Logo Synthesis and Manipulation with Clustered Generative Adversarial Networks
Supplementary Material

Alexander Sage
D-ITET, ETH Zurich
Switzerland
sagea@ee.ethz.ch

Eirikur Agustsson
D-ITET, ETH Zurich
Switzerland
aeirikur@vision.ee.ethz.ch

Radu Timofte
D-ITET, ETH Zurich
Merantix GmbH
radu.timofte@vision.ee.ethz.ch

Luc Van Gool
D-ITET, ETH Zurich
ESAT, KU Leuven
vangool@vision.ee.ethz.ch

1. Introduction

This is the supplementary material for our paper *Logo Synthesis and Manipulation with Clustered Generative Adversarial Networks*. The most recent version of this document can be found at

In Figure 1 we present an example interface of a logo generator which could be used to facilitate the design process of a new logo. The samples used here where generated by our iWGAN-LC model trained on LLD-logo at 64×64 pixels, clustered to 64 synthetic classes in the feature space of a ResNet classifier.

In Section 2, we list some additional quantitative results to back up our claims about image quality and diversity, followed by some details and statistics on the scraping outcome and final contents of LLD-icon and LLD-logo. After presenting two additional examples for vector arithmetic in latent space which could be useful for our logo application in Section 3, we proceed to show in Section 5, for each subset of our Large Logo Dataset, an excerpt of the collected data together with generated samples from selected GAN architectures and the clusters produced by the applied clustering methods. For CIFAR-10 we also show samples from our cluster-conditional CIFAR-10 models together with samples from the unconditional and supervised iWGAN variants in this section. Finally, we give some details on architecture and training hyper-parameters of our models in Section 6, including a larger reproduction of the our DCGAN architecture illustration in Figure 2.

2. Image quality and diversity scores

In Table 1 we give the CORNIA score as a measure of image quality for our LLD dataset (as Inception scores are not applicable) and MS-SSIM scores again to measure image diversity. As previously indicated, the numbers show that the image quality as well as the diversity both reach the same level as the original data on the respective dataset.

Method	Clusters	CORNIA score	Diversity (MS-SSIM)
DCGAN-LC with AE clustering	100	62.12±0.51	0.0475±0.0013
iWGAN-LC with AE clustering	100	60.24±0.61	0.0439±0.0010
*iWGAN		54.27±0.67	0.0488±0.0011
*iWGAN-LC with RC clustering	16	55.37±0.67	0.0490±0.0014
*iWGAN-LC with RC clustering	128	55.27±0.68	0.0484±0.0010
LLD-icon (original data)		61.00±0.62	0.0482±0.0014
*LLD-icon-sharp (original data)		55.37±0.67	0.0494±0.0011

Table 1: CORNIA scores and diversity scores for models trained on LLD-icon. The starred (*) models where trained on the subset LLD-icon-sharp where the vast majority of blurry icons have been removed.

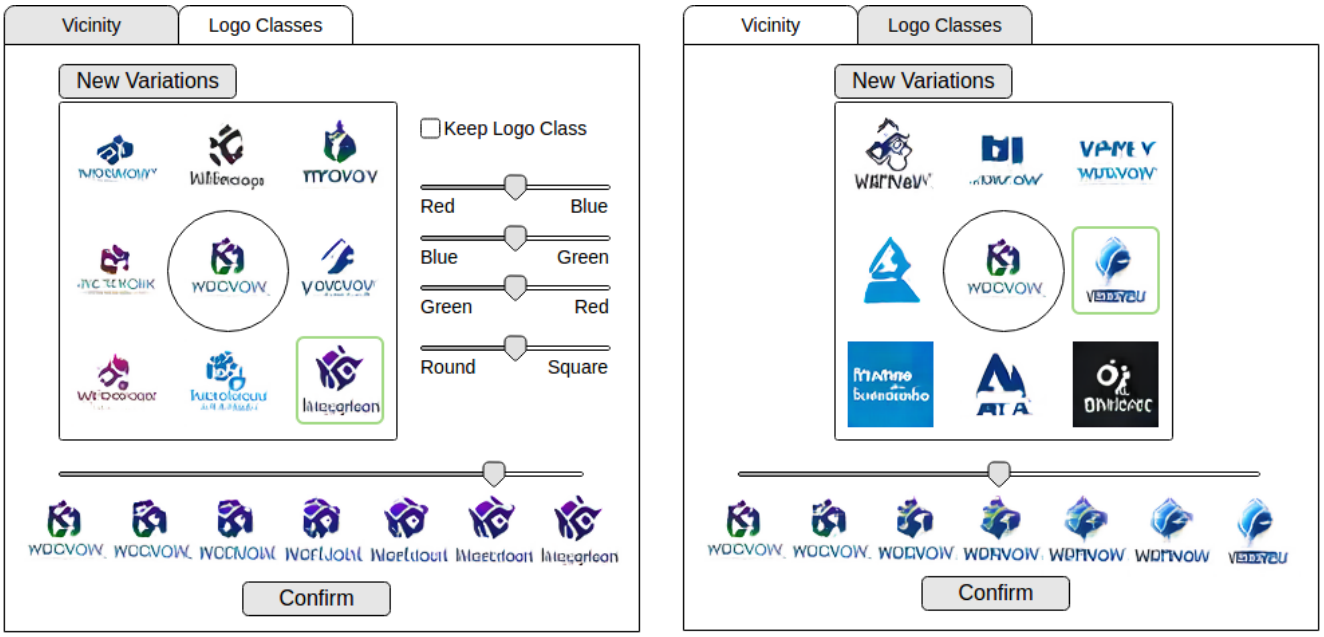


Figure 1: Logo generator interface. The user is able to choose either vicinity sampling or class transfer to modify the image in a chosen semantic direction. For both methods, 8 random variations are arranged around the current logo. Upon selecting the appropriate sample, the current logo can be modified by a variable amount using the slider at the bottom of the window. After confirming the selected modification, the process starts over again from the newly modified logo, until the desired appearance is reached. In addition to vicinity sampling within or across clusters, some pre-defined semantic modifications can be made using the sliders on the right hand side of the first view. The images used here are generated with iWGAN-LC trained at 6464 pixels on LLD-logo clustered to 64 different classes.

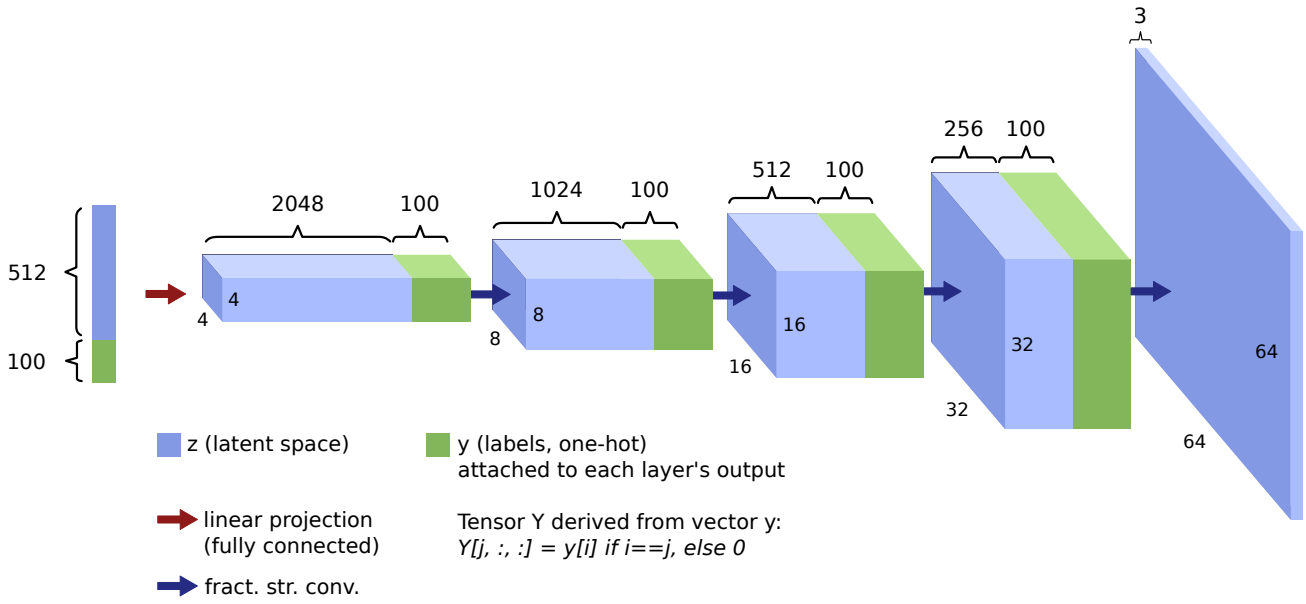


Figure 2: Generator network of the conditional DCGAN for 100 data clusters. The label information y is appended as a one-hot vector to the latent vector. It is also projected onto a set of feature maps consisting of all zeros except for the map corresponding to the class number, where all elements have value one. These additional feature maps are then appended to the output of each convolutional layer. The discriminator network works analog to this one, except that the data is flowing in the other direction, the fractionally strided convolutions are fully strided and there is only one output (real or fake).

3. Latent space exploration on LLD-logo

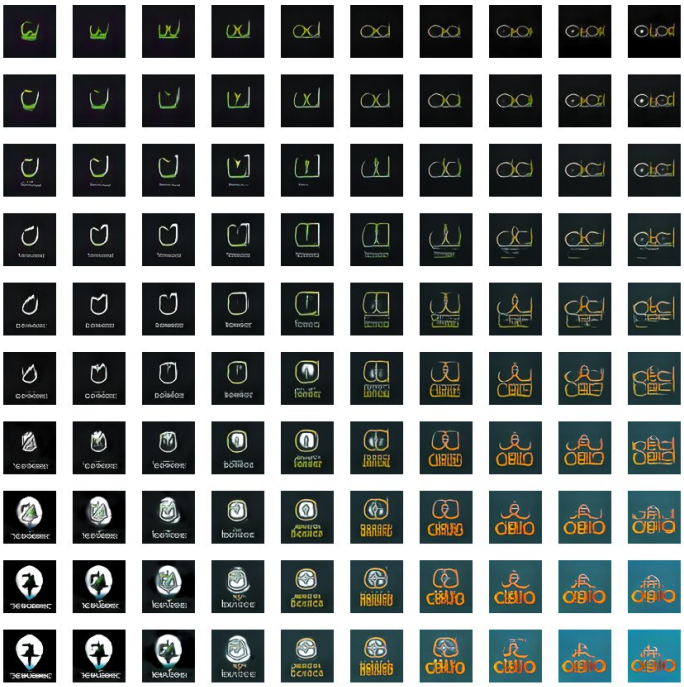
In this section, we present some interpolations on the LLD-logo dataset and perform two additional experiments with latent space operations.

Interpolation In Figure 3 we present two examples of interpolations between 4 different samples, representing a small section of the high-dimensional logo manifold created by the GAN.

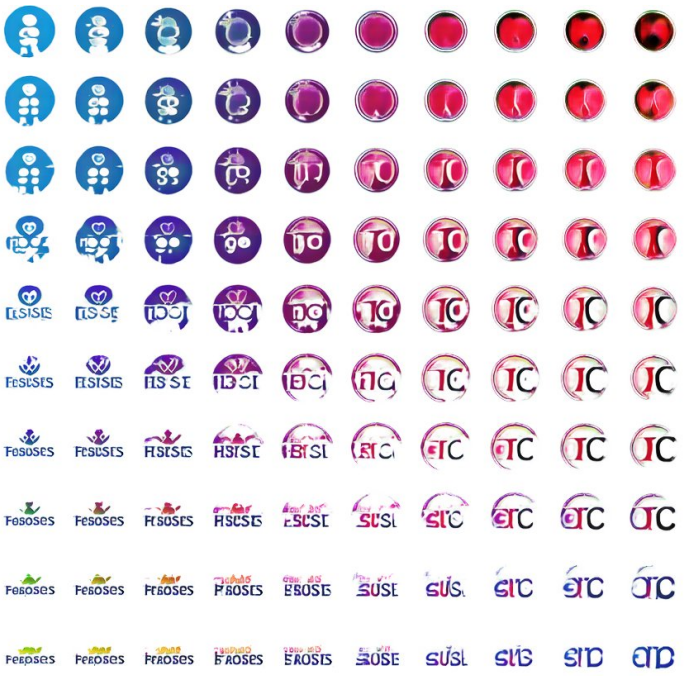
Vector arithmetic First, we define two desirable operations we would like to perform: (1) Color shifts from red to blue and blue to red and (2) Shape changes from square to round and round to square. For each of these semantic operations we identify a number (for our experiments around 30) of samples that match our criteria. To get operation (1) this means we select 30 red and 30 blue logos. We then construct a directional vector by subtracting the mean latent space vector of all blue logos from the mean latent space vector of all red logos, which gives us a directional vector from red to blue. Since some of these semantic attributes are expected to be encoded in the cluster labels as well, we can do the same with our one-hot encoded class vectors, which we can view as an additional cluster space. In Figure 4 we add this directional vector to a new random batch of generated logos. If we subtract the directional vector, we get a shift in the opposite direction, i.e. from blue to red. To find out how much of the color information is encoded in the latent representation and in the clusters respectively, we can perform the operation in only one of these domains. This is done in Figure 5 for the red-shift, where we observe a very similar behavior for both spaces, indicating that the color information is equally encoded in both latent space and labels.

Our second experiment is performed in the same way, and the directional vector is applied to the same batch of samples. Figure 6, again, shows the result for a simultaneous addition of both (latent and class) vectors in each direction, whereas each space is considered individually in Figure 7 for the directional vectors towards round logos. Here we can observe that some logos respond better to the change in latent space, while others seem more responsive to a changing cluster label. Overall, the label information seems to be a little stronger in this case.

In both experiments, the combined shift clearly performs best, and could provide a powerful tools for logo manipulation and other applications.



(a) Interpolation between 4 square logos.



(b) Interpolation between logos of different shape.

Figure 3: Four-point interpolation on LLD-logo.

(a) Random Sample, unmodified.

(b) Sample from (a) shifted towards blue logos

(c) Sample from (a) shifted towards red logos

Figure 4: Blue-red shift on a random batch. Directional vectors are both applied in latent space and in cluster label space.

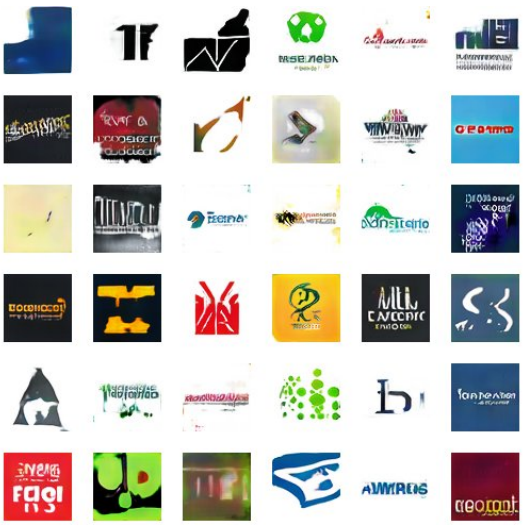
(a) Samples from Figure 4a shifted towards red logos only in latent vector space

(b) Samples from Figure 4a shifted towards red logos only in label vector space

Figure 5: Blue-red shift on a random batch performed in either latent representation or cluster labels.

(a) Random Sample, unmodified. (Same as Figure4a)

(b) Samples from Figure 4a shifted towards round logos.



(c) Samples from Figure 4a shifted towards square logos

Figure 6: Round-square shape shift on a random batch. Directional vectors are both applied in latent space and in cluster label space.

(a) Samples from Figure 6a shifted towards round logos only in latent vector space

(b) Samples from Figure 6a shifted towards round logos only in label vector space

Figure 7: Round-square shape shift on a random batch performed in either latent representation or cluster labels.

4. LLD crawling and image statistics

4.1. LLD-icon

When collecting the favicons for LLD-icon, our download script directly converted all icons found to a standardized 32x32 pixel resolution and RGB color space, discarding all non-square images. After acquiring the raw data from the web, we remove all exact duplicates and perform a three-stage clean-up process:

1. Sort all images by complexity by evaluating its PNG-compressed file size
2. Manually inspect and partition the resulting sorted list into three sections: Clean, mostly clean and mostly unwanted data. The last section is discarded, while the middle part (mostly clean) is further processed in the next step.
3. Sort the intermediate section according to the number of white pixels in each image and cut off at a certain point after inspection, discarding the images containing the least amount of white pixels.

Table 2 shows statistics on the crawling process, original image resolutions the icons where rescaled from, and numbers on content removed through our clean-up process.

4.2. LLD-logo

During the collection of LLD-logo on twitter, we use a face detector recognize faces and proceed to the next user in the search results if a face was detected. At the same time, we make use of twitters (relatively new) sensitive content flag to reject such flagged profiles. As the number of rejected profiles in Table 3 compared to the number of discarded images during cleanup (of which a substantial number where due to sensitive content) shows, this flag is only used very sporadically at this time, and is far from a reliable indicator. Figure 8 shows a histogram of image resolutions contained in LLD-icon (where no re-scaling was performed during data collection), with the top-5 image resolutions (amounting to 92% of images) given in Table 4.

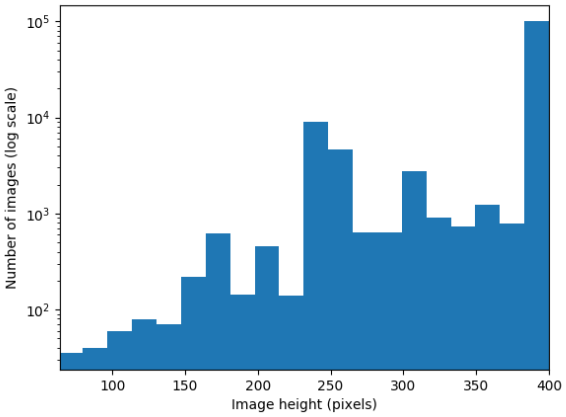


Figure 8: Histogram of image sizes in LLD-logo. There are a total of 329 different image resolutions contained in the dataset.

Failed requests	150,413	
Unreadable files	71,596	
Non-square images	36,401	
Unable to process	6	
Total images saved	662,273	
Image re-scaling		
Native 32 p	158,881	24.0%
Scaled up	355,260	53.6%
Scaled down	148,132	22.4%
Dataset cleanup		
Duplicates removed	114,063	17.2%
Discarded due to content	61,833	
Clean dataset size	486,377	

Table 2: Crawling statistics for LLD-icon

Flagged content ignored	1,066
Downloaded images	182,998
Discarded during cleanup	60,078
Final dataset size	122,920

Table 3: Crawling and clean-up statistics for LLD-icon

Image height (px)	Number of images	% of total
400	98,824	80.4%
240	8,625	7.0%
256	2,498	2.0%
300	2,143	1.7%
250	1,502	1.2%

Table 4: The 5 most prominent image resolutions in LLD-logo, covering 92.3% of the contained images.

5. Logo Data, clusters and generated samples

In this section, we will show a small sample from each of our introduced datasets and present generated icons from models trained on said dataset. Additionally, we show the data clusters produced by our clustering methods.

Since, due to the limited space available, it has not been featured in the main part of the paper, we start with LLD-logo. Figure 9 shows a sample of the original data collected (reduced to 64×64 pixels) next to the logos generated by an iWGAN model trained at 64×64 pixels. Compared to LLD-icon, these logos contain a lot more text and sometimes more detailed images. Both of these features are recreated nicely by the model, where the text is often (but not always) illegible while still of a realistic appearance. We expect the legibility of the text to be much higher if our data would not contain a lot of non-latin (e.g. Chinese) characters. Figure 10 contains the 64 clusters found by clustering with our RC method, showing very obvious semantic similarities within each cluster. It is not immediately noticeable that each block is composed of real (top half) and generated (bottom half) samples, which shows how well the GAN is able to reproduce the specific distributions inherent in each cluster.

In a similar way, Figures 11 and 12 present samples from LLD-icon and LLD-icon-sharp, respectively. Here we compare random samples from different trained models, containing both conditional and unconditional variants. Figure 13, 14, show the clusters found in LLD-icon by clustering in the latent space of an Autoencoder, while Figures 15 and 16 show clusters in LLD-icon-sharp from the feature-space of a ResNet classifier. A very noticeable difference originates from the fact that the Autoencoder was trained on gray-scale images and is thus relatively color-independent, while there are some very apparent single-color clusters in the RC-version, mostly containing green, blue or orange/red logos.

Finally, in Figure 17, we present some samples from our benchmarked CIFAR-10 Generators, together with the achieved inception score. Figures 18 and 19 compare the clusters found using our RC method with the original data labels, with noticeably more visually uniform classes using our synthetic labeling technique.

(a) Original data

(b) iWGAN-LC with 64 RC clusters

Figure 9: Random samples from LLD-logo data and trained iWGAN model using 64 RC clusters and a 64×64 pixel output resolution.

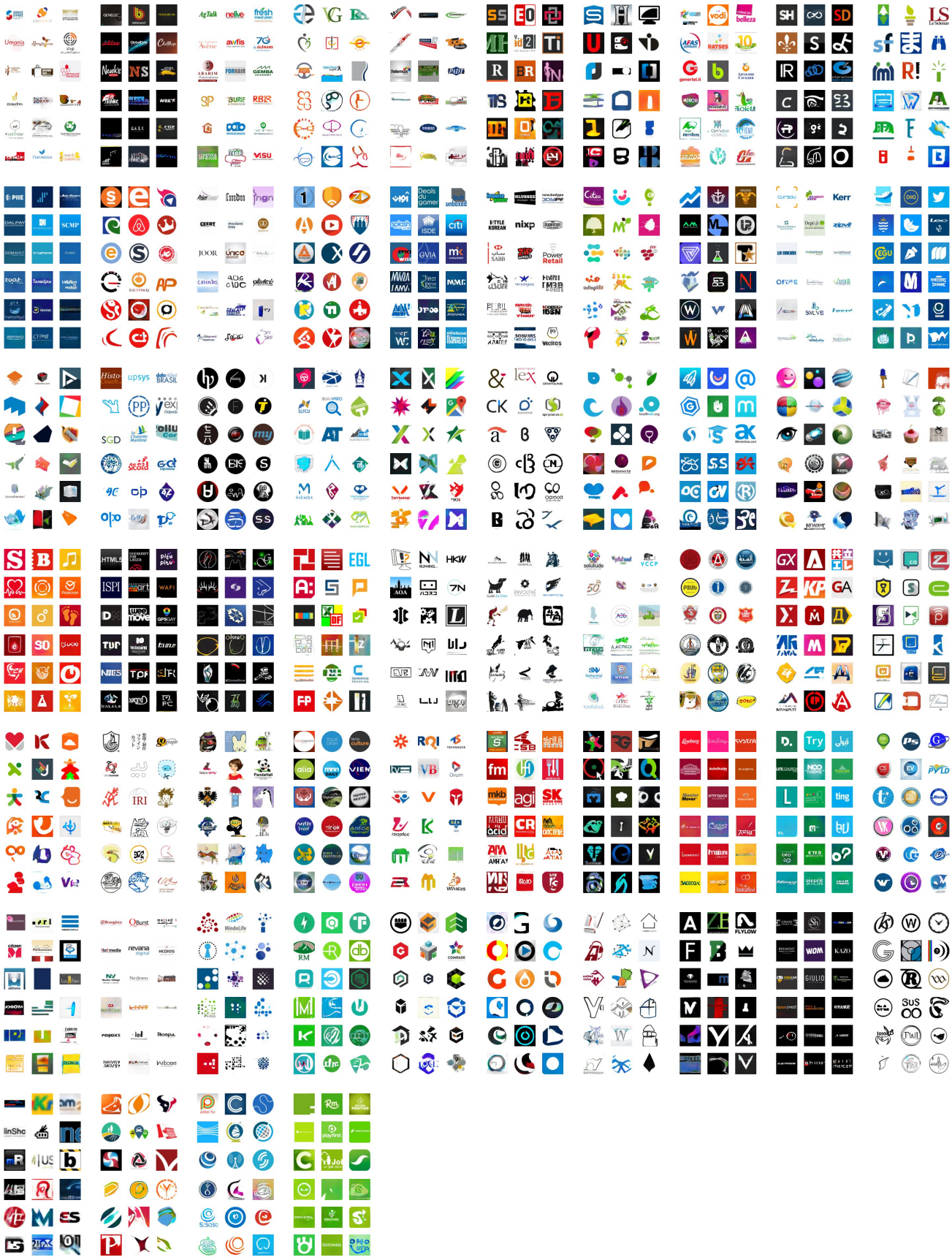
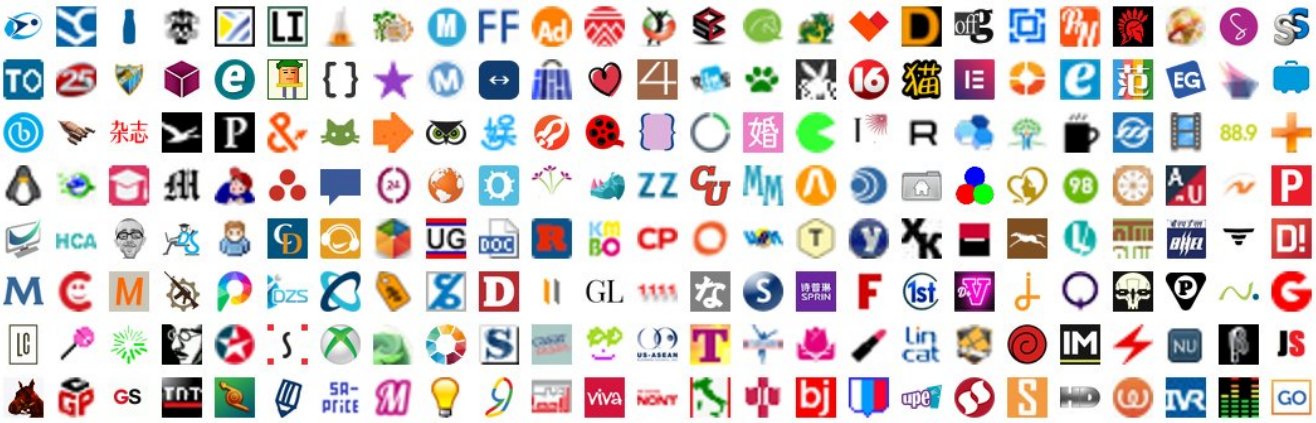


Figure 10: All 64 clusters of LLD-logo clustered with a ResNet classifier for 64 cluster centers. The top half of each block contains 9 random samples of original images from the cluster, while the bottom half contains 9 random samples from the iWGAN-LC Generator trained at 64×64 pixels. Best viewed as PDF at 400% magnification.



(a) Original data

(b) DCGAN-LC with 100 AE clusters

(c) iWGAN-LC with 100 AE clusters

(d) iWGAN-LC with 128 RC Clusters

Figure 11: Random samples from LLD-icon and generative models trained on this data.

(a) Original data

(b) Unconditional iWGAN

(c) iWGAN with 16 RC clusters

(d) iWGAN with 128 RC Clusters

Figure 12: Random samples from LLD-icon-sharp and generative models trained on this data.

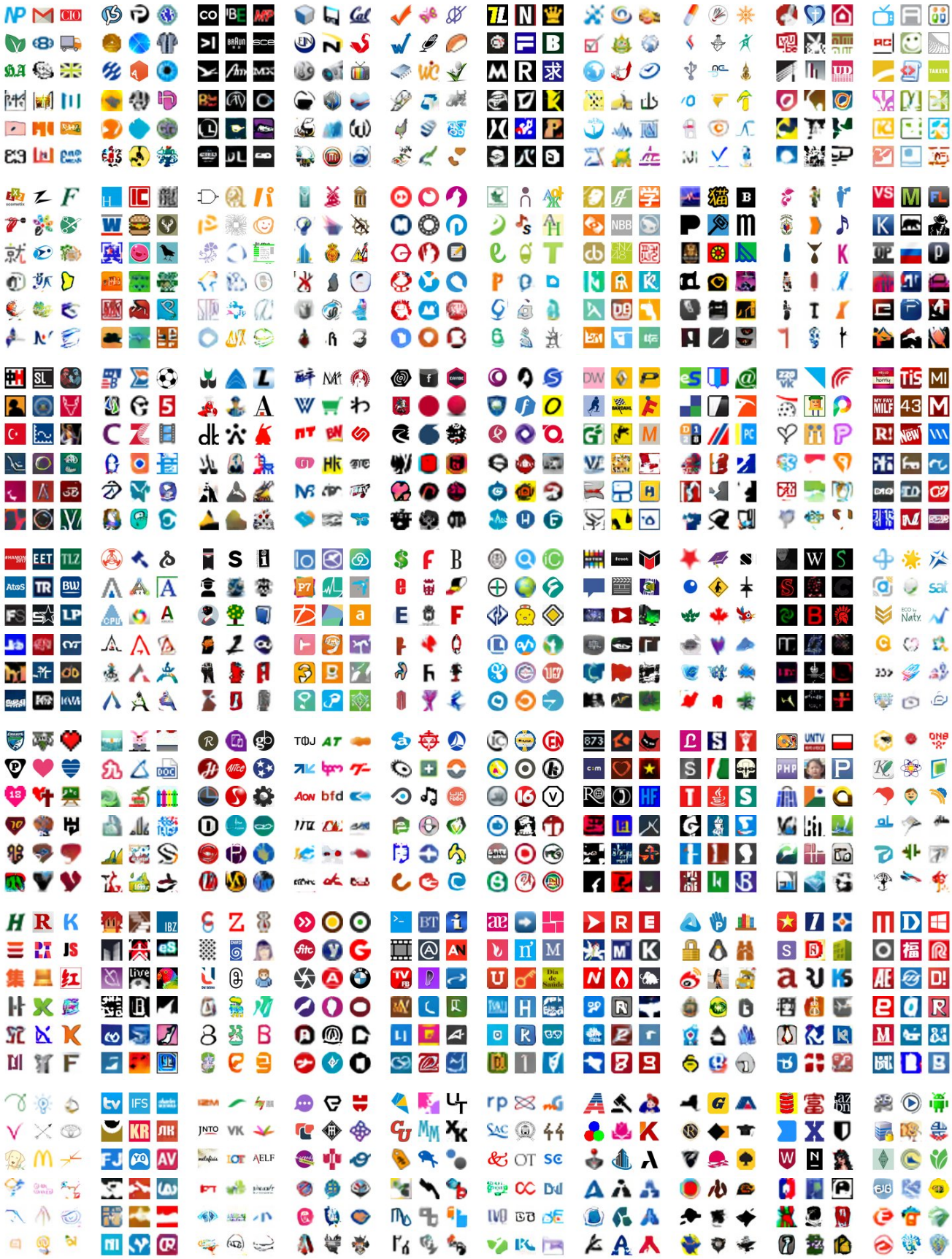


Figure 13: Clusters 1-70 of LLD-icon clustered in the latent space of an Autoencoder with 100 cluster centers. The top half of each block contains 9 random samples of original images from the cluster, while the bottom half contains 9 random samples from the DCGAN-LC Generator.

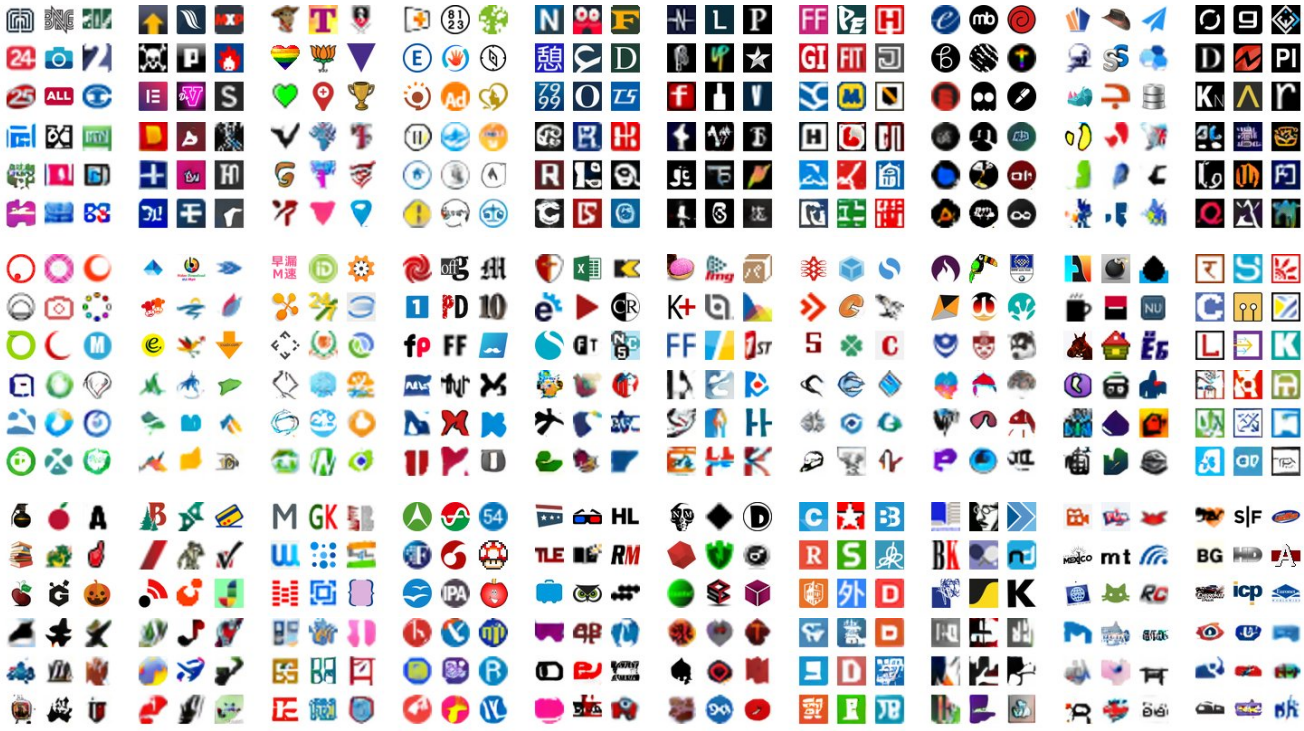


Figure 14: Clusters 71-128 of LLD-icon clustered in the latent space of an Autoencoder with 100 cluster centers. The top half of each block contains 9 random samples of original images from the cluster, while the bottom half contains 9 random samples from the DCGAN-LC Generator.

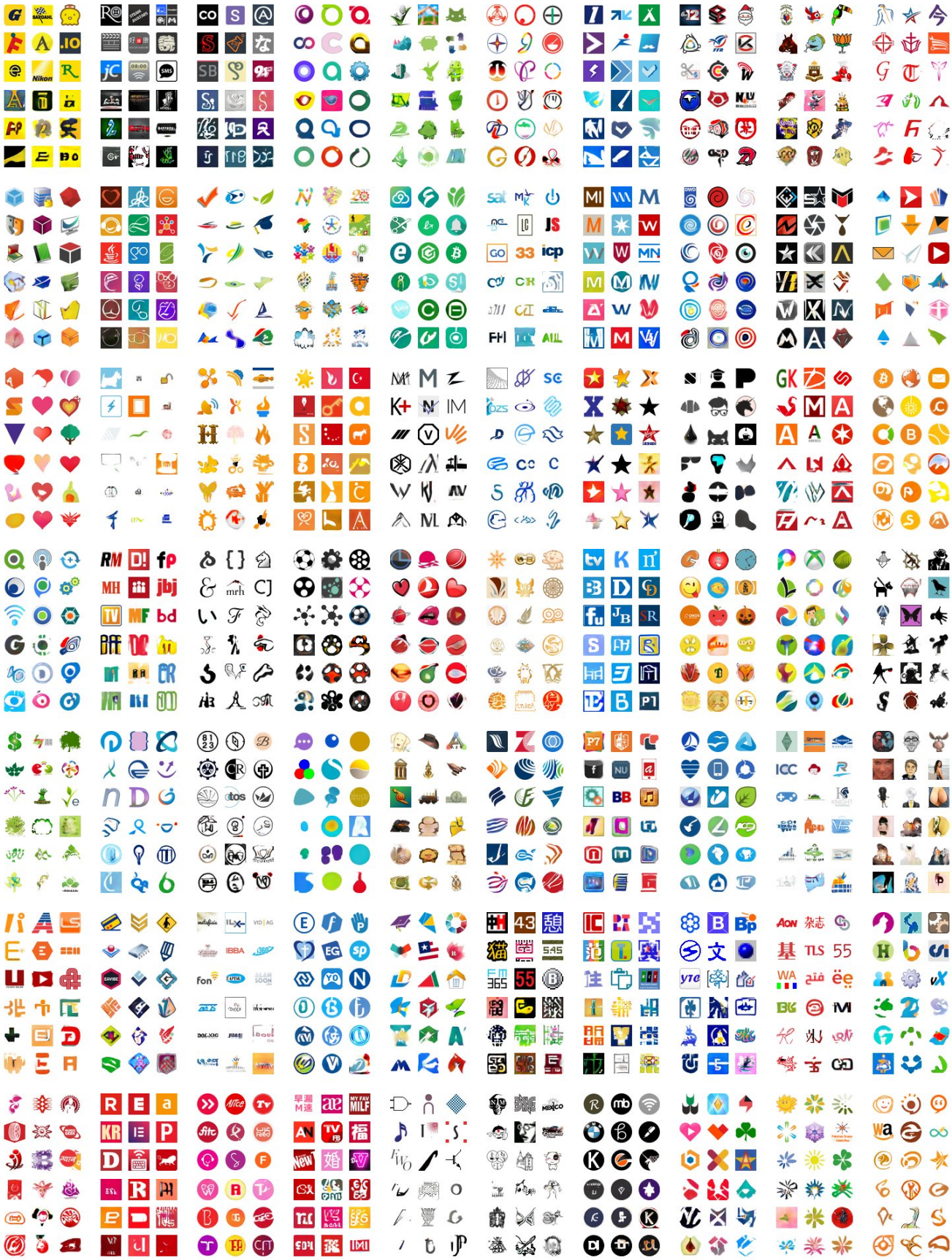
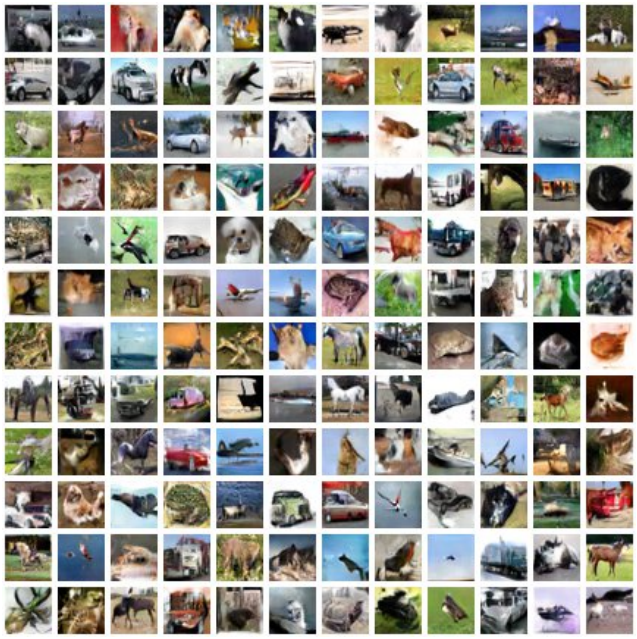
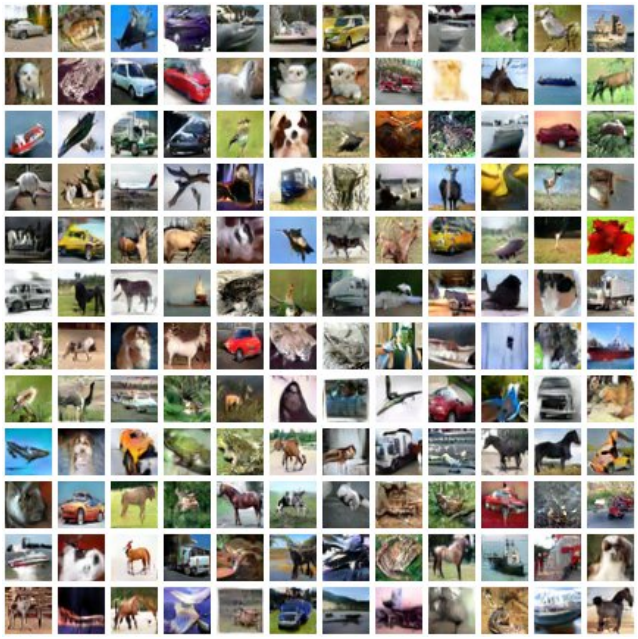


Figure 15: Clusters 71-100 of LLD-icon clustered in the latent space of an Autoencoder with 100 cluster centers. The top half of each block contains 9 random samples of original images from the cluster, while the bottom half contains 9 random samples from the DCGAN-LC Generator.

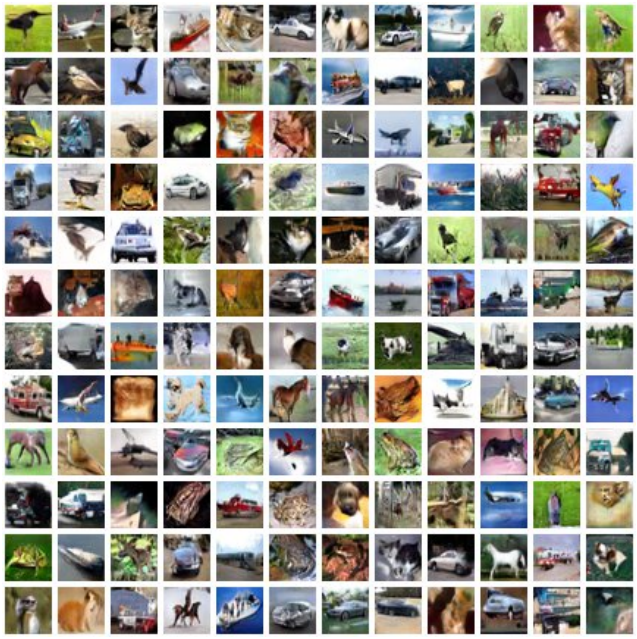
Figure 16: Clusters 71-128 of LLD-icon-sharp clustered with a ResNet Classifier and 128 cluster centers. The top half of each block contains 9 random samples of original images from the cluster, while the bottom half contains 9 random samples from the iWGAN-LC Generator.



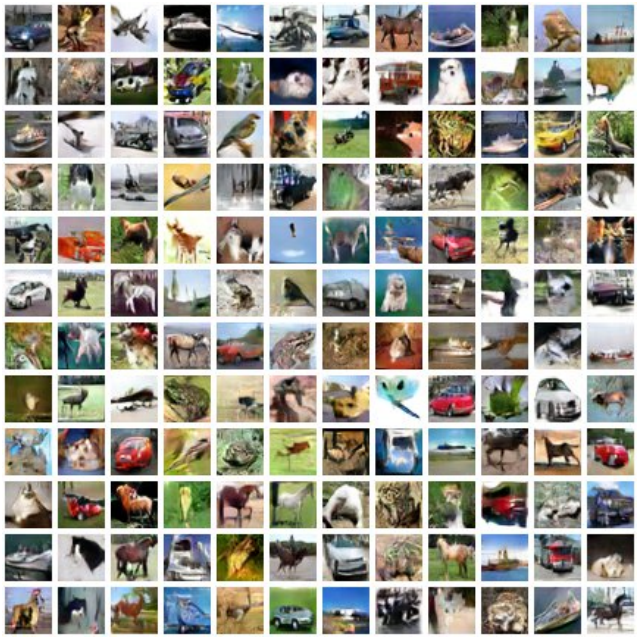
(a) iWgan unconditional. Inception score: 7.85



(b) iWGAN-AC with 32 RC clusters. Inception score: 8.67

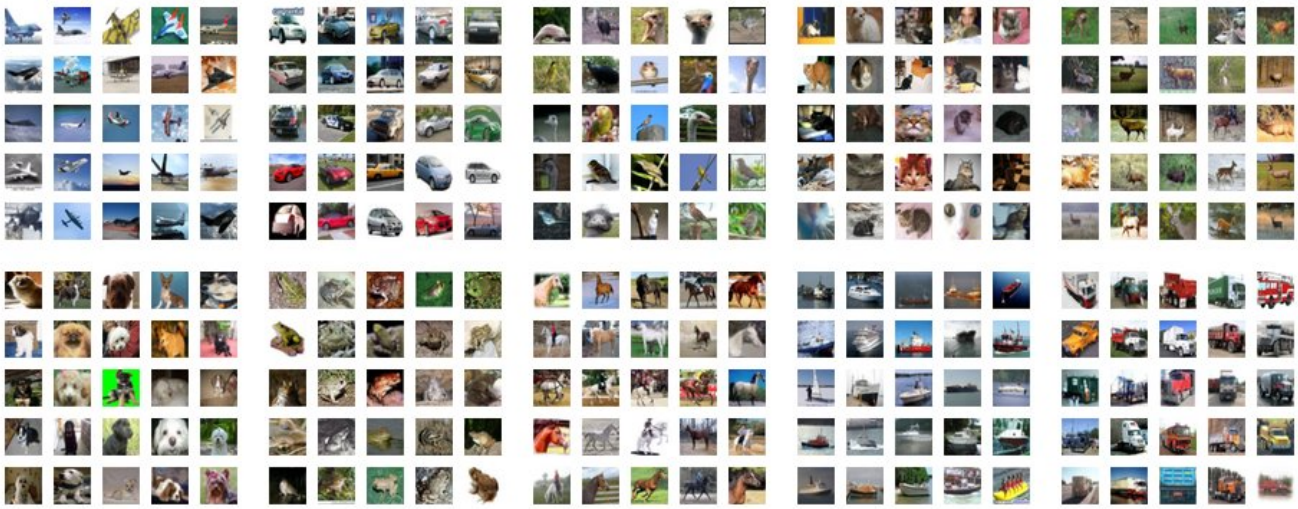


(c) iWGAN-AC with original labels. Inception score: 8.35

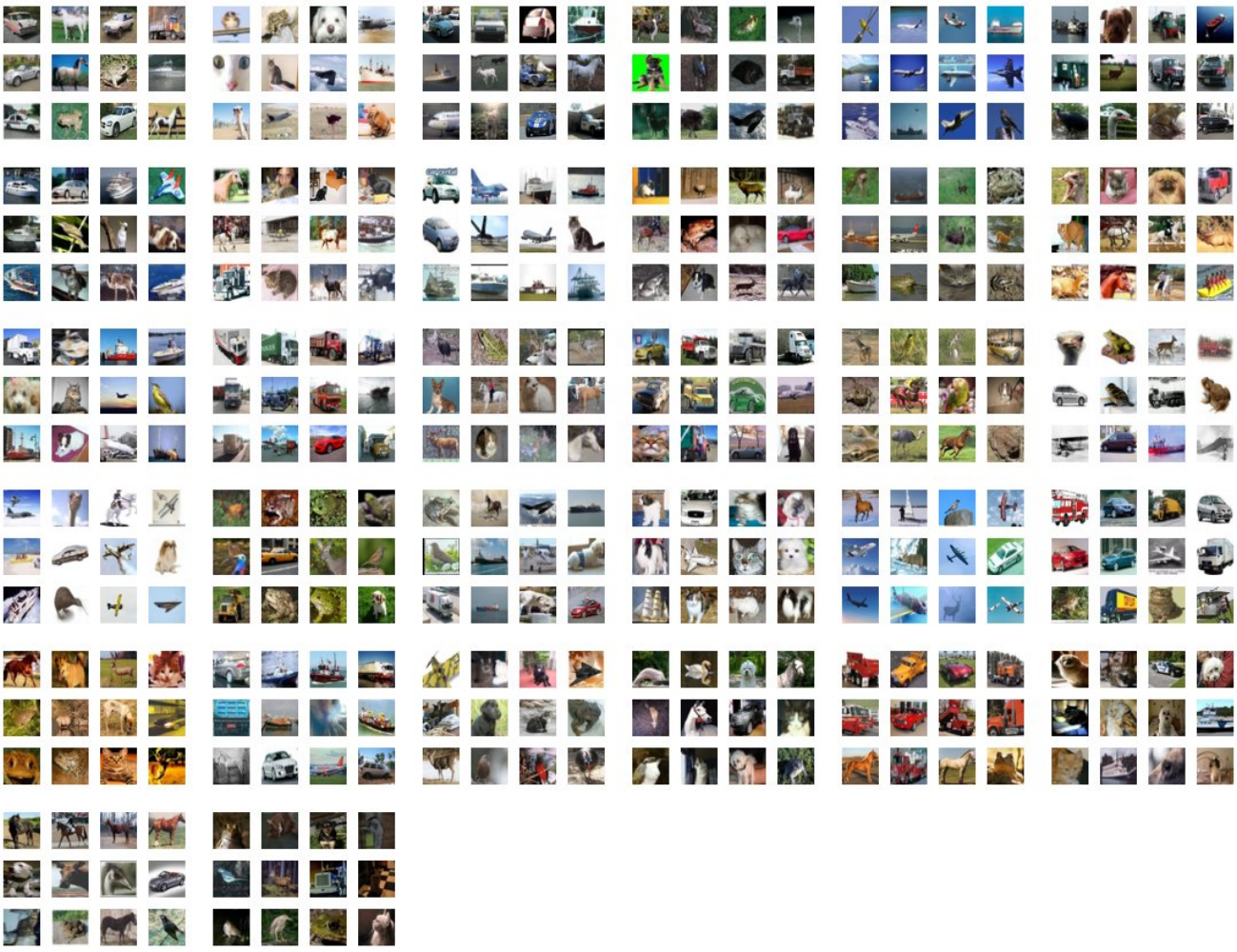


(d) iWGAN-LC with 32 RC clusters. Inception score: 7.83

Figure 17: Random samples from different iWGAN models trained on CIFAR-10 data.



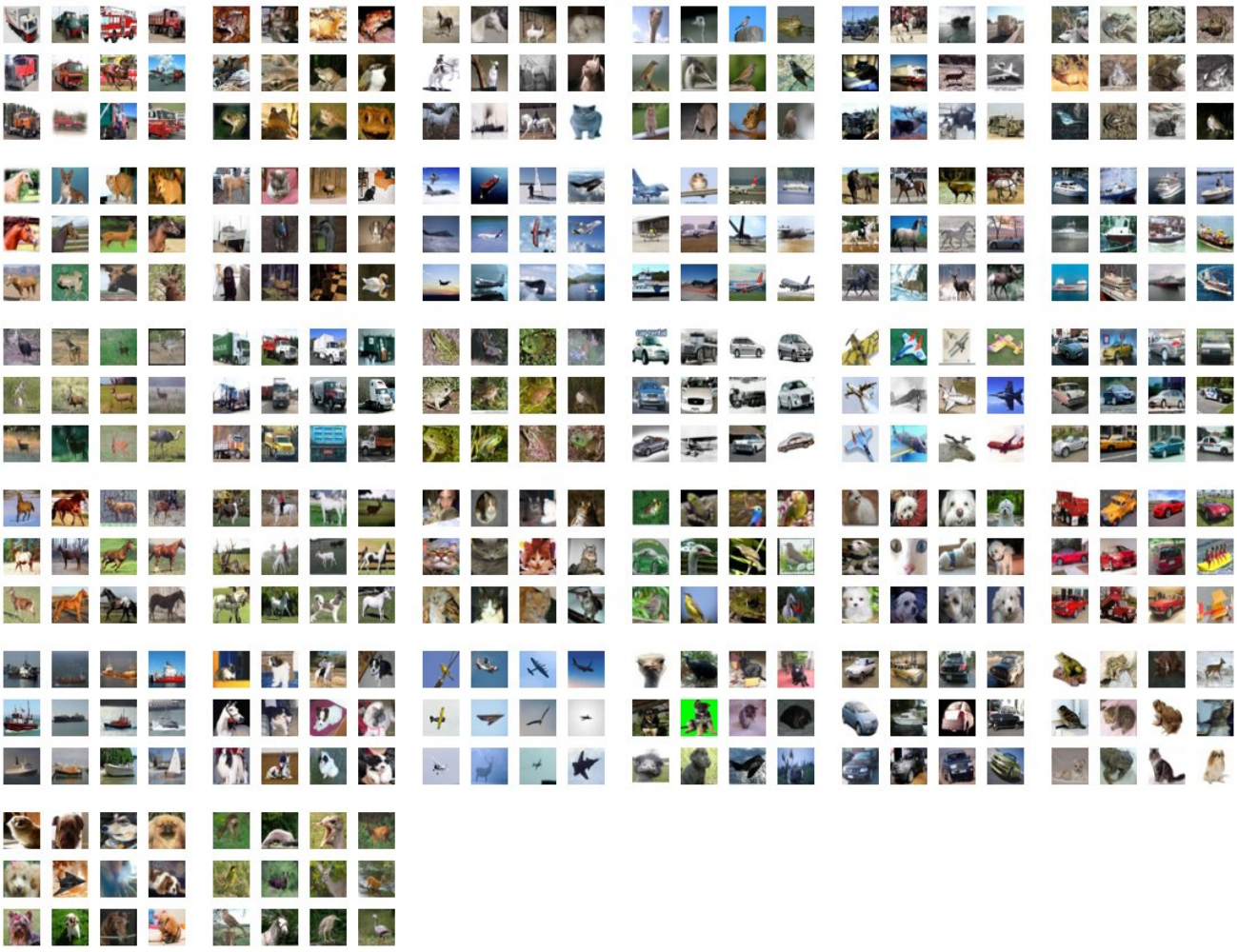
(a) Original data labels (10 categories)



(b) Clustering in Autoencoder space with 32 cluster centers

Figure 18: Original labels and 32 AE clusters. Note the strong variability in visual appearance within the semantic classes, pointing to a possible advantage of using a clustering more in-line with visual semantics. Our experiments with AE clustering produced clearly inferior results on the CIFAR-10 dataset (as compared to our own LLD data).

(a) Clustering in the CNN feature space of a ResNet classifier with 10 cluster centers



(b) Clustering in the CNN feature space of a ResNet classifier with 32 cluster centers

Figure 19: Resulting clusters using RC clustering with 10 and 32 cluster centers. Compared to the original labels in Figure 18, the 10 clusters shown here are more uniform in visual appearance, however increasing the number of clusters to 32 gives each of them an even more visually consistent appearance.

6. Architecture Details

In this section we specify the exact architectures and hyper-parameters used to train our models.

iWGAN for 32×32-pixel output We use the residual network architecture designed for CIFAR-10 described in [4] (Appendix C) for this model. For iWGAN-LC, each stage has an input shape of $[128 + k, ...]$ where k is the number of classes, i.e. the number of cluster centers used in our clustering approach. All training hyper-parameters remain untouched and we never use normalization in the Discriminator as this resulted in consistently superior Inception scores in our CIFAR-10 experiments. We use the exact same model and training parameters with our LLD-icon dataset.

iWGAN for 64x64-pixel output For LLD-logo at 64×64 pixels again the official TensorFlow implementation by Gulrajani *et al*. [4][1]. Again, the input for each stage is extended to have a shape of $[N + k, ...]$ where N is the size in the original model and k is the number of classes. The only change we made here is to only use $100,000$ iterations and linearly decay the learning rate over these iterations.

DCGAN For DCGAN, we deviate from some hyperparameters used in Taehoon Kim's TensorFlow implementation [2], namely:

- Higher number of feature maps: (128+k, 256+k, 512+k, 1024+k) for the Discriminator layers and (256+k, 512+k, 1034+k, 2048+k) for the Generator layers, with k again being the number of classes in the LC version.
- For each training iteration of the Discriminator, we train the Generator 3 times
- Reduced learning rate of 0.0004 (default: 0.002)
- Higher latent space dimensionality of 512 components (default: 100)
- Blur input images to Discriminator as detailed in Section 3.3 of our paper.

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

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

References

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

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

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

[4] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville. Improved training of wasserstein gans. *arXiv preprint arXiv:1704.00028*, 2017. 20

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

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

[7] 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.

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