Manifold Learning Benefits GANs (Supplementary Material)

Yao Ni[*,†], Piotr Koniusz[*,§,†], Richard Hartley[†,♦], Richard Nock[♦,♣,†]
[†]The Australian National University [§]Data61/CSIRO [♦]Google Research

firstname.lastname@anu.edu.au

Below is the contents of our supplementary material:

- §A explains our mathematical notation.
- §B provides further evaluations on CIFAR-10 and CIFAR-100 (32×32) datasets [30] given the large feature size of $d' = 1024$. We believe that it is valuable to demonstrate that our method can benefit from such a large d' while other methods cannot benefit much when applying $d' = 1024$ instead of the usual $d' = 512$.
- §C provides results for the limited data setting on CIFAR-10 and CIFAR-100.
- §D provides results on the ImageNet (32×32) dataset [9] in order to demonstrate that our LCSA-based pipeline benefits experiments on GAN for lower resolution images, which are favoured over high resolution images for instance when enriching training datasets of few-shot learning approaches. Note that results on ImageNet (64×64) and ImageNet (128×128) are given in our main submission.
- §E explains why FQGAN suffers from discriminator overfitting.
- Section F investigates the use of Denoising Auto-Encoder (DAE) [1] in place of LCSA, and concludes that LCSA is more beneficial.
- §G investigates the use of fixed β and γ rather than metaoptimized β and γ. The conclusion is that with mixing and proximity operators controlled by these parameters, we can reduce overfititng in the discriminator while training the manifold dictionary.
- §H illustrates the residual error between $\tilde{\boldsymbol{X}}$ and $h(\tilde{\boldsymbol{X}})$. The results support our belief that conv. encoder refines LCSA on difficult foreground objects while generating coarse-to-fine cov. features close to the manifold.
- §I details hyperparameters used in our experiments. We point out that the majority of parameters remain unchanged throughout many experiments.
- §J provides details of the feature encoding methods and the dictionary learning step.
- §K illustrates examples of images generated with or without the LCSA coder.
- §L offers proofs for claims of §5.

[*]Equal contribution. [♣] Brain team (richardnock@google.com).

A. Notations

Capitalized boldface symbols are matrices (*e.g.*, $\boldsymbol{X}$), lowercase boldface symbols are vectors (*e.g.*, $\boldsymbol{x}$), and regular fonts denote scalars *e.g.*, n, N, l, L, $X_{i,j}$ is a (i,j)-th coefficient of $\boldsymbol{X}$, x_i is an i-th coefficient of $\boldsymbol{x}$. We define a vector of all-ones as $\mathbf{1} = [1, \cdots, 1]^T$ and concatenation of α_i as $[\alpha_i]_{i=1,\cdots,I}$.

B. Experiments on CIFAR-10/100 ($d' = 1024$)

Below we demonstrate that utilizing LCSA in BigGAN and OmniGAN further enables them to utilize large feature sizes ($d' = 1024$), improving results further. Original BigGAN and OmniGAN struggle to attain good results for such a large feature size. Tables 7 verifies our pipelines on CIFAR-10 and CIFAR-100. Figures 9 and 10 show that our BigGAN+LCSA and OmniGAN+LCSA continue to learn while other models struggle to converge.

Model	CIFAR-10			CIFAR-100		
	IS ↑	tFID ↓	vFID ↓	IS ↑	tFID ↓	vFID ↓
BigGAN	9.64	10.71	14.86	11.54	15.11	19.97
FQGAN	9.44	12.59	16.69	**11.74**	8.49	13.51
BigGAN+LCSA	**9.81**	**3.51**	**7.55**	11.60	**5.49**	**10.37**
OmniGAN	9.92	12.11	16.29	12.42	10.11	14.85
OmniGAN+LCSA	**10.21**	**2.94**	**6.98**	**13.88**	**4.97**	**9.72**

Table 7. Results for different models on CIFAR-10 and CIFAR-100 with $d' = 1024$.

C. Data-limited Generation on CIFAR-10/100

Both DA [66] and ADA [20] limit discriminator overfitting by augmenting the real and generated images that are passed to the discriminator. However, in case of DA and ADA, augmentation artifacts leak into the generated images

Model	CIFAR-10			CIFAR-100		
	100%	20%	10%	100%	20%	10%
	IS↑ / tFID↓ / vFID↓	IS↑ / tFID↓ / vFID↓	IS↑ / tFID↓ / vFID↓	IS↑ / tFID↓ / vFID↓	IS↑ / tFID↓ / vFID↓	IS↑ / tFID↓ / vFID↓
BigGAN($d'=256$)	9.26 / 5.46 / 9.38	8.70 / 16.25 / 20.37	8.20 / 31.42 / 35.58	10.99 / 7.89 / 12.74	9.94 / 25.96 / 30.89	7.54 / 50.89 / 55.15
+DA	9.39 / 4.47 / 8.58	8.95 / 9.38 / 13.26	8.65 / 18.35 / 22.04	10.91 / 7.30 / 11.99	9.73 / 16.32 / 20.88	9.33 / 27.01 / 31.32
+LeCam	9.48 / 4.25 / 8.27	8.96 / 11.32 / 15.25	8.50 / 26.30 / 30.55	**11.44** / 6.53 / 11.23	10.00 / 20.82 / 25.77	8.10 / 39.33 / 44.23
+LCSA	**9.51** / **4.12** / **8.20**	**8.99** / **8.72** / **12.73**	**8.74** / **12.36** / **16.46**	11.02 / **6.38** / **11.13**	**10.38** / **13.22** / **18.06**	**10.08** / **21.13** / **25.87**
+LeCam+DA	9.47 / 4.29 / 8.29	9.05 / 7.53 / 11.46	8.84 / 12.15 / 15.92	11.15 / 6.55 / 11.34	10.42 / 13.01 / 17.65	9.92 / 21.73 / 26.19
+LCSA+DA	**9.50** / **3.75** / **7.83**	**9.08** / 7.29 / 11.31	8.86 / 10.86 / 14.78	**11.21** / **5.76** / **10.52**	10.55 / 12.03 / 16.92	**10.68** / 19.38 / 24.21
+LCSA+LeCam+DA	9.47 / 3.80 / 7.89	9.04 / **6.95** / **10.95**	**8.96** / **10.05** / **13.88**	11.17 / 5.85 / 10.64	**10.67** / **10.16** / **15.00**	10.28 / **18.24** / **23.12**
OmniGAN($d'=1024$)	9.98 / 6.89 / 10.76	8.62 / 37.86 / 42.18	6.59 / 54.04 / 58.71	12.61 / 8.43 / 13.21	10.11 / 40.57 / 44.86	6.87 / 63.41 / 67.46
+DA	10.10 / 4.26 / 8.15	9.47 / 13.56 / 17.34	8.96 / 19.59 / 23.60	12.96 / 7.48 / 12.11	11.42 / 17.72 / 22.49	10.21 / 32.61 / 36.86
+ADA	**10.27** / 5.03 / 9.15	9.44 / 27.20 / 31.05	7.72 / 41.82 / 45.36	13.47 / 6.13 / 10.88	12.18 / 13.66 / 18.34	8.81 / 46.74 / 51.03
+LCSA	10.20 / **2.58** / **6.65**	**9.97** / **5.15** / **9.01**	**9.74** / **7.66** / **11.47**	**13.74** / **4.94** / **9.67**	**12.86** / **11.15** / **15.90**	**11.85** / **13.82** / **18.56**
+LCSA+DA	10.22 / 2.52 / 6.60	9.93 / 5.01 / 8.89	9.75 / 7.46 / 11.35	13.79 / 4.58 / 9.46	12.92 / 11.06 / 15.73	11.88 / 13.68 / 18.44
+LCSA+ADA	**10.38** / **2.35** / **6.38**	**10.12** / **4.48** / **8.41**	**10.04** / **6.45** / **10.39**	**13.80** / **4.07** / **8.90**	**13.78** / **7.45** / **12.11**	**12.67** / **10.18** / **14.87**

Table 8. Comparison of our LCSA with DA, ADA and LeCam on CIFAR-10 and CIFAR-100 given different percentage of training data.

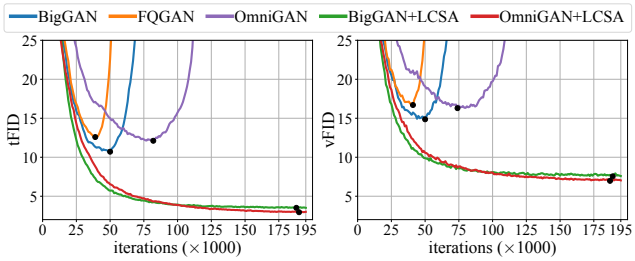


Figure 9. Evolution of tFID and vFID for different models on CIFAR-10 with $d'=1024$. Black dots indicate the minimum FID.

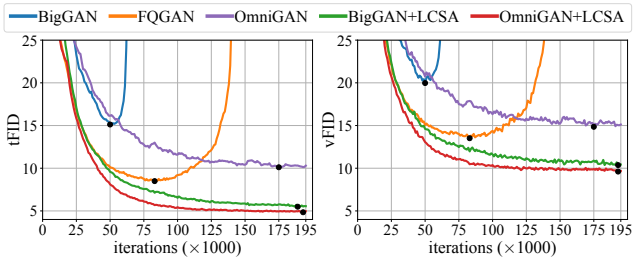
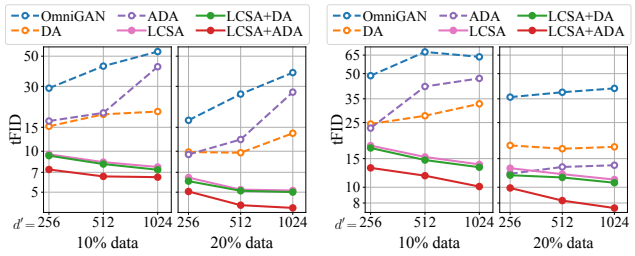


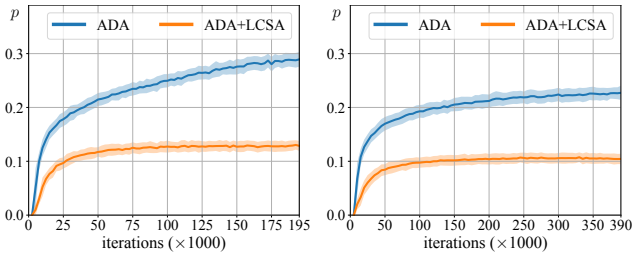
Figure 10. Evolution of tFID and vFID for different models on CIFAR-100 with d'=1024. Black dots indicate the minimum FID.



(a) CIFAR-10 (b) CIFAR-100

Figure 11. tFID w.r.t. different d' on CIFAR-10 and CIFAR-100 under the limited data setting.

as shown in Figure 24. The LeCamGAN [53] restricts discriminator overfitting by reducing the real/fake loss discrepancy at the output of the discriminator. However, it is not enough to limit overfitting by just operating at the output



(a) 10% CIFAR-10 (b) 20% CIFAR-10

Figure 12. Augmentation strength (p) of ADA *vs*. ADA+LCSA during training on 10%/20% CIFAR-10 data (OmniGAN d'=256).

of the discriminator (otherwise a well-designed loss would deal with all overfitting issues which is not the case in the literature). Therefore, we prevent the discriminator overfitting by modifying its blocks by encoding both the features of real and fake images into a common manifold and recovering their view from it, which has the ability to control the complexity of our feature space.

Below, we compare our LCSA with DA [66], ADA [20] on OmniGAN (and/or BigGAN) to variants without LCSA. We conduct experiments based on BigGAN to compare our LCSA with LeCamGAN [53] due to the issue of combining LeCam loss [53] with the multi-output loss of OmniGAN. Following [66], we augmented the real images with x-flip, and trained the OmniGAN and BigGAN for $1K$ epochs on the full data and $5K$ epochs on 10%/20% data setting. The DiffAug used translation and cutout augmentations. The ADA used 18 augmentations, including rotations and scaling. Details can be founded in [20].

Table 8 shows that our LCSA outperforms DA, ADA and LeCamGAN. LCSA can be always combined with them to improve their performance. Specifically, we achieve state-of-the-art results on CIFAR-10 (100%, 20% and 10%) and CIFAR-100 (20% and 10%) based on OmniGAN ($d'=1024$). In addition, in Figure 11, OmniGAN, DA and ADA become worse when increasing the d', while combining them with our LCSA improves the tFID given the larger model size, on which the discriminator can memo-

rize the training data easier, demonstrating that our LCSA has stronger performance while preventing overfitting better than ADA and DA alone. Moreover, we find that the DA and ADA leak the augmentation cues to the generated images, as shown in Figure 24, indicating that their discriminators overfit to the augmentation cues instead of learning meaningful discrimination boundary. ADA strives to control the strength of augmentations by a controller which observes the discriminator output and adjusts the desired augmentation strength. Thus, in Figure 12 we plot the augmentation strength of ADA and LCSA+ADA. We notice that **LCSA+ADA is able to limit the augmentation strength of ADA, thus reducing the leakage of augmentation artifacts.** This means LCSA can deal with the discriminator overfitting well. Concluding, LCSA enjoys better performance than ADA, DA, and LeCam for data-limited generation. LCSA gains further improvements with larger model size d', and LCSA lowers the risk of leaking augmentation artifacts. Thus, LCSA appears to have the superior capability of preventing discriminator overfitting than other strategies.

D. Experiments on ImageNet (32×32)

We preprocess images by center-cropping and downscaling them to 32×32 pixels. We train the network for 100 epochs. We set mini-batch size to 64, $\beta_0 = 0.$, $\gamma_0 = 0.1$, $\eta = 0.5$, $\sigma = 1.2$, whereas the dictionary learning rate is set to 1e-3. We equip each residual block of the discriminator with our LCSA module. We set $\Delta_\gamma = 1.2$ for $d' = 512$. For OmniGAN and OmniGAN+LCSA, we set weight decay of discriminator to be 1e-4 given $d' = 512$. Table 9 presents our results in the above setting.

Model	IS ↑	tFID ↓	vFID ↓
BigGAN	13.25	5.44	5.60
BigGAN+LCSA	**13.61**	**4.82**	**5.01**
OmniGAN	29.55	3.74	4.56
OmniGAN+LCSA	**30.53**	**3.53**	**4.37**

Table 9. Results for ImageNet (32×32).

E. A difference between FQGAN and Hard Assignment

As we note in §4.2, if $\boldsymbol{M}$ is formed by k-means clustering, Hard Assignment becomes an equivalent of the quantizer from FQGAN. However, the FQGAN suffers from discriminator overfitting, as we shown in Figures 8, 9 and 10. The FQGAN uses the proximity loss Eq(4), however, it has no intertwining step between the manifold learning step with blocks of discriminator. That is, for FQGAN, the feature input to the next layer from our pipeline Eq(3) is changed to:

$$\boldsymbol{X}_{l+1} = \tilde{\boldsymbol{X}}_l \tag{19}$$

Without the interwining step, without the adaptive mechanism, and without LCSA, the FQGAN cannot prevent blocks in discriminator from over-expressing the learned features, which may lie outside of the manifold, resulting in discriminator overfitting.

F. Replacing LCSA with Denoising Auto-Encoder (DAE)

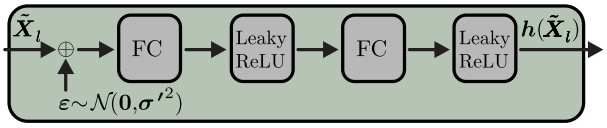


Figure 13. Architecture of DAE. We plug DAE in place of LCSA in Figure 2: we replace $\boldsymbol{h}(\cdot)$ based on LCSA with $\boldsymbol{h}(\cdot)$ based on DAE.

We conduct experiments in which we replace the LCSA coder with Denoising Auto-Encoder(DAE) [1]. As illustrated in Figure 13, two FC layers, each intertwined with LeakyReLU, form our DAE module. Moreover, we apply the mixing mechanism from Eq. (3) and the proximity operator from Eq. (4) which is an equivalent of the reconstruction loss of DAE given in Eq. (18). As Figure 13 indicates, DAE is equipped with a noise injector $\boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\sigma'}^2)$ with variance $\boldsymbol{\sigma'}^2$ (equivalent in Eq. (18)). We use the multivariate normal distribution which produces noise samples $\boldsymbol{\varepsilon}$ that are added to $\boldsymbol{X}'_l$, while $\boldsymbol{\sigma'}^2$ is a vector of on-diagonals equal σ'^2 (off-diagonal coefficients equal 0).

Instead of k (dictionary size), we vary the size of FC layers $(d' \times k_*)$ and $(k_* \times d')$, by varying the size of hidden representation k_*. We investigate $k_* \in \{8, 16, 32, 64, 128, 256\}$ and $\sigma'^2 \in \{0.01, 0.04, 0.25, 0.64, 1.0\}$. After searching for the best set of parameters, we obtain $k_* = 64$, $\sigma'^2 = 0.04$ for CIFAR-10 and $k_* = 32$, $\sigma'^2 = 0.04$ for CIFAR-100. The DAE module was updated once at each mini-batch.

Table 10 shows that OmniGAN+LCSA outperforms OmniGAN+DAE, which validates our assumptions on locality-constrained non-linear manifold learning and its ability to deal with the noise and regularization of discriminator.

Model	CIFAR-10			CIFAR-100		
	IS ↑	tFID ↓	vFID ↓	IS ↑	tFID ↓	vFID ↓
OmniGAN	9.70	6.88	10.65	12.78	9.13	13.82
OmniGAN+DAE	10.08	3.89	7.91	13.65	5.52	10.33
OmniGAN+LCSA	**10.09**	**3.29**	**7.31**	**13.73**	**5.12**	**9.91**

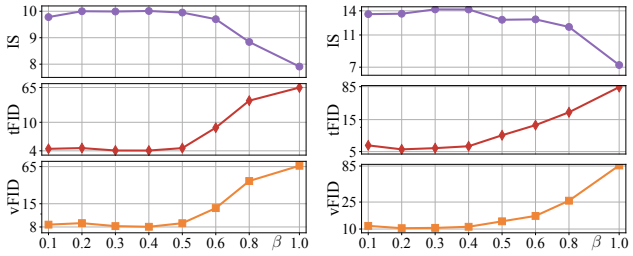
Table 10. OmniGAN+DAE *vs.* OmniGAN+LCSA ($d' = 512$).

G. Fixed β and γ

To verify that metaparameter learning of β and γ works better than setting fixed β and γ, Table 11 shows for the best possible fixed β. For each experiment, $\gamma = \Delta_\gamma \beta$ and $\Delta_\gamma = 1.2$ (same as for learnable experiment). For fixed β and γ, the best results are obtained with ($\beta = 0.4$, $\gamma = 0.48$) for CIFAR-10 and ($\beta = 0.2$, $\gamma = 0.24$) for CIFAR-100. Figure 14 provides further ablations.

Method	CIFAR-10			CIFARR-100		
	IS ↑	tFID ↓	vFID ↓	IS ↑	tFID ↓	vFID ↓
Fixed	10.01	4.06	8.03	13.64	5.46	10.29
Learnable	**10.09**	**3.29**	**7.31**	**13.73**	**5.12**	**9.91**

Table 11. Best fixed β and γ *vs*. best metalearnable β and γ.



(a) CIFAR-10 (b) CIFAR-100

Figure 14. Fixed β and γ, $\gamma = \Delta_\gamma \beta$ where $\Delta_\gamma = 1.2$ (as in the adaptive setting). We show results for several fixed values of β.

H. Illustration of the residual error

We randomly sample images from real dataset on CIFAR-10, and input them into the discriminator. We obtain $\|\boldsymbol{X}_1 - \tilde{\boldsymbol{X}}_1\|_F^2$ at the first residual block and normalize for visualization. Figure 15 shows LCSA coder perpetrates larger errors at locations of difficult objects and smaller ones at locations of backgrounds. As Eq. (3).

I. Network architecture and Hyperparameters

Unless specified otherwise, we set $k = 1024$, $\sigma = 1.2$, $\gamma_0 = 0$, $\beta_0 = 0.1$, and $\eta = 0.5$.

CIFAR-10. Below, we experiment with OmniGAN [69] which adopts the architecture of BigGAN [5]. We follow OmniGAN mini-batch size of 32 and train our networks for 500 epochs. To this end, we equip each residual block of the discriminator with the LCSA module. For our experiments on CIFAR-10, we apply weight decays of 1e-4, 1e-5 and 0 on generator given $d' = 256, 512, 1024$, weight decay 0 for the discriminator. To update the dictionary, we use the Adam optimizer [24] with learning rate of 0.002. We set We use Δ_γ equal 1.0, 1.2 and 2.5 given $d' = 256, 512, 1024$. For StyleGAN2+ADA+LCSA, we follow [20] and set $\eta = 0.6$. We set $\Delta_\gamma = 0.05$ due to the architecture differences between StyleGAN2 and BigGAN.

CIFAR-100. We use CIFAR-10 settings but we set the learning rate of DL to 0.0015, $k' = 32$, and weight decays of 5e-4, 2e-4, 5e-5 for discriminators with $d' = 256, 512, 1024$.

ImageNet (64×64). We equip BigGAN and OmniGAN with LCSA (the last 3 blocks $l \in \{3, 4, 5\}$, $d' = 384$). We use the architecture of BigGAN [5]. For OmniGAN and OmniGAN+LCSA, we set mini-batch size to 256, learning rates of generator/discriminator to 4e-4, and take 4 discriminator steps per generator step to prevent OmniGAN from diverging. Weight decays are 0 and 2e-5 for generator and discriminator. For BigGAN, FQGAN and BigGAN+LCSA, we apply settings of FQGAN [67] to set mini-batch size to 512, learning rates of generator and discriminator to 1e-4 and 4e-4. We set dictionary learning rate to 1e-3, $\Delta_\gamma = 1.5$, $k' = 8$. We set $\sigma = 0.8$, $\beta_0 = 0$, $\gamma_0 = 0.1$ given the discriminator does not overfit on this diverse dataset at the early training stage. All models are trained for 200 epochs.

ImageNet (128×128). We only equip OmniGAN with LCSA as OmniGAN consistently outperforms BigGAN. We use setting from ImageNet (64×64) but set dict. learning rate to 2e-4 and weight decay of discriminator to 5e-6. The two models are trained for 300 epochs.

Oxford-102 Flowers (256×256). We employ the state-of-the-art model for this dataset, MSG-StyleGAN [18], as our baseline. Following [18], we set the size of mini-batch to 32 while $d' = 512$. We set the number of iteration to $125K$, $k' = 32$, $\Delta_\gamma = 1$, $\eta = 0.7$, dictionary learning rate is 5e-4, and LCSA is applied to blocks 4 and 5 of discriminator.

FFHQ (256×256). Following [20], we set the mini-batch size to 64, d'=512 and train the networks until the discriminator had seen $25M$ real images. We equip StyleGAN2 with LCSA (the last 4 blocks $l \in \{4, 5, 6, 7\}$). We set k'=32, Δ_γ=0.05, η=0.9, dictionary learning rate is 5e-4.

Table 12 summarizes hyperparameters used in our experiments. We used a very limited range of these parameters (most par. do not change between experiments) *e.g.*, dict. learning rate equals 1e-3, 1.5e-3, 2e-3, 5e-4 or 2e-4, β_0 is set to 0 or 0.1, γ_0 is set to 0 or 0.1, Δ_γ is set to 0.05, 1, 1.2, 1.5 or 2.5, σ is set to 0.8 or 1.2, η is set to 0.5, 0.6, 0.7 or 0.9, and k' to 8, 32, or 128.

Table 13 provides number of parameters, FLOPs and the training time for models with/without the LCSA coder.

J. Coding Methods

Algorithms 1 and 2 detail our implementations of Non-Negative Sparse Coding (SC_+) and Sparse Coding (SC).

Algorithm 1 Non-negative Sparse Coding (SC_+).

Input: $\boldsymbol{X}, \boldsymbol{M}$ for a given forward pass, κ: the L_1 norm penalty, ι: the number of iterations, ω: the learning rate.

1: $\{\alpha_{in} \sim \mathcal{U}(1e-6, 1)\}_{i=1,\cdots,k \text{ and } n=1,\cdots,N'}$
2: $\boldsymbol{\alpha} \leftarrow \left[\frac{\boldsymbol{\alpha}_n}{\|\boldsymbol{\alpha}_n\|_1 + 1e-6}\right]_{n=1}^{N'}$ (concatenate $\boldsymbol{\alpha}_n$ into matrix $\boldsymbol{\alpha}$)
3: **for** $i = 1, \cdots, \iota$
4: $\quad \Delta\boldsymbol{\alpha} = 2\boldsymbol{M}^T(\boldsymbol{X} - \boldsymbol{M}\boldsymbol{\alpha}) + \kappa$ (gradient computation)
5: $\quad \boldsymbol{\alpha} \leftarrow \boldsymbol{\alpha} - \frac{\omega}{(1+i)^{0.3}} \Delta\boldsymbol{\alpha}$ (stochastic gradient descent)
6: $\quad \boldsymbol{\alpha} \leftarrow \text{ReLU}(\boldsymbol{\alpha})$ (reprojection into the feasible set)

Output: $\boldsymbol{\alpha}$: sparse codes

Algorithm 3 is our efficient implementation of the Orthogonal Matching Pursuit (OMP) which is based on the batch support of PyTorch. Operations *cat*, *squeeze*, *solve*

(a) car (b) $\|\tilde{\boldsymbol{X}}_1 - h(\tilde{\boldsymbol{X}}_1)\|_F^2$ of car (c) bird (d) $\|\tilde{\boldsymbol{X}}_1 - h(\tilde{\boldsymbol{X}}_1)\|_F^2$ of bird

(e) dog (f) $\|\tilde{\boldsymbol{X}}_1 - h(\tilde{\boldsymbol{X}}_1)\|_F^2$ of dog (g) horse (h) $\|\tilde{\boldsymbol{X}}_1 - h(\tilde{\boldsymbol{X}}_1)\|_F^2$ of horse

Figure 15. Real samples and the corresponding $\|\tilde{\boldsymbol{X}}_1 - h(\tilde{\boldsymbol{X}}_1)\|_F^2$ on CIFAR-10. Whiter pixels correspond to larger $\|\tilde{\boldsymbol{X}}_1 - h(\tilde{\boldsymbol{X}}_1)\|_F^2$.

dataset	d'	Model	dict LR	β_0	γ_0	Δ_γ	k'	σ	η	L	GWD	DWD
CIFAR-10	256	OmniGAN+LCSA	2e-3	0.1	0	1.0	128	1.2	0.5	{1,2,3,4}	1e-4	0
	512	BigGAN	–	–	–	–	–	–	–	–	0	0
		BigGAN+LCSA	2e-3	0.1	0	1.2	128	1.2	0.5	{1,2,3,4}	0	0
		OmniGAN	–	–	–	–	–	–	–	–	1e-5	0
		OmniGAN+LCSA	2e-3	0.1	0	1.2	128	1.2	0.5	{1,2,3,4}	1e-5	0
		StyleGAN2+ADA+LCSA	2e-3	0.1	0	0.05	128	1.2	0.6	{1,2,3,4}	0	0
	1024	BigGAN	–	–	–	–	–	–	–	–	0	0
		BigGAN+LCSA	2e-3	0.1	0	2.5	128	1.2	0.5	{1,2,3,4}	0	0
		OmniGAN	–	–	–	–	–	–	–	–	0	0
		OmniGAN+LCSA	2e-3	0.1	0	2.5	128	1.2	0.5	{1,2,3,4}	0	0
CIFAR-100	256	OmniGAN+LCSA	1.5e-3	0.1	0	1.0	32	1.2	0.5	{1,2,3,4}	1e-4	5e-4
	512	BigGAN	–	–	–	–	–	–	–	–	0	0
		BigGAN+LCSA	1.5e-3	0.1	0	1.2	32	1.2	0.5	{1,2,3,4}	0	0
		OmniGAN	–	–	–	–	–	–	–	–	1e-5	2e-4
		OmniGAN+LCSA	1.5e-3	0.1	0	1.2	32	1.2	0.5	{1,2,3,4}	1e-5	2e-4
	1024	BigGAN	–	–	–	–	–	–	–	–	0	0
		BigGAN+LCSA	1.5e-3	0.1	0	2.5	32	1.2	0.5	{1,2,3,4}	0	0
		OmniGAN	–	–	–	–	–	–	–	–	0	5e-5
		OmniGAN+LCSA	1.5e-3	0.1	0	2.5	32	1.2	0.5	{1,2,3,4}	0	5e-5
ImageNet 64×64	384	BigGAN	–	–	–	–	–	–	–	–	0	0
		BigGAN+LCSA	1e-3	0	0.1	1.5	8	0.8	0.5	{3,4,5}	0	0
		OmniGAN	–	–	–	–	–	–	–	–	0	2e-5
		OmniGAN+LCSA	1e-3	0	0.1	1.5	8	0.8	0.5	{3,4,5}	0	2e-5
ImageNet 128×128	384	OmniGAN	–	–	–	–	–	–	–	–	0	5e-6
		OmniGAN+LCSA	2e-4	0	0.1	1.5	8	0.8	0.5	{4,5,6}	0	5e-6
Oxford-102 Flowers	512	MSG-StyleGAN	–	–	–	–	–	–	–	–	0	0
		MSG-StyleGAN+LCSA	5e-4	0.1	0	1	32	1.2	0.7	{4,5}	0	0
FFHQ	512	StyleGAN2+LCSA	5e-4	0.1	0	0.05	32	1.2	0.9	{4,5,6,7}	0	0

Table 12. Hyperparameter used in our experiments. Abbreviation 'dict LR' denotes dictionary learning rate, 'GWD' and 'DWD' are weight decays of generator and discriminator, respectively.

Dataset	Baseline	GPUs	Baseline #Par. / FLOPs Training Time	+LCSA #Par. / FLOPs Training Time
CIFAR-10	OmniGAN (d'=512)	1	32.88M / 11.12G 27h11m	34.97M / 12.04G 30h54m
CIFAR-100	OmniGAN (d'=512)	1	33.48M / 11.12G 27h29m	35.57M / 12.04G 30h32m
ImageNet (64×64)	OmniGAN	4	115.69M / 18.84G 3d15h	118.44M / 19.10G 4d01h
ImageNet (128×128)	OmniGAN	4	158.36M / 31.45G 20d03h	162.29M / 31.86G 22d10h
Oxford-102 Flowers	MSG-StyleGAN	4	50.28M / 85.11G 22h46m	51.33M / 86.29G 23h09m
FFHQ	StyleGAN2	4	48.76M / 51.64G 58h42m	50.86M / 53.14G 59h51m

Table 13. Number of parameters/FLOPs (for both generator and discriminator) and training time for models with *vs*. without the LCSA encoder. Experiments were performed on NVIDA Tesla V100 GPUs.

Algorithm 2 Sparse Coding (SC).

Input: $\boldsymbol{X}, \boldsymbol{M}$ for a given forward pass, κ: the L_1 norm penalty, ι: the number of iterations, ω: the learning rate.

$\{\alpha_{in} \sim \mathcal{U}(-1,1)\}_{i=1,\cdots,k \text{ and } n=1,\cdots,N'}$
$\boldsymbol{\alpha}_+ = \text{ReLU}(\boldsymbol{\alpha}) + 1e{-}6$ and $\boldsymbol{\alpha}_- = \text{ReLU}(-\boldsymbol{\alpha}) + 1e{-}6$
$\boldsymbol{\alpha}_+ \leftarrow \left[\frac{\boldsymbol{\alpha}_{+,n}}{\|\boldsymbol{\alpha}_{+,n}\|_1 + 1e{-}6}\right]_{n=1}^{N'}$, $\boldsymbol{\alpha}_- \leftarrow \left[\frac{\boldsymbol{\alpha}_{-,n}}{\|\boldsymbol{\alpha}_{-,n}\|_1 + 1e{-}6}\right]_{n=1}^{N'}$
for $i = 1, \cdots, \iota$
  $\Delta\boldsymbol{\alpha} = 2\boldsymbol{M}^T(\boldsymbol{X} - \boldsymbol{M}(\boldsymbol{\alpha}_+ - \boldsymbol{\alpha}_-))$ (compute gradient)
  $\Delta\boldsymbol{\alpha}_+ = \Delta\boldsymbol{\alpha} + \kappa$ and $\Delta\boldsymbol{\alpha}_- = -\Delta\boldsymbol{\alpha} + \kappa$
  $\boldsymbol{\alpha}_+ \leftarrow \boldsymbol{\alpha}_+ - \frac{\omega}{(1+i)^{0.3}}\Delta\boldsymbol{\alpha}_+$ and $\boldsymbol{\alpha}_- \leftarrow \boldsymbol{\alpha}_- - \frac{\omega}{(1+i)^{0.3}}\Delta\boldsymbol{\alpha}_-$
  $\boldsymbol{\alpha}_+ \leftarrow \text{ReLU}(\boldsymbol{\alpha}_+)$ and $\boldsymbol{\alpha}_- \leftarrow \text{ReLU}(\boldsymbol{\alpha}_-)$
$\boldsymbol{\alpha} \leftarrow \boldsymbol{\alpha}_+ - \boldsymbol{\alpha}_-$

Output: $\boldsymbol{\alpha}$: sparse codes

and *one_hot* are equivalent to PyTorch methods with the same names while $\odot$ is the element-wise multiplication.

Algorithm 4 illustrates that Locality-constrained Linear Coding (LLC) is computed with operations which enjoy the batch support of PyTorch. Once more, operation *solve* is equivalent to the batch-wise linear solver of PyTorch.

Algorithm 5 is a sketch implementation of Locality-constrained Soft Assignment (LCSA). Function *idx_nn* firstly computes distances $\boldsymbol{D}$ of $\boldsymbol{X}$ to $\boldsymbol{M}$ as:

$$\boldsymbol{D} = [\|\boldsymbol{x}_1\|_2^2, \cdots, \|\boldsymbol{x}_{N'}\|_2^2] \cdot \boldsymbol{1}^T + \boldsymbol{1} \cdot [\|\boldsymbol{m}_1\|_2^2, \cdots, \|\boldsymbol{m}_k\|_2^2]^T - 2\boldsymbol{M}^T\boldsymbol{X}. \quad (20)$$

Subsequently, function *ktop* implemented in PyTorch selects k' indexes of k' smallest distances from $\boldsymbol{D}$ for each $\boldsymbol{x}_n$.

Algorithm 6 illustrates how we perform the Dictionary Learning (DL) step. While we could solve the Least

Algorithm 3 Orthogonal Matching Pursuit (OMP).

Input: $\boldsymbol{X}, \boldsymbol{M}$ for a given forward pass, τ: the number of non-zero elements (iterations).

$\boldsymbol{E} = \boldsymbol{X}$ (initialize matrix of residuals)
$\boldsymbol{M}^* = \emptyset$ (active dictionary tensor $\boldsymbol{M}^* \in \mathbb{R}^{d' \times 0 \times N'}$)
for $i = 1, \cdots, \tau$
  $\boldsymbol{P} = \boldsymbol{M}^T\boldsymbol{E}$ (projection of $\boldsymbol{X}$ onto residual $\boldsymbol{E}$)
  $\boldsymbol{\varphi^i} = \text{idx\_max}(|\boldsymbol{P}|)$ (index of maximum coefficient per column)
  $\boldsymbol{M}^* = \text{cat}(\boldsymbol{M}^*, [\boldsymbol{m}_{\varphi_1^i}, \cdots, \boldsymbol{m}_{\varphi_{N'}^i}]; 1)$ (active atoms are added to tensor $\boldsymbol{M}^*$ so that $\boldsymbol{M}^* \in \mathbb{R}^{d' \times i \times N'}$)
  for $n = 1, \cdots, N'$ (this loop is batch solved)
    $\bar{\boldsymbol{M}} = \text{squeeze}(\boldsymbol{M}_{:,:,n})$ (dict. $\bar{\boldsymbol{M}} \in \mathbb{R}^{d' \times i}$ for $\boldsymbol{x}_n$)
    $\boldsymbol{\alpha}_n = \text{solve}(\bar{\boldsymbol{M}}^T\boldsymbol{x}_n, \bar{\boldsymbol{M}}^T\bar{\boldsymbol{M}})$ (system of linear equations solved by batch solver torch.solve(·)
    $\boldsymbol{e}_n = \boldsymbol{x}_n - \bar{\boldsymbol{M}}\boldsymbol{\alpha}_n$ (residual solved by batch operations at once, that is, $\boldsymbol{E} \equiv [\boldsymbol{e}_1, \cdots, \boldsymbol{e}_{N'}]$)
$\boldsymbol{\alpha} = \text{one\_hot}([\boldsymbol{\varphi^1}, \cdots, \boldsymbol{\varphi^\tau}]) \odot [\boldsymbol{\alpha}_1, \cdots, \boldsymbol{\alpha}_{N'}]$

Output: $\boldsymbol{\alpha}$: sparse codes

Algorithm 4 Locality-constrained Linear Coding (LLC).

Input: $\boldsymbol{X}, \boldsymbol{M}$ for a given forward pass, k': the k' nearest neighbors, ρ: a small reg. constant equal $1e{-}6$.

for $n = 1, \cdots, N'$ (this loop is batch solved)
  $\boldsymbol{C}_n = (\boldsymbol{M} - \boldsymbol{1}\boldsymbol{x}_n^T)(\boldsymbol{M} - \boldsymbol{1}\boldsymbol{x}_n^T)^T$ (so-called data cov.)
  $\boldsymbol{\alpha}_n = \text{solve}(\boldsymbol{1}, \boldsymbol{C}_n + \text{diag}(\boldsymbol{1}) \cdot \rho)$ (matrix left div. by $\boldsymbol{1}$)
  $\boldsymbol{\alpha}_n \leftarrow \boldsymbol{\alpha}_n / \sum_j \alpha_{jn}$ (normalization)
$\boldsymbol{\alpha} = [\boldsymbol{\alpha}_1, \cdots, \boldsymbol{\alpha}_{N'}]$ (concatenation of vectors $\boldsymbol{\alpha}_n$ into matrix $\boldsymbol{\alpha}$)

Output: $\boldsymbol{\alpha}$: locality-constrained linear codes

Algorithm 5 Locality-constrained Soft Assignment (LCSA).

Input: $\boldsymbol{X}, \boldsymbol{M}$ for a given forward pass, k': the k' nearest neighbors, σ: the bandwidth controlling the slope rate.

for $n = 1, \cdots, N'$ (this loop is batch solved)
  $\boldsymbol{\varphi} = \text{idx\_nn}(\boldsymbol{x}_n; \boldsymbol{M}, k')$ (k' indexes of k' nearest $\boldsymbol{m}_j$ of $\boldsymbol{x}_n$)
  $\boldsymbol{\alpha}'_n = \boldsymbol{\alpha}'(\boldsymbol{x}_n; [\boldsymbol{m}_{\varphi_1}, \cdots, \boldsymbol{m}_{\varphi_{k'}}])$ (SA computed by Eq. (15) on local dictionary)
  $\boldsymbol{\alpha}_n = \pi(\boldsymbol{\alpha}'_n; \boldsymbol{\varphi}, k)$ (deploy coeff. of $\boldsymbol{\alpha}'_n$ to locations $\boldsymbol{\varphi}$ of k dim. zeroed vector)
$\boldsymbol{\alpha} = [\boldsymbol{\alpha}_1, \cdots, \boldsymbol{\alpha}_{N'}]$ (concat. vectors $\boldsymbol{\alpha}_n$ into matrix $\boldsymbol{\alpha}$)

Output: $\boldsymbol{\alpha}$: locality-constrained soft-assigned codes

Squares Problem in the closed-form, we noted that a simple one step of SGD per mini-batch is sufficient to obtain good results. Of course, where needed and/or requested by reviewers, we will refine our claims and proofs further.

(a) OmniGAN, d'=256 (b) OmniGAN+LCSA, d'=256 (c) OmniGAN+LCSA, d'=512 (d) OmniGAN+LCSA, d'=1024

Figure 16. Examples of images generated after training on CIFAR-10 (32×32).

Algorithm 6 Dictionary Learning (DL).

Input: $\boldsymbol{X}, \boldsymbol{M}, \boldsymbol{\alpha}$ for a given forward pass, ω: the learning rate.

1: $\{m_{ij} \sim \mathcal{U}(-1,1)\}_{i=1,\cdots,d' \text{ and } j=1,\cdots,k}$

2: $\boldsymbol{M} \leftarrow \left[\frac{\boldsymbol{m}_j}{\|\boldsymbol{m}_j\|_1 + 1e\text{-}6}\right]_{j=1}^{k}$ (concat. of $\boldsymbol{m}_j$ into matrix $\boldsymbol{M}$)

3: **for** $i = 1, \cdots, \iota$

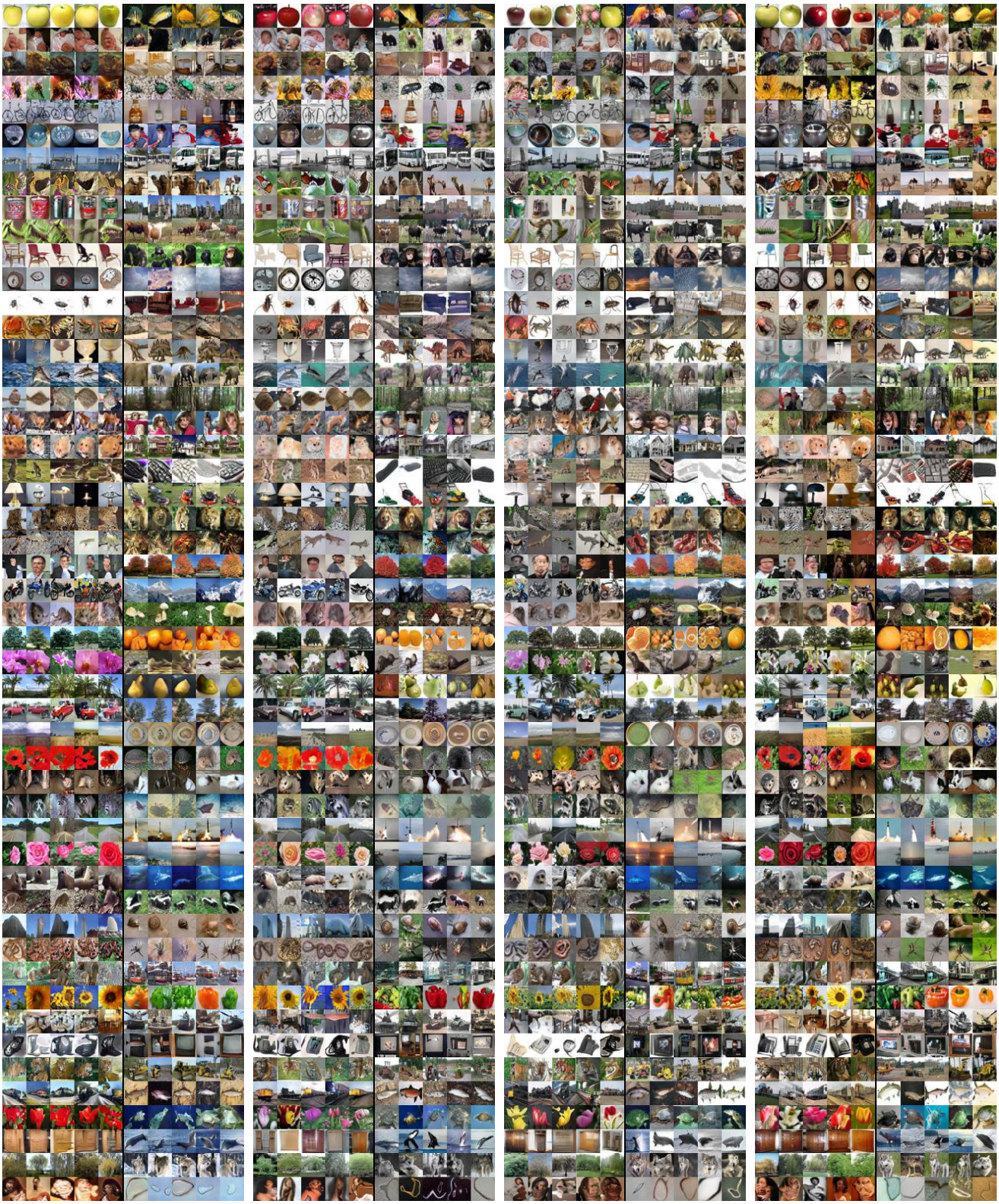
4: $\quad \Delta \boldsymbol{M} = 2(\boldsymbol{X} - \boldsymbol{M}\boldsymbol{\alpha})\boldsymbol{\alpha}^T$ (gradient computation)

5: $\quad \boldsymbol{M} \leftarrow \boldsymbol{M} - \frac{\omega}{(1+i)^{0.3}} \Delta \boldsymbol{M}$ (gradient descent)

Output: $\boldsymbol{M}$: dictionary atoms

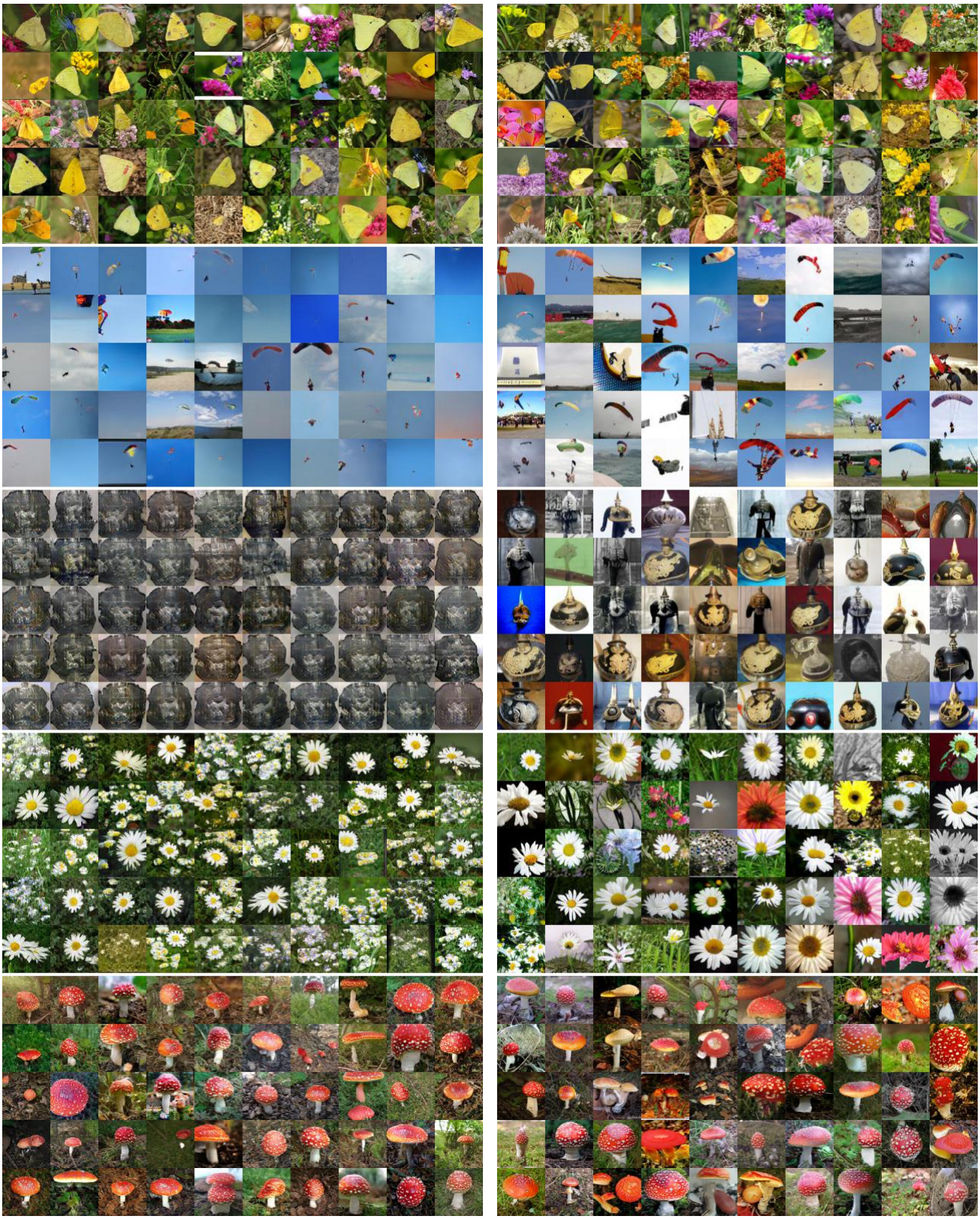
K. Generated Images

Figures 16, 17, 18, 19, and 20 show examples of images generated with OminGAN *vs*. OmniGAN+LCSA after training on CIFAR-10 (32×32), CIFAR-100 (32×32), ImageNet (64×64) (we also include BigGAN & BigGAN+LCSA) and ImageNet (128×128). Figure 21 shows examples of images generated with MSG-StyleGAN *vs*. MSG-StyleGAN+LCSA after training on Oxford-102 Flowers (256×256). Figures 22 and 23 shows examples of images generated with StyleGAN2 *vs*. StyleGAN2+LCSA for training on FFHQ (256×256).



(a) OmniGAN, d'=256 (b) OmniGAN+LCSA, d'=256 (c) OmniGAN+LCSA, d'=512 (d) OmniGAN+LCSA, d'=1024

Figure 17. Examples of images generated after training on CIFAR-100 (32×32). Each row contains samples from two classes. We note that OmniGAN+LCSA produces more diverse images than OmniGAN.

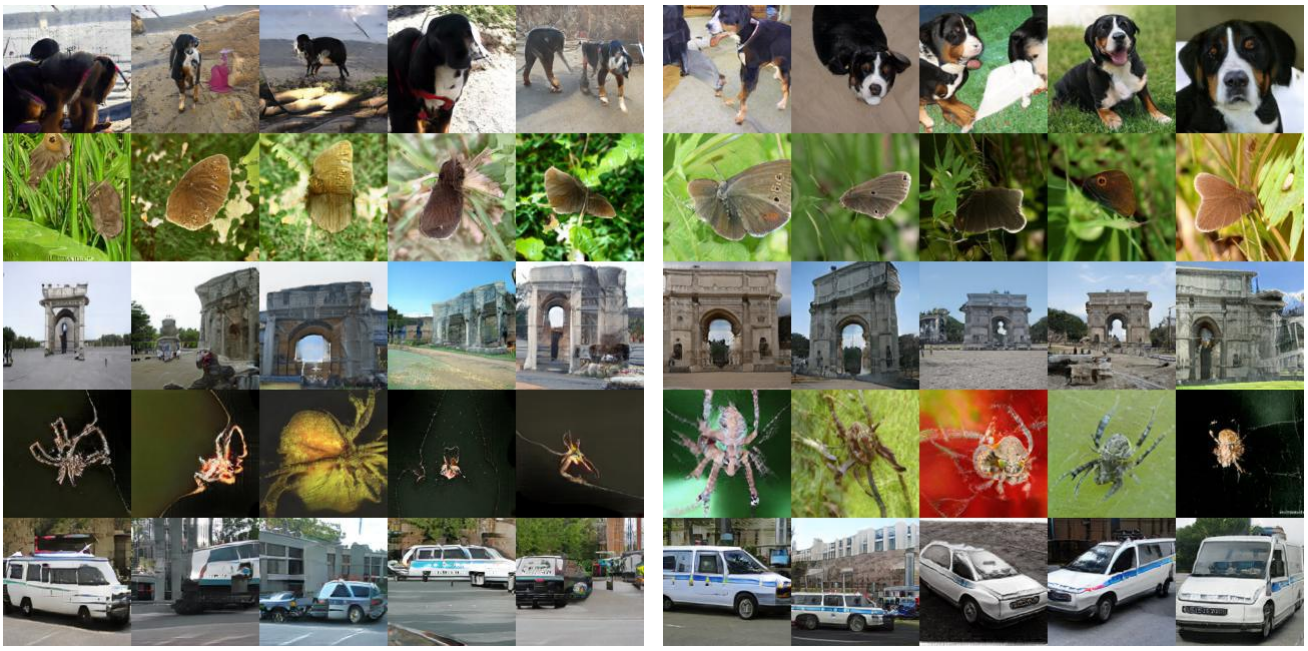


(a) BigGAN (b) BigGAN+LCSA

Figure 18. Images generated by BigGAN and BigGAN+LCSA on ImageNet (64×64). Our BigGAN+LCSA can generate more diverse and realistic images than BigGAN. We also note that the BigGAN has completely failed on class *pickelhaube* (the 3^{rd} row).

(a) OmniGAN (b) OmniGAN+LCSA

Figure 19. Images generated by OmniGAN and OmniGAN+LCSA on ImageNet (64×64). Our OmniGAN+LCSA produces more diverse and accurate images than OmniGAN alone.



(a) OmniGAN (b) OmniGAN+LCSA

Figure 20. Generated images on ImageNet (128×128) dataset. Our OmniGAN+LCSA can generate more diverse and realistic images than OmniGAN.

(a) MSG-StyleGAN (b) MSG-StyleGAN+LCSA

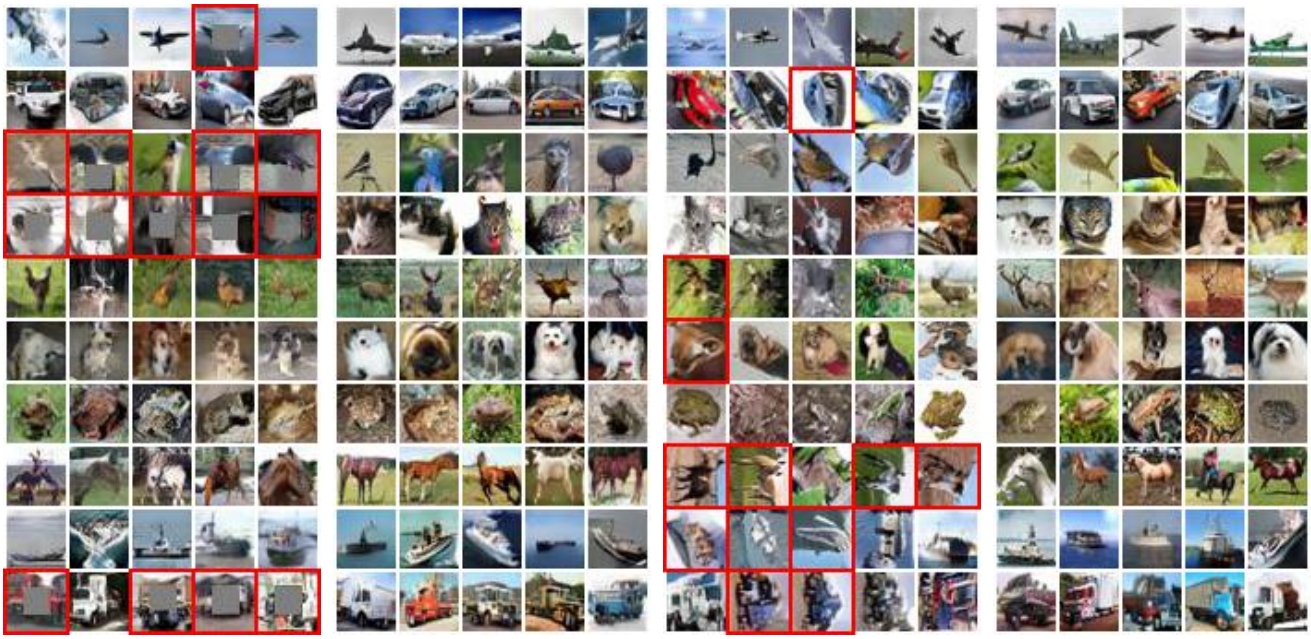
Figure 21. Generated images using the mixing of different input noises on Oxford-102 Flowers (256 × 256). Two sets of images are generated from their respective latent codes (the first row and the first column, respectively). The top left image (red box) is generated using the averaged latent inputs from images in the first row and the first column. The rest of images are generated by averaging the two latent inputs of images chosen from the first row and the first column *i.e.*, $\mathbf{I}_{i,j} = G((\boldsymbol{z}_{i,1} + \boldsymbol{z}_{1,j})/2)$. We input the same noise codes to both of the two models. Our MSG-StyleGAN+LCSA can generate more diverse and accurate flowers than MSG-StyleGAN alone.

(a) StyleGAN2 (b) StyleGAN2+LCSA

Figure 22. Examples of images generated by training on FFHQ (256×256) $70K$ dataset with truncation trick using $\psi = 0.5$ [21].

(a) StyleGAN2 (b) StyleGAN2+LCSA

Figure 23. Examples of images generated by training on FFHQ (256×256) $70K$ dataset without using the truncation trick. We notice that our StyleGAN2+LCSA can generate more realistic images than StyleGAN2 alone.



(a) DA (leak cutout) (b) DA+LCSA (c) ADA (leak rotation) (d) ADA+LCSA

Figure 24. Generated images of different methods limiting overfitting on 10% CIFAR-10 data. **DA leaks the cutout augmentation artifacts.** See images enclosed with red frames. **ADA also leaks the rotation augmentation artifacts. Combining LCSA with ADA or DA prevents the leakage of augmentation artifacts.**

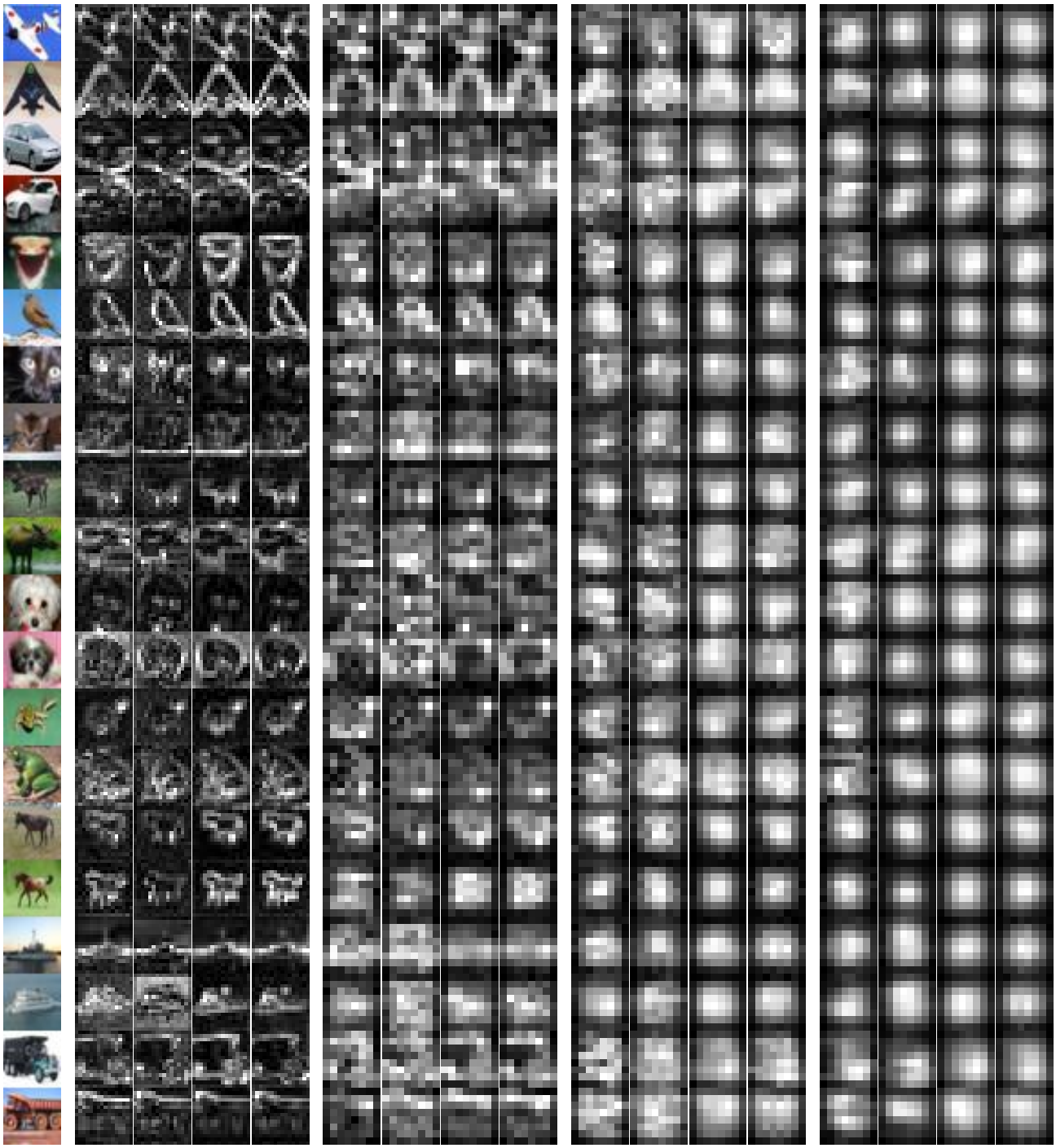
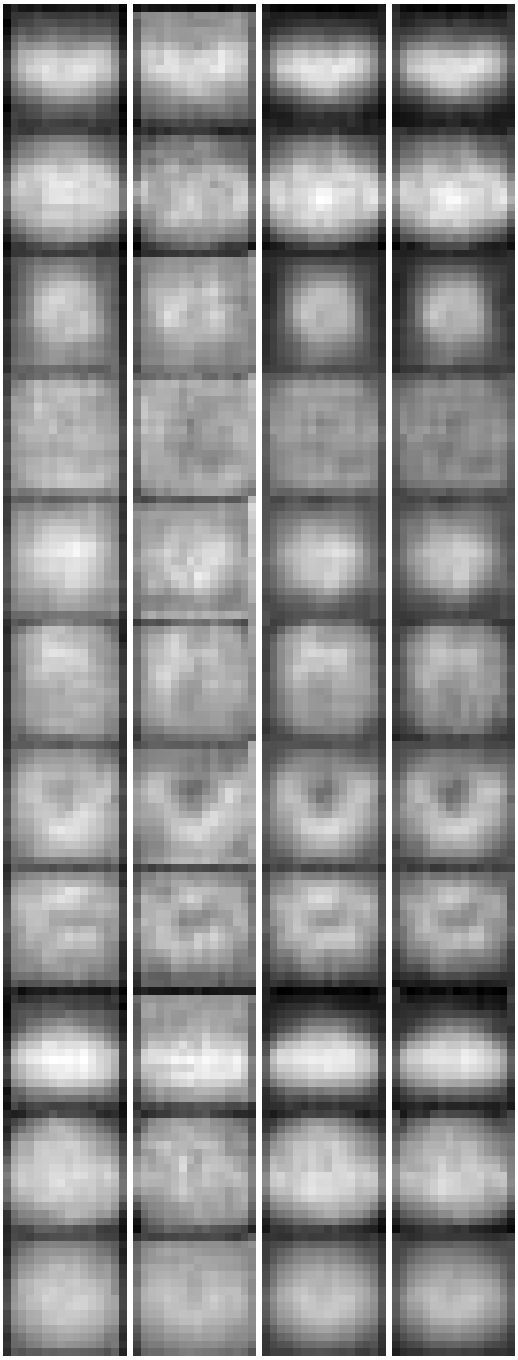


Figure 25. B1(LCSA LLC NNSC OMP), B2(LCSA LLC NNSC OMP), B3(LCSA LLC NNSC OMP), B4(LCSA LLC NNSC OMP). B1, B2, B3 and B4 denote the consecutive blocks of the discriminator. The first column shows the chosen images and remaining columns show the variance of coefficients of each $\boldsymbol{\alpha}$. Notice that LCSA results in a slightly larger variance on foregrounds and edges of objects than other coders while still maintaining a low variance on fairly basic visual appearances such as uniform backgrounds or simple textures. **We believe this is the example of the ability of LCSA to perform the quantization (backgrounds) *vs*. the approximate linear coding (foregrounds) trade-off which is controlled with the Lipschitz constant K (de facto σ).**



(a) LCSA (b) LLC (c) NNSC (d) OMP

Figure 26. Mean over variance of each $\boldsymbol{\alpha}$ on block B1. The first 10 rows correspond to the mean over variance of coefficients of each $\boldsymbol{\alpha}$ codes of each class, the last row is the mean over variance of coefficients of each $\boldsymbol{\alpha}$ over all classes. Notice that the combined variance of LCSA is again somewhat larger compared to NNSC and OMP but also lower than LLC in the areas of background. This indicates a good trade-off between quantization and reconstruction capacity of LCSA.

L. Sketch Proofs of Claims from Section 5

Below we provide proofs and proof sketches of properties of LCSA. We note that some of mathematics for $\boldsymbol{\alpha}'$ functions is cumbersome, thus we resort to key intuitions where necessary. We use one column display to ease readability of some math formulas. Before embarking on the proofs, we develop key quantities about $\boldsymbol{\alpha}'$. Since it lives on the probability simplex, we compute Shannon's entropy $S(\cdot) = -\mathbb{E}_{\alpha'}[\log \alpha']$ of $\boldsymbol{\alpha}'$ ($\boldsymbol{\alpha}$ being $\boldsymbol{\alpha}'$ composed with a linear operator, its fundamental geometric properties follow from $\boldsymbol{\alpha}'$). We recall

$$\boldsymbol{\alpha}'(\boldsymbol{x}; \boldsymbol{M}, \sigma) \quad = \quad \frac{1}{Z(\boldsymbol{x}; \boldsymbol{M}, \sigma)} \left[e^{-\frac{1}{2\sigma^2}||\boldsymbol{x}-\boldsymbol{m}_1||_2^2}, \cdots, e^{-\frac{1}{2\sigma^2}||\boldsymbol{x}-\boldsymbol{m}_k||_2^2} \right]^T \tag{21}$$

After some algebra that we skip for readability, we arrive at the statistical form of the gradient ∇S and Hessian $\mathrm{H}S$ at any $\boldsymbol{x}$ (skipped from notations for readability)

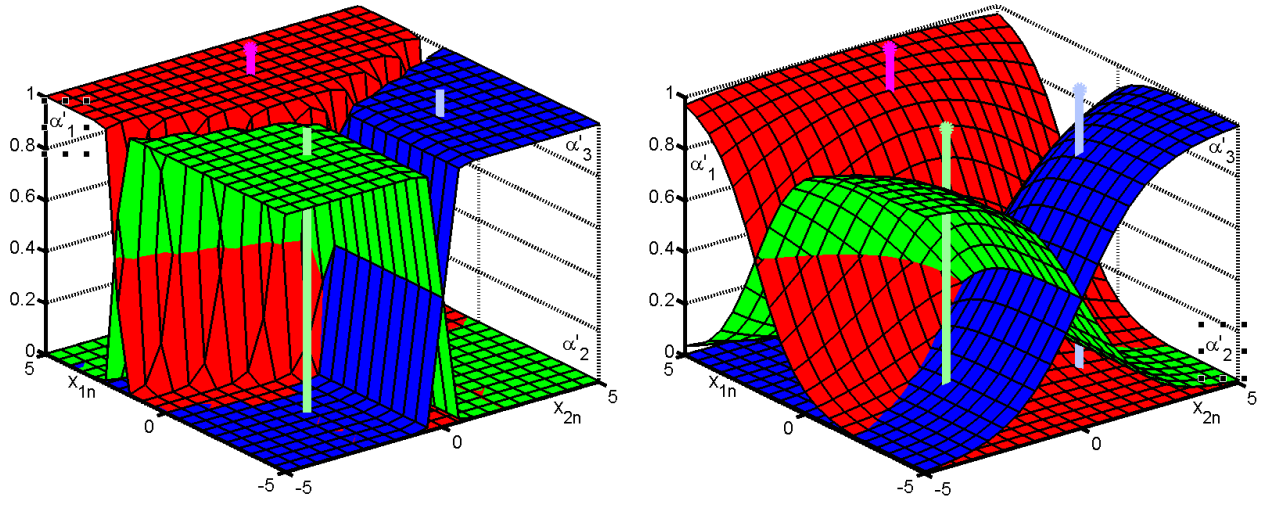
$$\begin{aligned}
\nabla S(\boldsymbol{x}) \quad &= \quad \frac{\mathbb{E}_p[\delta]}{2\sigma^2} \cdot (\mathbb{E}_q[\boldsymbol{m}] - \mathbb{E}_p[\boldsymbol{m}]), \\
\mathrm{H}S(\boldsymbol{x}) \quad &= \quad \frac{\mathbb{E}_p[\delta]}{2\sigma^4} \cdot \mathbb{E}_q[(\boldsymbol{m} - \mathbb{E}_p[\boldsymbol{m}])(\boldsymbol{m} - \mathbb{E}_p[\boldsymbol{m}])^T] - \frac{2 + \mathbb{E}_p[\delta]}{2\sigma^4} \cdot \mathbb{E}_p[(\boldsymbol{m} - \mathbb{E}_p[\boldsymbol{m}])(\boldsymbol{m} - \mathbb{E}_p[\boldsymbol{m}])^T],
\end{aligned}$$

where expectations are with respect to dictionary's columns, and for any such column $\boldsymbol{m}_i$, we let $\delta_i = \frac{\|\boldsymbol{x}-\boldsymbol{m}_i\|_2^2}{\sigma^2}$ and p, q are the distributions in the k-dim probability simplex with $p_i \propto \exp\left(-\frac{\|\boldsymbol{x}-\boldsymbol{m}_i\|_2^2}{2\sigma^2}\right)$ and $q_i \propto p_i \delta_i$. From the shape operator of the entropy (see *e.g.* [41]),

$$\mathrm{s}S \quad = \quad -\frac{1}{\sqrt{1 + \|\nabla S\|_2^2}} \cdot (\mathrm{I} + \nabla S \nabla S^T)^{-1} \mathrm{H}S,$$

we deduce the general formula for the mean curvature of S (using Sherman-Morrison Lemma to get rid of the inverse and properties of the trace),

$$\begin{aligned}
hS \quad &= \quad \frac{1}{k\sqrt{1 + \|\nabla S\|_2^2}} \cdot \mathrm{tr}\left(\left(\mathrm{I} - \frac{1}{1 + \|\nabla S\|_2^2} \cdot \nabla S \nabla S^T \right) \cdot -\mathrm{H}S \right) \\
&= \quad \frac{1}{k(1 + \|\nabla S\|_2^2)^{\frac{3}{2}}} \cdot \left[\nabla S^T \mathrm{H}S \nabla S - \|\nabla S\|_2^2 \cdot \mathrm{tr}(\mathrm{H}S) - \mathrm{tr}(\mathrm{H}S) \right].
\end{aligned}$$



(a) $\sigma^2 = 1$ (b) $\sigma^2 = 9$

Figure 27. Illustration of $\boldsymbol{\alpha}$ of LCSA. Two-dimensional anchors $[\boldsymbol{m}_1, \boldsymbol{m}_2, \boldsymbol{m}_3]$ were used (three poles). LCSA was applied w.r.t. $\boldsymbol{x} \in [-5, 5]^2$. Responses $\boldsymbol{\alpha}' = [\alpha'_1, \alpha'_2, \alpha'_3]$ are given in three colors.

For any vector $\boldsymbol{a}$ on the k-dim probability simples, let us use the shorthand $\boldsymbol{\mu}_a = \mathbb{E}_a[\boldsymbol{m}]$. Let us define, for any two such vectors $\boldsymbol{a}, \boldsymbol{b}$,

$$J(\boldsymbol{a},\boldsymbol{b}) \quad = \quad \mathbb{E}_a[\|\boldsymbol{m}\|_2^2]\|\boldsymbol{\mu}_b - \boldsymbol{\mu}_a\|_2^2 - \mathbb{E}_a[((\boldsymbol{\mu}_a - \boldsymbol{\mu}_b)^T\boldsymbol{m})^2] + (\boldsymbol{\mu}_b^\top \boldsymbol{\mu}_a)^2 - \|\boldsymbol{\mu}_a\|_2^2\|\boldsymbol{\mu}_b\|_2^2. \tag{22}$$

Notice that Cauchy-Schwartz inequality brings

$$\begin{aligned} \mathbb{E}_a[((\boldsymbol{\mu}_a - \boldsymbol{\mu}_b)^T\boldsymbol{m})^2] \quad &\leq \quad \mathbb{E}_a[\|\boldsymbol{m}\|_2^2]\|\boldsymbol{\mu}_b - \boldsymbol{\mu}_a\|_2^2, \qquad (23) \\ (\boldsymbol{\mu}_b^\top \boldsymbol{\mu}_a)^2 \quad &\leq \quad \|\boldsymbol{\mu}_a\|_2^2\|\boldsymbol{\mu}_b\|_2^2, \qquad (24) \end{aligned}$$

so we get a useful interval to which $J(\boldsymbol{a},\boldsymbol{b})$ belongs: $J(\boldsymbol{a},\boldsymbol{b}) \in \mathrm{I}(\boldsymbol{x};\boldsymbol{M},\sigma)$ with:

$$\mathrm{I}(\boldsymbol{x};\boldsymbol{M},\sigma) \quad = \quad \left[-\left(\|\boldsymbol{\mu}_a\|_2^2\|\boldsymbol{\mu}_b\|_2^2 - (\boldsymbol{\mu}_b^\top \boldsymbol{\mu}_a)^2\right), \mathbb{E}_a[\|\boldsymbol{m}\|_2^2]\|\boldsymbol{\mu}_b - \boldsymbol{\mu}_a\|_2^2 - \mathbb{E}_a[((\boldsymbol{\mu}_a - \boldsymbol{\mu}_b)^T\boldsymbol{m})^2]\right], \tag{25}$$

and we note that those Cauchy-Schwartz inequalities (23), (24) also yield

$$0 \quad \in \quad \mathrm{I}(\boldsymbol{x};\boldsymbol{M},\sigma). \tag{26}$$

After some more derivations, we arrive at the expression

$$\begin{aligned} hS \quad &= \quad \frac{\left\{ \begin{array}{l} \left(\frac{\mathbb{E}_p[\delta]}{2\sigma^2}\right)^2 \cdot \frac{2+\mathbb{E}_p[\delta]}{2\sigma^4} \cdot \left(\mathbb{E}_p[\|\boldsymbol{m}\|_2^2]\|\boldsymbol{\mu}_q - \boldsymbol{\mu}_p\|_2^2 - \mathbb{E}_p[((\boldsymbol{\mu}_p - \boldsymbol{\mu}_q)^T\boldsymbol{m})^2] + (\boldsymbol{\mu}_q^\top \boldsymbol{\mu}_p)^2 - \|\boldsymbol{\mu}_p\|_2^2\|\boldsymbol{\mu}_q\|_2^2\right) \\ -\left(\frac{\mathbb{E}_p[\delta]}{2\sigma^2}\right)^2 \cdot \frac{\mathbb{E}_p[\delta]}{2\sigma^4} \cdot \left(\mathbb{E}_q[\|\boldsymbol{m}\|_2^2]\|\boldsymbol{\mu}_q - \boldsymbol{\mu}_p\|_2^2 - \mathbb{E}_q[((\boldsymbol{\mu}_p - \boldsymbol{\mu}_q)^T\boldsymbol{m})^2] + (\boldsymbol{\mu}_p^T \boldsymbol{\mu}_q)^2 - \|\boldsymbol{\mu}_p\|_2^2\|\boldsymbol{\mu}_q\|_2^2\right) \end{array} \right\}}{k\left(1 + \left(\frac{\mathbb{E}_p[\delta]}{2\sigma^2}\right)^2 \cdot \|\boldsymbol{\mu}_q - \boldsymbol{\mu}_p\|_2^2\right)^{\frac{3}{2}}} \\ &= \quad \frac{(\mathbb{E}_p[\delta])^2}{k\sigma^2} \cdot \frac{(2+\mathbb{E}_p[\delta]) \cdot J(\boldsymbol{p},\boldsymbol{q}) - \mathbb{E}_p[\delta] \cdot J(\boldsymbol{q},\boldsymbol{p})}{\left(4\sigma^4 + (\mathbb{E}_p[\delta])^2 \cdot \|\boldsymbol{\mu}_q - \boldsymbol{\mu}_p\|_2^2\right)^{\frac{3}{2}}}. \end{aligned} \tag{27}$$

We are now ready to develop the proofs (or proof sketches) to our various Claims:

Proof of Claim 1. Since

$$\frac{1}{(z+a)^{\frac{3}{2}}} \quad = \quad \frac{1}{a^{\frac{3}{2}}} - \frac{3z}{2a^{\frac{5}{2}}} + o(z),$$

we get the Taylor expansion of hS in (27) as $\sigma \to 0$ ($Q, R \neq 0$):

$$hS \quad = \quad \frac{Q}{k\sigma^2} + \sigma^2 R + o(\sigma^2).$$

So hS diverges at rate $1/\sigma^2$ as $\sigma \to 0$, which shows that the entropy's support approaches a vertex of the probability simplex (where its curvature is maximal) and we get the HA solution.

Proof of Claim 2. We analyze cases for which $hS \to 0$. Looking at (27), we see several scenarii: (i) $\mathbb{E}_p[\delta]$ is small. This only happens when all the dictionary columns are close to each other *and* $\boldsymbol{x}$ is close to them. In Fig. 27, this would amount to brings all poles close to each other and would thus bring regions of linearity, i.e. the intersection of Voronoi cells, close to each other. Another case, (ii), is when $\|\boldsymbol{\mu}_q - \boldsymbol{\mu}_p\|_2 \leq \epsilon$ with a decreasing ϵ, because then both bounds of interval $\mathrm{I}(\boldsymbol{x};\boldsymbol{M},\sigma)$ are $O(\epsilon^2)$ and so the mean curvature vanishes as well. Looking at $\boldsymbol{\mu}_q - \boldsymbol{\mu}_p$, one can see that the norm of their differences vanishes in particular when there are two sets of dictionary anchors, one which are far away from $\boldsymbol{x}$, and thus with low weight on both p and q, and a second set, closer to $\boldsymbol{x}$ and such that all norms $\|\boldsymbol{m}_\cdot - \boldsymbol{x}\|_2$ are approximately constant, which precisely mean that $\boldsymbol{x}$ is close to the center of the Voronoi cell defined by those anchors. There is also a simple visual argument: Figure 27 shows that the rapid slope change takes place at $\boldsymbol{\mu}_{12} = \frac{1}{2}(\boldsymbol{m}_1 + \boldsymbol{m}_2)$, $\boldsymbol{\mu}_{13} = \frac{1}{2}(\boldsymbol{m}_1 + \boldsymbol{m}_3)$, $\boldsymbol{\mu}_{23} = \frac{1}{2}(\boldsymbol{m}_2 + \boldsymbol{m}_3)$ and $\boldsymbol{\mu}_{123} = \frac{1}{3}(\boldsymbol{m}_1 + \boldsymbol{m}_2 + \boldsymbol{m}_3)$. In fact, maximizing over the absolute value of derivatives of each α'_i w.r.t. $\boldsymbol{x}$, that is, $\mathrm{argmax}_{t\boldsymbol{x}'} \left|\frac{\partial \alpha'_i}{\partial \boldsymbol{x}'}\right|$, yields maximum at

locations $\boldsymbol{\mu}_{\Lambda(k')}$, where operator argmax$_t$ returns top $t=|\Lambda|$ maxima, Λ returns all possible ordered subsets of non-zero sizes of anchor indexes $i=1,\cdots,k'$ *e.g.*, for $k'=3$ we have $\{(1,2),(1,3),(2,3),(1,2,3)\}=\Lambda(3)$. At $\{\boldsymbol{\mu}_{\Lambda(k')}\}$ locations, the maximum indicates the largest slope changes which coincide with the linear slope regime of sigmoid function.

Proof of Claim 3. To this end, we seek the stationary points of $\epsilon^2=\|\boldsymbol{x}-\sum_k \boldsymbol{m}_k\alpha_k(\boldsymbol{x};\boldsymbol{M},\sigma)\|_2^2$ where

$$\alpha_k=\frac{\exp(-\|\boldsymbol{x}-\boldsymbol{m}_k\|^2/2\sigma^2)}{Z},$$

just as in (21) and $\boldsymbol{m}_k$ are dictionary atoms defining the Voronoi cell of $\boldsymbol{x}$. By setting $\frac{\partial\epsilon^2}{\partial\sigma}=0$ we obtain

$$-\frac{2}{\sigma^3}(\boldsymbol{x}-\boldsymbol{M}\boldsymbol{\alpha}(x))\Big(\sum_i \boldsymbol{m}_i\alpha_i(\boldsymbol{x})\big(\|\boldsymbol{x}-\boldsymbol{m}_i\|_2^2\big)-c\big)\Big)=0$$

where $c=\sum_k\|\boldsymbol{x}-\boldsymbol{m}_k\|_2^2\alpha_k(x)$. It is trivial to see that the first maximum is at the stationary point $\sigma=\infty$. Moreover, for $\sigma=0$ we have also the stationary point as

$$-\frac{2}{\sigma^3}(\boldsymbol{x}-\boldsymbol{M}\boldsymbol{\alpha}(x))\Big(\sum_i \boldsymbol{m}_i\alpha_i(\boldsymbol{x})\big(\|\boldsymbol{x}-\boldsymbol{m}_i\|_2^2\big)-c\big)\Big)=0 \tag{28}$$

$$-\frac{2}{\sigma^3}\big(\boldsymbol{x}-\boldsymbol{M}\boldsymbol{\alpha}(x)\big)\Big(\big(\sum_i \boldsymbol{m}_i\alpha_i(\boldsymbol{x})\|\boldsymbol{x}-\boldsymbol{m}_i\|_2^2\big)-\big(\sum_i \boldsymbol{m}_i\alpha_i^2(\boldsymbol{x})\|\boldsymbol{x}-\boldsymbol{m}_i\|_2^2\big)\Big)=0 \tag{29}$$

because if $\sigma=0$ then

$$\sum_i \boldsymbol{m}_i\alpha_i(\boldsymbol{x})\|\boldsymbol{x}-\boldsymbol{m}_i\|_2^2=\sum_i \boldsymbol{m}_i\alpha_i^2(\boldsymbol{x})\|\boldsymbol{x}-\boldsymbol{m}_i\|_2^2. \tag{30}$$

The last condition follows from the simplification of c due to the fact that for $\sigma=0$, $\boldsymbol{\alpha}$ codes obey the Hard Assignment, that is only one coefficient of $\boldsymbol{\alpha}$ is equal one, the rest are equal zero.

It is easy to see the other stationary point occurs if $\boldsymbol{x}=\boldsymbol{M}\boldsymbol{\alpha}(x)$ which we already identified as the case of perfect reconstruction.

The last set of stationary points needs to satisfy

$$\sum_i \boldsymbol{m}_i\alpha_i(\boldsymbol{x})\Big(d_i^2-\sum_k d_k^2\alpha_k(\boldsymbol{x})\Big)=0,$$

where $d_i^2=\|\boldsymbol{x}-\boldsymbol{m}_i\|_2^2$ (for d_k^2 just substitute k in place of i), and for these points we have checked via simulations that they yield the minimum reconstruction error.

Proof of Claim 4 (Lipschitz condition). Define $\boldsymbol{y}=\boldsymbol{M}\boldsymbol{\alpha}(\boldsymbol{x})=h(\boldsymbol{x})$, where $\boldsymbol{\alpha}$ is the LCSA mapping. The task is to compute the Jacobian matrix $\partial\boldsymbol{y}/\partial\boldsymbol{x}$. Therefore we compute $\partial y_i/\partial x_j$.

We have

$$\boldsymbol{y}=\sum_k \boldsymbol{m}_k\alpha_k \tag{31}$$

where

$$\alpha_k=\frac{\exp(-\|\boldsymbol{x}-\boldsymbol{m}_k\|^2/2\sigma^2)}{Z}$$

and Z is defined so that $\sum_k\alpha_k=1$. From this we get

$$\begin{aligned}\frac{\partial(\alpha_k Z)}{\partial x_j}&=\exp(-\|\boldsymbol{x}-\boldsymbol{m}_k\|^2/2\sigma^2)\ \ (m_{kj}-x_j)/\sigma^2\\&=Z\alpha_k(m_{kj}-x_j)/\sigma^2,\end{aligned} \tag{32}$$

where m_{kj} is the j-th component of $\boldsymbol{m}_k$.

In addition, since $Z = \sum_k \alpha_k Z$, summing (32) we get

$$\frac{\partial Z}{\partial x_j} = \sum_k Z\alpha_k(m_{kj} - x_j)/\sigma^2 \,. \tag{33}$$

From (31) we have,

$$y_i = \frac{\sum_k m_{ki}(\alpha_k Z)}{Z} \,,$$

where we have multiplied top and bottom by Z, for convenience. We differentiate this as a quotient, using (32) and (33), giving

$$\begin{aligned}
\sigma^2 \frac{\partial y_i}{\partial x_j} &= \frac{Z\big(\sum_k m_{ki} Z\alpha_k(m_{kj} - x_j)\big) - \big(\sum_k m_{ki}(\alpha_k Z)\big)\big(\sum_k Z\alpha_k(m_{kj} - x_j)\big)}{Z^2} \\
&= \sum_k m_{ki}\alpha_k(m_{kj} - x_j) - \Big(\sum_k m_{ki}\alpha_k\Big)\Big(\sum_k \alpha_k(m_{kj} - x_j)\Big) && \text{cancelling } Z^2 \\
&= \sum_k m_{ki}\alpha_k(m_{kj} - x_j) - \Big(\sum_k m_{ki}\alpha_k\Big)\Big((\sum_k \alpha_k m_{kj}) - x_j\Big) && \text{since } \sum_k \alpha_k x_j = x_j \\
&= \sum_k m_{ki}\alpha_k m_{kj} - \Big(\sum_k m_{ki}\alpha_k\Big)\Big(\sum_k \alpha_k m_{kj}\Big) && \text{cancelling } x_j \\
&= \sum_k \alpha_k m_{ki} m_{kj} - y_i y_j \,.
\end{aligned} \tag{34}$$

Finally, writing $\tilde{\boldsymbol{m}}_k = \boldsymbol{m}_k - \boldsymbol{y}$, this gives

$$\frac{\partial y_i}{\partial x_j} = \frac{1}{\sigma^2}\sum_k \alpha_k \tilde{m}_{ki}\tilde{m}_{kj} \,. \tag{35}$$

This shows the interesting fact that the Jacobian is symmetric positive-semidefinite, and can be written in a way depending only on $\boldsymbol{y} = h(\boldsymbol{x})$, but not on $\boldsymbol{x}$ directly.

A further interesting result can be derived from (35). Let $d\boldsymbol{x}$ be an infinitessimal change in $\boldsymbol{x}$. The corresponding infinitessimal change in $\boldsymbol{y}$ is computed by

$$\begin{aligned}
dy_i &= \frac{\partial y_i}{\partial \boldsymbol{x}}\, d\boldsymbol{x} \\
&= \sum_j \frac{\partial y_i}{\partial x_j}\, dx_j \\
&= \frac{1}{\sigma^2}\sum_k \alpha_k \tilde{m}_{ki} \sum_j \tilde{m}_{kj} dx_j && \text{after interchanging summation order} \\
&= \frac{1}{\sigma^2}\sum_k \alpha_k \tilde{m}_{ki} \left\langle \tilde{\boldsymbol{m}}_k, d\boldsymbol{x} \right\rangle \,.
\end{aligned}$$

Now, if $d\boldsymbol{x}$ is perpendicular to the simplex Δ, then it is perpendicular to each $\tilde{\boldsymbol{m}}_k$, so the inner product vanishes. Consequently, dy_i vanishes for all i, and $d\boldsymbol{y} = 0$. This shows that if $\boldsymbol{x}$ varies in a direction perpendicular to the simplex, then the value of $\boldsymbol{y} = h(\boldsymbol{x})$ does not change. This is a verification of the fact that $h(\boldsymbol{x})$ is constant on *fibres* perpendicular to the simplex Δ.

The formula (34) can be written neatly using matrices. Suppose that $\boldsymbol{M}$ is the matrix with columns $\boldsymbol{m}_k$. Then we have

$$\begin{aligned}
\sigma^2 \frac{\partial \boldsymbol{y}}{\partial \boldsymbol{x}} &= \boldsymbol{M}\mathrm{diag}(\boldsymbol{\alpha})\boldsymbol{M}^T - \boldsymbol{y}\boldsymbol{y}^T \\
&= \boldsymbol{M}(\mathrm{diag}(\boldsymbol{\alpha}) - \boldsymbol{\alpha}\boldsymbol{\alpha}^T)\boldsymbol{M}^T \,,
\end{aligned} \tag{36}$$

Lipschitz condition – L_1. A Lipschitz constant for a function $\boldsymbol{y} = h(\boldsymbol{x})$ with respect to some norm $\|\cdot\|$ is a constant K such that

$$\|\boldsymbol{y} - \boldsymbol{y}'\| \leq K\|\boldsymbol{x} - \boldsymbol{x}'\| \,.$$

For a differentiable function a Lipschitz constant for the L_1 vector norm is given by

$$K = \max_j \|\partial \boldsymbol{y}/\partial x_j\|_1$$

evaluated over the domain of the function (a single Voronoi cell for instance).

From (35) we have

$$\begin{aligned}
K &= (1/\sigma^2) \max_j \Big\| \sum_k \alpha_k \tilde{\boldsymbol{m}}_k \, \tilde{m}_{kj} \Big\|_1 \\
&\leq (1/\sigma^2) \max_j \sum_k \|\alpha_k \tilde{\boldsymbol{m}}_k \tilde{m}_{kj}\|_1 && \text{triangle inequality} \\
&\leq (1/\sigma^2) \sum_k \max_j \|\alpha_k \tilde{\boldsymbol{m}}_k \tilde{m}_{kj}\|_1 && \text{interchanging } \max \text{ and sum} \\
&= (1/\sigma^2) \sum_k \max_j \alpha_k |\tilde{m}_{kj}| \; \|\tilde{\boldsymbol{m}}_k\|_1 && \text{taking scalars outside the norm} \\
&= (1/\sigma^2) \sum_k \alpha_k \|\tilde{\boldsymbol{m}}_k\|_1 \max_j |\tilde{m}_{kj}| && \text{taking terms outside } \max_j \\
&= (1/\sigma^2) \sum_k \alpha_k \|\tilde{\boldsymbol{m}}_k\|_1 \; \|\tilde{\boldsymbol{m}}_k\|_\infty \\
&\leq (1/\sigma^2) \max_k \|\tilde{\boldsymbol{m}}_k\|_1 \; \|\tilde{\boldsymbol{m}}_k\|_\infty && \text{since } \sum_k \alpha_k = 1
\end{aligned}$$

This leads to

$$K \leq \frac{\max_k \|\tilde{\boldsymbol{m}}_k\|_1^2}{\sigma^2}$$

Since $\tilde{\boldsymbol{m}}_k = \boldsymbol{m}_k - \boldsymbol{y}$ and $\boldsymbol{y}$ can vary, we see that $\|\tilde{\boldsymbol{m}}_k\|^2 \leq D^2$ where D (with respect to the given norm) is the diameter of the simplex with vertices $\boldsymbol{m}_k$. (The diameter of a set is the supremum of distances between two points in the set.) This gives finally

$$K \leq \frac{D^2}{\sigma^2} \,.$$

This will be true for any norm that is greater than the ∞-norm for all points. However, it therefore applies to any norm that is greater than a multiple of the ∞-norm. Therefore we can conclude Proposition 2.4 as follows.

Proposition 2.4: For any norm $\|\cdot\|$ on $\mathbb{R}^d$ equivalent to the 2-norm, a Lipschitz constant for the LCSA mapping on a given Voronoi region is given by

$$K = \frac{D^2}{\sigma^2}$$

where D is the diameter of the simplex Δ.

Lipschitz condition – L_2. For the L_2 norm a Lipschitz constant is given by

$$K = \max_{\|v\|} \frac{\|J\boldsymbol{v}\|}{\|\boldsymbol{v}\|}$$

where $\boldsymbol{v}$ is a vector and $J = \partial y_i/\partial x_j$ is the Jacobian. In the case where J is symmetric positive definite (such as in the present case), this is equal to

$$K = \max_{\|v\|=1} \boldsymbol{v}^T J \boldsymbol{v} \,. \tag{37}$$

With $J = \partial y_i / \partial x_j = \sum_k \alpha_k \tilde{m}_{ki} \tilde{m}_{kj} / \sigma^2$ as in (35) this becomes

$$\begin{aligned} \boldsymbol{v}^T J \boldsymbol{v} &= (1/\sigma^2) \sum_k \alpha_k \langle \tilde{\boldsymbol{m}}_k, \boldsymbol{v} \rangle^2 \\ &\leq (1/\sigma^2) \max_k \langle \tilde{\boldsymbol{m}}_k, \boldsymbol{v} \rangle^2 . \end{aligned}$$

This quantity is maximized, over vectors $\boldsymbol{v}$ of norm 1 when $\boldsymbol{v} = \arg\max_k \|\tilde{\boldsymbol{m}}_k\|$ (up to scale), in which case it is equal to $\max_k \|\tilde{\boldsymbol{m}}_k\|^2 / \sigma^2$. This shows that when $\|\boldsymbol{v}\| = 1$,

$$\begin{aligned} \boldsymbol{v}^T J \boldsymbol{v} &\leq \max_k \|\tilde{\boldsymbol{m}}_k\|^2 / \sigma^2 \\ &\leq D^2 / \sigma^2. \end{aligned} \tag{38}$$

This shows that the same Lipschitz constant $K = D^2/\sigma^2$ holds for the L_2 as for the L_1 Lipschitz condition.

Proof of Claim 5. The LCSA encoding $\boldsymbol{M}\boldsymbol{\alpha}(\boldsymbol{x})$ is non-continuous at the boundaries of the Voronoi regions simply because the set of k' nearest neighbor anchors of $\boldsymbol{x}$ changes as $\boldsymbol{x}$ crosses from one Voronoi region to another the Voronoi region.

Proof of Claim 7. We penalise the spectral norm of Jacobian matrix $\left\| \frac{\partial \boldsymbol{M}\boldsymbol{\alpha}(\boldsymbol{x})}{\partial \boldsymbol{x}} \right\|_2 = K$ via de facto controlling the Lipschitz constant $K = D^2/\sigma^2$ because this is shown in (37) and (38).

Claims 1–8 highlight that LCSA can perform the feature quantization in some regions of the feature space (proximity of dictionary atoms), as well as it can perform the approximate linear coding akin to linear coding methods (the proximity of the mean of atoms of Voronoi cell). The Lipschitz constant highlights that our σ controls denoising achieved via LCSA due to the mechanism akin to DAE (whose denoising effect is controlled by its σ'). Our blocks in the GAN discriminator can guide the dictionary atoms and conv. features towards quantization or reconstruction with higher linear fidelity, also selecting some intermediate denoising hypothesis as the Lipschitz constant poses the upper bound controlling denoising effect.