Logo Synthesis and Manipulation with Clustered Generative Adversarial Networks

Anonymous CVPR submission

Paper ID 2998

Abstract

Designing a logo for a new brand is a lengthy and tedious back-and-forth process between a designer and a client. In this paper we explore to what extent machine learning can solve the creative task of the designer. For this, we build a dataset – LLD – of 600k+ logos crawled from the world wide web. Training Generative Adversarial Networks (GANs) for logo synthesis on such multi-modal data is not straightforward and results in mode collapse for some state-of-the-art methods. We propose the use of synthetic labels obtained through clustering to disentangle and stabilize GAN training. We are able to generate a high diversity of plausible logos and we demonstrate latent space exploration techniques to ease the logo design task in an interactive manner. Moreover, we validate the proposed clustered GAN training with synthetic labels on CIFAR10 and achieve state-of-the-art Inception scores. GANs can cope with multi-modal data by means of synthetic labels achieved through clustering and our results show the creative potential of such techniques for logo synthesis and manipulation. Our dataset and models will be made publicly available.

1. Introduction and related work

Logo design Designing a logo for a new brand usually is a lengthy and tedious process, both for the client and the designer. A lot of ultimately unused drafts are produced, from which the client selects his favorites, followed by multiple cycles refining the logo to the clients needs and wishes. Especially for those clients without a specific idea of the end product this results in a procedure that is not only time, but also cost intensive.

The goal of this work is to provide a framework towards a system with the ability to generate (virtually) infinitely many variations of logos (some are shown in Figure 1) to facilitate and expedite such a process. To this end, the prospective client should be able to modify a prototype logo according to specific parameters like shape, color etc. or shift it a certain amount towards the characteristics of another prototype. Such a system could help both designer and client to get an idea of a potential logo, which a professional designer could then build upon, even if the system itself was not (yet) able to output production-quality designs.

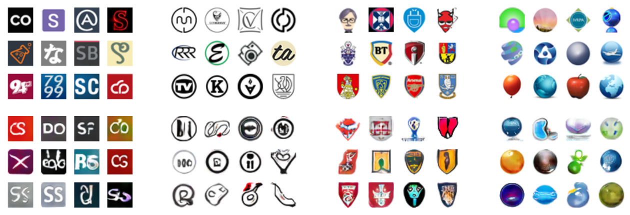


Figure 1: Original and generated images from four selected clusters from our LLD-icon-sharp dataset. The top three rows consist of original logos, followed by logos generated using our iWGAN-LC trained on 128 RC clusters.

Logo image data Existing research literature focused mostly on retrieval, detection, and recognition of a reduced number of logos [13, 16, 29, 31, 33, 41] and, consequently, a number of datasets were introduced. The most representative large public logo datasets are shown in Table 1. Due to the low diversity of the contained logos, these datasets are not suitable for learning and validating automatic logo generators. At the same time a number of web pages allow (paid) access to a large number of icons, such as iconsdb.com (4135+ icons), icons8.com (59900+), iconfinder.com (7473+), iconarchive.com (450k+) and thenounproject.com (1m+). However, the diversity of these icons is limited by the number of sources, namely designers/artists, themes (categories) and design patterns (many are black and white icons). Therefore, we crawl a highly diverse dataset – the Large Logo Dataset (LLD) – of real logos 'in the wild' from the Internet. As shown in Table 1 our LLD proposes thousands of times more logos than the largest public logo dataset to date, WebLogo-2M [33].

In contrast to popularly used natural image datasets such as ImageNet [30], CIFAR-10 [20] and LSUN [40], face datasets like CelebA [22] and the relatively easily modeled handwritten digits of MNIST [21], logos are: (1) *Artificial*, yet strongly multimodal and thus challenging for generative models; (2) *Applied*, as there is an obvious real-world demand for synthetically generated, unique logos since they are expensive to produce; (3) *Hard to label*, as there are very few categorical properties which manifest themselves in a logo's visual appearance. While the logos are easily obtainable in large quantities, they are specifically designed to be unique, which ensures the diversity of a large logo dataset. We argue that all these characteristics make logos a very attractive domain for machine learning research in general, and generative modeling in particular.

Generative models Recent advances in generative modeling have provided viable frameworks for making such a system possible. The current state-of-the-art is made up mainly of two types of generative models, namely Variational Autoencoders (VAEs) [15, 18, 19] and Generative Adversarial Networks (GANs) [1, 9, 10]. Both of these models generate their images from a high-dimensional latent space that can act as a sort of "design space" in which a user is able to modify the output in a structured way. VAEs have the advantage of directly providing embeddings of any given image in the latent space, so modifications could be made to its reconstruction, but tend to suffer from blurry output owed to the nature of the pixel-wise L_2 loss used during training. GANs on the other hand, which consist of a separate generator and discriminator network trained simultaneously on opposing objectives in a competitive manner, are known to provide realistic looking, crisp images but are notoriously unstable to train. To address this difficulty, a number of improvements in the architecture and training methods of GANs have been suggested [32], such as using deep convolutional layers [27] or modified loss functions e.g. based on least-squares [23] or the Wasserstein distance between probability distributions [2, 3, 11].

Conditional models The first extension of GANs with class-conditional information [24] followed shortly after its inception, generating MNIST digits conditioned on class labels provided to both generator and discriminator during training. Since then it has been proven for supervised datasets, that class-conditional variants of generative networks very often produce superior results compared to their unconditional counterparts [11, 14, 25]. By adding an encoder to map a real image into the latent space, it was shown to be feasible to generate a modified version of the original image by changing class attributes on faces [5, 26] and other natural images [35]. Other notable applications include the generation of images from a high-level description such as various visual attributes [38] or text descriptions [28].

Dataset	Logos	Images
FlickLogos-27 [17]	27	1080
FlickLogos-32 [29]	32	8240
BelgaLogos [16]	37	10000
LOGO-Net [13]	160	73414
WebLogo-2M [33]	194	1867177
LLD-icon (ours)	486377	486377
LLD-logo (ours)	122920	122920
LLD (ours)	486377+	609297

Table 1: Logo datasets. Our LLD provides orders of magnitude more logos than the existing public datasets.

Our contributions In this work we train GANs on our own highly multi-modal logo data as a first step towards user-manipulated artificial logo synthesis. Our main contributions are:

- LLD - a novel dataset of 600k+ logo images.
- Methods to successfully train GAN models on multi-modal data. Our proposed clustered GAN training achieves state-of-the-art Inception scores on CIFAR10 dataset.
- An exploration of GAN latent space for logo synthesis.

The remainder of this paper is structured as follows. We introduce a novel Large Logo Dataset (LLD) in Section 2. We describe the proposed clustered GAN training, the clustering methods, as well as the GAN architectures used and perform quantitative experiments in Section 3. Then we demonstrate logo synthesis by latent space exploration operations in Section 4. Finally, we draw the conclusions in Section 5.

2. LLD: Large Logo Dataset

In the following we introduce a novel dataset based on website logos, called the Large Logo Dataset (LLD). It is the largest logo dataset to date (see Table 1). LLD dataset consists of two parts, a low resolution (32×32 pixel) favicon subset (LLD-icon) and the higher-resolution (400×400 pixel) twitter subset (LLD-logo). In the following we will briefly describe the acquisition, properties and possible use-cases for each.

2.1. LLD-icon: Favicons

For generative models like GANs, the difficulty of keeping the network stable during training increases with image resolution. Thus when starting to work with a new type of data it makes sense to start off with a variant which is inherently low-resolution. Luckily, in the domain of logo images there is a category of such inherently low-resolution, low-complexity images: Favicons, the small icons representing a website e.g. in browser tabs or favorite lists. We decided to crawl the web for such favicons using the largest resource of high quality website URLs we could find, Alexa's top 1-million website list[1]. To this end we use the Python package Scrapy[2] in conjunction with our own download script which directly converts all icons found to a standardized 32×32 pixel resolution and RGB color space, discarding all non-square images.

After acquiring the raw data from the web, we remove all exact duplicates (of which there are a surprisingly high number of almost 20 %). Visual inspection of the raw data reveals a non-negligible number of images that do not comply to our initial dataset criteria and often are not even remotely logo-like, such as faces and other natural images. In an attempt to get rid of this unwanted data, we (i) sort all images by PNG-compressed file size – an image complexity indicator; (ii) manually inspect and partition the resulting sorted list into three sections: clean and mostly clean data which are kept, and mostly unwanted data which is discarded; (iii) discard the mostly clean images containing the least amount of white pixels.

The result of this process is a mostly clean set of 486,377 images of uniform 32×32 pixel size, making it very easy to use. The disadvantage of this standardized size is that 54 % of images appear blurry because they where scaled up from a lower resolution. For this reason we will also be providing (the indices for) a subset of the data containing only sharp images, which we will refer to as icons-sharp.

2.2. LLD-logo: Twitter

For training generative networks at an increased resolution, additional high-resolution data is needed, which favicons cannot provide. One possible option would be to crawl the respective websites directly to look for the website or company logo. However, (i) it might not always be straightforward to find the logo and distinguish it from other images on the website and (ii) the aspect ratio and resolution of logos obtained in this way will be very varied, which would necessitate extensive cropping and resizing, potentially degrading the quality of a large portion of logos.

By crawling twitter instead of websites, we are able to acquire standardized square 400×400 pixel profile images which can easily be accessed through the twitter API without the need for web scraping. We use the Python wrapper tweepy to search for the (sub-) domain names contained in the alexa list and match the original URL with the website provided in the twitter profile to make sure that we have found the right twitter user. The images are then run through a face detector to reject any personal twitter accounts and the remaining images are saved together with the twitter meta data such as user name, number of followers and description. For this part of the dataset, all original resolutions are kept as-is, where 80% are 400×400 pixel and the rest some lower resolution (details given in supplementary material).

The acquired images are analyzed and sorted with a combination of automatic and manual processing in order to get rid of unwanted and possibly sensitive images, resulting in 122,920 usable high-resolution logos of consistent quality with rich meta data from the respective twitter accounts. These logo images form the LLD-logo dataset.

3. Clustered GAN Training

We propose a method for stabilizing GAN training and achieving higher quality samples on unlabeled datasets by means of clustering (a) in the latent space of an autoencoder trained on the same data or (b) in the CNN feature space of a ResNet classifier trained on ImageNet. With both methods we are able to produce semantically meaningful clusters which improve GAN training.

In this Section we review the GAN architectures used in our study, describe the clustering methods based on Autoencoder latent space and ResNet features and discuss the quantitative experimental results.

3.1. GAN architectures

Our generative models are based on Deep Convolutional Generative Adversarial Networks (DCGAN) of Radford *et al.* [27] and improved Wasserstein GAN with gradient penalty (iWGAN) as proposed by Gulrajani *et al.* [11].

DCGAN For our DCGAN experiments, we use Taehoon Kim's TensorFlow implementation [3]. We train DCGAN exclusively on the low-resolution LLD-icon subset, for which it proved to be inherently unstable without using our clustering approach. We use the input blurring explained in the next section in all our DCGAN experiments. For details on hyper-parameters used, we refer the interested reader to the supplementary material.

iWGAN All our iWGAN experiments are based on the official TensorFlow repository by Gulrajani *et al.* [11][4]. We kept the default settings as provided by the authors. We exclusively use both 32- and 64-pixel ResNet architectures provided in the repository with the only major modifications being our conditioning method as described below. We also use linear learning rate decay (from the initial value to zero over the full training iterations) for all our experiments.

[1]now officially retired, formerly available at https://www.alexa.com

[2]https://scrapy.org/

[3]https://github.com/carpedm20/DCGAN-tensorflow

[4]https://github.com/igul222/improved_wgan_training

3.2. Clustering

As mentioned in the introductory Section 1, training a conditional GAN with labels is beneficial in terms of improved output quality over an unsupervised setting. In particular, we found DCGAN to be unstable with our icon dataset (LLD-icon) for resolutions higher than 10×10, and could stabilize it by introducing synthetic labels as described in this section. In addition to stabilizing GAN training, we are able to achieve state-of-the-art Inception scores (as proposed by Salimans *et al*. [32]) on CIFAR-10 using iWGAN with our synthetic labels produced by RC clustering as described below, and thus demonstrate quantitative evidence of a quality improvement using this approach in Section 3.4. Furthermore, the cluster labels subsequently allow us to have some additional control over the generated logos by generating samples from individual clusters or transforming an particular logo to inherit the specific attributes of another cluster as detailed in Section 4.

AE: AutoEncoder Clustering Our first proposed method for producing synthetic data labels is by means of clustering in the latent space z of an Autoencoder. We construct an Autoencoder consisting of a modified version of the GAN discriminator with $dim(z)$ outputs instead of one, and the unmodified GAN Generator acting as a decoder for the reconstruction of the image from the latent representation, as illustrated in Figure 2. This Autoencoder is trained using a simple L_2 loss between original and reconstructed image. The image data is then encoded to latent vectors, followed by a PCA dimensionality reduction and finally clustered using (minibatch) k-means. Using this approach with our logo data leads to clusters that are both semantically meaningful, as they are based on high-level AE features, and recognizable by the GAN because they where created using the same general network topology.

RC: ResNet Classifier Clustering For our second clustering method we leverage the learned features of a ImageNet classifier such as ResNet-50 by He *et al*. [12]. We feed our images to the classifier and extract the output of the final pooling layer of the network to get a 2048-dimensional feature vector. After a PCA dimensionality reduction we can cluster our data in this feature space with (minibatch) k-means. The obtained clusters are considerably superior than those of AE clustering method for CIFAR-10, where one could argue that we are benefiting from the similarity in categories between ImageNet and CIFAR-10 and are thus indirectly using labeled data. However, the clustering is very meaningful even for our logo dataset which has a very different content and does not consist of natural images like ImageNet, proving the generality of this approach.

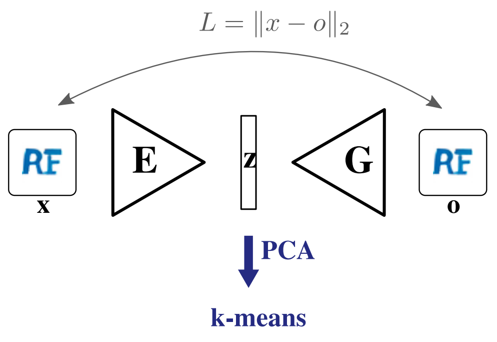


Figure 2: Autoencoder used for AE clustering. The generator G is equivalent to the one used in the GAN, while the encoder E consists of the GAN discriminator D with a higher number of outputs to match the dimensionality of the latent space z. It is trained using a simple L_2 loss function.

3.3. Conditional GAN Training Methods

In this section we describe the conditional GAN models use to leverage our synthetic data labels, as well the input blurring applied to DCGAN.

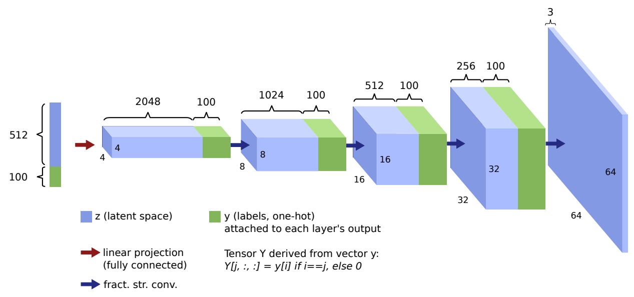

512
100
2048
100
4
4
1024
100
8
8
512
100
16
16
256
100
32
32
3
64
64
z (latent space)
y (labels, one-hot) attached to each layer's output
linear projection (fully connected)
Tensor Y derived from vector y:
Y[j, :, :] = y[i] if i==j, else 0
fract. str. conv.

Figure 3: Generator network as used for our layer conditional DCGAN (DCGAN-LC). 100 labels y are appended as a one-hot vector to the latent vector. It is also projected onto a set of feature maps consisting of all zeros except for the map corresponding to the class number, where all elements have value one. These additional feature maps are then appended to the input of each convolutional layer.

LC: Layer Conditional GAN In our layer-conditional models, the cluster label for each training sample is fed to all convolutional and linear layers of both generator and discriminator. For linear layers it is simply appended to the input as a one-hot vector. For convolutional layers the labels are projected onto "one-hot feature maps" with as many channels as there are clusters, where the one corresponding to the cluster number contains only ones while the rest are zero. These additional feature maps are appended to the input of every convolutional layer, such that every layer can

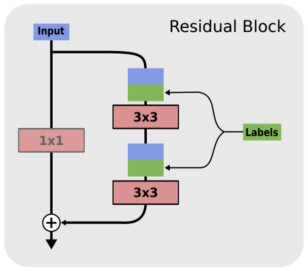


Figure 4: Layer Conditional Residual block as used in our iWGAN-LC. The label information is appended to the convolutional layer input in the same way as described in Figure 3. The skip connections remain unconditional.

directly access the label information. This is illustrated in Figure 3 for DCGAN and Figure 4 for ResNet as used in our iWGAN model. Even though the labels are provided to every layer, there is no explicit mechanism forcing the network to use this information. In case the labels are random or meaningless, they can simply be ignored by the network. However, as soon as the discriminator starts adjusting its criteria for different clusters, it forces the generator to produce images which comply with the different requirements for each class. Our experiments confirm that visually meaningful clusters are always picked up by the model, while the network simply falls back to the unconditional state for random labels. This type of class conditioning has some nice properties such as the ability to interpolate between different classes and is less prone to failure in producing class-conditional samples compared to the AC conditioning described below. However, it does come with the drawback of adding a significant number of parameters, especially to low-resolution networks, when there are a large number of classes. This effect diminishes with larger networks containing more feature maps, as the number of added parameters remains constant.

AC: Auxiliary Classifier GAN With iWGAN we also use the Auxiliary Classifier proposed by Odena *et al*. [25] as implemented by Glurajani *et al*. [11]. While this method does not allow us to interpolate between clusters and is thus slightly more limited from an application perspective, it does avoid adding parameters to the convolutional layers which in general results in a network with fewer parameters. iWGAN-AC was our method of choice for CIFAR-10, as it delivers the highest Inception scores.

Gaussian Blur During our experiments we noticed how blurring the input image helps the network remain stable during training, which in the end lead us to apply a Gaussian blur on all images presented to the discriminator (training data as well as samples from the Generator), like it has been previously implemented by Susmelj *et al*. [34]. The method

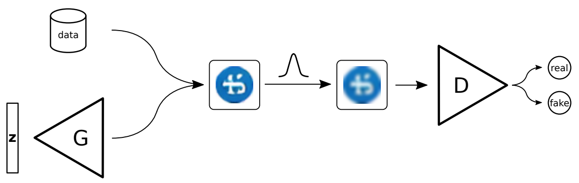


Figure 5: Generative Adversarial Net with blurred Discriminator input. Both original and generated images are blurred using a Gaussian filter of fixed strength.

is schematically illustrated in Figure 5. Upscaling the images to 64×64 pixel resolution before convolving them with the Gaussian kernel enables us to train with blurred images while preserving almost all of the image's sharpness when scaled back down to the original resolution of 32×32 pixel. When generating image samples from the trained Generator without applying the blur filter, there is some noticeable noise in the images, which becomes imperceptible after resizing to the original data resolution while producing perfectly sharp output images. Based on our experimental experience we believe this to produce higher quality samples and help stability, it is however not strictly necessary to achieve stability with DCGAN when using clustered training.

3.4. Quantitative evaluation and state-of-the-art

In order to quantitatively assess the performance of our solutions on the commonly used CIFAR-10 dataset we report Inception scores [32] and diversity scores based on MS-SSIM [36] as suggested in [25] over a set of 50000 randomly generated images. In Table 2 we summarize results for different configurations in supervised (using CIFAR class labels) and unsupervised settings in LC and AC conditional modes, including reported scores from the literature.

Clustering On CIFAR-10, increasing the number of RC clusters from 1 to 128 leads to better diversity scores for iWGAN-AC, at the same time the Inception score peaks above 32 clusters. We note that using RC clustering leads to better performance than using AE clustering.

Performance and state-of-the-art Our best Inception score of 8.67 achieved with iWGAN-AC and 32 RC clusters is significantly higher than 8.09 by Salimans *et al*. [32] with their Improved GAN method, the best score reported in the literature for unsupervised methods. Surprisingly, our best result, achieved with unsupervised synthetic labels provided by RC clustering, is comparable to 8.59 of the Stacked GANs approach by Huang *et al*. [14], the best

	Method	clusters	Inception score	Diversity (MS-SSIM)
unsupervised	Infusion training[4]		4.62±0.06	
	ALI [8](from[37])		5.34±0.05	
	Impr.GAN(-L+HA)[32]		6.86±0.06	
	EGAN-Ent-VI [6]		7.07±0.10	
	DFM [37]		7.72±0.13	
	iWGAN [11]		7.86±0.07	
	iWGAN		7.853±0.072	0.0504±0.0017
	iWGAN-LC with AE clustering	32	7.300±0.072	0.0507±0.0016
	iWGAN-LC with RC clustering	32	7.831±0.072	0.0491±0.0015
	iWGAN-AC with AE clustering	32	7.885±0.083	0.0504±0.0014
	iWGAN-AC with RC clustering	10	8.433±0.068	0.0505±0.0016
	iWGAN-AC with RC clustering	32	**8.673±0.075**	0.0500±0.0016
	iWGAN-AC with RC clustering	128	**8.625±0.109**	**0.0465±0.0015**
supervised	iWGAN-LC		7.710±0.084	0.0510±0.0013
	Impr.GAN [32]		8.09±0.07	
	iWGAN-AC [11]		8.42±0.10	
	AC-GAN [25]		8.25±0.07	
	SGAN [14]		8.59±0.12	
	CIFAR-10 (original data)		11.237±0.116	0.0485±0.0016

Table 2: Inception scores and diversity scores comparison on CIFAR-10 benchmark. The unsupervised methods do not use the CIFAR-10 class labels. Note that our unsupervised methods achieve state-of-the-art performance comparable to the best supervised approaches.

score reported for supervised methods.

Image quality Complementary to the Inception and the diversity scores we also measured the image quality using CORNIA, a robust no-reference image quality assessment method proposed by Ye and Doermann [39]. On both CIFAR-10 and LLD-icon our generative models obtained CORNIA scores equivalent to those of the original images from each dataset. This result is in-line with the findings in [34], where the studied GANs also converge in terms of CORNIA scores towards the data image quality at GAN convergence.

LC vs. AC for conditional GANs Our AC-GAN variants are better than their LC counterparts in terms of Inception scores, but comparable in terms of diversity for CIFAR-10. We believe that this is owed to fact that AC-GAN enforces the generation of images which can easily be classified to the provided clusters, which in turn could raise the classifier-based Inception score. Even though the numbers indicate a qualitative advantage of AC- over LC-GAN, we prefer the latter for our logo application as it allows smooth interpolation even in-between different clusters. This is not possible in the standard AC-GAN implementation since the cluster labels are discrete integer values and thus all our desirable latent space operations would be constrained to be performed within a specific data cluster, which does not match our intended use.

4. Logo synthesis by latent space exploration

As mentioned in the previous section, layer conditioning allows for smooth transitions in the latent space from one class to another, which is critical for logo synthesis and manipulation by exploration of the latent space. Therefore, we work with two configurations: iWGAN-LC with 128 RC clusters and DCGAN-LC with 100 AE clusters. Their Inception, diversity and CORNIA scores are comparable on LLD-icon dataset.

4.1. Sampling

In generative models like GANs [10] and VAEs [19], images are generated from a high-dimensional latent vector (with usually somewhere between 50 and 1000 dimensions), also commonly referred to as z-vector. During training, each component of this vector is randomly sampled from a Uniform or Gaussian distribution, so that the generator after training produces a reasonable output for any random vector sampled from the same distribution. The space spanned by these latent vectors, called the latent space, is often highly structured, such that latent vectors can be deliberately manipulated in order to achieve certain properties in the output [5, 7, 27].

Using DCGAN-LC with 100 AE clusters on the same data, Figure 6 contains samples from a specific cluster next to a sample of the respective original data. This shows how the layer conditional DCGAN is able to pick up on the data distribution and produce samples which are very easy to attribute to the corresponding cluster and are often hard to distinguish from the originals at first glance. For comparison we also show results for iWGAN-LC with 128 RC clusters trained on the LLD-icon-sharp dataset in Figure 1.

4.2. Interpolations

To show that a generator does not simply learn to reproduce samples from the training set, but is in fact able to produce smooth variations of its output images, it is common practice [9] to perform interpolations between two points in the latent space and to show that the outcome is a smooth transition between the two corresponding generated images, with all intermediate images exhibiting the same distribution and quality. Interpolation also provides an effective tool for a logo generator application, as the output image can be manipulated in a controlled manner towards a certain (semantically meaningful) direction in the latent space.

An example with 64 interpolation steps to showcase the smoothness of such an interpolation is given in Figure 7 where we interpolate between 4 sample points, producing believable logos at every step. As it is the case in this example, the interpolation works very well even between logos of different clusters, even though the generator was never trained for mixed cluster attributes.

Some more interpolations between different kinds of logos both within a single cluster and between logos of different clusters are shown in Figure 8, this time between 2 endpoints and with only 8 interpolation steps.

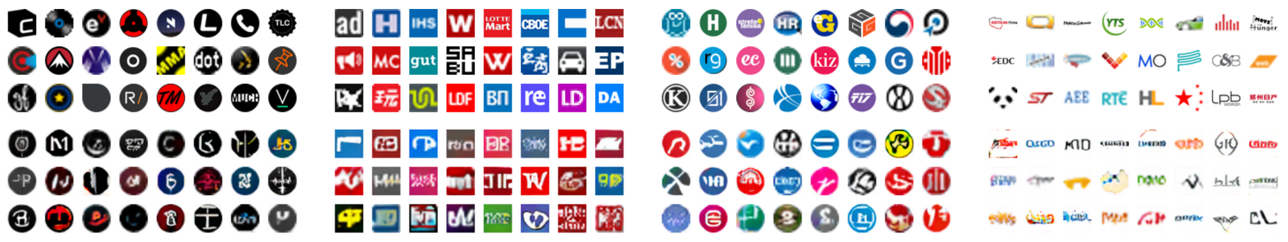


Figure 6: The first four (random) clusters of LLD-icon as attained with our AE-Clustering method using 100 cluster centers. The top half of each example contains a random selection of original images, while the bottom half consists of samples generated by DCGAN-LC for the corresponding cluster. The very strong visual correspondence demonstrates the network's ability to capture the data distributions inherent the classes produced by our clustering method.

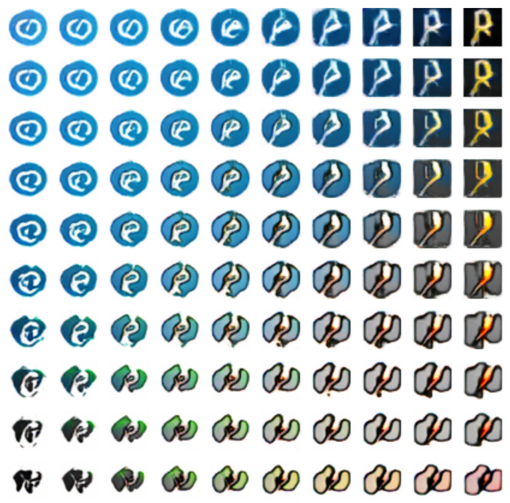


Figure 7: Interpolation between 4 selected logos of distinct classes using DCGAN-LC with 100 AE clusters on LLD-icon, showcasing smooth transitions and interesting intermediate samples in-between all of them.

4.3. Class transfer

As the one-hot class vector representing the logo cluster is separate from our latent vector, it is also possible to keep the latent space representation constant and only change the cluster of a generated logo. Figure 9 contains 11 logos (top row) that are being transformed to a particular cluster class in each subsequent row. This shows very nicely how the general appearance such as color and contents are encoded in the z-vector while the cluster label transforms these attributes into a form which conforms with the contents of the respective cluster. Here, again, interpolation could be used to create intermediate versions as desired.

4.4. Vicinity sampling

Another powerful tool to explore the latent space is vicinity sampling, where we perturb a given sample in random directions in latent space. This could be useful to present the user of a logo generator application with a choice of possible variants, allowing him to modify his logo

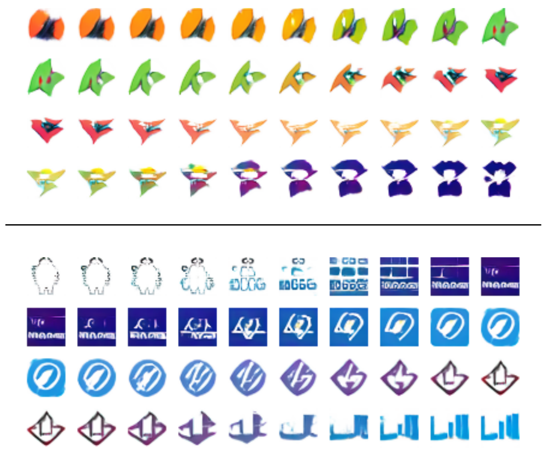


Figure 8: Continuous interpolation between 5 random points each within one cluster (top) and in-between distinct clusters (bottom) in latent space using iWGAN-LC with 128 RC clusters on icon-sharp. We observe smooth transitions and logo-like samples in all of the sampled subspace.

step by step into directions of his choice. In Figure 10 we present an example of a 2-step vicinity sampling process, where we interpolate one-third towards random samples to produce a succession of logo variants.

4.5. Vector arithmetic example: Sharpening

For models trained on our LLD-icon data, some of the generated icons are blurry because roughly half of the logos in this dataset are upscaled from a lower resolution. However, by averaging over the z-vector of a number of blurry samples and subtracting from this the average of a number of sharp samples, it is possible to construct a "sharpening" vector which can be added to blurry logos to transform them into sharp ones. This works very well even if the directional vector is calculated exclusively from samples in one cluster and then applied samples of another, showing that the blurriness is in fact nothing more than another feature embedded in the latent space. The result of such a transformation is shown in Figure 11, where such a sharpening vector was calculated from 40 sharp and 42 blurry samples manually

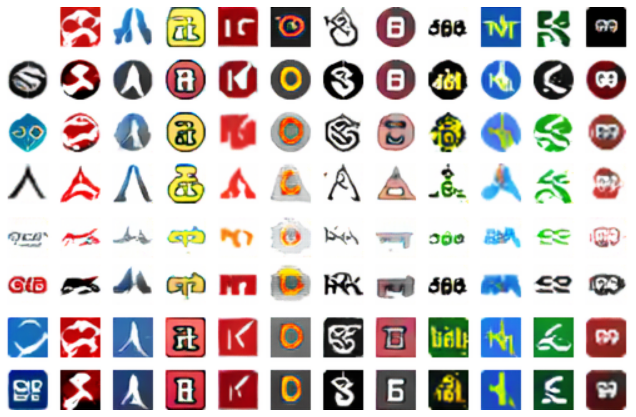


Figure 9: Logo class transfer using DCGAN-LC on LLD-icon with 100 AE clusters. The logos of 1st row get transferred to the class (cluster) of the logos in the 1st column (to the left). Hereby the latent vector is kept constant within each column and the class label is kept constant within each row (except for the 1st ones, resp.). The original samples have been hand-picked for illustrative purposes.

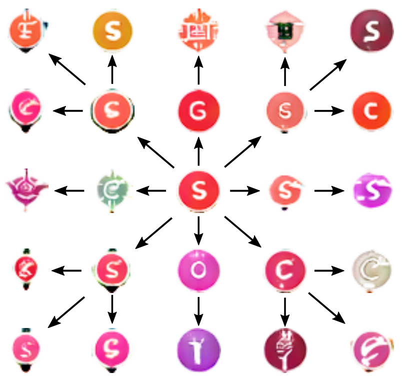


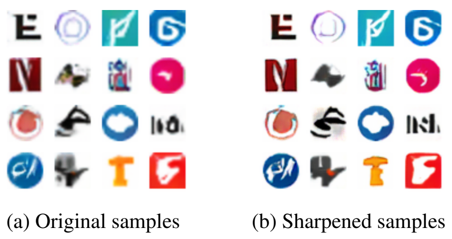
Figure 10: Vicinity Sampling using iWGAN-LC on LLD-icon-sharp with 128 RC clusters.

selected from two random batches of the same cluster. The resulting vector is then applied equally to all blurry samples. The quality of the result, while already visually convincing, could be further optimized by adding individually adjusted fractions of this sharpening vector to each logo.

This example of adding a sharpening vector to the latent representation is only one of many latent space operations one could think of, such as directed manipulation of form, color and other image content.

4.6. Supplementary material

Due to page length limit we invite the reader to check the supplementary material for more visual results of our approaches, including higher resolution results from our models trained on the LLD-logo subset and results for operations such as vicinity sampling and vector arithmetic examples. Furthermore, we provide an example for a user interface for a logo generator application which implements latent space operations for an easy manipulation of logo attributes.



(a) Original samples (b) Sharpened samples

Figure 11: Sharpening of logos in the latent space by adding an offset calculated from the latent vectors of sharp and blurry samples. We used DCGAN-LC and 100 AE clusters.

5. Conclusions

In this paper we tackled the problem of logo design by synthesis and manipulation with generative models:

- **(i)** We introduced a Large Logo Dataset (LLD) crawled from Internet with orders of magnitude more logos than the existing datasets.
- **(ii)** In order to cope with the high multi-modality and to stabilize GAN training on such data we proposed clustered GANs, that is GANs conditioned with synthetic labels obtained through clustering. We performed clustering in the latent space of an Autoencoder or in the CNN features space of a ResNet classifier and conditioned DCGAN and improved WGAN utilizing either an Auxiliary Classifier or Layer Conditional model.
- **(iii)** We quantitatively validated our clustered GAN approaches on a CIFAR-10 benchmark where we set a clear state-of-the-art Inception score for unsupervised generative models, showcasing the benefits of meaningful synthetic labels obtained through clustering in the CNN features space of a ResNet classifier.
- **(iv)** We showed that the latent space of the networks trained on our logo data is smooth and highly structured, thus having interesting properties exploitable by performing vector arithmetic in that space.
- **(v)** We showed that the synthesis and manipulation of (virtually) infinitely many variations of logos is possible through latent space exploration equipped with a number of operations such as interpolations, sampling, class transfer or vector arithmetic in latent space like our sharpening example.

Our solutions ease logo design task in an interactive manner and are significant steps towards a fully automatic logo design system.

For more results, operations, and settings the reader is invited to consult the supplementary material.

References

[1] M. Arjovsky and L. Bottou. Towards principled methods for training generative adversarial networks. *arXiv preprint arXiv:1701.04862*, 2017. 2

[2] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein generative adversarial networks. In D. Precup and Y. W. Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 214–223, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR. 2

[3] D. Berthelot, T. Schumm, and L. Metz. Began: Boundary equilibrium generative adversarial networks. *arXiv preprint arXiv:1703.10717*, 2017. 2

[4] F. Bordes, S. Honari, and P. Vincent. Learning to generate samples from noise through infusion training. *arXiv preprint arXiv:1703.06975*, 2017. 6

[5] A. Brock, T. Lim, J. M. Ritchie, and N. Weston. Neural photo editing with introspective adversarial networks. *arXiv preprint arXiv:1609.07093*, 2016. 2, 6

[6] Z. Dai, A. Almahairi, P. Bachman, E. Hovy, and A. Courville. Calibrating energy-based generative adversarial networks. *arXiv preprint arXiv:1702.01691*, 2017. 6

[7] A. Dosovitskiy, J. Tobias Springenberg, and T. Brox. Learning to generate chairs with convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1538–1546, 2015. 6

[8] V. Dumoulin, I. Belghazi, B. Poole, A. Lamb, M. Arjovsky, O. Mastropietro, and A. Courville. Adversarially learned inference. *arXiv preprint arXiv:1606.00704*, 2016. 6

[9] I. Goodfellow. Nips 2016 tutorial: Generative adversarial networks. *arXiv preprint arXiv:1701.00160*, 2016. 2, 6

[10] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014. 2, 6

[11] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville. Improved training of wasserstein gans. *arXiv preprint arXiv:1704.00028*, 2017. 2, 3, 5, 6

[12] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 4

[13] S. C. HOI, X. WU, H. LIU, Y. Wu, H. Wang, H. Xue, and Q. Wu. Large-scale deep logo detection and brand recognition with deep region-based convolutional networks. In *arXiv*, 2015. 1, 2

[14] X. Huang, Y. Li, O. Poursaeed, J. Hopcroft, and S. Belongie. Stacked generative adversarial networks. *arXiv preprint arXiv:1612.04357*, 2016. 2, 5, 6

[15] D. Jimenez Rezende, S. Mohamed, and D. Wierstra. Stochastic backpropagation and approximate inference in deep generative models. *arXiv preprint arXiv:1401.4082*, 2014. 2

[16] A. Joly and O. Buisson. Logo retrieval with a contrario visual query expansion. In *Proceedings of the 17th ACM international conference on Multimedia*, pages 581–584. ACM, 2009. 1, 2

[17] Y. Kalantidis, L. G. Pueyo, M. Trevisiol, R. van Zwol, and Y. Avrithis. Scalable triangulation-based logo recognition. In *Proceedings of the 1st ACM International Conference on Multimedia Retrieval*, page 20. ACM, 2011. 2

[18] D. P. Kingma, S. Mohamed, D. J. Rezende, and M. Welling. Semi-supervised learning with deep generative models. In *Advances in Neural Information Processing Systems*, pages 3581–3589, 2014. 2

[19] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013. 2, 6

[20] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. 2009. 1

[21] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 2

[22] Z. Liu, P. Luo, X. Wang, and X. Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, 2015. 2

[23] X. Mao, Q. Li, H. Xie, R. Y. Lau, Z. Wang, and S. P. Smolley. Least squares generative adversarial networks. *arXiv preprint ArXiv:1611.04076*, 2016. 2

[24] M. Mirza and S. Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014. 2

[25] A. Odena, C. Olah, and J. Shlens. Conditional image synthesis with auxiliary classifier gans. 2017. 2, 5, 6

[26] G. Perarnau, J. van de Weijer, B. Raducanu, and J. M. Álvarez. Invertible conditional gans for image editing. *arXiv preprint arXiv:1611.06355*, 2016. 2

[27] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015. 2, 3, 6

[28] S. Reed, Z. Akata, X. Yan, L. Logeswaran, B. Schiele, and H. Lee. Generative adversarial text to image synthesis. In M. F. Balcan and K. Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1060–1069, New York, New York, USA, 20–22 Jun 2016. PMLR. 2

[29] S. Romberg, L. G. Pueyo, R. Lienhart, and R. Van Zwol. Scalable logo recognition in real-world images. In *Proceedings of the 1st ACM International Conference on Multimedia Retrieval*, page 25. ACM, 2011. 1, 2

[30] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. 1

[31] H. Sahbi, L. Ballan, G. Serra, and A. Del Bimbo. Context-dependent logo matching and recognition. *IEEE Transactions on Image Processing*, 22(3):1018–1031, 2013. 1

[32] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved techniques for training gans. In *Advances in Neural Information Processing Systems*, pages 2234–2242, 2016. 2, 4, 5, 6

[33] H. Su, S. Gong, and X. Zhu. Weblogo-2m: Scalable logo detection by deep learning from the web. In *The IEEE Inter-*

national Conference on Computer Vision (ICCV), Oct 2017. 1, 2

[34] I. Susmelj, E. Agustsson, and R. Timofte. ABC-GAN: Adaptive blur and control for improved training stability of generative adversarial networks. 5, 6

[35] A. van den Oord, N. Kalchbrenner, L. Espeholt, O. Vinyals, A. Graves, et al. Conditional image generation with pixelcnn decoders. In *Advances in Neural Information Processing Systems*, pages 4790–4798, 2016. 2

[36] Z. Wang, E. P. Simoncelli, and A. C. Bovik. Multiscale structural similarity for image quality assessment. In *The Thrity-Seventh Asilomar Conference on Signals, Systems Computers, 2003*, volume 2, pages 1398–1402 Vol.2, Nov 2003. 5

[37] D. Warde-Farley and Y. Bengio. Improving generative adversarial networks with denoising feature matching. In *International Conference on Learning Representations*, 2017. 6

[38] X. Yan, J. Yang, K. Sohn, and H. Lee. *Attribute2Image: Conditional Image Generation from Visual Attributes*, pages 776–791. Springer International Publishing, Cham, 2016. 2

[39] P. Ye and D. Doermann. No-reference image quality assessment using visual codebooks. *IEEE Transactions on Image Processing*, 21(7):3129–3138, 2012. 6

[40] F. Yu, A. Seff, Y. Zhang, S. Song, T. Funkhouser, and J. Xiao. Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv:1506.03365*, 2015. 1

[41] G. Zhu and D. Doermann. Automatic document logo detection. In *Document Analysis and Recognition, 2007. ICDAR 2007. Ninth International Conference on*, volume 2, pages 864–868. IEEE, 2007. 1