508 In this supplementary, we provide:

509 ◇ Predictable hyperparameters introduction.

510 ◇ Training and implementation details.

511 ◇ Additional results of model parsing and CNN-generated image detection.

512 ◇ The construction of RED140 dataset.

513 ◇ Hyperparameter ground truth and the model parsing performance for each GM

A   Predictable Hyperparameters Introduction

515 We investigate 37 hyperparameters that exhibit the predictability according to Asnani *et al.* [1].
516 These hyperparameters are categorized into three groups: (1) Loss Function (Tab. 4), (2) Discrete
517 Architecture Hyperparameters (Tab. 5), (3) Continuous Architecture Hyperparameters (Tab. 6). We
518 report our proposed method performance of parsing hyperparameters in these three groups via Fig. 7a,
519 Fig. 7b, and Fig. 7c, respectively. Moreover, in the main paper's Eq. 8 and Fig. 4a, we employ the
520 assignment hierarchy $\mathbf{M}^s$ to group different hyperparameters together, which supervises the learning
521 of the matching matrix $\mathbf{M}$. The construction of such the $\mathbf{M}^s$ is also based on Tab. 4, 5, and 6, which
522 not only define three coarse-level categories, but also fine-grained categories such as pixel-level
523 objective (loss) function (*e.g.*, L1 and MSE) in Tab. 4, and normalization methods (*e.g.*, ReLu and
524 Tanh) as well as nonlinearity functions (*e.g.*, Layer Norm. and Batch Norm.) in Tab. 5.

B   Training and Implementation Details

526 **Training Details** Given the directed graph $\mathbf{A} \in \mathbb{R}^{C \times C}$, which contains C graph nodes. We
527 empirically set C as 55, as mentioned in the main paper Sec. 3.3. In the training, LGPN takes the
528 given image $\mathbf{I}$ and output the refined feature $\mathbf{V} \in \mathbb{R}^{55 \times D}$, which contains learned features for each
529 graph node, namely, $\mathbf{V} = \{\mathbf{v}_0, \mathbf{v}_1, ..., \mathbf{v}_{54}\}$. As a matter of fact, we can view $\mathbf{V}$ as three separate
530 sections: $\mathbf{V}^l = \{\mathbf{v}_0, \mathbf{v}_1, ..., \mathbf{v}_9\}$, $\mathbf{V}^d = \{\mathbf{v}_{10}, \mathbf{v}_{11}, ..., \mathbf{v}_{27}\}$, and $\mathbf{V}^c = \{\mathbf{v}_{28}, \mathbf{v}_{29}, ..., \mathbf{v}_{54}\}$, which
531 denote learned features for graph nodes of 10 loss functions (*e.g.*, L1 and MSE), 18 discrete architecture
532 hyperparameter (*e.g.*, Batch Norm. and ReLU), and 9 continuous architecture hyperparameter (*i.e.*,
533 Parameter Num.), respectively. Note $\mathbf{V}^c$ represents learned features of 9 continuous architecture
534 hyperparameters because each continuous hyperparameter is represented by 3 graph nodes, as
535 illustrated in Fig. 1c of the main paper. Furthermore, via Eq. 7 in the main paper, we use $\mathbf{V}$ to obtain
536 the corresponding probability score $\mathbf{p} = \{p_0, p_1, ..., p_{54}\}$ for each graph node. Similar to $\mathbf{V}$, this
537 $\mathbf{p}$ can be viewed as three sections: $\mathbf{p}^l \in \mathbb{R}^{10}$, $\mathbf{p}^d \in \mathbb{R}^{18}$ and $\mathbf{p}^c \in \mathbb{R}^{27}$ for loss functions, discrete
538 architecture hyperparameters, continuous architecture hyperparameters, respectively. In the end,
539 we use $\mathbf{p}$ to help optimize LGPN via the graph node classification loss (Eq. 7). After the training
540 converges, we further apply individual fully connected layers on the top of frozen learned features
541 of continuous architecture hyperparameters (*e.g.*, $\mathbf{V}^c$). via minimizing the $\mathcal{L}_1$ distance between
542 predicted and ground truth value.

543 In the inference (the main paper Fig. 4b), for loss function and discrete architecture hyperparameters,
544 we use output probabilities (*e.g.*, $\mathbf{p}^l$ and $\mathbf{p}^d$) of discrete value graph nodes, for the binary "used v.s.
545 not" classification. For the continuous architecture hyperparameters, we first concatenate learned
546 features of corresponding graph nodes. We utilize such a concatenated feature with pre-trained, fully
547 connected layers to estimate the continuous hyperparameter value.

548 **Model Parsing Implementation Details** Denote the output feature from the Generation Trace
549 Capturing Network as $\mathbf{f} \in \mathbb{R}^{2048}$. To transform $\mathbf{f}$ into a set of features $\mathbf{H} = \{\mathbf{h}_0, \mathbf{h}_1, ..., \mathbf{h}_{54}\}$ for the
550 55 graph nodes, 55 independent linear layers (Θ) are employed. Each feature $\mathbf{h}_i$ is of dimension
551 $\mathbb{R}^{128}$. The $\mathbf{H}$ is fed to the GCN refinement block, which contains 5 GCN blocks, each of which has 2
552 stacked GCN layers. In other words, the GCN refinement block has 10 layers in total. We use the
553 correlation graph $\mathbf{A} \in \mathbb{R}^{55 \times 55}$ (Fig. 9) to capture the hyperparameter dependency and during the
554 training the LGPN pools $\mathbf{A}$ into $\mathbf{A}_1 \in \mathbb{R}^{18 \times 18}$ and $\mathbf{A}_2 \in \mathbb{R}^{6 \times 6}$ as the Fig. 2 of the main paper. The
555 LGPN is implemented using the PyTorch framework. During training, a learning rate of 3e-2 is used.
556 The training is performed with a batch size of 400, where 200 images are generated by various GMs
557 and 200 images are real.

14

Table 4: Loss Function types used by all GMs. We group the 10 loss functions into three categories. We use the binary representation to indicate the presence of each loss type in training the respective GM.

Category	Loss Function
Pixel-level	L_1
	L_2
	Mean squared error (MSE)
	Maximum mean discrepancy (MMD)
	Least squares (LS)
Discriminator	Wasserstein loss for GAN (WGAN)
	Kullback–Leibler (KL) divergence
	Adversarial
	Hinge
Classification	Cross-entropy (CE)

Table 5: Discrete Architecture Hyperparameters used by all GMs. We group the 18 discrete architecture hyperparameters into 6 categories. We use the binary representation to indicate the presence of each hyperparameter type in training the respective GM.

Category	Discrete Architecture Hyperparameters
Normalization	Batch Normalization
	Instance Normalization
	Adaptive Instance Normalization
	Group Normalization
Nonlinearity in the Last Layer	ReLU
	Tanh
	Leaky_ReLU
	Sigmoid
	SiLU
Nonlinearity in the Last Block	ELU
	ReLU
	Leaky_ReLU
	Sigmoid
	SiLU
Up-sampling	Nearest Neighbour Up-sampling
	Deconvolution
Skip Connection	Feature used
Down-sampling	Feature used

Table 6: Continuous Architecture Hyperparameters used by all GMs, where "[" denotes inclusive and "(" denotes exclusive intervals. We report the range for 9 continuous hyperparameters.

Category	Range	Discrete Architecture Hyperparameters
Layer Number	$(0—717]$	Layers Number
	$[0—289]$	Convolution Layer Number
	$[0—185]$	Fully-connected Layer Number
	$[0—46]$	Pooling Layer Number
	$[0—235]$	Normalization Layer Number
	$(0—20]$	Layer Number per Block
Unit Number	$(0—8,365]$	Filter Number
	$(0—155]$	Block Number
	$(0—56,008,488]$	Parameter Number

Implementation Details for Detection and Attributions We validate GTC's effectiveness in capturing the generation trace in Fig. 6. Specifically, Fig. 8 shows the detailed implementation. We employ FC layers to convert output feature $\mathbf{f} \in \mathbb{R}^{2048}$ to $\mathbf{f}_{det.} \in \mathbb{R}^2$ and $\mathbf{f}_{att.} \in \mathbb{R}^5$ for CNN-generated image detection and image attribution respectively.

C Additional Results

We report the detailed performance on RED116 via Tab. 9, demonstrating that our proposed LGPN achieves the best performance on both datasets. Also, In Fig. 5a of the main paper, we visualize the

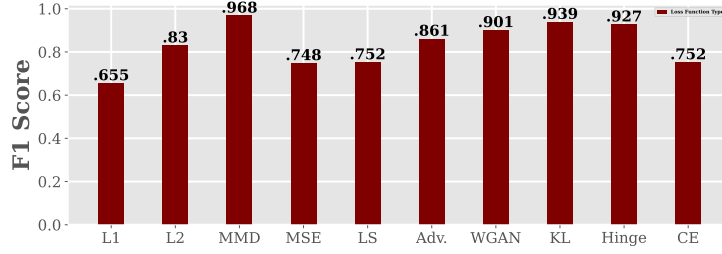
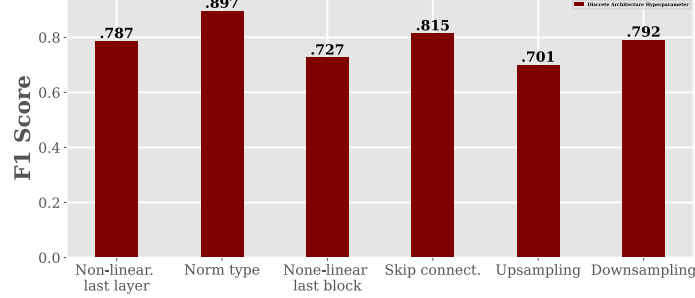
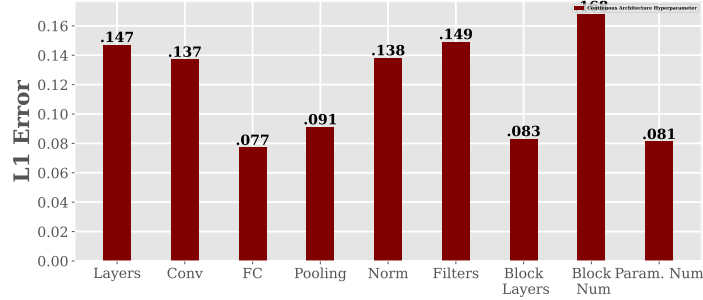
15

(a)

(b)

(c)

Figure 7: (a) The F1score on the loss function reveals that `MMD` and `KL` are two easiest loss functions to predict. (b) The F1 score on the discrete architecture hyperparameters demonstrates that predicting these hyperparameters is more challenging than predicting the loss function. This finding aligns with the empirical results reported in the previous work [1]. (c) The L1 error on the continuous architecture hyperparameters indicates that it is challenging to predict `Block Num.` and `Filter Num.`.

Table 7: Using different n graph nodes for the continuous hyperparameter regression.

n Value	2	3	4	5	6
L1 Error	0.123	0.120	0.124	0.132	0.130

correlation graph similarities among different GMs in the first test set. In this section, we offer a similar visualization (*e.g.,* Fig. 10 of the supplementary) for other test sets. Supplementary Fig. 11 shows the relationship between graph node classification and GM correlation graph similarity for GMs. In the main paper's Sec 3.3, we use n graph nodes for each continuous hyperparameter, and n is set as 3. Tab. 7 shows the advantage of choice, which shows the lowest $\mathcal{L}_1$ regression error is achieved when n is 3. Furthermore, in addition to Fig. 6 a of the main paper, we provide the detailed CNN-generated image detection performance measured by in Supplementary's Tab. 8.

16

(a) CNN-generated image detection   (b) Image attribution.

Figure 8: We construct simple classifiers based on GTC. Then, we train these two classifiers for CNN-generated image detection and image attribution, respectively. Please note that GTC only leverages ImageNet pre-trained weights as the initialization, same as the previous method [54]. For a fair comparison, no model parsing datasets such as RED116 and RED140 are used for pre-training.

| Detection method | Generative Adversarial Networks | | | | | | Deep-fakes | Low level vision | | Perceptual loss | | Avg. |
	Pro-GAN	Cycle-GAN	Big-GAN	Style-GAN	Gau-GAN	Star-GAN	FF++	SITD	SAN	CRN	IMLE	
Wang. *et al* [54]	**100.0**	96.83	88.24	98.29	98.09	95.44	66.27	86.0	61.2	98.94	99.52	89.89
DE-FAKE [46]	96.03	97.74	89.93	78.09	98.15	98.34	81.43	88.71	**90.14**	90.01	95.14	91.52
HiFi-Net [20]	96.00	89.50	96.31	90.05	88.00	96.40	84.71	79.90	81.00	90.00	62.00	87.51
Ojha. *et al* [41]	**100.0**	99.46	**99.59**	97.24	**99.98**	99.60	82.45	61.32	79.02	96.72	99.00	92.21
NPR. [49]	**100.0**	**99.50**	94.50	99.80	88.80	**100.0**	86.20	75.32	82.72	94.72	98.11	93.30
Ours	**100.0**	97.03	97.76	**99.96**	98.97	99.65	**93.50**	**98.42**	85.00	**99.60**	**99.90**	**97.27**

Table 8: To show the effectiveness of the GTC, we report CNN-generated image detection performance measured by Average Precision on 11 forgery methods containing face and non-face images. Previous methods' quantitative results are taken from [41]. [**Bold**: best result].

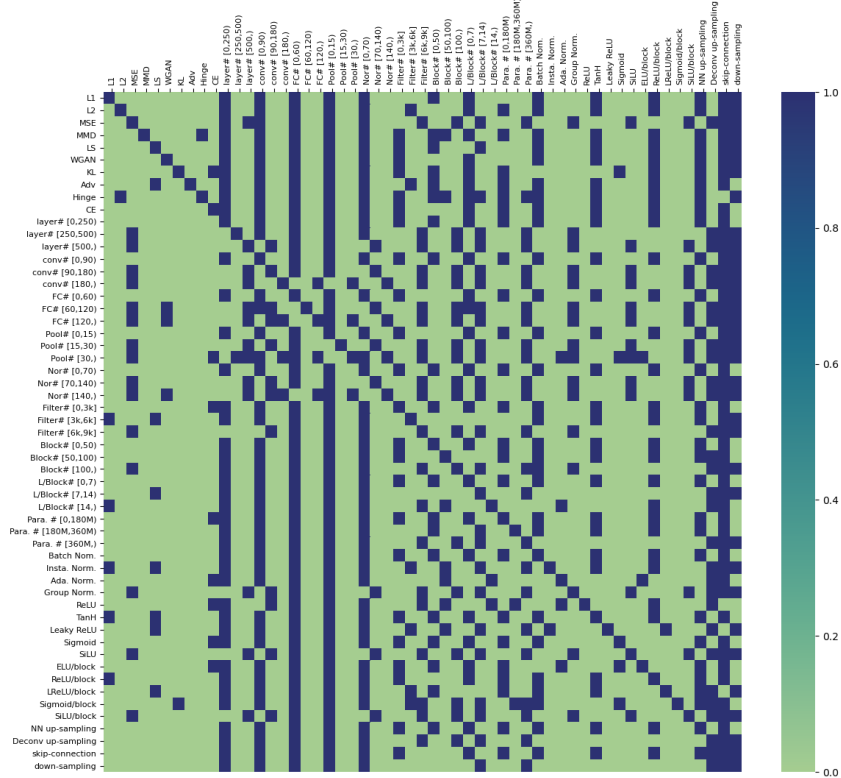



Figure 9: The initial correlation graph **A** that we construct based on the probability table in Sec. 3.1 of the main paper. The optimum threshold we use is 0.45.

17

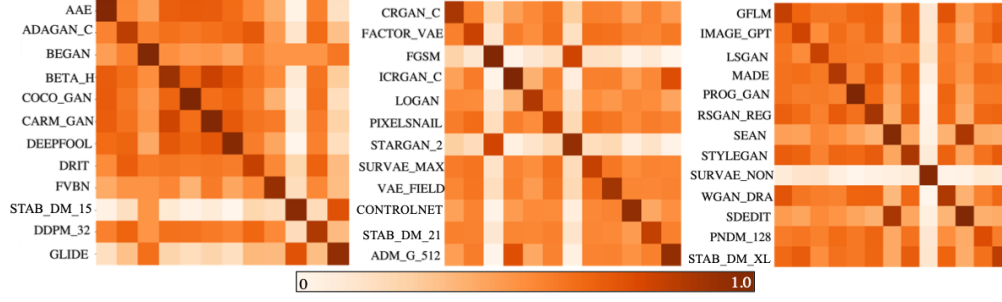



Figure 10: Each element of these two matrices is the average cosine similarities of $2,000$ pairs of generated correlation graphs $\mathbf{A}'_0$ from corresponding GMs in the second, third and forth test sets.

Method	Loss Function		Dis. Archi. Para.		Con. Archi. Para.
	F1 ↑	Acc. ↑	F1 ↑	Acc. ↑	L1 error ↓
Random GT [1]	0.636	0.716	0.529	0.575	0.184
FEN-PN [1]	0.813	0.792	0.718	0.706	0.149
FEN-PN* [1]	0.801	0.811	0.701	0.708	0.146
GTC w MLP	0.778	0.801	0.689	0.701	0.169
GTC w Stacked -GCN	0.790	0.831	0.698	0.720	0.145
LGPN	**0.841**	**0.833**	**0.727**	**0.755**	**0.130**

Table 9: The model parsing performance on RED116. In the last row, our proposed LGPN that contains GTC and GCN Refinement block, which achieves the best model parsing performance in all metrics. [**Key**: GCN refine.: GCN refinement block; **Bold**: best.].

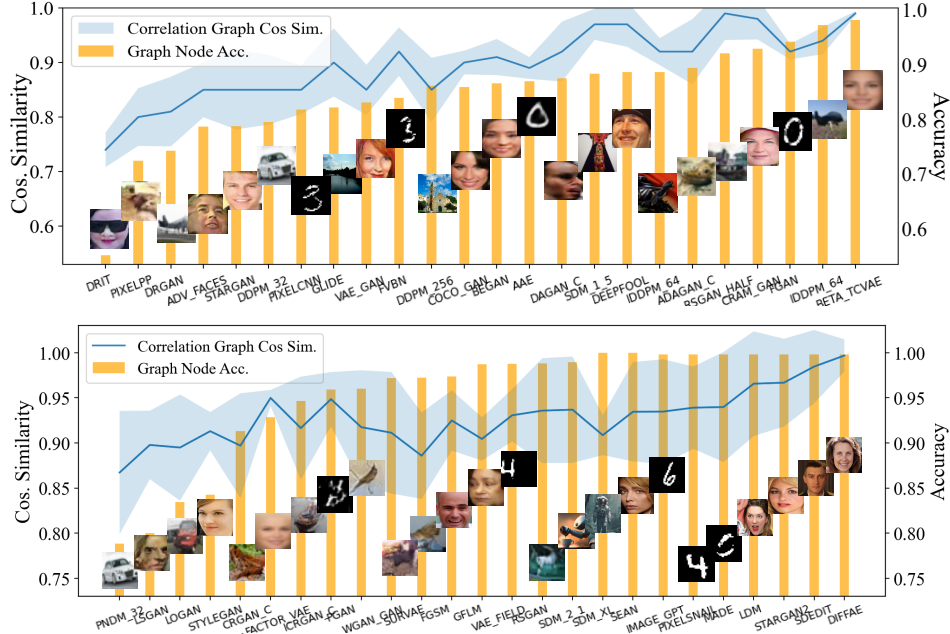



Figure 11: For each GM in four test sets. The upper figure corresponds to the first and second test sets, while the lower figure corresponds to the third and fourth test sets. The positive correlation between the cosine similarity between image pairs' correlation graphs (*i.e.*, $\mathbf{A}'_0$) and the graph node classification accuracy achieved by LGPN on that particular GM. This positive correlation demonstrates that the correlation graph is important to the performance of the graph node classification, which encourages the LGPN to learn the representation for each hyperparameter.

18

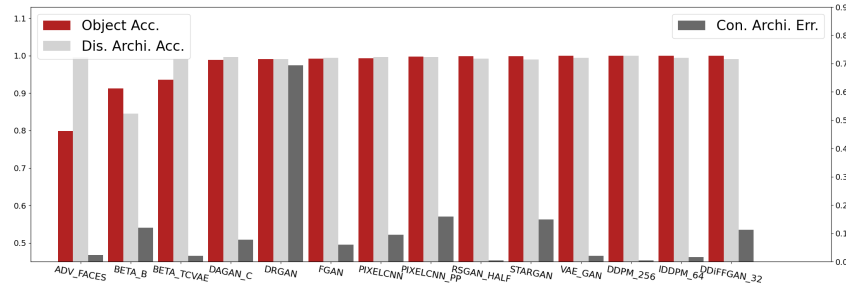
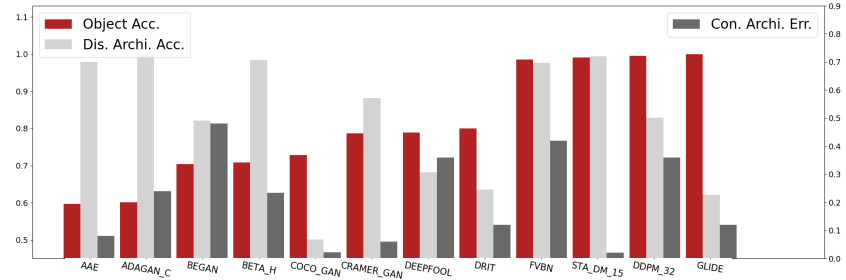
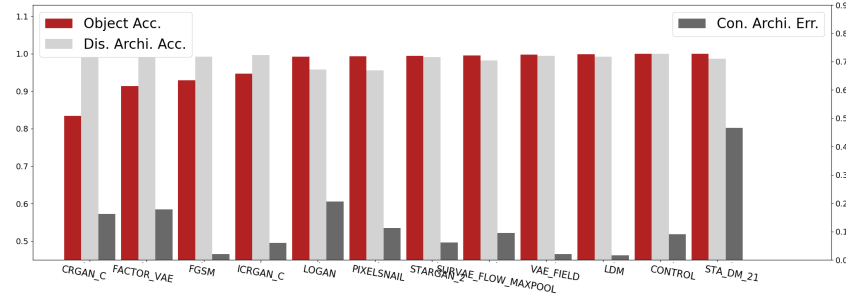
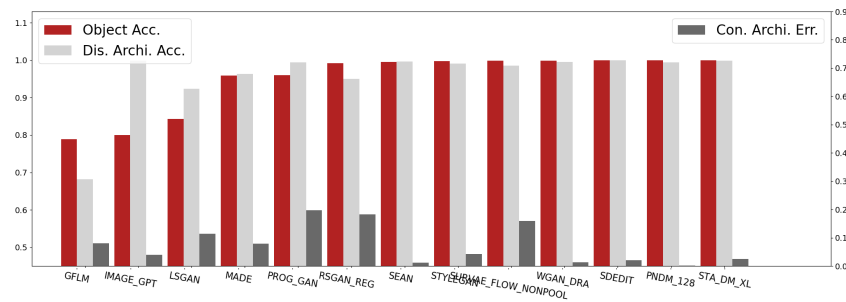



Figure 12: We report detailed model parsing results on different GMs in each test set. These results include loss function and discrete architecture hyperparameter prediction accuracy, as well as the L1 error on the continuous architecture hyperparameter prediction. Specifically, (a), (b), (c), and (d) are the performance for GMs in the 1st, 2nd, 3rd, and 4th test sets, respectively.

19

Table 10: Test sets used for evaluation. Each set contains generative models from GAN, DM, VAE, AR (Auto-Regressive), AA (Adversarial Attack), and NF (Normalizing Flow). All test sets contain face and non-face in the image content. [Keys: R means GM is used in the test set of RED116 but is not used in RED140.]

Set 1	Set 2	Set 3	Set 4
ADV_FACES	AAE	BICYCLE_GAN (R)	GFLM
BETA_B	ADAGAN_C	BIGGAN_512 (R)	IMAGE_GPT
BETA_TCVAE	BEGAN	CRGAN_C	LSGAN
BIGGAN_128 (R)	BETA_H	FACTOR_VAE	MADE
DAGAN_C	BIGGAN_256 (R)	FGSM	PIX2PIX (R)
DRGAN	COCOGAN	ICRGAN_C	PROG_GAN
FGAN	CRAMERGAN	LOGAN	RSGAN_REG
PIXEL_CNN	DEEPFOOL	MUNIT (R)	SEAN
PIXEL_CNN++	DRIT	PIXEL_SNAIL	STYLE_GAN
RSGAN_HALF	FAST_PIXEL(R)	STARGAN_2	SURVAE_FLOW_NONPOOL
STARGAN	FVBN	SURVAE_FLOW_MAXPOOL	WGAN_DRA
VAEGAN	SRFLOW (R)	VAE_FIELD	YLG (R)
DDPM_256	STABLE_DM_21	LDM	PNDM_32
IDDPM_64	DDPM_32	CONTROLNET	STABLE_DM_XL
Denoise_GAN_32	GLIDE	STABLE_DM_15	SEDdit

(a)   (b)   (c)

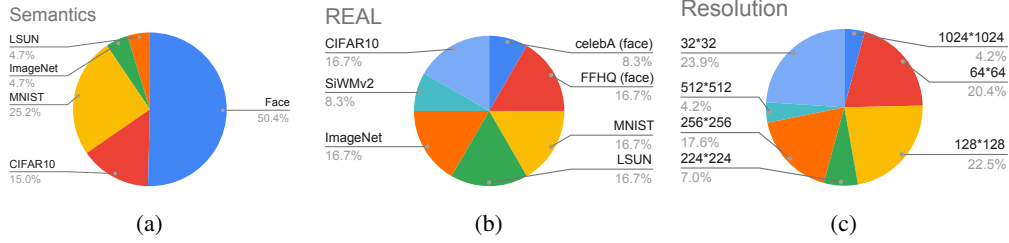



Figure 13: RED140 statistics. (a) The dataset is trained on various image contents or semantics. (b) The real-image category contains many real-image datasets that GMs are trained on [29, 28, 13, 60, 21, 32, 33]. (c) The GM has various image resolutions.

D   RED140 Dataset

In this section, we provide an overview of the RED140 dataset, which is used for both model parsing and coordinated attack detection. Note that, for the experiment reported in Tab. **??** of the supplementary, we follow the test sets defined in RED116 [1]. When we construct RED140, we use images from ImageNet, FFHQ, CelebHQ, CIFAR10, and LSUN as the real-images category of RED140. We exclude GM that is not trained in the real-image category of RED140. In addition, both RED116 and RED140 contain various image content and resolution, and the details about RED140 are uncovered in Fig. 13 of the supplementary. For test sets (Tab. 10 of the supplementary), we follow the dataset partition of RED116, excluding the GMs that are trained on real images, which RED140 does have. For example, JFT-300M is used to train BigGAN, so we remove `BIGGAN_128`, `BIGGAN_256` and `BIGGAN_512` in the first, second, and third test sets.

E   GM Hyperparameter Ground Truth

In this section, we report the ground truth vector of different hyperparameters of each GM contained in the RED140. Specifically, Tab. 11 and Tab. 12 report the loss function ground truth vector for each GM. Tab. 13 and Tab. 14 report the discrete architecture hyperparameter ground truth for each GM. Tab. 15 and Tab. 16 report the continuous architecture hyperparameter ground truth for each GM. The detailed model parsing performance on each GM is reported in the supplementary's Fig. 12.

20

Table 11: Ground truth Loss Function feature vector used for prediction of loss type for all GMs. The loss function ground truth is in (Tab. 4).

GM	L_1	L_2	MSE	MMD	LS	WGAN	KL	Adversarial	Hinge	CE
AAE	1	0	0	0	0	0	0	0	0	1
ACGAN	1	0	0	0	0	0	0	0	0	1
ADAGAN_C	0	0	0	0	1	0	0	0	0	1
ADAGAN_P	0	0	0	0	1	0	0	0	0	0
ADM_G_128	0	0	1	0	0	0	0	0	0	0
ADM_G_256	0	0	1	0	0	0	0	0	0	0
ADV_FACES	1	0	1	0	1	0	0	0	0	0
ALAE	0	0	1	0	1	0	0	0	0	0
BEGAN	1	0	0	0	0	0	0	0	0	0
BETA_B	0	0	0	0	0	0	1	0	0	1
BETA_H	0	0	0	0	0	0	1	0	0	1
BETA_TCVAE	1	0	0	0	0	0	1	0	0	1
BGAN	0	0	0	0	1	0	0	0	0	1
BICYCLE_GAN	1	0	1	0	0	0	1	0	0	0
BIGGAN_128	1	0	0	0	0	0	0	0	0	0
BIGGAN_256	1	0	0	0	0	0	0	0	0	0
BIGGAN_512	1	0	0	0	0	0	0	0	0	0
Blended_DM	0	0	1	0	0	0	0	0	0	0
CADGAN	0	0	0	1	0	0	0	0	0	0
CCGAN	0	0	0	0	1	0	0	1	0	0
CGAN	0	0	1	0	1	0	0	0	0	0
CLIPDM	0	0	1	0	0	0	0	0	0	0
COCO_GAN	1	1	0	0	0	1	0	0	1	0
COGAN	0	0	0	0	1	0	0	0	0	0
COLOUR_GAN	1	0	0	0	1	0	0	0	0	0
CONT_ENC	0	1	0	0	1	0	0	0	0	0
CONTRAGAN	1	0	0	0	0	0	0	1	0	1
CONTROLNET	0	0	1	0	0	0	0	0	0	0
COUNCIL_GAN	1	0	1	0	1	0	0	0	0	0
CRAMER_GAN	0	0	0	0	0	1	0	0	0	0
CRGAN_C	1	1	0	0	0	0	0	0	0	1
CRGAN_P	1	1	0	0	0	0	0	0	0	0
CYCLEGAN	1	0	0	0	1	0	0	0	0	0
DAGAN_C	1	0	0	0	0	0	0	0	0	1
DAGAN_P	1	0	0	0	0	0	0	0	0	0
DCGAN	0	0	0	0	0	0	0	0	0	1
DDPM_32	0	0	1	0	0	0	0	0	0	0
DDPM_256	0	0	1	0	0	0	0	0	0	0
DDiFFGAN_32	0	0	1	0	0	1	0	0	0	0
DEEPFOOL	1	1	0	0	0	0	0	0	0	0
DFCVAE	0	1	0	0	0	0	1	0	0	1
DIFFAE_256	0	0	1	0	0	0	0	0	0	0
DIFFAE_LATENT	0	0	1	0	0	0	0	0	0	0
DIFF-ProGAN	0	0	0	0	0	0	1	0	0	0
DIFF-StyleGAN	0	0	0	0	0	0	1	0	0	0
DIFF-ISGEN	0	0	0	0	0	0	1	0	0	0
DISCOGAN	1	0	0	0	1	0	0	0	0	0
DRGAN	0	0	0	0	1	0	0	0	0	1
DRIT	1	0	0	0	1	0	0	0	0	1
DUALGAN	1	0	0	0	0	1	0	0	0	0
EBGAN	0	1	0	0	1	0	0	1	1	0
ESRGAN	1	0	0	0	1	0	0	0	0	0
FACTOR_VAE	1	0	0	0	0	0	1	0	0	1
Fast pixel	0	0	0	0	0	0	0	0	0	1
FFGAN	1	1	0	0	1	0	0	0	0	1
FGAN	0	0	0	0	1	0	0	1	0	0
FGAN_KL	1	0	0	0	0	0	0	0	0	0
FGAN_NEYMAN	0	1	0	0	0	0	0	0	0	0
FGAN_PEARSON	0	0	1	0	0	0	0	0	1	0
FGSM	0	0	0	0	1	0	0	0	0	0
FPGAN	1	1	0	0	1	0	0	0	0	1
FSGAN	1	0	0	0	1	0	0	0	0	1
FVBN	0	0	0	0	0	0	0	0	0	1
GAN_ANIME	1	1	0	0	0	1	0	0	1	0
Gated_pixel_cnn	0	0	0	0	0	0	0	0	0	1
GDWCT	1	0	1	0	0	0	0	0	1	0
GFLM	0	0	1	0	0	0	0	0	0	1
GGAN	1	0	0	0	0	0	0	0	0	0
GLIDE	0	0	1	0	0	0	0	0	0	0
ICRGAN_C	1	1	0	0	0	0	0	0	0	1
ICRGAN_P	1	1	0	0	0	0	0	0	0	0

21

Table 12: Ground truth Loss Function feature vector used for prediction of loss type for all GMs. The loss function ground truth is in (Tab. 4).

GM	L_1	L_2	MSE	MMD	LS	WGAN	KL	Adversarial	Hinge	CE
IDDPM_32	0	0	1	0	0	0	0	0	0	0
IDDPM_64	0	0	1	0	0	0	0	0	0	0
IDDPM_256	0	0	1	0	0	0	0	0	0	0
ILVER_256	0	0	1	0	0	0	0	0	0	0
Image_GPT	0	0	0	0	0	0	0	0	0	1
INFOGAN	0	0	1	0	1	0	0	0	0	1
LAPGAN	0	0	0	0	1	0	0	0	0	0
LDM	0	0	1	0	0	0	0	0	0	0
LDM_CON	0	0	1	0	0	1	0	0	0	0
Lmconv	0	0	0	0	0	0	0	0	0	1
LOGAN	1	1	0	0	0	0	0	1	0	0
LSGAN	0	0	1	0	0	0	0	0	1	0
MADE	0	0	0	0	0	0	0	0	0	1
MAGAN	0	0	1	0	0	0	0	0	0	0
MEMGAN	0	0	0	0	1	0	0	0	0	0
MMD_GAN	1	0	0	1	0	0	0	0	0	0
MRGAN	0	0	1	0	1	0	0	0	0	0
MSG_STYLE_GAN	0	0	0	0	1	0	0	0	0	0
MUNIT	1	0	0	0	1	0	0	0	0	0
NADE	0	0	0	0	0	0	0	0	0	1
OCFGAN	0	0	0	1	0	0	0	0	1	0
PGD	1	1	0	0	0	0	0	0	0	0
PIX2PIX	1	0	0	0	1	0	0	0	0	0
PixelCNN	0	0	0	0	0	0	0	0	0	1
PixelCNN++	0	0	0	0	0	0	0	0	0	1
PIXELDA	0	0	0	0	1	0	0	0	1	1
PixelSnail	0	0	0	0	0	0	0	0	0	1
PNDM_32	0	0	1	0	0	0	0	0	0	0
PNDM_256	0	0	1	0	0	0	0	0	0	0
PROG_GAN	0	0	0	0	0	1	0	0	1	0
RGAN	0	0	0	0	0	1	0	0	0	0
RSGAN_HALF	0	0	0	0	0	0	0	0	0	1
RSGAN_QUAR	0	0	0	0	0	0	0	0	0	1
RSGAN_REG	0	0	0	0	0	0	0	0	0	1
RSGAN_RES_BOT	0	0	0	0	0	0	0	0	0	1
RSGAN_RES_HALF	0	0	0	0	0	0	0	0	0	1
RSGAN_RES_QUAR	0	0	0	0	0	0	0	0	0	1
RSGAN_RES_REG	0	0	0	0	0	0	0	0	0	1
SAGAN	0	0	0	0	1	0	0	0	0	0
SCOREDIFF_256	0	0	1	0	0	0	0	0	0	0
SDEdit_256	0	0	1	0	0	0	0	0	0	0
SEAN	1	0	0	0	1	0	0	0	0	0
SEMANTIC	0	1	0	0	1	0	0	0	0	0
SGAN	0	0	0	0	1	0	0	0	0	1
SNGAN	0	0	0	0	1	0	0	1	0	0
SOFT_GAN	0	0	0	0	1	0	0	0	0	0
SRFLOW	1	0	0	0	0	0	0	0	0	1
SRRNET	0	1	1	0	1	0	0	0	0	1
STANDARD_VAE	0	0	0	0	0	0	1	0	0	1
STARGAN	1	0	0	0	1	0	0	0	0	1
STARGAN_2	1	0	0	0	1	0	0	0	0	0
STA_DM_15	0	0	1	0	0	0	0	0	0	0
STA_DM_21	0	0	1	0	0	0	0	0	0	0
STA_DM_XL	0	0	1	0	0	0	0	0	0	0
STGAN	1	0	0	0	1	1	0	0	0	0
STYLEGAN	0	1	0	0	0	1	0	0	0	0
STYLEGAN_2	0	1	0	0	1	0	0	0	1	0
STYLEGAN2_ADA	0	1	0	1	1	0	0	0	1	0
SURVAE_FLOW_MAXPOOL	0	0	0	0	0	0	1	0	0	1
SURVAE_FLOW_NONPOOL	0	0	0	0	0	0	1	0	0	1
TPGAN	1	0	0	0	0	1	0	0	0	0
UGAN	0	0	0	0	1	0	0	0	0	0
UNIT	0	0	0	0	1	0	1	0	0	0
VAE_field	0	0	0	0	0	0	1	0	0	1
VAE_flow	0	0	0	0	0	0	1	0	0	1
VAEGAN	1	0	0	0	1	0	1	0	0	0
VDVAE	0	0	0	0	0	0	1	0	0	1
WGAN	0	0	0	0	0	1	0	0	0	0
WGAN_DRA	0	0	1	0	0	1	0	0	0	0
WGAN_WC	0	0	0	0	0	1	0	0	0	0
WGANGP	0	1	0	0	0	1	0	0	0	0
YLG	0	0	0	0	0	1	0	0	0	0

22

Table 13: Ground truth feature vector used for prediction of Discrete Architecture Hyperparameters for all GMs. The discrete architecture hyperparameter ground truth is defined in (Tab. 5). **A — D** are Batch Norm., Instance Norm., Adaptive Instance Norm., and Group Norm., respectively. **E — I** are non-linearity in the last layer, and they are ReLU, Tanh, Leaky_ReLu, Sigmoid, and SiLU. **J — N** are non-linearity in the last block, and they are ELU, ReLU, Leaky_ReLu, Sigmoid, and SiLU. **O** and **P** are Nearest Neighbour and Deconvolution Upsampling. **Q** and **L** are Skip Connection and Downsampling.

GM	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	L
AAE	1	0	0	0	0	1	0	0	0	1	0	0	0	0	1	0	1	0
ACGAN	1	0	0	0	0	1	0	0	0	0	1	0	0	0	1	0	1	0
ADAGAN_C	1	0	0	0	0	1	0	0	0	0	1	0	0	0	1	0	1	0
ADAGAN_P	1	0	0	0	0	1	0	0	0	0	1	0	0	0	1	0	1	0
ADM_G_128	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	1	1	1
ADM_G_256	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	1	1	1
ADV_FACES	0	1	0	0	0	1	0	0	0	0	1	0	0	0	1	0	1	0
ALAE	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	1	0	1
BEGAN	1	0	0	0	0	1	0	0	0	1	0	0	0	0	1	0	0	0
BETA_B	0	0	0	0	0	0	0	1	0	0	1	0	0	0	1	0	1	1
BETA_H	0	0	0	0	0	0	0	1	0	0	1	0	0	0	1	0	1	1
BETA_TCVAE	0	0	0	0	0	0	0	1	0	0	1	0	0	0	1	0	1	1
BGAN	1	0	0	0	0	1	0	0	0	0	0	1	0	0	1	0	0	0
BICYCLE_GAN	1	0	0	0	0	1	0	0	0	0	1	0	0	0	1	0	0	0
BIGGAN_128	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	1	1
BIGGAN_256	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	1	1
BIGGAN_512	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	1	1
Blended_DM	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	1	1	1
CADGAN	1	0	0	0	0	1	0	0	0	0	1	0	0	0	1	0	1	1
CCGAN	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	1	1
CGAN	1	0	0	0	0	1	0	0	0	0	0	1	0	0	1	0	0	0
CLIPDM	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	1	1
COCO_GAN	1	0	0	0	0	1	0	0	0	0	1	0	0	0	1	0	0	0
COGAN	1	0	0	0	0	1	0	0	0	0	0	1	0	0	1	0	1	1
COLOUR_GAN	1	0	0	0	0	1	0	0	0	0	1	0	0	0	1	0	1	1
CONT_ENC	1	0	0	0	0	1	0	0	0	0	0	1	0	0	1	0	1	1
CONTRAGAN	1	0	0	0	0	1	0	0	0	0	1	0	0	0	1	0	1	0
CONTROLNET	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	1	1	1
COUNCIL_GAN	0	1	0	0	0	1	0	0	0	0	1	0	0	0	1	0	1	0
CRAMER_GAN	1	0	0	0	0	1	0	0	0	0	1	0	0	0	1	0	1	0
CRGAN_C	1	0	0	0	0	1	0	0	0	0	1	0	0	0	1	0	1	0
CRGAN_P	1	0	0	0	0	1	0	0	0	0	1	0	0	0	1	0	1	0
CYCLEGAN	0	1	0	0	0	1	0	0	0	0	1	0	0	0	0	1	1	1
DAGAN_C	1	0	0	0	0	1	0	0	0	0	1	0	0	0	1	0	1	0
DAGAN_P	1	0	0	0	0	1	0	0	0	0	1	0	0	0	1	0	1	0
DCGAN	1	0	0	0	0	1	0	0	0	0	1	0	0	0	1	0	1	0
DDiFFGAN_32	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	1	1	1
DDPM_32	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	1	1
DDPM_256	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	1	1
DEEPFOOL	0	0	1	0	1	0	0	0	0	0	1	0	0	0	0	1	0	0
DFCVAE	1	0	0	0	0	0	0	1	0	0	1	0	0	0	1	0	0	1
DIFF_ISGEN	1	0	0	0	0	0	0	1	0	0	0	0	1	0	0	1	1	0
DIFF_PGAN	1	0	0	0	0	0	0	1	0	0	0	1	0	0	1	0	1	0
DIFF_SGAN	1	0	0	0	0	0	0	1	0	0	0	1	0	0	1	0	1	0
DIFFAE	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	1	1	1
DIFFAE_LATENT	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	1	1	1
DISCOGAN	0	1	0	0	0	1	0	0	0	0	1	0	0	0	0	1	1	1
DRGAN	1	0	0	0	0	1	0	0	0	1	0	0	0	0	1	0	1	1
DRIT	0	1	0	0	0	1	0	0	0	0	1	0	0	0	0	1	1	1
DUALGAN	1	0	0	0	0	1	0	0	0	0	1	0	0	0	1	0	0	0
EBGAN	1	0	0	0	0	1	0	0	0	0	0	1	0	0	1	0	0	1
ESRGAN	0	0	1	0	0	0	1	0	0	0	0	1	0	0	0	1	0	0
FACTOR_VAE	0	0	0	0	0	0	0	1	0	0	1	0	0	0	1	0	1	1
FASTPIXEL	1	0	0	0	0	0	0	1	0	1	0	0	0	0	1	0	1	0
FFGAN	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	1	1
FGAN	1	0	0	0	0	0	0	1	0	0	1	0	0	0	1	0	1	0
FGAN_KL	1	0	0	0	0	0	0	1	0	0	1	0	0	0	1	0	1	0
FGAN_NEYMAN	1	0	0	0	0	0	0	1	0	0	1	0	0	0	1	0	1	0
FGAN_PEARSON	1	0	0	0	0	0	0	1	0	0	1	0	0	0	1	0	1	0
FGSM	0	0	1	0	1	0	0	0	0	0	1	0	0	0	0	1	0	0
FPGAN	0	1	0	0	0	1	0	0	0	0	1	0	0	0	1	0	0	1
FSGAN	1	0	0	0	1	0	0	0	0	0	1	0	0	0	0	1	1	1
FVBN	0	0	1	0	0	0	0	1	0	1	0	0	0	0	1	0	1	0
GAN_ANIME	0	1	0	0	0	1	0	0	0	0	1	0	0	0	1	0	1	1
GATED_PIXEL_CNN	0	0	1	0	0	0	0	1	0	0	0	1	0	0	0	1	1	0
GDWCT	0	1	0	0	0	1	0	0	0	0	1	0	0	0	1	0	0	1
GFLM	0	0	1	0	1	0	0	0	0	0	1	0	0	0	0	1	0	0
GGAN	1	0	0	0	0	1	0	0	0	0	1	0	0	0	1	0	1	1
GLIDE	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	1	1	1

23

Table 14: Ground truth feature vector used for prediction of Discrete Architecture Hyperparameters for all GMs. The discrete architecture hyperparameter ground truth is defined in (Tab. 5). **A — D** are Batch Norm., Instance Norm., Adaptive Instance Norm., and Group Norm., respectively. **E — I** are non-linearity in the last layer and they are ReLU, Tanh, Leaky_ReLu, Sigmoid, and SiLU. **J — N** are non-linearity in the last block and they are ELU, ReLU, Leaky_ReLu, Sigmoid, and SiLU. **O** and **P** are Nearest Neighbour and Deconvolution Upsampling. **Q** and **L** are Skip Connection and Downsampling.

GM	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	L
ICRGAN_C	1	0	0	0	0	1	0	0	0	0	1	0	0	0	1	0	1	0
ICRGAN_P	1	0	0	0	1	0	0	0	0	0	1	0	0	0	1	0	1	0
IDDPM_32	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	1	1	1
IDDPM_64	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	1	1	1
IDDPM_256	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	1	1	1
IMAGE_GPT	1	0	0	0	0	0	0	1	0	0	0	1	0	0	0	1	1	1
INFOGAN	1	0	0	0	0	1	0	0	0	0	0	1	0	0	1	0	0	1
ILVER_256	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	1	1
LAPGAN	0	0	1	0	0	1	0	0	0	0	1	0	0	0	0	1	1	0
LDM	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	1	1	1
LDM_CON	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	1	1	1
LMCONV	0	0	1	0	0	0	0	1	0	1	0	0	0	0	1	0	1	1
LOGAN	1	0	0	0	0	1	0	0	0	0	1	0	0	0	1	0	1	0
LSGAN	1	0	0	0	0	1	0	0	0	0	1	0	0	0	1	0	0	0
MADE	0	0	1	0	0	0	0	1	0	1	0	0	0	0	1	0	1	0
MAGAN	1	0	0	0	0	1	0	0	0	0	1	0	0	0	1	0	1	0
MEMGAN	1	0	0	0	0	1	0	0	0	0	1	0	0	0	1	0	1	0
MMD_GAN	1	0	0	0	0	1	0	0	0	0	1	0	0	0	1	0	1	0
MRGAN	1	0	0	0	0	1	0	0	0	0	1	0	0	0	1	0	1	0
MSG_STYLE_GAN	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	1	0	1
MUNIT	0	1	0	0	1	0	0	0	0	0	1	0	0	0	0	1	1	1
NADE	0	0	1	0	0	0	0	1	0	1	0	0	0	0	1	0	1	0
OCFGAN	1	0	0	0	0	1	0	0	0	0	1	0	0	0	1	0	1	0
PGD	0	0	1	0	1	0	0	0	0	0	1	0	0	0	0	1	0	0
PIX2PIX	0	1	0	0	0	1	0	0	0	0	0	1	0	0	0	1	1	1
PIXELCNN	1	0	0	0	0	0	0	1	0	1	0	0	0	0	1	0	1	0
PIXELCNN_PP	0	0	1	0	0	0	0	1	0	1	0	0	0	0	1	0	1	1
PIXELDA	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0
PIXELSNAIL	0	0	1	0	1	0	0	0	0	0	0	0	1	0	1	0	1	0
PNDM_32	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	1	1	1
PNDM_256	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	1	1	1
PROG_GAN	1	0	0	0	0	0	0	1	0	0	0	1	0	0	1	0	0	1
RGAN	1	0	0	0	0	1	0	0	0	0	1	0	0	0	1	0	0	1
RSGAN_HALF	1	0	0	0	0	1	0	0	0	0	1	0	0	0	1	0	1	0
RSGAN_QUAR	1	0	0	0	0	1	0	0	0	0	1	0	0	0	1	0	1	0
RSGAN_REG	1	0	0	0	0	1	0	0	0	0	1	0	0	0	1	0	1	0
RSGAN_RES_BOT	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	1	0
RSGAN_RES_HALF	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	1	0
RSGAN_RES_QUAR	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	1	0
RSGAN_RES_REG	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	1	0
SAGAN	1	0	0	0	0	1	0	0	0	0	0	1	0	0	1	0	0	0
SCOREDIFF_256	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	1	1
SDEDIT	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	1	1
SEAN	0	0	0	0	0	1	0	0	0	0	1	0	0	0	1	0	1	0
SEMANTIC	0	1	0	0	0	1	0	0	0	0	1	0	0	0	1	0	0	1
SGAN	1	0	0	0	0	1	0	0	0	0	0	1	0	0	1	0	0	1
SNGAN	1	0	0	0	0	1	0	0	0	0	1	0	0	0	1	0	1	0
SOFT_GAN	1	0	0	0	0	1	0	0	0	0	1	0	0	0	1	0	0	0
SRFLOW	0	0	1	0	0	0	1	0	0	1	0	0	0	0	0	1	0	0
SRRNET	1	0	0	0	0	1	0	0	0	0	1	0	0	0	1	0	1	1
STANDARD_VAE	0	0	0	0	0	0	1	0	0	1	0	0	0	0	1	0	1	1
STA_DM_15	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	1	1	0
STA_DM_21	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	1	1	0
STA_DM_XL	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	1	1	0
STARGAN	0	1	0	0	0	1	0	0	0	0	1	0	0	0	1	0	0	1
STARGAN_2	0	1	0	0	0	1	0	0	0	0	1	0	0	0	1	0	0	1
STGAN	1	0	0	0	0	1	0	0	0	0	1	0	0	0	1	0	1	1
STYLEGAN	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	1	0	1
STYLEGAN_2	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	1	0	1
STYLEGAN_ADA	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	1	0	1
SURVAE_M	0	0	1	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0
SURVAE_N	0	0	1	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0
TPGAN	1	0	0	0	0	0	0	1	0	0	0	1	0	0	1	0	1	1
UGAN	1	0	0	0	0	0	0	1	0	0	1	0	0	0	1	0	1	0
UNIT	0	1	0	0	0	1	0	0	0	0	1	0	0	0	0	1	1	1
VAE_FIELD	0	0	1	0	0	0	0	1	0	1	0	0	0	0	1	0	0	0
VAE_FLOW	0	0	1	0	0	0	0	1	0	1	0	0	0	0	1	0	0	0
VAE_GAN	1	0	0	0	0	1	0	0	0	0	1	0	0	0	1	0	0	0
VDVAE	0	0	1	0	1	0	0	0	0	0	0	1	0	0	0	1	1	1
WGAN	1	0	0	0	0	1	0	0	0	0	1	0	0	0	1	0	0	0
WGAN_DRA	1	0	0	0	0	1	0	0	0	0	1	0	0	0	1	0	1	0
WGAN_WC	1	0	0	0	0	1	0	0	0	0	1	0	0	0	1	0	1	0
WGANGP	1	0	0	0	0	1	0	0	0	0	1	0	0	0	1	0	0	0
YLG	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	1	1

24

Table 15: Ground truth feature vector used for prediction of Continuous Architecture Hyperparameters for all GMs. The discrete architecture hyperparameter ground truth is defined in (Tab. 6). F1: # layers, F2: # convolutional layers, F3: # fully connected layers, F4: # pooling layers, F5: # normalization layers, F6: #filters, F7: # blocks, F8:# layers per block, and F9: # parameters.

GM	Layer #	Conv. #	FC #	Pool #	Norm. #	Filter #	Block #	Block Layer #	Para. #
AAE	9	0	7	0	2	0	0	0	1, 593, 378
ACGAN	18	10	1	0	7	2, 307	5	3	4, 276, 739
ADAGAN_C	35	14	13	1	7	4, 131	9	3	9, 416, 196
ADAGAN_P	35	14	13	1	7	4, 131	9	3	9, 416, 196
ADM_G_128	223	90	37	8	88	N/A	34	7	421, 529, 606
ADM_G_256	266	107	45	11	103	N/A	39	7	553, 838, 086
ADV_FACES	45	23	1	1	20	2, 627	4	6	30, 000, 000
ALAE	33	25	8	0	0	4, 094	3	8	50, 200, 000
BEGAN	10	9	1	0	0	515	2	4	7, 278, 472
BETA_B	7	4	3	0	0	99	1	3	469, 173
BETA_H	7	4	3	0	0	99	1	3	469, 173
BETA_TCVAE	7	4	3	0	0	99	1	3	469, 173
BGAN	8	0	5	0	3	0	2	3	1, 757, 412
BICYCLE_GAN	25	14	1	0	10	4, 483	2	10	23, 680, 256
BIGGAN_128	63	21	1	0	41	6, 123	6	10	50, 400, 000
BIGGAN_256	75	25	1	0	49	7, 215	6	12	55, 900, 000
BIGGAN_512	87	29	1	0	57	8, 365	6	14	56, 200, 000
Blended_DM	266	107	45	11	103	N/A	39	7	553, 838, 086
CADGAN	8	4	1	0	3	451	3	2	3, 812, 355
CCGAN	22	12	0	0	10	3, 203	2	9	29, 257, 731
CGAN	8	0	5	0	3	0	2	3	1, 757, 412
COCO_GAN	19	9	1	0	9	2, 883	3	4	50, 000, 000
COGAN	9	5	0	0	4	259	2	2	1, 126, 790
COLOUR_GAN	19	10	0	0	9	2, 435	2	9	19, 422, 404
CONT_ENC	19	11	0	0	8	5, 987	2	8	40, 401, 187
CONTRAGAN	35	14	13	1	7	4, 131	9	3	9, 416, 196
CONTROLNET	427	121	132	0	174	N/A	56	7	39, 726, 979
COUNCIL_GAN	62	30	3	0	29	6, 214	2	10	69, 616, 944
CLIPDM	226	120	34	0	72	N/A	38	3	113, 673, 219
CRAMER_GAN	9	4	1	0	4	454	2	3	9, 681, 284
CRGAN_C	35	14	13	1	7	4, 131	9	3	9, 416, 196
CRGAN_P	35	14	13	1	7	4, 131	9	3	9, 416, 196
CYCLEGAN	47	24	0	0	23	2, 947	4	9	11, 378, 179
DAGAN_C	35	14	13	1	7	4, 131	9	3	9, 416, 196
DAGAN_P	35	14	13	1	7	4, 131	9	3	9, 416, 196
DCGAN	9	4	1	0	4	454	2	3	9, 681, 284
DDiFFGAN_32	289	80	91	0	118	N/A	40	2	48, 432, 515
DDPM_32	164	89	24	0	51	N/A	28	7	35, 746, 307
DDPM_256	225	120	34	0	71	N/A	39	7	113, 673, 219
DEEPFOOL	95	92	1	2	0	7, 236	4	10	22, 000, 000
DFCVAE	45	22	2	0	21	4, 227	4	7	2, 546, 234
DIFF_ISGEN	88	24	56	0	8	N/A	8	6	30, 276, 583
DIFF_PGAN	45	20	0	3	13	N/A	11	4	105, 684, 175
DIFF_SGAN	48	20	28	0	8	N/A	8	6	24, 767, 458
DIFFAE	712	263	171	45	233	N/A	118	7	336, 984, 582
DIFFAE_LATENT	717	264	172	46	235	N/A	155	6	445, 203, 974
DISCOGAN	21	12	0	0	9	3, 459	2	9	29, 241, 731
DRGAN	44	28	1	1	14	4, 481	3	8	18, 885, 068
DRIT	19	10	0	0	9	1, 793	4	3	9, 564, 170
DUALGAN	25	14	1	0	10	4, 483	2	10	23, 680, 256
EBGAN	6	3	1	0	2	195	2	2	738, 433
ESRGAN	66	66	0	0	0	4, 547	5	4	7, 012, 163
FACTOR_VAE	7	4	3	0	0	99	1	3	469, 173
Fast pixel	17	9	0	0	8	768	2	8	4, 600, 000
FFGAN	39	19	1	1	19	3, 261	0	0	50, 000, 000
FGAN	5	0	3	0	2	0	2	2	2, 256, 401
FGAN_KL	5	0	3	0	2	0	2	2	2, 256, 401
FGAN_NEYMAN	5	0	3	0	2	0	2	2	2, 256, 401
FGAN_PEARSON	5	0	3	0	2	0	2	2	2, 256, 401
FGSM	95	92	1	2	0	7, 236	4	10	22, 000, 000
FPGAN	23	12	0	0	11	2, 179	2	6	53, 192, 576
FSGAN	37	20	0	1	16	2, 863	4	8	94, 669, 184
FVBN	28	0	28	0	0	0	1	1	307, 721
GAN_ANIME	25	18	0	0	7	2, 179	4	6	8, 467, 854
Gated_pixel_cnn	32	32	0	0	0	5, 433	3	10	3, 364, 161
GDWCT	79	27	40	1	11	5, 699	2	4	51, 965, 832
GFLM	95	92	1	2	0	7, 236	4	10	22, 000, 000
GGAN	8	4	1	0	3	451	3	2	3, 812, 355
GLIDE	331	93	103	6	129	N/A	74	5	385, 030, 726
ICRGAN_C	35	14	13	1	7	4, 131	9	3	9, 416, 196
ICRGAN_P	35	14	13	1	7	4, 131	9	3	9, 416, 196

25

Table 16: Ground truth feature vector used for prediction of Continuous Architecture Hyperparameters for all GMs. The discrete architecture hyperparameter ground truth is defined in (Tab. 6). F1: # layers, F2: # convolutional layers, F3: # fully connected layers, F4: # pooling layers, F5: # normalization layers, F6: #filters, F7: # blocks, F8:# layers per block, and F9: # parameters.

GM	Layer #	Conv. #	FC #	Pool #	Norm. #	Filter #	Block #	Block Layer #	Para. #
IDDPM_32	193	85	32	0	76	N/A	45	2	52, 546, 438
IDDPM_64	195	87	32	0	76	N/A	45	2	27, 3049, 350
IDDPM_256	201	96	34	0	71	N/A	40	5	113, 676, 678
ILVER_256	266	107	45	11	103	N/A	39	7	553, 838, 086
Image_GPT	59	42	0	0	17	4, 673	7	8	401, 489
INFOGAN	7	3	1	0	3	195	2	2	1, 049, 985
LAPGAN	11	6	5	0	0	262	4	2	2, 182, 857
LDM	255	127	24	0	104	N/A	65	4	329, 378, 945
LDM_CON	503	159	184	8	152	N/A	65	8	456, 755, 873
Lmconv	105	60	10	35	0	7, 156	15	5	46, 000, 000
LOGAN	35	14	13	1	7	4, 131	9	3	9, 416, 196
LSGAN	9	5	0	0	4	1, 923	2	4	23, 909, 265
MADE	2	0	2	0	0	0	1	2	12552784
MAGAN	9	5	0	0	4	963	2	3	11, 140, 934
MEMGAN	14	7	1	0	6	1, 155	3	4	4, 128, 515
MMD_GAN	9	4	1	0	4	454	2	3	9, 681, 284
MRGAN	9	4	1	0	4	451	3	2	15, 038, 350
MSG_STYLE_GAN	33	25	8	0	0	4, 094	3	8	50, 200, 000
MUNIT	18	15	0	0	3	3, 715	2	6	10, 305, 035
NADE	1	0	1	0	0	0	1	1	785, 284
OCFGAN	9	4	1	0	4	454	2	3	9, 681, 284
PGD	95	92	1	2	0	7, 236	4	10	22, 000, 000
PIX2PIX	29	16	0	0	13	5, 507	2	13	54, 404, 099
PixelCNN	17	9	0	0	8	768	2	8	4, 600, 000
PixelCNN++	105	60	10	35	0	7, 156	15	5	46, 000, 000
PIXELDA	27	14	1	0	12	835	4	6	483, 715
PixelSnail	90	90	0	0	0	4, 051	3	10	40, 000, 000
PNDM_32	164	89	24	0	51	N/A	28	7	35, 746, 307
PNDM_256	266	107	45	11	103	N/A	39	7	553, 838, 086
PROG_GAN	26	25	1	0	0	4, 600	3	8	46, 200, 000
RGAN	7	3	1	0	3	195	2	2	1, 049, 985
RSGAN_HALF	8	4	1	0	3	899	3	2	13, 129, 731
RSGAN_QUAR	8	4	1	0	3	451	3	2	3, 812, 355
RSGAN_REG	8	4	1	0	3	1, 795	3	2	48, 279, 555
RSGAN_RES_BOT	15	7	1	0	7	963	3	4	758, 467
RSGAN_RES_HALF	15	7	1	0	7	1, 155	3	4	1, 201, 411
RSGAN_RES_QUAR	15	7	1	0	7	579	3	4	367, 235
RSGAN_RES_REG	15	7	1	0	7	2, 307	3	4	4, 270, 595
SAGAN	11	6	1	0	4	139	2	4	16, 665, 286
SEAN	19	16	0	0	0	5, 062	2	7	266, 907, 367
SEMANTIC	23	12	0	0	11	2, 179	2	6	53, 192, 576
SGAN	7	3	1	0	3	195	2	2	1, 049, 985
SCOREDIFF_256 225	120	34	0	11	71	N/A	39	7	113, 673, 219
SDEdit	226	120	34	0	72	N/A	38	3	113, 673, 219
SNGAN	23	11	1	0	11	3, 871	4	5	10, 000, 000
SOFT_GAN	8	0	5	0	3	0	2	3	1, 757, 412
SRFLOW	66	66	0	0	2	4, 547	5	4	7, 012, 163
SRRNET	74	36	1	0	37	2, 819	4	16	4, 069, 955
STANDARD_VAE	7	4	3	0	0	99	1	3	469, 173
STARGAN	23	12	0	0	11	2, 179	2	6	53, 192, 576
STARGAN_2	67	26	12	4	25	4, 188	4	12	94, 008, 488
STA_DM_15	503	159	184	8	152	N/A	65	8	456, 755, 873
STA_DM_21	503	159	184	8	152	N/A	65	8	456, 755, 873
STA_DM_XL	601	184	201	12	185	N/A	74	9	618, 997, 638
STGAN	19	10	0	0	9	2, 953	2	5	25, 000, 000
STYLEGAN	33	25	8	0	0	4, 094	3	8	50, 200, 000
STYLEGAN_2	33	25	8	0	0	4, 094	3	8	59, 000, 000
STYLEGAN2_ADA	33	25	8	0	0	4, 094	3	8	59, 000, 000
SURVAE_FLOW_MAX	95	90	0	5	0	6, 542	2	20	25, 000, 000
SURVAE_FLOW_NON	90	90	0	0	0	6, 542	2	20	25, 000, 000
TPGAN	45	31	2	1	11	5, 275	0	0	27, 233, 200
UGAN	9	4	1	0	4	771	2	3	4, 850, 692
UNIT	43	22	0	0	21	4, 739	4	8	13, 131, 779
VAE_field	6	0	6	0	0	0	1	3	300, 304
VAE_flow	14	0	14	0	0	0	2	4	760, 448
VAEGAN	17	7	2	0	8	867	2	6	26, 396, 740
VDVAE	48	42	0	6	0	3, 502	3	13	41, 000, 000
WGAN	9	5	0	0	4	1, 923	2	4	23, 909, 265
WGAN_DRA	18	10	1	0	7	2, 307	5	3	4, 276, 739
WGAN_WC	18	10	1	0	7	2, 307	5	3	4, 276, 739
WGANGP	9	5	0	0	4	1, 923	2	4	23, 905, 841
YLG	33	20	1	2	10	5, 155	5	5	42, 078, 852

26