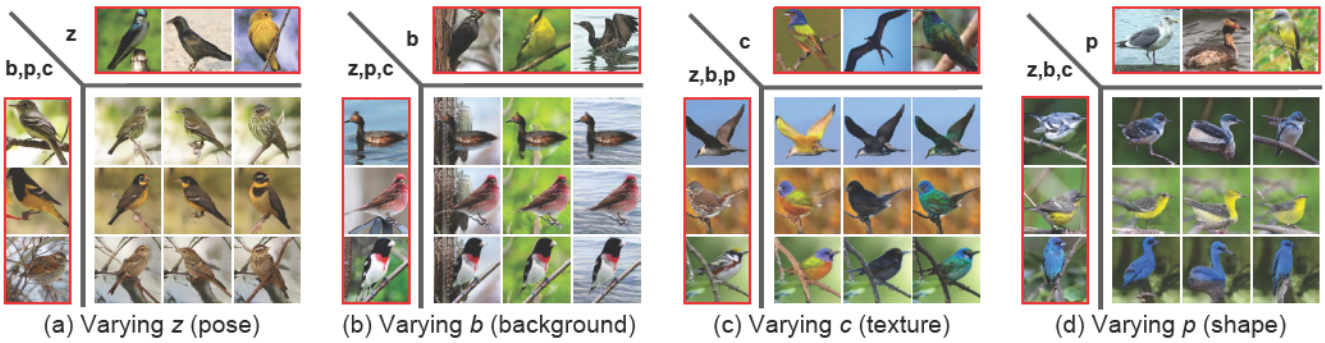
Specifically, we take our learned parent stage generator G_p (see Fig. 2 (b)), and use it to train a new shape and pose feature extractor S, which takes as input a real image x and outputs feature $S(x)$. G_p takes as input codes p and z to generate the parent stage image, which captures the object's shape. Let's denote its intermediate feature as $\phi(p, z)$. We use the standard adversarial loss [12] to train S so that the distribution of $S(x)$ matches that of $\phi(p, z)$ (i.e., only S is learned and $\phi(p, z)$ is produced from the fixed pretrained G_p); see figure above. Ultimately, this trains S to produce features that match those sampled from the $\phi(p, z)$ distribution, which already has learned to encode shape and pose. To enforce S to preserve instance-specific shape and pose details of x (i.e., so that the resulting generated image using $S(x)$ is spatially-aligned to x), we randomly sample codes z, b, p, c to generate fake images using the full generator G, and for each fake image $G(z, b, p, c)$, we enforce an L1 loss between the feature $\phi(p, z)$ and the feature $S(G(z, b, p, c))$.

Once trained, we can use this *feature mode* to extract the pixel-aligned pose and shape feature $S(x)$ from an input image x, and combine it with the background b and texture c codes extracted from (up to) two reference images, to perform conditional mix-and-match image generation.

4. Experiments

We evaluate MixNMatch's conditional mix-and-match image generation results, its ability to disentangle each latent factor, and its learned representation for fine-grained object clustering of real images. We also showcase sketch2color, cartoon2img, and img2gif applications.

Datasets. (1) **CUB** [39]: 11,788 bird images from 200 classes; (2) **Stanford Dogs** [21]: 12,000 dog images from 120 classes; (3) **Stanford Cars** [22]: 8,144 car images from 196 classes. We set the prior latent code distributions following FineGAN [36][1]. The only supervision we use is bounding boxes to model background during training.



(a) Varying *z* (pose) (b) Varying *b* (background) (c) Varying *c* (texture) (d) Varying *p* (shape)

Figure 4: **Varying a single factor.** Real images are indicated with red boxes. For (a-d), the reference images on the left/top provide three/one factors. The center 3x3 images are generations. For example, in (a) the top row yellow bird has an upstanding pose with its head turned to the right, and the resulting images have the same pose.

Baselines. We compare to a number of state-of-the-art GAN, disentanglement, and clustering methods. For all methods, we use the authors' public code. The code for SC-GAN [20] only has the unconditional version, so we implement its BiGAN [6] variant following the paper details.

Implementation details. We train and generate 128 × 128 images. In feature mode (2nd stage) training, $\phi(y)$ is a learned distribution from the code mode (1st stage) and may not model the entire real feature distribution (e.g., due to mode collapse). Thus, we assume that patch-level features are better modeled, and apply a patch discriminator. For our feature mode, since the predicted object masks are often highly accurate, we can optionally directly stitch the foreground (if only changing background) or background (if only changing texture) from the corresponding reference image. When optimizing Eqn. 2, we add noise to D since the sampled c, p, b are one hot, while predicted $\hat{c}$, $\hat{p}$, $\hat{b}$ will never be one-hot. Full training details are in the supp.

4.1. Qualitative Results

Conditional mix-and-match image generation. We show results on CUB, Stanford Cars, and Stanford Dogs; see Fig. 3. The first three rows show the background, texture, and shape + pose reference (real) images from which our model extracts b, c, and p & z, respectively, while the fourth and fifth rows show MixNMatch's feature mode and code mode generation results, respectively.

Our feature mode results (4th row) demonstrate how well MixNMatch preserves shape and pose information from the reference images (3th rows), while transferring background and texture information (from 1st and 2nd rows). For example, the generated bird in the second column preserves the exact pose and shape of the bird standing on the pipe (3rd row) and transfers the brownish bark background and rainbow object texture from the 1st and 2nd row images, respectively. Our code mode results (5th row) also capture the different factors from the reference images well, though not as well as the feature mode for pose and shape. Thus, this mode is more useful for applications in which inexact instance-level pose and shape transfer is acceptable (e.g., generating a completely new instance which captures the factors at a high-level). Overall, these results highlight how well MixNMatch disentangles and encodes factors from real images, and preserves them in the generation.

Note that here we take both z and p from the same reference images (row 3) in order to perform a direct comparison between the code and feature modes. We next show results of disentangling all four factors, including z and p.

Disentanglement of factors. Here we evaluate how well MixNMatch disentangles each factor (background b, texture c, pose z, shape p). Fig. 4 shows our disentanglement of each factor on CUB (results for Dogs and Cars are in the supp.). For each subfigure, the images in the top row and leftmost column (with red borders) are real reference images. The specific factors taken from each image are indicated in the top-left corner; e.g., in (a), pose is taken from the top row, while background, shape, texture are taken from the leftmost column. Note how we can make (a) a bird change poses by varying z, (b) change just the background by varying b, (c) colorize by varying c, and (d) change shape by varying p (e.g., see the duck example in 3rd column). As described in Sec. 3.4, our feature mode can preserve pixel-level shape+pose information from a reference image (i.e., both p and z are extracted from it) in the generation. Thus, for this experiment, (b) and (c) are results of feature mode, while (a) and (d) are results of code mode.

sketch2color / cartoon2img. We next try adapting MixNMatch to other domains not seen during training; sketch (Fig. 5) and cartoon (Fig. 6). Here we use our feature mode as it can preserve pixel-level shape+pose information. Interestingly, the results indicate that MixNMatch learns *part* information without supervision. For example, in Fig. 6 column 2, it can correctly transfer the black, white, and red part colors to the rubber duck.

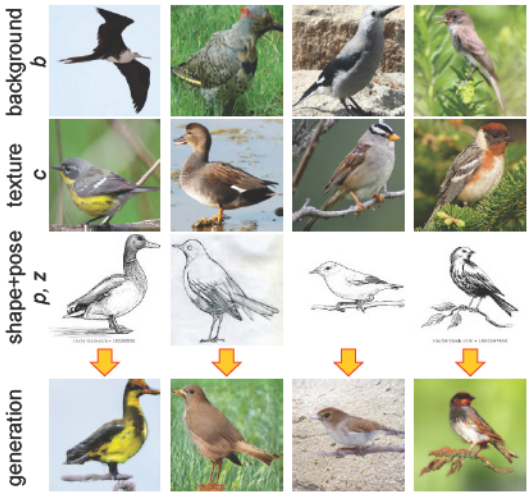


Figure 5: **sketch2color.** First three rows are real reference images. Last row shows generation results of adding background and texture to the sketch images.

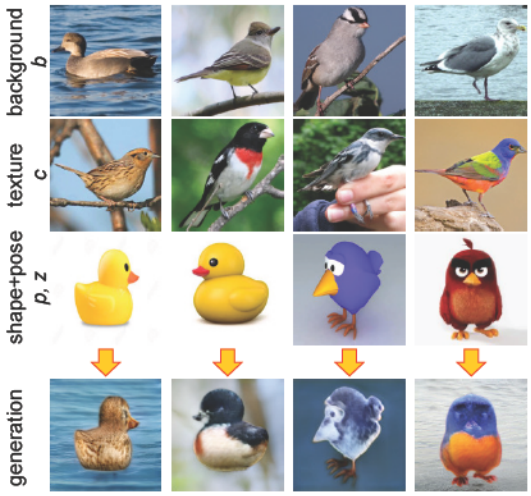


Figure 6: **cartoon2img.** MixNMatch automatically learns part semantics, *without supervision*; e.g., in the 2nd column, the colors of the texture reference are accurately transferred.

	Inception Score			FID		
	Birds	Dogs	Cars	Birds	Dogs	Cars
Simple-GAN	31.85 ± 0.17	6.75 ± 0.07	20.92 ± 0.14	16.69	261.85	33.35
InfoGAN [4]	47.32 ± 0.77	43.16 ± 0.42	28.62 ± 0.44	13.20	29.34	17.63
LR-GAN [45]	13.50 ± 0.20	10.22 ± 0.21	5.25 ± 0.05	34.91	54.91	88.80
StackGANv2 [48]	43.47 ± 0.74	37.29 ± 0.56	$\mathbf{33.69 \pm 0.44}$	13.60	31.39	16.28
FineGAN [36]	$\mathbf{52.53} \pm 0.45$	46.92 ± 0.61	32.62 ± 0.37	11.25	25.66	16.03
MixNMatch (Ours)	50.05 ± 0.75	$\mathbf{46.97 \pm 0.51}$	31.12 ± 0.62	**9.17**	**24.24**	**6.48**

Table 1: **Image quality & diversity.** IS (↑ better) and FID (↓ better). MixNMatch generates diverse, high-quality images that compare favorably to state-of-the-art baselines.

	NMI			Accuracy		
	Birds	Dogs	Cars	Birds	Dogs	Cars
JULE [46]	0.204	0.142	0.232	0.045	0.043	0.046
JULE-ResNet-50 [46]	0.203	0.148	0.237	0.044	0.044	0.050
DEPICT [10]	0.290	0.182	0.329	0.061	0.052	0.063
DEPICT-Large [10]	0.297	0.183	0.330	0.061	0.054	0.062
FineGAN [10]	0.403	0.233	0.354	0.126	0.079	0.078
MixNMatch (Ours)	**0.422**	**0.324**	**0.357**	**0.136**	**0.089**	**0.079**

Table 2: **Fine-grained object clustering.** Our approach outperforms state-of-the-art clustering methods.

img2gif. MixNMatch can also be used to animate a static image; see Fig. 7 (code mode result) and supp. video.

4.2. Quantitative Results

Image diversity and quality. We compute Inception Score [34] and FID [14] over 30K randomly generated images. We condition the generation only on sampled latent codes (by sampling z, p, c, b from their prior distributions; see Footnote 1), and not on real image inputs, for a fair comparison to the baselines. Table 1 shows that MixNMatch generates diverse and realistic images that are competitive to state-of-the-art unconditional GAN methods.

Fine-grained object clustering. We next evaluate MixNMatch's learned representation for clustering real images into fine-grained object categories. We compare to state-of-the-art deep clustering methods: **FineGAN** [36], **JULE** [46], and **DEPICT** [10], and their stronger variants [36]: **JULE-Res50** and **DEPICT-Large**. For evaluation metrics, we use Normalized Mutual Information (**NMI**) [43] and **Accuracy** [10], which measures the best mapping between predicted and ground truth labels. All methods cluster the same bounding box cropped images.

To cluster real images, we use MixNMatch's p (shape) and c (texture) encoders as fine-grained feature extractors. For each image, we concatenate its L2-normalized penultimate features, and run k-means clustering with k = # of ground-truth classes. MixNMatch's features lead to significantly more accurate clusters than the baselines; see Table 2. JULE and DEPICT focus more on background and rough shape information instead of fine grained details, and thus have relatively low performance. FineGAN performs much better, but it trains the encoders post-hoc on *fake images* to repredict their corresponding latent codes (as it cannot directly condition its generator on real images) [36]. Thus, there is a domain gap to the real image domain. In contrast, MixNMatch's encoders are trained to extract features from both real and fake images, so it does not suffer from domain differences.

Shape and texture disentanglement. In order to quantitatively evaluate MixNMatch's disentanglement of shape and texture, we propose the following evaluation metric: We randomly sample 5000 image pairs (A, B) and generate new images C, which take texture and background (codes c, b) from image A, and shape and pose from image B (codes p, z). If a model disentangles these factors well *and* preserves them in the generated images, then the spatial posi-

Figure 7: **image2gif.** MixNMatch can combine the pose factor z from a reference video (top row), with the other factors in a static image (1st column) to animate the object.

tion of part keypoints (e.g., beak, tail) in B should be close to that in C, while the texture around those keypoints in A should be similar to that in C; see Fig. 8.

To measure how well shape is preserved, we train a keypoint detector [13] on CUB, and use it to detect 15 keypoints in generated images C. We then calculate the L2-distance (in x,y coordinate space) to the corresponding visible keypoints in B. To measure how well texture is preserved, for each keypoint in A and C, we first crop a 16x16 patch centered on it. We then compute the χ^2-distance between the L1-normalized color histograms of the corresponding patches in A and C. See supp. for more details.

Table 3 (top) shows the results averaged over all 15 keypoints among all 5000 image triplets. We compare to **FineGAN** [36], **SC-GAN** [20], a generative model that disentangles style (texture) and content (geometrical information), and **Deforming AE** [35], a generative autoencoder that disentangles shape and texture from real images via unsupervised deformation constraints. Fig. 8 shows qualitative comparisons. Clearly, MixNMatch better disentangles and preserves shape and texture compared to the baselines. SC-GAN does not differentiate background and foreground and uses a condensed code space to model content and style, so it has difficulty transferring texture and shape accurately. Deforming AE fails because its assumption that an image can be factorized into a canonical template and a deformation field is difficult to realize in complicated shapes such as birds. FineGAN performs better, but it again is hindered by the domain gap. Finally, our feature mode has the best performance for shape disentanglement due to its ability of preserving instance-specific shape and pose details.

Ablation studies. Finally, we study MixNMatch's various components: 1) **no paired image-code adversarial loss**, where we do not have Eqn. 2, instead we directly feed the predicted code from encoder to the generator, and apply an L1 loss between the generated and real images; 2) **without code reprediction loss**, where we do not apply Eqn. 3; 3) **with code reprediction loss but with code constraints**, where during generating fake images, we keep FineGAN's

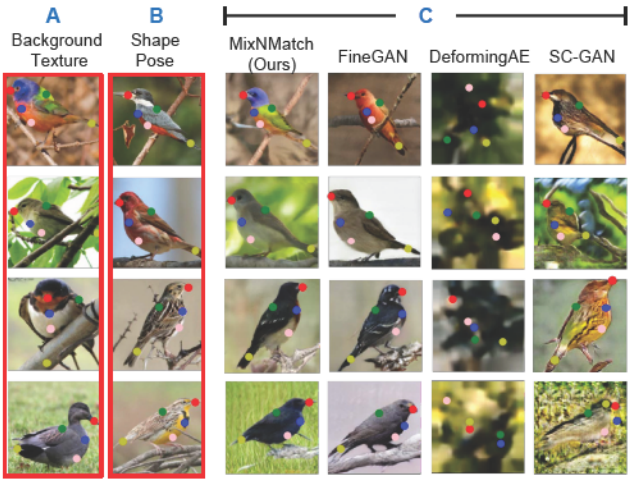


Figure 8: **Shape & texture disentanglement.** Our approach preserves shape, texture better than strong baselines.

	Shape	Texture
Deforming AE [35]	69.97	0.792
SC-GAN [20]	32.37	0.641
FineGAN [36]	21.04	0.602
MixNMatch (code mode)	20.57	**0.540**
MixNMatch (feature mode)	**16.29**	0.565
Code mode w/o paired adv loss	60.41	0.798
Code mode w/o code reprediction	47.67	0.724
Code mode w/ code constraint	26.95	0.601
Feature mode w/o L1 loss	61.76	0.575
Feature mode w/o adv loss	17.61	0.572

Table 3: **Shape & texture disentanglement.** (Top) Comparisons to baselines. (Bottom) Ablation studies. We report keypoint L2-distance and color histogram χ^2-distance for measuring shape and texture disentanglement (↓ better).

code constraints; 4) **without feature mode L1 loss**, where we only apply an adversarial loss between $S(x)$ and $\phi(y)$; 5) **without feature mode adversarial loss**, where we only have the L1 loss in feature mode training.

Table 3 (bottom) shows that all losses are necessary in code mode training; otherwise, disentanglement cannot be learned properly. In feature mode training, both adversarial and L1 losses are helpful, as they adapt the model to the real image domain to extract precise shape + pose information.

Discussion. There are some limitations worth discussing. First, our generated background may miss large structures, as we use a patch-level discriminator. Second, the feature mode training, depends on, and is sensitive to, how well the model is trained in the code mode. Finally, for reference images whose background and object texture are very similar, our model can fail to produce a good object mask, and thus generate an incomplete object.

Acknowledgments. This work was supported in part by NSF CAREER IIS-1751206, IIS-1748387, IIS-1812850, AWS ML Research Award, Adobe Data Science Research Award, and Google Cloud Platform research credits.

References

[1] Guha Balakrishnan, Amy Zhao, Adrian Dalca, Fredo Durand, and John Guttag. Synthesizing images of humans in unseen poses. In *CVPR*, 2018.

[2] Jianmin Bao, Dong Chen, Fang Wen, Houqiang Li, and Gang Hua. Towards open-set identity preserving face synthesis. In *CVPR*, 2018.

[3] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale GAN training for high fidelity natural image synthesis. In *ICLR*, 2019.

[4] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *NeurIPS*, 2016.

[5] Emily Denton and Vighnesh Birodkar. Unsupervised learning of disentangled representations from video. In *NeurIPS*, 2017.

[6] Jeff Donahue, Philipp Krähenbühl, and Trevor Darrell. Adversarial feature learning. In *ICLR*, 2017.

[7] Jeff Donahue and Karen Simonyan. Large scale adversarial representation learning. In *NeurIPS*, 2019.

[8] Vincent Dumoulin, Ishmael Belghazi, Ben Poole, Alex Lamb, Martin Arjovsky, Olivier Mastropietro, and Aaron Courville. Adversarially learned inference. In *ICLR*, 2017.

[9] Patrick Esser, Ekaterina Sutter, and Björn Ommer. A variational u-net for conditional appearance and shape generation. In *CVPR*, 2018.

[10] Kamran Ghasedi Dizaji, Amirhossein Herandi, Cheng Deng, Weidong Cai, and Heng Huang. Deep clustering via joint convolutional autoencoder embedding and relative entropy minimization. In *ICCV*, 2017.

[11] Abel Gonzalez-Garcia, Joost van de Weijer, and Yoshua Bengio. Image-to-image translation for cross-domain disentanglement. In *NeurIPS*, 2018.

[12] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NeurIPS*, 2014.

[13] Kaiming He, Georgia Gkioxari, Piotr Dollar, and Ross Girshick. Mask r-cnn. In *ICCV*, 2017.

[14] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, Gunter Klambauer, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *NeurIPS*, 2017.

[15] Qiyang Hu, Attila Szabó, Tiziano Portenier, Paolo Favaro, and Matthias Zwicker. Disentangling factors of variation by mixing them. In *CVPR*, 2018.

[16] Xun Huang, Ming-Yu Liu, Serge J. Belongie, and Jan Kautz. Multimodal unsupervised image-to-image translation. In *ECCV*, 2018.

[17] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. Image-to-image translation with conditional adversarial networks. *CVPR*, 2017.

[18] Donggyu Joo, Doyeon Kim, and Junmo Kim. Generating a fusion image: One's identity and another's shape. In *CVPR*, 2018.

[19] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *CVPR*, 2019.

[20] Hadi Kazemi, Seyed Mehdi Iranmanesh, and Nasser M. Nasrabadi. Style and content disentanglement in generative adversarial networks. In *WACV*, 2018.

[21] Aditya Khosla, Nityananda Jayadevaprakash, Bangpeng Yao, and Li Fei-Fei. Novel dataset for fine-grained image categorization. In *First Workshop on Fine-Grained Visual Categorization*, 2011.

[22] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *IEEE Workshop on 3D Representation and Recognition (3dRR-13)*, 2013.

[23] Hsin-Ying Lee, Hung-Yu Tseng, Jia-Bin Huang, Maneesh Singh, and Ming-Hsuan Yang. Diverse image-to-image translation via disentangled representations. In *ECCV*, 2018.

[24] Zejian Li, Yongchuan Tang, and Yongxing He. Unsupervised disentangled representation learning with analogical relations. In *IJCAI*, 2018.

[25] Alexander Liu, Yen-Cheng Liu, Yu-Ying Yeh, and Yu-Chiang Frank Wang. A unified feature disentangler for multi-domain image translation and manipulation. In *NeurIPS*, 2018.

[26] Dominik Lorenz, Leonard Bereska, Timo Milbich, and Björn Ommer. Unsupervised part-based disentangling of object shape and appearance. In *CVPR*, 2019.

[27] Liqian Ma, Qianru Sun, Stamatios Georgoulis, Luc Van Gool, Bernt Schiele, and Mario Fritz. Disentangled person image generation. In *CVPR*, 2018.

[28] Takeru Miyato and Masanori Koyama. cgans with projection discriminator. In *ICLR*, 2018.

[29] Augustus Odena, Christopher Olah, and Jonathon Shlens. Conditional image synthesis with auxiliary classifier gans. In *ICML*, 2017.

[30] Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. Semantic image synthesis with spatially-adaptive normalization. In *CVPR*, 2019.

[31] Xi Peng, Xiang Yu, Kihyuk Sohn, Dimitris N Metaxas, and Manmohan Chandraker. Reconstruction-based disentanglement for pose-invariant face recognition. In *ICCV*, 2017.

[32] Albert Pumarola, Antonio Agudo, Aleix M Martinez, Alberto Sanfeliu, and Francesc Moreno-Noguer. Ganimation: Anatomically-aware facial animation from a single image. In *ECCV*, 2018.

[33] Scott E. Reed, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran, Bernt Schiele, and Honglak Lee. Generative adversarial text to image synthesis. In *ICML*, 2016.

[34] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *NeurIPS*, 2016.

[35] Zhixin Shu, Mihir Sahasrabudhe, Riza Alp Güler, Dimitris Samaras, Nikos Paragios, and Iasonas Kokkinos. Deforming autoencoders: Unsupervised disentangling of shape and appearance. In *ECCV*, 2018.

[36] Krishna Kumar Singh, Utkarsh Ojha, and Yong Jae Lee. FineGAN: Unsupervised hierarchical disentanglement for

fine-grained object generation and discovery. In *CVPR*, 2019.

[37] Luan Tran, Xi Yin, and Xiaoming Liu. Disentangled representation learning gan for pose-invariant face recognition. In *CVPR*, 2017.

[38] Sergey Tulyakov, Ming Yu-Liu, Xiadong Wang, and Jan Kautz. Mocogan: Decomposing motion and content for video generation. *CVPR*, 2018.

[39] Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The Caltech-UCSD Birds-200-2011 Dataset. Technical Report CNS-TR-2011-001, 2011.

[40] Fanyi Xiao, Haotian Liu, and Yong Jae Lee. Identity from here, pose from there: Self-supervised disentanglement and generation of objects using unlabeled videos. In *ICCV*, 2019.

[41] Xianglei Xing, Ruiqi Gao, Tian Han, Song-Chun Zhu, and Ying Nian Wu. Deformable generator network: Unsupervised disentanglement of appearance and geometry. In *CVPR*, 2018.

[42] Tao Xu, Pengchuan Zhang, Qiuyuan Huang, Han Zhang, Zhe Gan, Xiaolei Huang, and Xiaodong He. Attngan: Fine-grained text to image generation with attentional generative adversarial networks. In *CVPR*, 2018.

[43] Wei Xu, Xin Liu, and Yihong Gong. Document clustering based on non-negative matrix factorization. In *SIGIR*, 2003.

[44] Xinchen Yan, Jimei Yang, Kihyuk Sohn, and Honglak Lee. Attribute2image: Conditional image generation from visual attributes. In *ECCV*, 2016.

[45] Jianwei Yang, Anitha Kannan, Dhruv Batra, and Devi Parikh. Lr-gan: Layered recursive generative adversarial networks for image generation. *ICLR*, 2017.

[46] Jianwei Yang, Devi Parikh, and Dhruv Batra. Joint unsupervised learning of deep representations and image clusters. In *CVPR*, 2016.

[47] Guojun Yin, Bin Liu, Lu Sheng, Nenghai Yu, Xiaogang Wang, and Jing Shao. Semantics disentangling for text-to-image generation. In *CVPR*, 2019.

[48] Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaogang Wang, Xiaolei Huang, and Dimitris Metaxas. Stackgan++: Realistic image synthesis with stacked generative adversarial networks. *arXiv: 1710.10916*, 2017.

[49] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. *ICCV*, 2017.