Reverse Engineering of Generative Models: Inferring Model Hyperparameters from Generated Images – Supplementary material –

Vishal Asnani, Xi Yin, Tal Hassner, Xiaoming Liu

1 Test sets for evaluation

The experiments described in the text were performed on four different test sets, each set containing twelve different GMs for the leave out testing. For test sets, we follow the distribution of GMs as follows: six GANs, two VAEs, two ARs, one NF and one AA model. We select this distribution because of the number of GMs of each type in our dataset which has 81 GANs, 13 VAEs, 11 ARs, 5 NFs and 6 AAs. The sets considered are shown in Table 1.

2 Ground truth for GMs

We collected a fake face dataset of 116 GMs, each of them with $1,000$ generated images. We also collect the ground truth hyperparameters for network architecture and loss function types. Table 2 shows the ground truth representation of the network architecture where different hyperparameters are of different data types. Therefore, we apply min-max normalization for the continuous type parameters to make all values in the range of $[0,1]$. For multi-class and binary labels, we further show the feature value for different labels in Table 3. Note that some parameters share the same values but with different meanings. For example, F14 and F15 represent skip connection and down-sampling respectively. Table 4 shows the ground truth representation of the loss function types used to train each GM where all these values are binary indicating whether the particular loss type was used or not.

3 Network architecture

Figure 5 shows the network architecture used in different experiments. For GM parsing, our FEN has two stem convolution layers and 15 convolution blocks with each block having convolution, batch normalization and ReLU activation to estimate the fingerprint. The encoder in the PN has five convolution blocks with each block having convolution, pooling and ReLU activation. This is followed by two fully connected layers to output a 512 dimension feature vector which is further given as input to multiple branches to output different predictions. For continuous type parameters, we use two fully connected layers to output a 9-D network architecture. For discrete type parameters and loss function parameters, we use separate classifiers with three fully connected layers for every parameter to perform multi-class or binary classification.

For the deepfake detection task, we change the architecture of our FEN network as current deepfake manipulation detection requires much deeper networks. Thus, our FEN architecture has two stem convolution layers and 29 convolution blocks to estimate the fingerprint. For further classification, we use a shallow network of five convolution blocks followed by two fully connected layers.

For the image attribution task, we use the same FEN as used in model parsing, and a shallow network of two convolution blocks and two fully connected layers to perform multi-class classification.

4 Feature heatmaps

Every hyperparameter defined for network architecture and loss function type prediction may depend on certain region of the input image. To find out which region of the input image our model is looking at to predict each hyperparameter, we mask out 5×5 region from the input image. For the continuous type parameters, we compute the L_1 error between every predicted hyperparameter and its ground truth. This value of error will tell us how important is this 5 region in the input image to predict a particular hyperparameter. The higher the value of this error, the higher is the importance of that region in the prediction of the corresponding hyperparameter. For discete type parameters in network architecture and loss function, we estimate the probability of the ground truth label for every parameter. We subtract this probability from one to estimate the heatmap of the respective feature. Important regions will not affect the probability of the ground truth label for a particular feature. To obtain a stable heatmap, we do the above experiment on 100 randomly chosen images across the different GMs and then calculate the average heatmap.

Figure 1, 2 and 3 show the feature heatmaps for every hyperparameter of network architecture and loss type feature vector for Face, MNIST and CIFAR data respectively. For each hyperparmater, there are certain regions of the input that are more important than others. Each type of data has different type of heatmaps indicating different regions of importance. For face and CIFAR, these regions lie mostly in the central part but for MNIST, many of the features depend on the regions closer to edges. There are also some similarities between these heatmaps

TABLE 1: Test sets used for evaluation. Each set contains six GANs, two VAEs, two ARs, one AA and one NF.

GM	Set 1	Set 2	Set 3	Set 4
GM 1	ADV_FACES	AAE	BICYCLE_GAN	GFLM
GM 2	BETA_B	ADAGAN_C	BIGGAN_512	IMAGE_GPT
GM 3	BETA_TCVAE	BEGAN	CRGAN_C	LSGAN
GM 4	BIGGAN_128	BETA_H	FACTOR_VAE	MADE
GM 5	DAGAN_C	BIGGAN_256	FGSM	PIX2PIX
GM 6	DRGAN	COCOGAN	ICRGAN_C	PROG_GAN
GM 7	FGAN	CRAMERGAN	LOGAN	RSGAN_REG
GM 8	PIXEL_CNN	DEEPFOOL	MUNIT	SEAN
GM 9	PIXEL_CNN++	DRIT	PIXEL_SNAIL	STYLE_GAN
GM 10	RSGAN_HALF	FAST_PIXEL	STARGAN_2	SURVAE_FLOW_NONPOOL
GM 11	STARGAN	FVBN	SURVAE_FLOW_MAXPOOL	WGAN_DRA
GM 12	VAEGAN	SRFLOW	VAE_FIELD	YLG

for a particular type of data. This can indicate the similarity of these hyperparameters.

References

[1] A. Makhzani, J. Shlens, N. Jaitly, I. Goodfellow, and B. Frey, "Adversarial autoencoders," in *ICLR*, 2016.

[2] A. Odena, C. Olah, and J. Shlens, "Conditional image synthesis with auxiliary classifier GANs," in *ICLR*, 2017.

[3] R. D. Hjelm, A. P. Jacob, A. Trischler, G. Che, K. Cho, and Y. Bengio, "Boundary seeking GANs," in *ICLR*, 2018.

[4] J.-Y. Zhu, R. Zhang, D. Pathak, T. Darrell, A. A. Efros, O. Wang, and E. Shechtman, "Toward multimodal image-to-image translation," in *NeurIPS*, 2017.

[5] A. Brock, J. Donahue, and K. Simonyan, "Large scale GAN training for high fidelity natural image synthesis," in *ICLR*, 2019.

[6] W. Jitkrittum, P. Sangkloy, M. W. Gondal, A. Raj, J. Hays, and B. Schölkopf, "Kernel mean matching for content addressability of GANs," in *ICML*, 2019.

[7] E. Denton, S. Gross, and R. Fergus, "Semi-supervised learning with context-conditional generative adversarial networks," *arXiv preprint arXiv:1611.06430*, 2016.

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

[9] M.-Y. Liu and O. Tuzel, "Coupled generative adversarial networks," in *NeurIPS*, 2016.

[10] K. Nazeri, E. Ng, and M. Ebrahimi, "Image colorization using generative adversarial networks," in *AMDO*, 2018.

[11] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros, "Context encoders: Feature learning by inpainting," in *CVPR*, 2016.

[12] H. Zhang, Z. Zhang, A. Odena, and H. Lee, "Consistency regularization for generative adversarial networks," in *ICLR*, 2020.

[13] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *ICCV*, 2017.

[14] M. Kang and J. Park, "ContraGAN: Contrastive learning for conditional image generation," in *NeurIPS*, 2020.

[15] T. Kim, M. Cha, H. Kim, J. K. Lee, and J. Kim, "Learning to discover cross-domain relations with generative adversarial networks," in *ICML*, 2017.

[16] H. Y. Lee, H. Y. Tseng, Q. Mao, J. B. Huang, Y. D. Lu, M. Singh, and M. H. Yang, "DRIT++: Diverse image-to-image translation via disentangled representations," *International Journal of Computer Vision*, vol. 128, no. 10-11, pp. 2402–2417, 2020.

[17] Z. Yi, H. Zhang, P. Tan, and M. Gong, "DualGAN: Unsupervised dual learning for image-to-image translation," in *ICCV*, 2017.

[18] J. Zhao, M. Mathieu, and Y. LeCun, "Energy-based generative adversarial networks," in *ICLR*, 2017.

[19] X. Wang, K. Yu, S. Wu, J. Gu, Y. Liu, C. Dong, Y. Qiao, and C. Change Loy, "ESRGAN: Enhanced super-resolution generative adversarial networks," in *ECCV*, 2018.

[20] A. Pumarola, A. Agudo, A. M. Martinez, A. Sanfeliu, and F. Moreno-Noguer, "GANimation: Anatomically-aware facial animation from a single image," in *ECCV*, 2018.

[21] J. H. Lim and J. C. Ye, "Geometric GAN," *arXiv preprint arXiv:1705.02894*, 2017.

[22] R. Sun, T. Fang, and A. Schwing, "Towards a better global loss landscape of GANs," *NeurIPS*, 2020.

[23] Z. Zhao, S. Singh, H. Lee, Z. Zhang, A. Odena, and H. Zhang, "Improved consistency regularization for GANs," *arXiv preprint arXiv:2002.04724*, 2020.

[24] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel, "InfoGAN: Interpretable representation learning by information maximizing generative adversarial nets," in *NeurIPS*, 2016.

[25] Y. Wu, J. Donahue, D. Balduzzi, K. Simonyan, and T. Lillicrap, "LOGAN: Latent optimisation for generative adversarial networks," *arXiv preprint arXiv:1912.00953*, 2019.

[26] Y. Kim, M. Kim, and G. Kim, "Memorization precedes generation: Learning unsupervised GANs with memory networks," in *ICLR*, 2018.

[27] X. Huang, M.-Y. Liu, S. Belongie, and J. Kautz, "Multimodal unsupervised image-to-image translation," in *ECCV*, 2018.

[28] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in *CVPR*, 2017.

[29] K. Bousmalis, N. Silberman, D. Dohan, D. Erhan, and D. Krishnan, "Unsupervised pixel-level domain adaptation with generative adversarial networks," in *CVPR*, 2017.

[30] A. Jolicoeur-Martineau, "The relativistic discriminator: a key element missing from standard GAN," in *ICLR*, 2019.

[31] A. Odena, "Semi-supervised learning with generative adversarial networks," in *ICMLW*, 2016.

[32] M. Lin, "Softmax gan," *arXiv preprint arXiv:1704.06191*, 2017.

[33] Y. Jin, J. Zhang, M. Li, Y. Tian, H. Zhu, and Z. Fang, "Towards the automatic anime characters creation with generative adversarial networks," *arXiv preprint arXiv:1708.05509*, 2017.

[34] M.-Y. Liu, T. Breuel, and J. Kautz, "Unsupervised image-to-image translation networks," in *NeurIPS*, 2017.

[35] M. Arjovsky and L. Bottou, "Towards principled methods for training generative adversarial networks," in *ICLR*, 2017.

[36] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville, "Improved training of wasserstein GANs," in *NeurIPS*, 2017.

[37] N. Kodali, J. Abernethy, J. Hays, and Z. Kira, "On convergence and stability of GANs," *arXiv preprint arXiv:1705.07215*, 2017.

[38] G. Daras, A. Odena, H. Zhang, and A. G. Dimakis, "Your local GAN: Designing two dimensional local attention mechanisms for generative models," in *CVPR*, 2020.

[39] T. Karras, M. Aittala, J. Hellsten, S. Laine, J. Lehtinen, and T. Aila, "Training generative adversarial networks with limited data," in *NeurIPS*, 2020.

[40] S. Nowozin, B. Cseke, and R. Tomioka, "f-GAN: training generative neural samplers using variational divergence minimization," in *NeurIPS*, 2016.

[41] E. Denton, S. Chintala, A. Szlam, and R. Fergus, "Deep generative image models using a laplacian pyramid of adversarial networks," in *NeurIPS*, 2015.

[42] L. Metz, B. Poole, D. Pfau, and J. Sohl-Dickstein, "Unrolled generative adversarial networks," in *ICLR*, 2017.

[43] T. Karras, M. Aittala, J. Hellsten, S. Laine, J. Lehtinen, and T. Aila, "Training generative adversarial networks with limited data," in *NeurIPS*, 2020.

[44] X. Mao, Q. Li, H. Xie, R. Y. Lau, Z. Wang, and S. Paul Smolley, "Least squares generative adversarial networks," in *ICCV*, 2017.

[45] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *ICML*, 2017.

[46] O. Nizan and A. Tal, "Breaking the cycle - colleagues are all you need," in *CVPR*.

[47] T. Xiao, J. Hong, and J. Ma, "DNA-GAN: Learning disentangled representations from multi-attribute images," *ICLRW*, 2018.

[48] M. M. Rahman Siddiquee, Z. Zhou, N. Tajbakhsh, R. Feng, M. B. Gotway, Y. Bengio, and J. Liang, "Learning fixed points in generative adversarial networks: From image-to-image translation to disease detection and localization," in *ICCV*, 2019.

TABLE 2: Ground truth feature vector used for prediction of network architecture for all GMs. F1: # layers, F2: # convolutional layers, F3: # fully connected layers, F4: # pooling layers, F5: # normalization layers, F6: #filters, F7: # blocks, F8:# layers per block, F9: # parameters, F10: normalization type, F11: non-linearity type in last layer, F12: nonlinearity type in blocks, F13: up-sampling type, F14: skip connection, F15: downsampling

GM	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14	F15
AAE	9	0	7	0	2	0	0	0	1593378	0	1	0	0	1	0
ACGAN	18	10	1	0	7	2307	5	3	4276739	0	1	1	0	1	0
ADAGAN_C	35	14	13	1	7	4131	9	3	9416196	0	1	1	0	1	0
ADAGAN_P	35	14	13	1	7	4131	9	3	9416196	0	1	1	0	1	0
ADV_FACES	45	23	1	1	20	2627	4	6	30000000	1	1	1	0	1	0
ALAE	33	25	8	0	0	4094	3	8	50200000	1	2	2	1	0	1
BEGAN	10	9	1	0	0	515	2	4	7278472	0	1	0	0	0	0
BETA_B	7	4	3	0	0	99	1	3	469173	3	3	1	0	1	1
BETA_H	7	4	3	0	0	99	1	3	469173	3	3	1	0	1	1
BETA_TCVAE	7	4	3	0	0	99	1	3	469173	3	3	1	0	1	1
BGAN	8	0	5	0	3	0	2	3	1757412	0	1	2	0	0	0
BICYCLE_GAN	25	14	1	0	10	4483	2	10	23680256	0	1	1	0	0	0
BIGGAN_128	63	21	1	0	41	6123	6	10	50400000	0	1	1	1	1	1
BIGGAN_256	75	25	1	0	49	7215	6	12	55900000	0	1	1	1	1	1
BIGGAN_512	87	29	1	0	57	8365	6	14	56200000	0	1	1	1	1	1
CADGAN	8	4	1	0	3	451	3	2	3812355	0	1	1	0	1	1
CCGAN	22	12	0	0	10	3203	2	9	29257731	0	1	1	1	1	1
CGAN	8	0	5	0	3	0	2	3	1757412	0	1	2	0	0	0
COCO_GAN	19	9	1	0	9	2883	3	4	50000000	0	1	1	0	0	0
COGAN	9	5	0	0	4	259	2	2	1126790	0	1	2	0	1	1
COLOUR_GAN	19	10	0	0	9	2435	2	9	19422404	0	1	1	0	1	1
CONT_ENC	19	11	0	0	8	5987	2	8	40401187	0	1	2	0	1	1
CONTRAGAN	35	14	13	1	7	4131	9	3	9416196	0	1	1	0	1	0
COUNCIL_GAN	62	30	3	0	29	6214	2	10	69616944	1	1	1	0	1	0
CRAMER_GAN	9	4	1	0	4	454	2	3	9681284	0	1	1	0	1	0
CRGAN_C	35	14	13	1	7	4131	9	3	9416196	0	1	1	0	1	0
CRGAN_P	35	14	13	1	7	4131	9	3	9416196	0	1	1	0	1	0
CYCLEGAN	47	24	0	0	23	2947	4	9	11378179	1	1	1	1	1	1
DAGAN_C	35	14	13	1	7	4131	9	3	9416196	0	1	1	0	1	0
DAGAN_P	35	14	13	1	7	4131	9	3	9416196	0	1	1	0	1	0
DCGAN	9	4	1	0	4	454	2	3	9681284	0	1	1	0	1	0
DEEPFOOL	95	92	1	2	0	7236	4	10	22000000	2	0	1	1	0	0
DFCVAE	45	22	2	0	21	4227	4	7	2546234	0	3	2	0	0	1
DISCOGAN	21	12	0	0	9	3459	2	9	29241731	1	1	2	1	1	1
DRGAN	44	28	1	1	14	4481	3	8	18885068	0	1	0	0	1	1
DRIT	19	10	0	0	9	1793	4	3	9564170	1	1	1	1	1	1
DUALGAN	25	14	1	0	10	4483	2	10	23680256	0	1	1	0	0	0
EBGAN	6	3	1	0	2	195	2	2	738433	0	1	2	0	0	1
ESRGAN	66	66	0	0	0	4547	5	4	7012163	2	2	2	1	0	0
FACTOR_VAE	7	4	3	0	0	99	1	3	469173	3	3	1	0	1	1
Fast pixel	17	9	0	0	8	768	2	8	4600000	0	3	0	0	1	0
FFGAN	39	19	1	1	19	3261	0	0	50000000	0	1	1	1	1	1
FGAN	5	0	3	0	2	0	2	2	2256401	0	3	1	0	1	0
FGAN_KL	5	0	3	0	2	0	2	2	2256401	0	3	1	0	1	0
FGAN_NEYMAN	5	0	3	0	2	0	2	2	2256401	0	3	1	0	1	0
FGAN_PEARSON	5	0	3	0	2	0	2	2	2256401	0	3	1	0	1	0
FGSM	95	92	1	2	0	7236	4	10	22000000	2	0	1	1	0	0
FPGAN	23	12	0	0	11	2179	2	6	53192576	1	1	1	0	0	1
FSGAN	37	20	0	1	16	2863	4	8	94669184	0	0	1	1	1	1
FVBN	28	0	28	0	0	0	1	1	307721	2	3	0	0	1	0
GAN_ANIME	25	18	0	0	7	2179	4	6	8467854	1	1	1	0	1	1
Gated_pixel_cnn	32	32	0	0	0	5433	3	10	3364161	2	3	2	1	1	0
GDWCT	79	27	40	1	11	5699	2	4	51965832	1	1	1	0	0	1
GFLM	95	92	1	2	0	7236	4	10	22000000	2	0	1	1	0	0
GGAN	8	4	1	0	3	451	3	2	3812355	0	1	1	0	1	1
ICRGAN_C	35	14	13	1	7	4131	9	3	9416196	0	1	1	0	1	0
ICRGAN_P	35	14	13	1	7	4131	9	3	9416196	0	1	1	0	1	0
Image_GPT	59	42	0	0	17	4673	7	8	401489	0	3	2	1	1	1
INFOGAN	7	3	1	0	3	195	2	2	1049985	0	1	2	0	0	1
LAPGAN	11	6	5	0	0	262	4	2	2182857	2	1	1	1	1	0
Lmconv	105	60	10	35	0	7156	15	5	46000000	2	3	0	1	1	1
LOGAN	35	14	13	1	7	4131	9	3	9416196	0	1	1	0	1	0
LSGAN	9	5	0	0	4	1923	2	4	23909265	0	1	1	0	0	0
MADE	2	0	2	0	0	0	1	2	12552784	2	3	0	0	1	0
MAGAN	9	5	0	0	4	963	2	3	11140934	0	1	1	0	1	0
MEMGAN	14	7	1	0	6	1155	3	4	4128515	0	1	1	0	1	0
MMD_GAN	9	4	1	0	4	454	2	3	9681284	0	1	1	0	1	0
MRGAN	9	4	1	0	4	451	3	2	15038350	0	1	1	0	1	0
MSG_STYLE_GAN	33	25	8	0	0	4094	3	8	50200000	1	2	2	1	0	1
MUNIT	18	15	0	0	3	3715	2	6	10305035	1	0	1	1	1	1
NADE	1	0	1	0	0	0	1	1	785284	2	3	0	0	1	0
OCFGAN	9	4	1	0	4	454	2	3	9681284	0	1	1	0	1	0
PGD	95	92	1	2	0	7236	4	10	22000000	2	0	1	1	0	0
PIX2PIX	29	16	0	0	13	5507	2	13	54404099	1	1	2	1	1	1
PixelCNN	17	9	0	0	8	768	2	8	4600000	0	3	0	0	1	0
PixelCNN++	105	60	10	35	0	7156	15	5	46000000	2	3	0	1	1	1
PIXELDA	27	14	1	0	12	835	4	6	483715	0	1	1	1	0	0
PixelSnail	90	90	0	0	0	4051	3	10	40000000	2	0	3	0	1	0
PROG_GAN	26	25	1	0	0	4600	3	8	46200000	0	3	3	0	0	1
RGAN	7	3	1	0	3	195	2	2	1049985	0	1	2	0	0	1
RSGAN_HALF	8	4	1	0	3	899	3	2	13129731	0	1	1	0	1	0
RSGAN_QUAR	8	4	1	0	3	451	3	2	3812355	0	1	1	0	1	0
RSGAN_REG	8	4	1	0	3	1795	3	2	48279555	0	1	1	0	1	0
RSGAN_RES_BOT	15	7	1	0	7	963	3	4	758467	0	1	1	1	1	0
RSGAN_RES_HALF	15	7	1	0	7	1155	3	4	1201411	0	1	1	1	1	0
RSGAN_RES_QUAR	15	7	1	0	7	579	3	4	367235	0	1	1	1	1	0
RSGAN_RES_REG	15	7	1	0	7	2307	3	4	4270595	0	1	1	1	1	0
SAGAN	11	6	1	0	4	139	2	4	16665286	0	1	2	0	0	0
SEAN	19	16	0	0	0	5062	2	7	266907367	3	1	1	0	1	0
SEMANTIC	23	12	0	0	11	2179	2	6	53192576	1	1	1	0	0	1
SGAN	7	3	1	0	3	195	2	2	1049985	0	1	2	0	0	1
SNGAN	23	11	1	0	11	3871	4	5	10000000	0	1	1	0	1	0
SOFT_GAN	8	0	5	0	3	0	2	3	1757412	0	1	2	0	0	0
SRFLOW	66	66	0	0	2	4547	5	4	7012163	2	2	0	1	0	0
SRRNET	74	36	1	0	37	2819	4	16	4069955	0	1	1	0	1	1
STANDARD_VAE	7	4	3	0	0	99	1	3	469173	3	3	1	0	1	1
STARGAN	23	12	0	0	11	2179	2	6	53192576	1	1	1	0	0	1
STARGAN_2	67	26	12	4	25	4188	4	12	94008488	1	2	2	0	0	1
STGAN	19	10	0	0	9	2953	2	5	25000000	0	1	2	1	1	1
STYLEGAN	33	25	8	0	0	4094	3	8	50200000	1	2	2	1	0	1
STYLEGAN_2	33	25	8	0	0	4094	3	8	59000000	1	2	2	1	0	1
STYLEGAN2_ADA	33	25	8	0	0	4094	3	8	59000000	1	2	2	1	0	1
SURVAE_FLOW_MAXPOOL	95	90	0	5	0	6542	2	20	25000000	2	0	0	0	0	0
SURVAE_FLOW_NONPOOL	90	90	0	0	0	6542	2	20	25000000	2	0	0	0	0	0
TPGAN	45	31	2	1	11	5275	0	0	27233200	0	3	3	0	1	1
UGAN	9	4	1	0	4	771	2	3	4850692	0	3	1	0	1	0
UNIT	43	22	0	0	21	4739	4	8	13131779	1	1	1	1	1	1
VAE_field	6	0	6	0	0	0	1	3	300304	2	3	0	0	0	0
VAE_flow	14	0	14	0	0	0	2	4	760448	2	3	0	0	0	0
VAEGAN	17	7	2	0	8	867	2	6	26396740	0	1	1	0	0	1
VDVAE	48	42	0	6	0	3502	3	13	41000000	2	0	2	1	1	1
WGAN	9	5	0	0	4	1923	2	4	23909265	0	1	1	0	0	0
WGAN_DRA	18	10	1	0	7	2307	5	3	4276739	0	1	1	0	1	0
WGAN_WC	18	10	1	0	7	2307	5	3	4276739	0	1	1	0	1	0
WGANGP	9	5	0	0	4	1923	2	4	23905841	0	1	1	0	0	0
YLG	33	20	1	2	10	5155	5	5	42078852	0	1	1	1	1	1

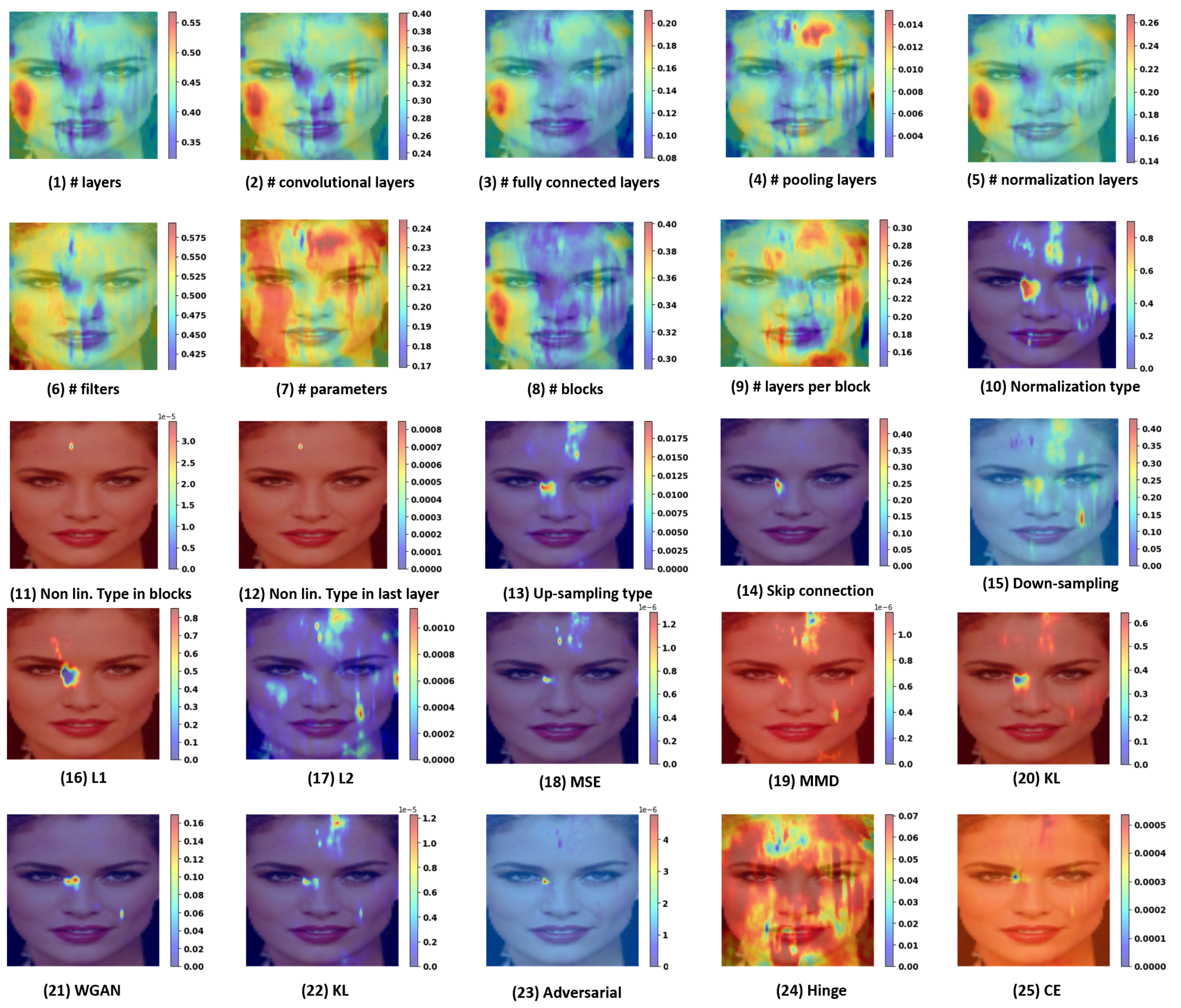

Fig. 1: Feature heatmap for each feature in network architecture and loss function predicted feature vector for face data. Each heatmap provides the importance of the region in the estimation of the respective parameter.

TABLE 3: Feature value for different labels of multi-class and binary features.

Feature	Label	Value
Normalization type	0	Batch Normalization
	1	Instance Normalization
	2	Adaptive Instance Normalization
	3	No Normalization
Non-linearity type in last layer	0	ReLU
	1	Tanh
	2	Leaky_ReLU
	3	Sigmoid
Non-linearity type in blocks	0	ELU
	1	ReLU
	2	Leaky_ReLU
	3	Sigmoid
Upsampling type	0	Nearest Neighbour
	1	Deconvolution
Skip connection and downsampling	0	Feature used
	1	Feature not used

[49] W. Cho, S. Choi, D. K. Park, I. Shin, and J. Choo, "Image-to-image translation via group-wise deep whitening-and-coloring transformation," in *CVPR*, 2019.
[50] A. Karnewar and O. Wang, "MSG_GAN: Multi-scale gradients for generative adversarial networks," in *CVPR*, 2020.
[51] S. Pidhorskyi, D. A. Adjeroh, and G. Doretto, "Adversarial latent autoencoders," in *CVPR*, 2020.
[52] N. Papernot, F. Faghri, N. Carlini, I. Goodfellow, R. Feinman, A. Kurakin, C. Xie, Y. Sharma, T. Brown, A. Roy, A. Matyasko, V. Behzadan, K. Hambardzumyan, Z. Zhang, Y.-L. Juang, Z. Li, R. Sheatsley, A. Garg, J. Uesato, W. Gierke, Y. Dong, D. Berthelot, P. Hendricks, J. Rauber, and R. Long, "Technical report on the cleverhans v2.1.0 adversarial examples library," *arXiv preprint arXiv:1610.00768*, 2018.
[53] D. Deb, J. Zhang, and A. K. Jain, "Advfaces: Adversarial face synthesis," in *IJCB*, 2019.
[54] A. Dabouei, S. Soleymani, J. Dawson, and N. Nasrabadi, "Fast geometrically-perturbed adversarial faces," in *WACV*, 2019.
[55] H. Qiu, C. Xiao, L. Yang, X. Yan, H. Lee, and B. Li, "Semanticadv: Generating adversarial examples via attribute-conditioned image editing," in *ECCV*, 2020.
[56] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," in *ICLR*, 2018.
[57] X. Yuan, P. He, Q. Zhu, and X. Li, "Adversarial examples: Attacks and defenses for deep learning," *IEEE transactions on neural networks and learning systems*, vol. 30, no. 9, pp. 2805–2824, 2019.

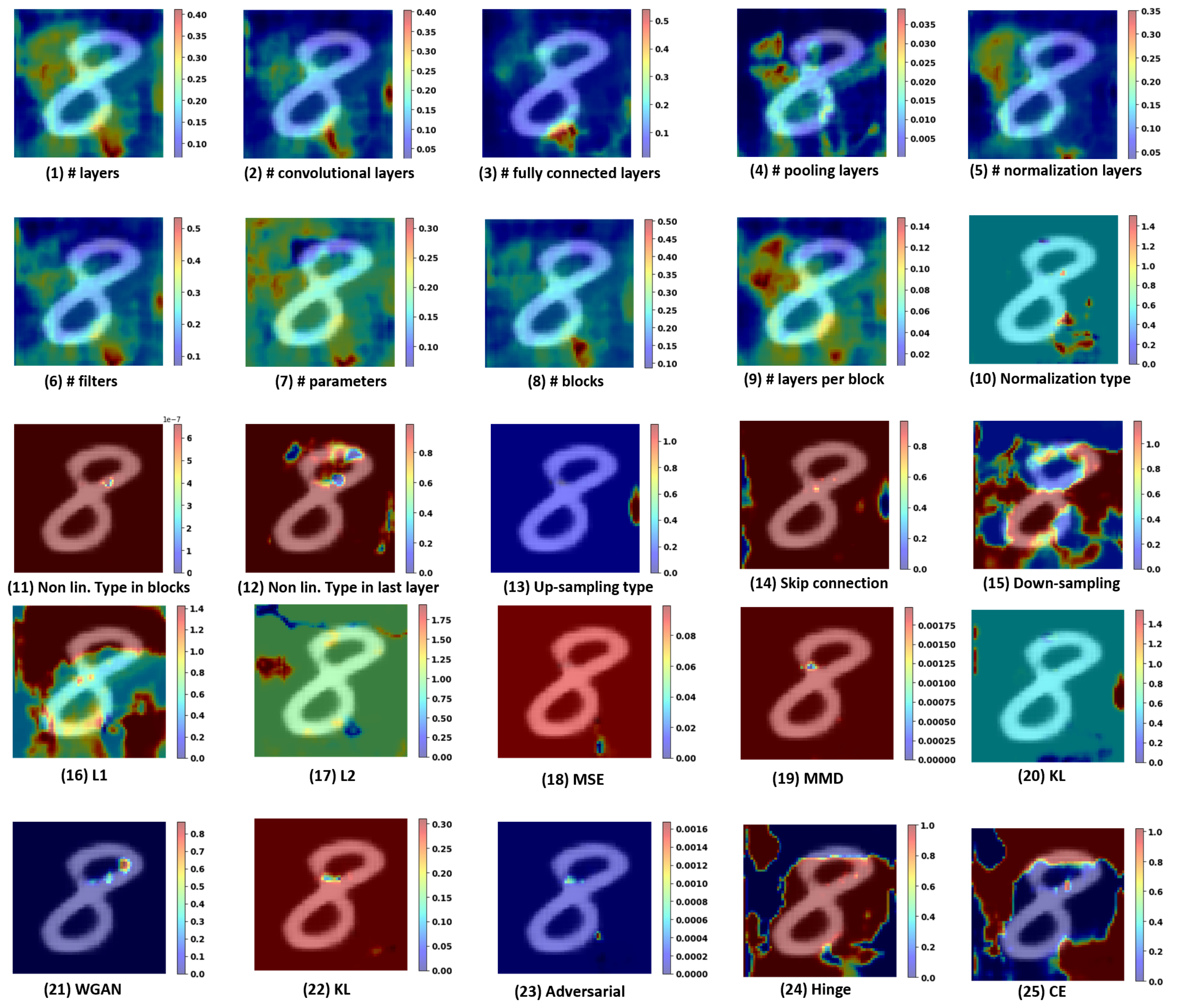

Fig. 2: Feature heatmap for each feature in network architecture and loss function predicted feature vector for MNIST data.

[58] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, “DeepFool: a simple and accurate method to fool deep neural networks,” in *CVPR*, 2016.

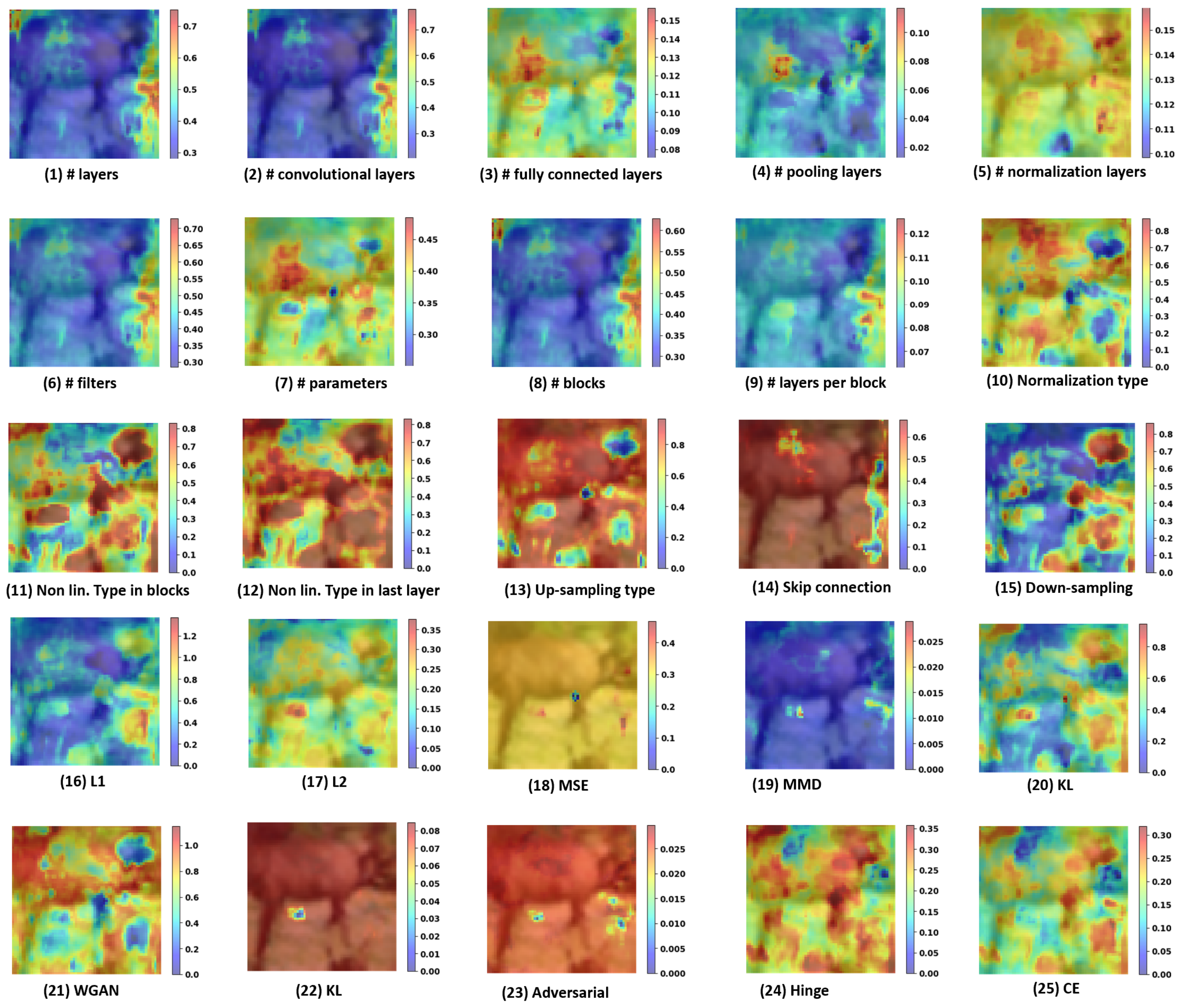

Fig. 3: Feature heatmap for each feature in network architecture and loss function predicted feature vector for CIFAR data.

TABLE 4: Ground truth feature vector used for prediction of loss type for all GMs.

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
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
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
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
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
DEEPFOOL	1	1	0	0	0	0	0	0	0	0
DFCVAE	0	1	0	0	0	0	1	0	0	1
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
ICRGAN_C	1	1	0	0	0	0	0	0	0	1
ICRGAN_P	1	1	0	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
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
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
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
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

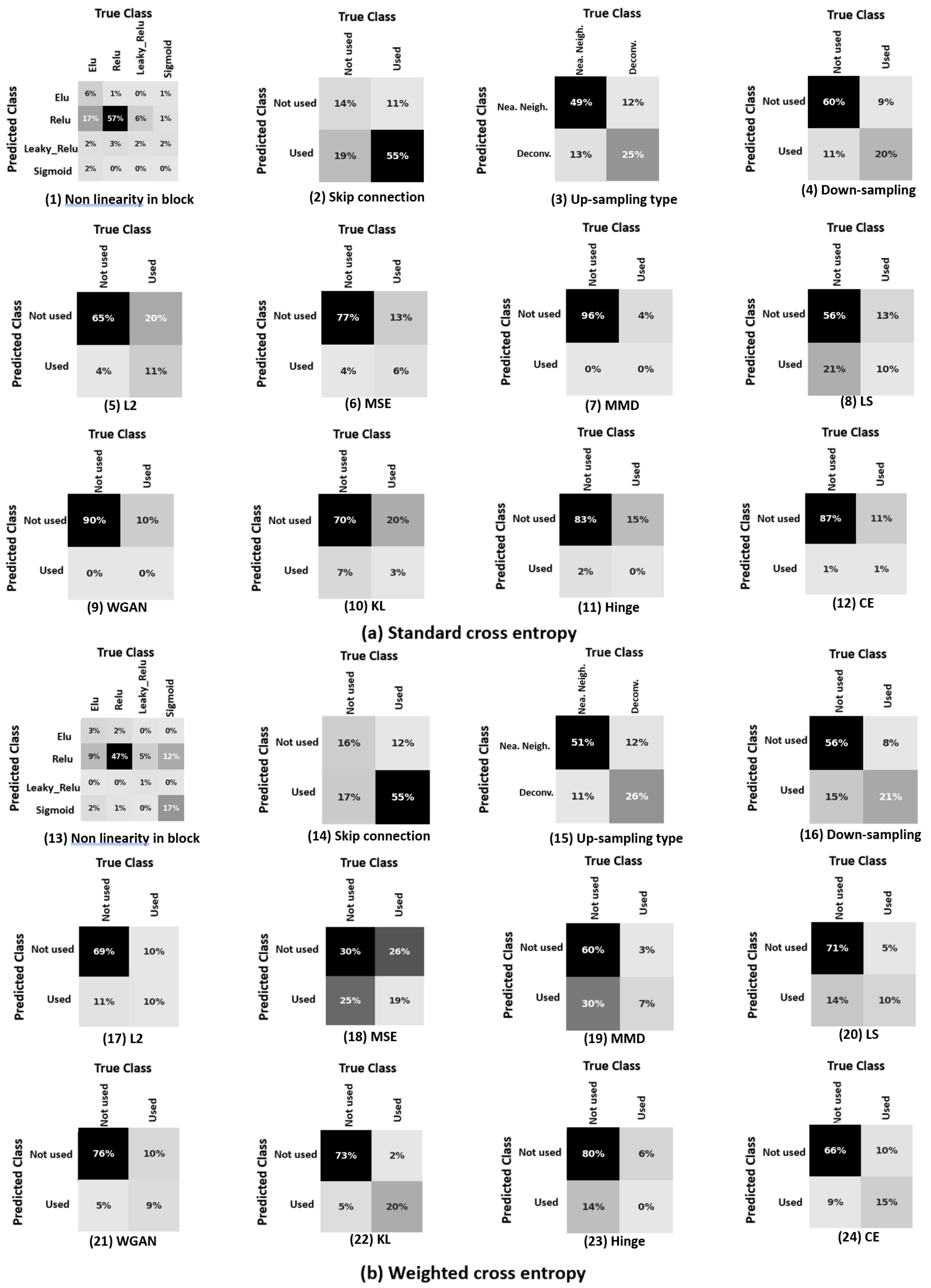

Fig. 4: Confusion matrix in the estimation of remaining parameters which were not shown in paper for network architecture and loss function. (1)-(12): Standard cross-entropy and (12)-(24): Weighted cross entropy. Weighted cross entropy handles imbalance of data much better than the standard cross entropy which usually predicts one class.

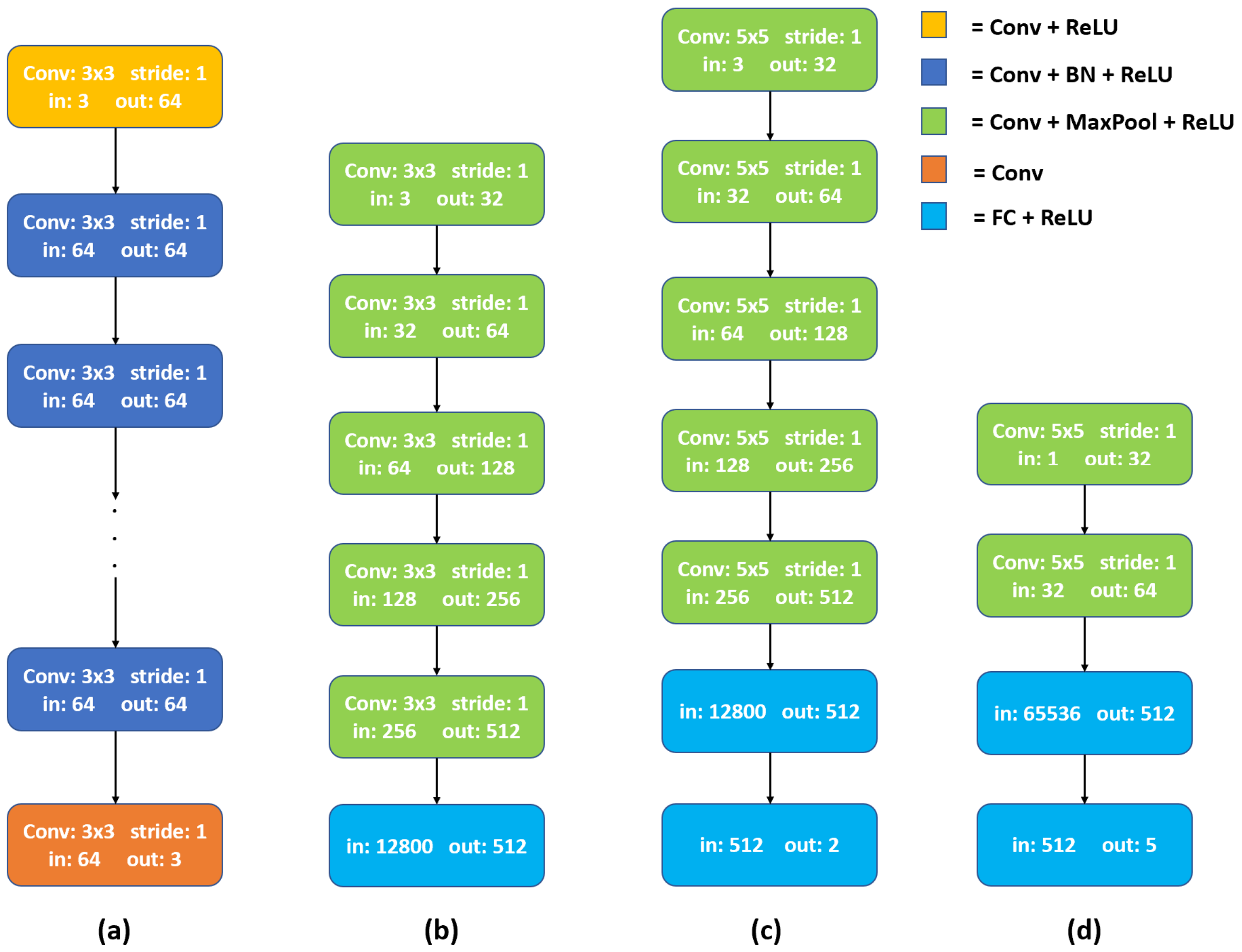
Conv: 3x3 stride: 1 in: 3 out: 64
Conv: 3x3 stride: 1 in: 64 out: 64
Conv: 3x3 stride: 1 in: 64 out: 64
Conv: 3x3 stride: 1 in: 64 out: 64
Conv: 3x3 stride: 1 in: 64 out: 3
Conv: 3x3 stride: 1 in: 3 out: 32
Conv: 3x3 stride: 1 in: 32 out: 64
Conv: 3x3 stride: 1 in: 64 out: 128
Conv: 3x3 stride: 1 in: 128 out: 256
Conv: 3x3 stride: 1 in: 256 out: 512
in: 12800 out: 512
Conv: 5x5 stride: 1 in: 3 out: 32
Conv: 5x5 stride: 1 in: 32 out: 64
Conv: 5x5 stride: 1 in: 64 out: 128
Conv: 5x5 stride: 1 in: 128 out: 256
Conv: 5x5 stride: 1 in: 256 out: 512
in: 12800 out: 512
in: 512 out: 2
Conv: 5x5 stride: 1 in: 1 out: 32
Conv: 5x5 stride: 1 in: 32 out: 64
in: 65536 out: 512
in: 512 out: 5
= Conv + ReLU
= Conv + BN + ReLU
= Conv + MaxPool + ReLU
= Conv
= FC + ReLU
(a)
(b)
(c)
(d)

Fig. 5: Network architecture for various components of our method. (a) FEN (b) Mean and instance parser in PN (c) Shallow network for deepfake detection (d) Shallow network for image attribution.