Supplementary material for FIANCEE: Faster Inference of Adversarial Networks via Conditional Early Exits

Karpikova Polina*[1,2], Radionova Ekaterina*[1], Yaschenko Anastasia*[1,2], Spiridonov Andrei*[1]
Kostyushko Leonid[3], Fabbricatore Riccardo[1], Ivakhnenko Aleksei[1†]
[1]Samsung AI Center - Moscow, Russia [2]Higher School of Economics - Moscow, Russia
[3]Lomonosov Moscow State University - Russia

Contents

S1Qualitative results S1
S1.1The OASIS pipeline S1
S1.1.1 Architectures and dimensions . . . S1
S1.1.2 Training details S2
S1.2The MegaPortraits pipeline S3
S1.2.1 Architectures and dimensions . . . S3
S1.2.2 Training details S3

S2Comparisons S5

S3Complexity analysis S6

S1. Qualitative results

S1.1. The OASIS pipeline

S1.1.1 Architectures and dimensions

The original OASIS [13] generative DNN consists of an initial 2D convolutional layer, followed by 6 SPADE-ResBlock modules [11] and a final Conv2D, LeakyRelu, and TanH. Its total number of parameters is 74M. We appended 4 branches to it, after ResBlock 1, 2, 3, and 4 respectively. Each branch consisted of the same number of ResBlock modules as the remaining part of the backbone. In order to create lighter computational paths, we decreased the number of channels of the branches' modules. To do it in a coherent manner, we decided to scale down all channels uniformly by multiplying them by a *scale factor* (SF). Since such scaling with arbitrary coefficients may produce channel numbers too small to be of use, we restrained its effect by imposing a minimum number of channels, under which no scaling was forced. In other words, if the minimum number is 64, and we enforce factor of 1/3 starting from 128 , the new channel number will be 64, instead of 43. We explored a plethora of different scale factors and minimum channels, which we report in Table S8.

The database we employed was created using 500 semantic maps randomly chosen from the training dataset,

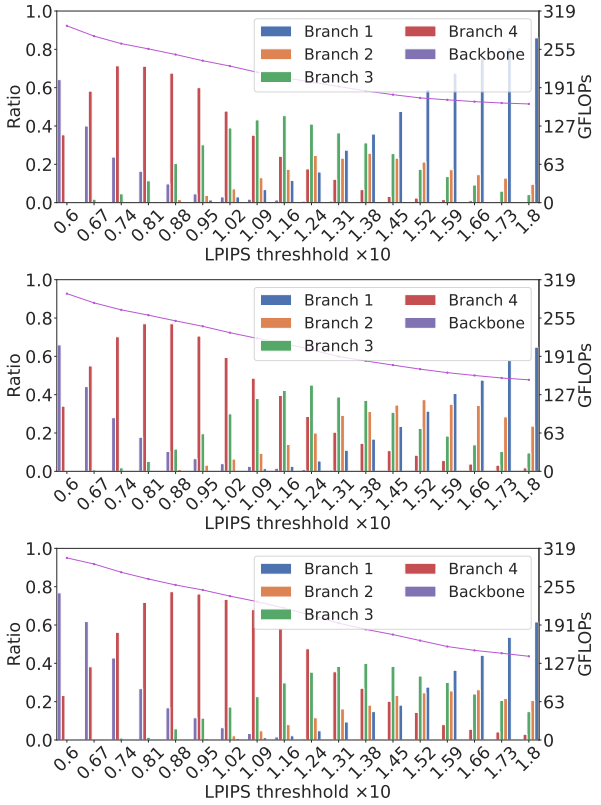

Figure S1. OASIS pipeline, comparison between the efficacy of different scale factors. The minimum number of channels is 64. From top to bottom: SF = 1/2, 1/3, 1/4.

each concatenated with 100 different 3D noise tensors to produce a variety of inputs, that were processed and divided into 128 non-overlapping patches, yielding a total of $500 \times 100 \times 128 = 6.4$M key-value pairs. Since redundancy in the key space is rather probable, we extracted from this multitude of pairs only up to 5K for each semantic class using FPS sampling [4], for a total of 122 100 pairs. Each key is a 1024-dimensional vector, and each value consists of a

SF	Bank	Branch 1		Branch 2		Branch 3		Branch 4	
		FID↓	mIOU↑	FID↓	mIOU↑	FID↓	mIOU↑	FID↓	mIOU↑
1/2	✗	64.2	59.8	59.3	62.6	55.9	62.2	50.1	64.2
	✓	52.8	67.5	51.8	68.7	49.5	68.5	48.1	69.3
1/3	✗	65.9	61.4	59.5	61.6	57.2	65.2	53.1	69.4
	✓	54.1	65.5	53.6	69.6	50.6	68.8	48.4	69.6
1/4	✗	69.6	57.5	62.2	61.8	56.4	65.5	53.0	68.3
	✓	54.9	65.5	54.4	67.1	53.0	66.7	49.7	69.4
Backbone								47.7	69.3

Table S1. Quantitative results for the OASIS pipeline. The minimum number of channels is 64.

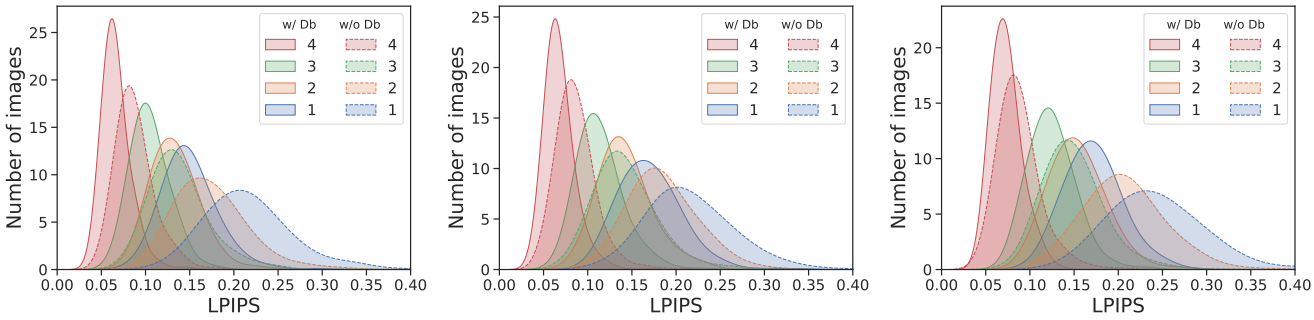

Figure S2. OASIS pipeline, comparison between the database effect to quality distribution for different scale factors. The minimum number of channels is 64. Left to right: SF $= 1/2,\ 1/3,\ 1/4$.

float32 tensor of dimensions $(512, 4, 4)$. The total size of stored parameters is thus 1.1G.

During the retrieval, the guiding features are taken after the first Conv2D and ResNet blocks of the backbone. Then, for each on the $N \in [1, 35]$ semantic classes present in the input, these features are cut into 128 patches and their 1024-dimensional space is scanned in order to find the closest key from the database with corresponding semantic class. This search is performed quite rapidly thanks to the FAISS library [7], and thus does not burden computations.

Once retrieved all 128 patches, a guiding feature is constructed by gluing them together. This feature is concatenated to the input of each branch, and for this reason their number of channels must be increased. When employing the database, the input channels for the first ResBlocks in each branch, reported in Tab. S8, are multiplied by 1.5. The memory overhead from the MegaPortrait's database amounts to 21 Mb. We expect it to be uploaded to the GPU memory, since its size is much lower than that of the network, weighting 131 Mb. On a desktop GPU (P40), the retrieval latency is up to 5 ms, which constitutes 16% of the smallest branch's inference time. We assume our method will be used to speedup networks used in real-time applications, which usually run on edge devices, and use a batch of size 1 to minimize latency.

The last key component of our pipeline is the Predictor. It's architecture is summarized in Table S7.

S1.1.2 Training details

For the implementation of our method we had to train all branches and the predictor.

Branches for OASIS were trained by competing against copies of the original OASIS discriminator. Alongside, we also imposed VGG [6] and LPIPS [16] losses using as ground truth the image synthesized by the backbone,

$$\mathcal{L}_{\text{Branch}} = \mathcal{L}_{\text{OASIS}} + \alpha\mathcal{L}_{\text{VGG}} + \beta\mathcal{L}_{\text{LPIPS}}, \qquad (1)$$

where the overall learning rate was set to 4×10^{-4} and the coefficients were set to $\alpha = 10$ and $\beta = 5$ in order to equalize the losses' contribution. The discriminators retained their original losses. Both the generator and the discriminators were trained via Adam optimization [8] with $\beta_1 = 0$, $\beta_2 = 0.999$. The computations were performed using distributed data parallel from the PyTorch library [12] onto 2 P40 NVIDIA GPUs with batch $= 2$ and lasted approximately 6 days. The resultant qualities can be found in Tab. S1 and Tab. S2.

The OASIS predictor was trained to output images' quality for each branch. We did it by imposing minimum

squared error loss between its predictions and the actual qualities:

$$\mathcal{L}_{\text{Pred}}(z, c; S) = \|P(z, c) - S\|^2. \quad (2)$$

The learning rate was set to 0.01, the loss was optimized via stochastic gradient descent with cosine scheduler [9]. The choice of training set for the predictor was not trivial, since the pipeline inputs consist of a semantic map concatenated to a 3D noise tensor. Due to the high dimensionality of the noise space, sampling uniformly from it does not guarantee any convergence for the learning process. Instead, we randomly extracted 100 3D noise tensors and combined them with 500 semantic maps from the Cityscapes [1] training set, thus obtaining $50\ 000$ examples. We then tested this technique by using 300 and 500 noise tensors. Once trained, we measured the predictor's error by using 500 images from Cityscapes' validation set combined with the same noises used for the training and with new noises. The results are reported, respectively, in Table S3 and Table S4.

Noises	B 1	B 2	B 3	B 4	Mean error
100	5%	6%	6%	7%	6%
300	5%	5%	6%	6%	5.5%
500	5%	5%	6%	6%	5.5%

Table S3. Validation error for the OASIS predictor. The validation set was created joining the noises used for the training to the 500 semantic maps from the validation set of the Cityscapes dataset.

Noises	B 1	B 2	B 3	B 4	Mean error
100	14%	14%	13%	16%	14%
300	10%	11%	11%	15%	12%
500	10%	10%	10%	13%	11%

Table S4. Test error for the OASIS predictor. The test set was created joining random noises to the 500 semantic maps from the validation set of the Cityscapes dataset.

S1.2. The MegaPortraits pipeline

S1.2.1 Architectures and dimensions

The original MegaPortraits [3] generative DNN for images of resolution 512×512 pixels consists of a set of modules predicting a volumetric representation and another set, called G2D, that renders an output image from a processed volume. Its total number of parameters is 32M. We appended our branches after ResBlock2D modules $2,\ 4,\ 6$. Their respective length is $7,\ 5,\ 3$. Just as before, we created lighter computational paths by scaling down all channels uniformly. The new channel numbers were obtained multiplying the original ones by a scale factor. As before, we restricted the effect of this scaling by imposing a minimum number of channels equal to 24, under which no further scaling was forced. We enforced a plethora of different scale factors, which we report in Table S9. For this task, we used a database containing 960 key-value pairs. The values consisted of RGB images of the source subject, uniformly covering the space of head rotations and expressions. The keys were obtained exploiting the MegaPortraits initial modules, the so-called encoders, that yield the Euler angles at which a head is rotated, as well as a multitude of parameters encoding face expressions. Each key encoded 3 angles and a 512-dimensional vector for the expressions. The total size of stored parameters is therefore 0.9G.

The database was searched for the closest key during the inference phase with the aid of the FAISS library [7]. Each retrieved image was subsequently concatenated to the input of all ResBlock2D modules in every branch, thus when employing the database 3 channels must be added to all input channels in Table S9. The architecture of the MegaPortraits predictor is summarized in Table S7.

S1.2.2 Training details

For the MegaPortraits pipeline, we trained our branches using hinge adversarial loss, each branch competing against a copy of multi-scale patch discriminator [17]. Additionally, we imposed feature matching [14], VGG19 perceptual [6], L1 and MS-SSIM [15] losses. We also use a specialized gaze loss computed with a VGG16 network that distills gaze detection (RT-GENE, [5]) and blink detection (RT-BENE, [2]) systems into one model. More details on the losses can be found in MegaPortraits [3]. All losses are computed in relation to the backbone images and using only foreground regions. Overall, the total loss is

$$\mathcal{L}_{\text{Branch}} = \mathcal{L}_{\text{Adv}} + c_1\mathcal{L}_{\text{VGG}} + c_2\mathcal{L}_{\text{MS-SSIM}} + c_3\mathcal{L}_{\text{L1}} + c_4\mathcal{L}_{\text{FM}} + c_5\mathcal{L}_{\text{GL}} \quad (3)$$

with the following weights: $c_1 = 18$, $c_2 = 0.84$, $c_3 = 0.16$, $c_4 = 40$, and $c_5 = 5$. Branches and discriminators were trained using AdamW optimizers [10] with $\beta_1 = 0.05$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$, weight decay $= 10^{-2}$ and initial learning rate $= 2 \times 10^{-4}$. Cosine learning rate schedulers were employed during training with minimum learning rate of 10^{-6}. Computations were done via PyTorch distributed data parallel. The model was trained in mixed precision on 2 P40 NVIDIA GPUs with effective batch size 6 for approximately 3 days. The resultant qualities can be found in Tab. S5.

For each input, the Predictor estimates LPIPS for all branches. To train it, we imposed MAE loss between predicted and state of truth similarity: $\mathcal{L}_{\text{Pred}}(z, c; S) = |P(z, c) - S|$. We employed the AdamW optimizer with

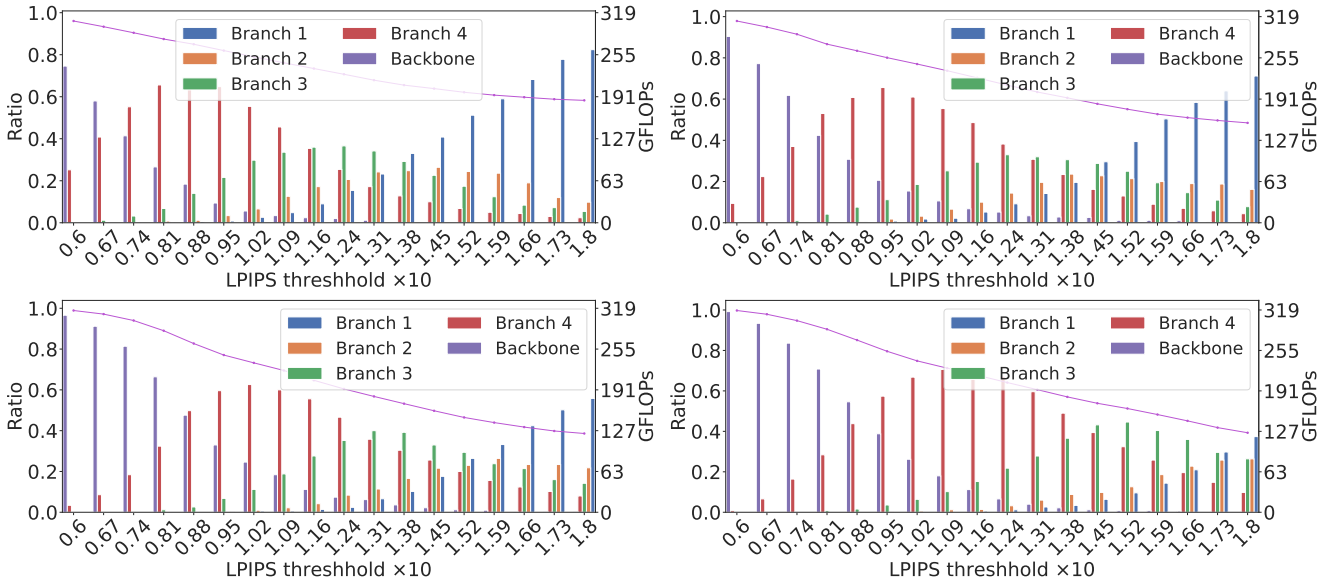

Figure S3. OASIS pipeline, min channels=32, comparison between the efficacy of different scale factors. Top to bottom, left to right: SF $= 1/2,\ 1/3,\ 1/4,\ 1/6$.

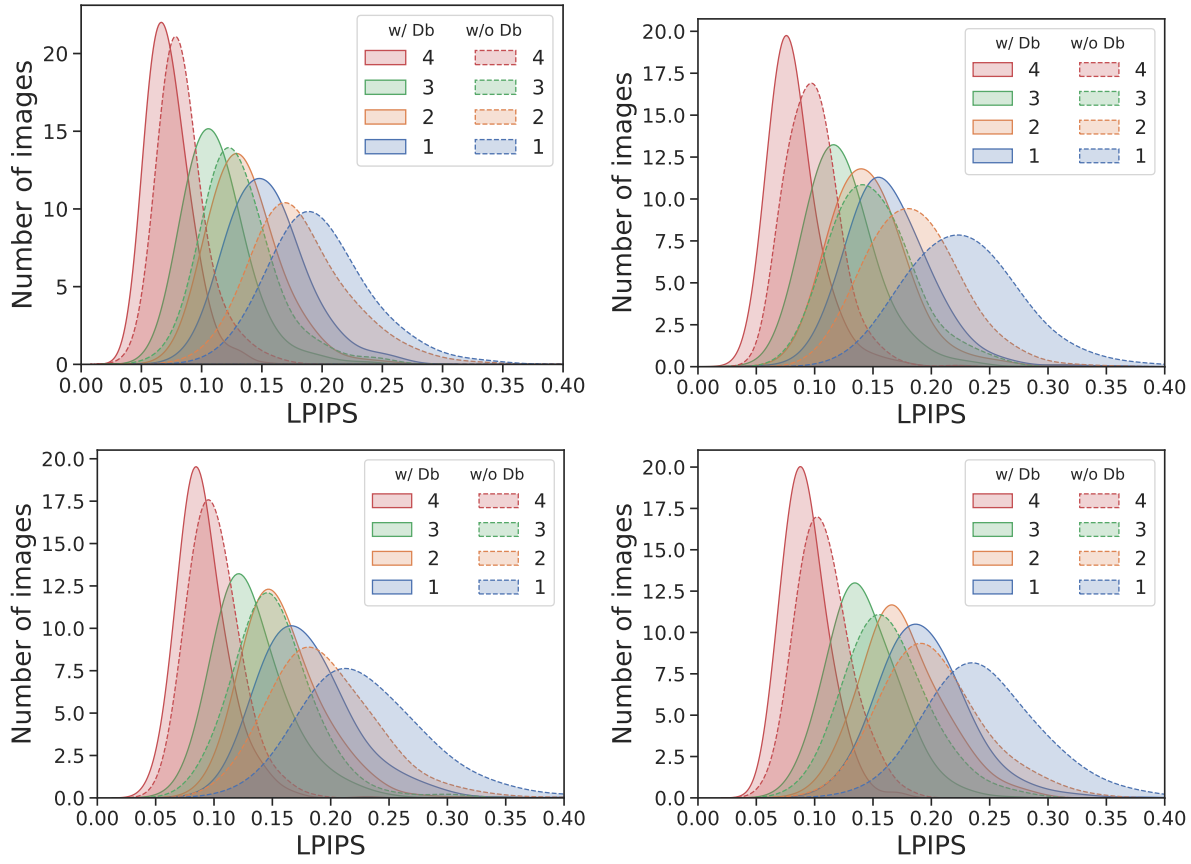

Figure S4. OASIS pipeline, comparison between database effect to quality distribution for min channels=32 and different scale factors. Top to bottom, left to right: SF $= 1/2,\ 1/3,\ 1/4,\ 1/6$.

SF	Bank	Branch 1		Branch 2		Branch 3		Branch 4	
		FID↓	mIOU↑	FID↓	mIOU↑	FID↓	mIOU↑	FID↓	mIOU↑
1/2	✗	58.8	62.8	58.3	63.3	53.2	66.9	51.3	69.0
	✓	52.3	65.6	51.4	67.8	49.6	67.3	48.6	67.8
1/3	✗	66.8	57.1	60.7	62.6	52.8	65.9	51.9	66.7
	✓	55.2	66.7	54.1	66.1	53.1	67.0	51.9	68.3
1/4	✗	69.5	59.7	60.8	61.4	58.4	65.1	54.4	67.4
	✓	57.7	65.9	57.4	66.7	55.3	67.5	51.2	67.7
1/6	✗	69.5	56.7	65.2	62.1	61.9	65.1	54.0	66.4
	✓	60.6	67.6	58.1	66.8	57.6	67.0	51.4	68.9
Backbone								47.7	69.3

Table S2. Quantitative results for the OASIS pipeline at different scale factors. The minimum number of channels is 32.

		Cross-reenactment		
SF	Bank	Branch 1 FID↓	Branch 2 FID↓	Branch 3 FID↓
1/3	✗	56.05	52.77	49.08
	✓	54.60	52.40	50.44
1/6	✗	61.30	55.58	51.00
	✓	59.01	54.08	50.84
1/8	✗	61.84	55.66	50.88
	✓	57.94	54.88	50.96
1/15	✗	66.87	61.75	51.56
	✓	57.25	57.70	51.85
Backbone				50.28

Table S5. Quantitative results for the MegaPortraits pipeline, cross-reenactment.

B 1	B 2	B 3	Mean error
1%	1%	2%	1%

Table S6. Test error for the MegaPortraits predictor for SF = 1/8.

$\beta_1 = 0.05$, $\beta_2 = 0.999$ and initial learning rate 2×10^{-4} alongside cosine learning rate scheduler.

S2. Comparisons

We implemented all architectures listed in Table S8 and Table S9. The overall results for the OASIS pipeline can be compared in Fig. S1 and Fig. S3, while for the MegaPortraits pipeline they are shown in Fig. S6 . We can see how different scale factors yield different branch distributions. The effect of the database on the branches of all scale factors is reported in Fig. S2.

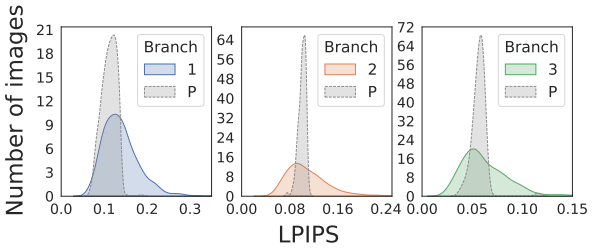

Figure S5. Comparison between quality distributions of single MegaPortraits branches, and quality distributions obtained by use of the predictor (P). The predictor was set to enforce thresholds equal to the branches' mean quality. LPIPS were obtained by comparing images of branches for SF=1/8 with backbones' images. The curves are the result of kernel density estimation with bandwidth 0.3.

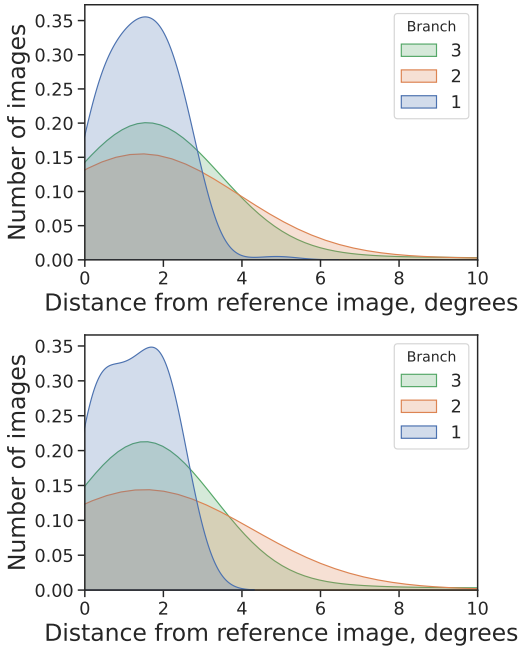

Figure S8. MegaPortraits pipeline, distribution of images routed to different branches in relation to their head rotation angle. First row SF $= 1/8$, second row SF $= 1/15$.

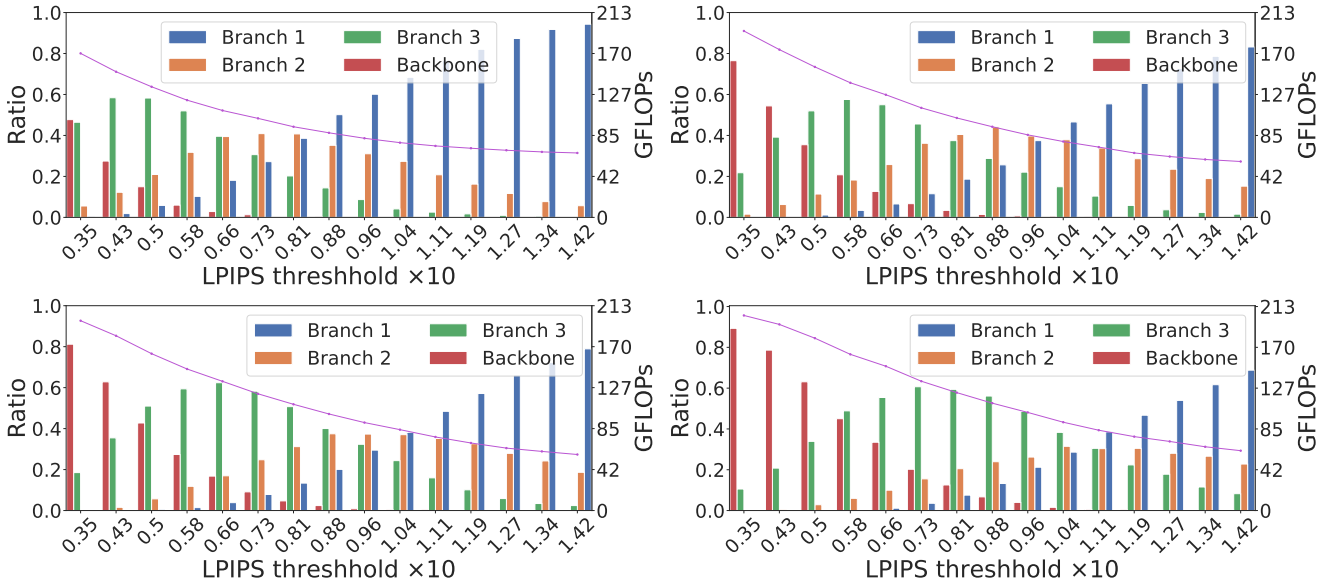

Figure S6. MegaPortraits pipeline, comparison between the efficacy of different scale factors. From left to right: SF $= 1/3,\ 1/6,\ 1/8,\ 1/15$.

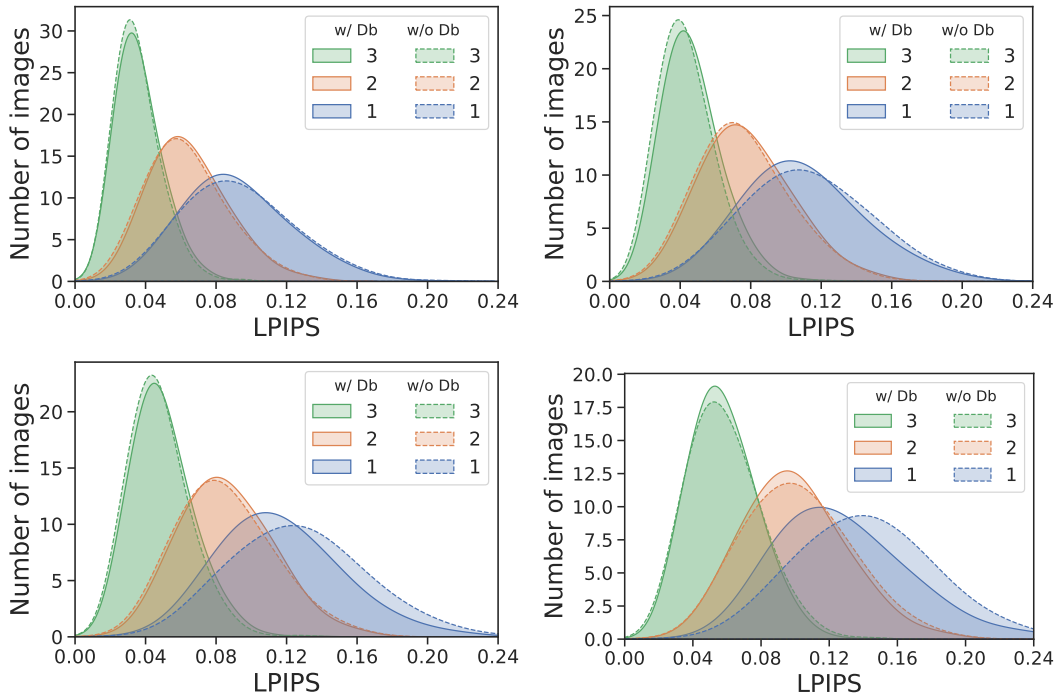

Figure S7. MegaPortraits pipeline, comparison between database effect to quality distribution for different scale factors. Left to right: SF $= 1/3,\ 1/6,\ 1/8,\ 1/15$.

S3. Complexity analysis

For the MegaPortraits pipeline, the quality of synthesized images seems to correlate with the angle at which the head is rotated. This is reflected in our method as well. Indeed, heads rotated at higher angles have greater probability of being routed to a later branch, as evidenced by Fig. S8.

MegaPortraits Predictor	
Module	(in, out)
Flatten	
Linear + LeakyReLu	(1584, 512)
Linear + LeakyReLu	(512, 256)
Linear + LeakyReLu	(256, 128)
Linear + LeakyReLu	(128, 64)
Linear + LeakyReLu	(64, 3)
Total number of parameters = 1M	FLOPs = 1M

OASIS Predictor	
Module	(in, out)
Conv2D + ReLu	(1024, 512)
ResBlock	(512, 512)
Flaten	
Linear + ReLu	(10752, 4096)
Linear + ReLu	(4096, 1024)
Linear + ReLu	(1024, 512)
Linear + ReLu	(512, 128)
Linear	(128, 5)
Total number of parameters = 58M	FLOPs = 250M

Table S7. Architecture of the MegaPortraits predictor. Dimensions are in the form (input channels, output channels).

OASIS branches	SF=1/2	Min. channels = 64		
Module	Branch 1	Branch 2	Branch 3	Branch 4
SPADE-ResBlock	(1024, 512, 16, 32)	(1024, 256, 32, 64)	(512, 128, 64, 128)	(256, 64, 128, 256)
SPADE-ResBlock	(512, 256, 32, 64)	(256, 128, 64, 128)	(128, 64, 128, 256)	(64, 64, 256, 512)
SPADE-ResBlock	(256, 128, 64, 128)	(128, 64, 128, 256)	(64, 64, 256, 512)	
SPADE-ResBlock	(128, 64, 128, 256)	(64, 64, 256, 512)		
SPADE-ResBlock	(64, 64, 256, 512)			
Conv2D, Tanh	(64, 3, 256, 512)	(64, 3, 256, 512)	(64, 3, 256, 512)	(64, 3, 256, 512)
Total number of parameters	w/o bank 45.4M	w/ bank 55.7M		
OASIS branches	SF=1/3	Min. channels = 64		
Module	Branch 1	Branch 2	Branch 3	Branch 4
SPADE-ResBlock	(1024, 336, 16, 32)	(1024, 168, 32, 64)	(512, 84, 64, 128)	(256, 64, 128, 256)
SPADE-ResBlock	(336, 168, 32, 64)	(168, 84, 64, 128)	(84, 64, 128, 256)	(64, 64, 256, 512)
SPADE-ResBlock	(168, 84, 64, 128)	(84, 64, 128, 256)	(64, 64, 256, 512)	
SPADE-ResBlock	(84, 64, 128, 256)	(64, 64, 256, 512)		
SPADE-ResBlock	(64, 64, 256, 512)			
Conv2D, Tanh	(64, 3, 256, 512)	(64, 3, 256, 512)	(64, 3, 256, 512)	(64, 3, 256, 512)
Total number of parameters	w/o bank 35.6M	w/ bank 44.5M		
OASIS branches	SF=1/4	Min. channels = 64		
Module	Branch 1	Branch 2	Branch 3	Branch 4
SPADE-ResBlock	(1024, 256, 16, 32)	(1024, 128, 32, 64)	(512, 64, 64, 128)	(256, 64, 128, 256)
SPADE-ResBlock	(256, 128, 32, 64)	(128, 64, 64, 128)	(64, 64, 128, 256)	(64, 64, 256, 512)
SPADE-ResBlock	(128, 64, 64, 128)	(64, 64, 128, 256)	(64, 64, 256, 512)	
SPADE-ResBlock	(64, 64, 128, 256)	(64, 64, 256, 512)		
SPADE-ResBlock	(64, 64, 256, 512)			
Conv2D, Tanh	(64, 3, 256, 512)	(64, 3, 256, 512)	(64, 3, 256, 512)	(64, 3, 256, 512)
Total number of parameters	w/o bank 30.9M	w/ bank 39.1M		

Table S8. Dimensions of modules for all branches in the form of (input channels, output channels, image height, image width). In all branches, after each SPADE-ResBlock but the last, we also applied 2D nearest-neighbour upsampling, thus doubling the height and width. When employing the database, the input channels for the first ResBlock in each branch, are multiplied by 1.5.

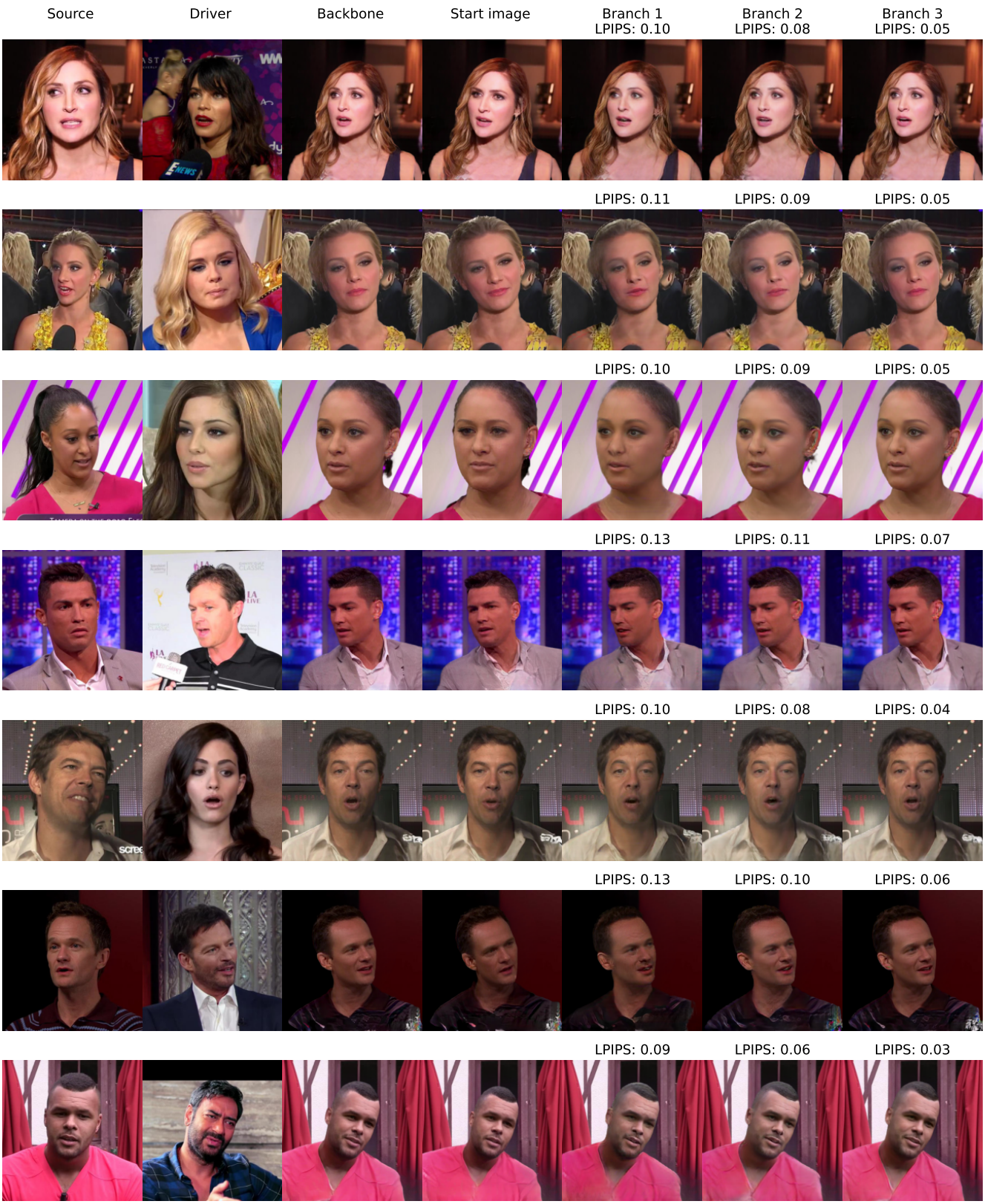

Figure S9. Samples for the MegaPortraits pipeline on SF = 1/8. The background is inpainted.

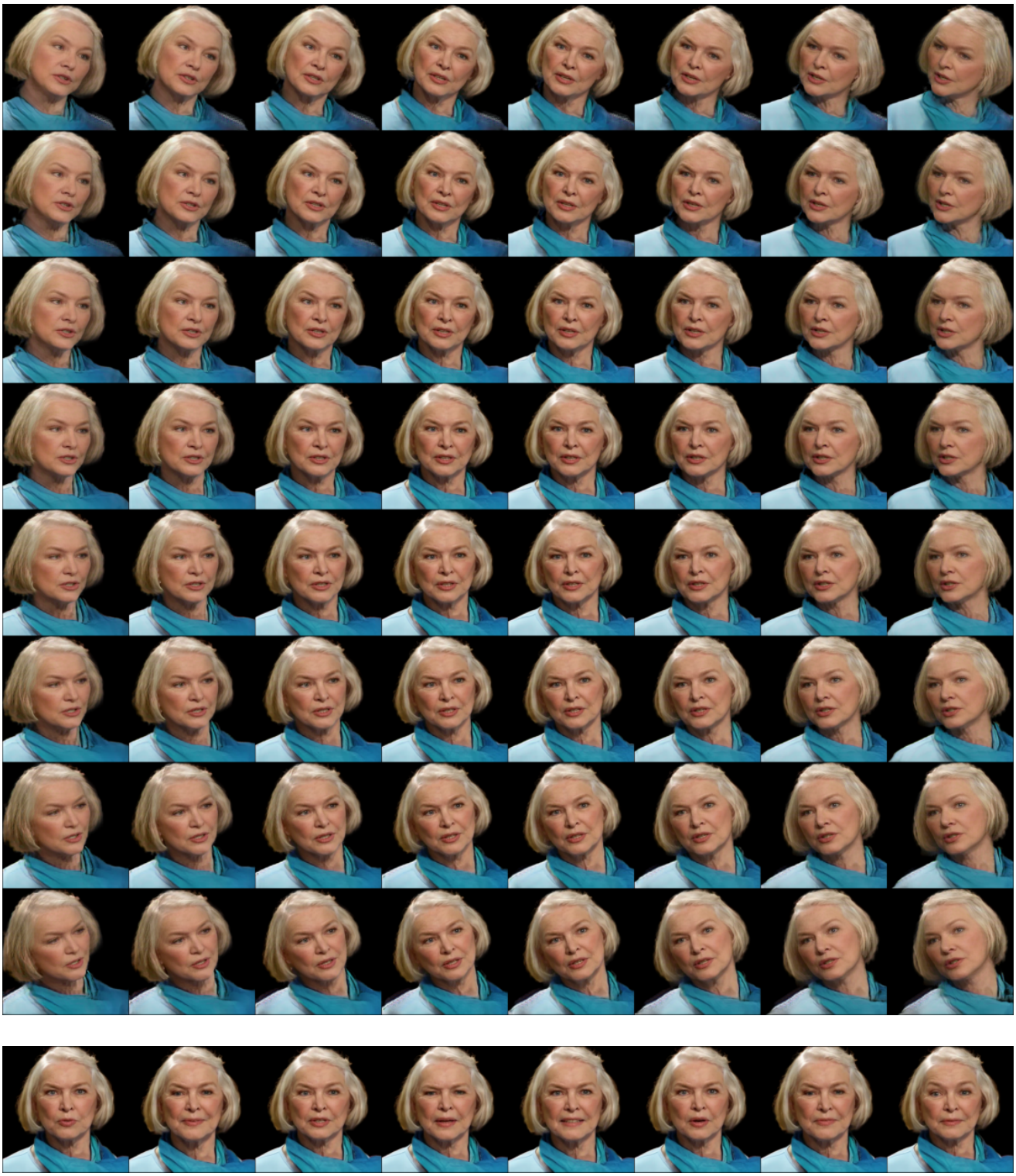

Figure S10. Examples of bank images for different rotations (top) and expressions (bottom).

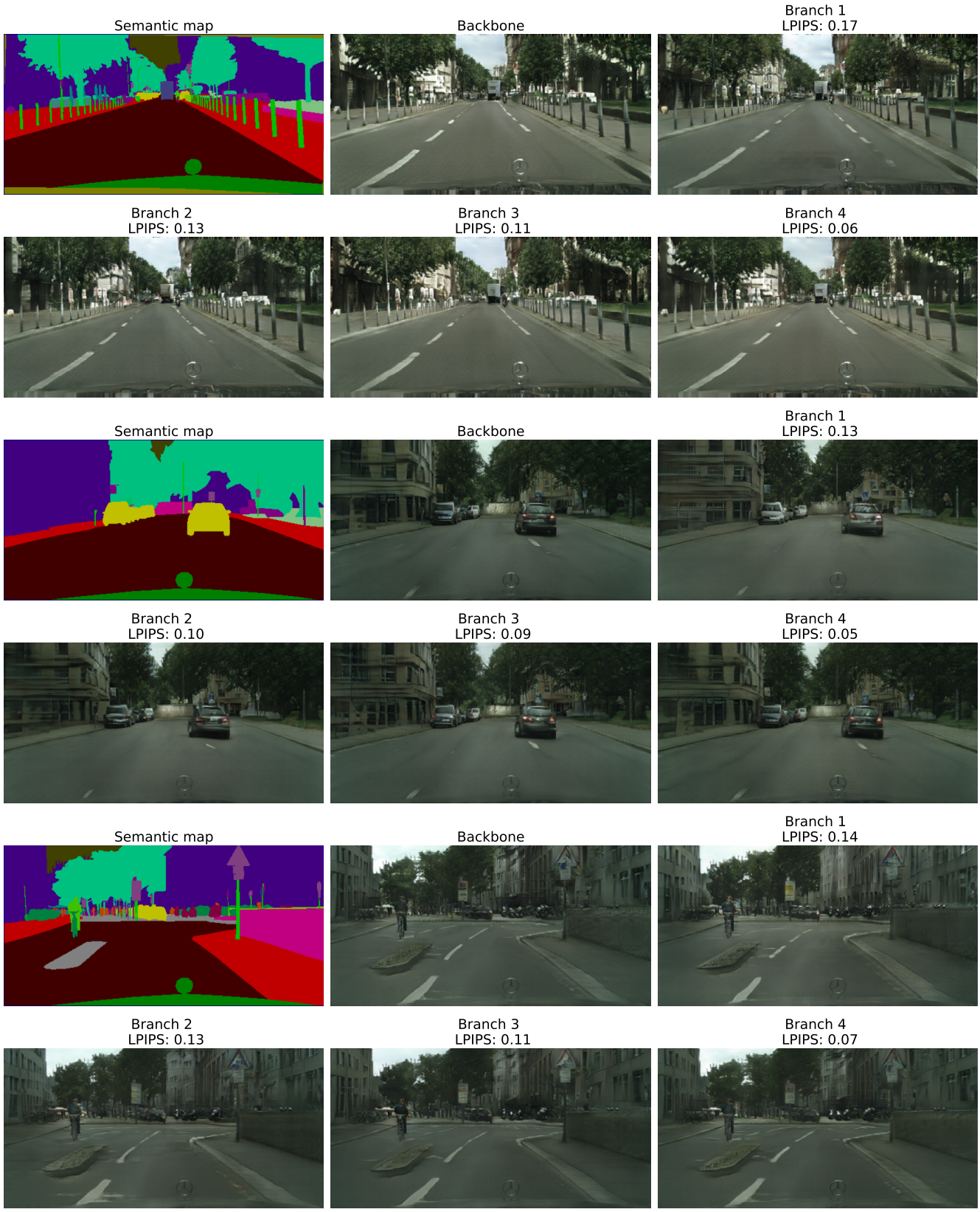

Figure S11. Samples for the OASIS pipeline on SF = 4

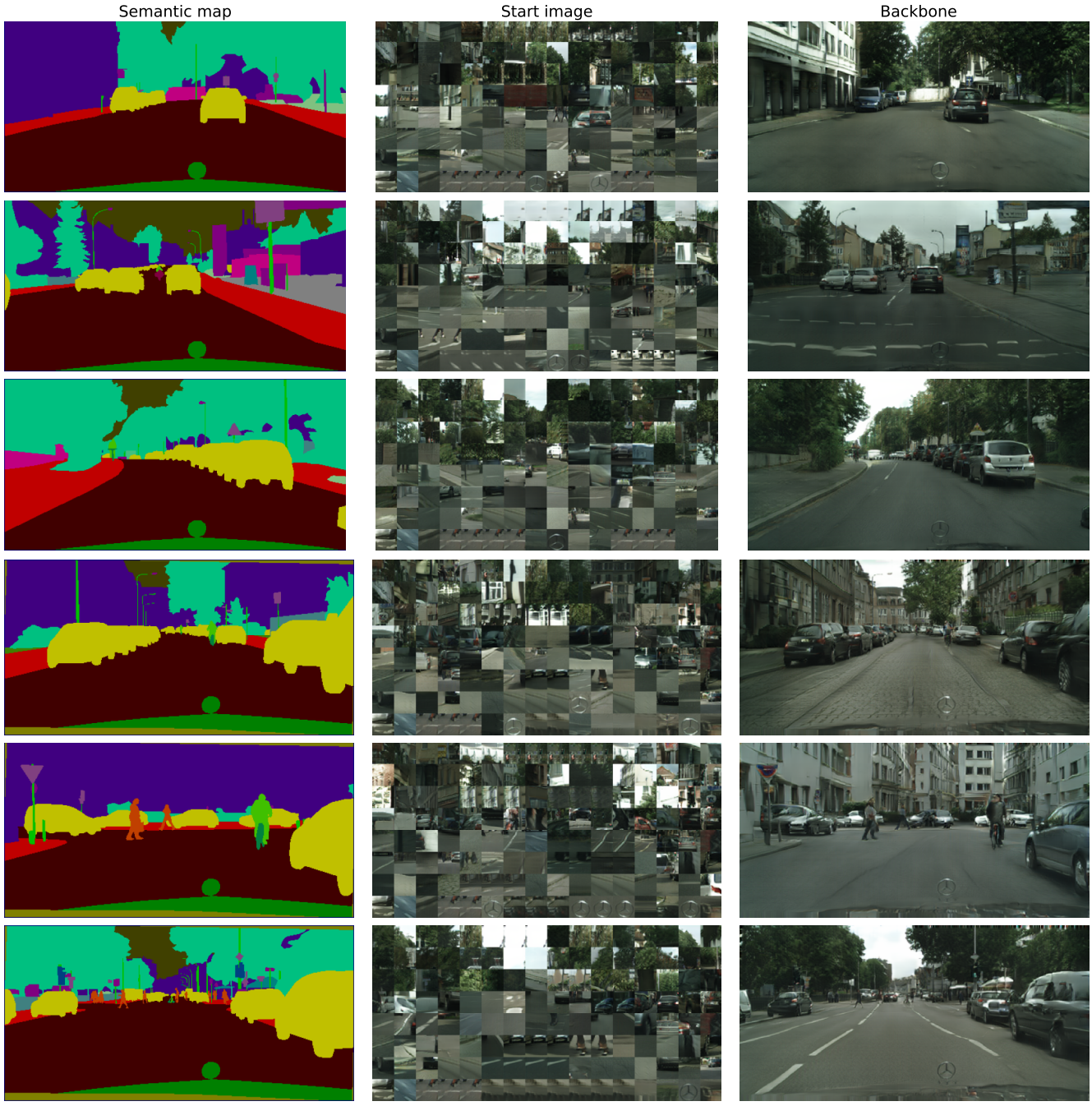

Figure S12. Examples of start images visualizations for the OASIS pipeline. For each feature patch we found the closest one in the bank and then showed corresponding RGB patch of backbone output.

MegaPortraits branches	SF=1/3	Min. channels = 24	
Module	Branch 1	Branch 2	Branch 3
ResBlock2D	(512, 170)	(512, 170)	(512, 170)
ResBlock2D	(170, 170)	(170, 170)	(170, 85)
ResBlock2D	(170, 170)	(170, 85)	(85, 42)
ResBlock2D	(170, 170)	(85, 42)	
ResBlock2D	(170, 85)	(42, 24)	
ResBlock2D	(85, 42)		
ResBlock2D	(42, 24)		
ReLu, Conv2D, Tanh	(24, 3)	(24, 3)	(24, 3)
Total number of parameters	w/ bank 5.6M		
MegaPortraits branches	SF=1/6	Min. channels = 24	
Module	Branch 1	Branch 2	Branch 3
ResBlock2D	(512, 85)	(512, 85)	(512, 85)
ResBlock2D	(85, 85)	(85, 85)	(85, 42)
ResBlock2D	(85, 85)	(85, 42)	(42, 24)
ResBlock2D	(85, 85)	(42, 24)	
ResBlock2D	(85, 42)	(24, 24)	
ResBlock2D	(42, 24)		
ResBlock2D	(24, 24)		
ReLu, Conv2D, Tanh	(24, 3)	(24, 3)	(24, 3)
Total number of parameters	w/ bank 2.3M		

MegaPortraits branches	SF=1/8	Min. channels = 24	
Module	Branch 1	Branch 2	Branch 3
ResBlock2D	(512, 64)	(512, 64)	(512, 64)
ResBlock2D	(64, 64)	(64, 64)	(64, 32)
ResBlock2D	(64, 64)	(64, 32)	(32, 24)
ResBlock2D	(64, 64)	(32, 24)	
ResBlock2D	(64, 32)	(24, 24)	
ResBlock2D	(32, 24)		
ResBlock2D	(24, 24)		
ReLu, Conv2D, Tanh	(24, 3)	(24, 3)	(24, 3)
Total number of parameters	w/ bank 1.6M		
MegaPortraits branches	SF=1/15	Min. channels = 24	
Module	Branch 1	Branch 2	Branch 3
ResBlock2D	(512, 34)	(512, 34)	(512, 34)
ResBlock2D	(34, 34)	(34, 34)	(34, 24)
ResBlock2D	(34, 34)	(34, 24)	(24, 24)
ResBlock2D	(34, 34)	(24, 24)	
ResBlock2D	(34, 24)	(24, 24)	
ResBlock2D	(24, 24)		
ResBlock2D	(24, 24)		
ReLu, Conv2D, Tanh	(24, 3)	(24, 3)	(24, 3)
Total number of parameters	w/ bank 0.8M		

Table S9. MegaPortraits pipeline. Dimensions of modules for all branches in the form of (input channels, output channels). The ResBlock2D are made of layers BatchNorm2D, h-swish, Conv2D, BatchNorm2D, h-swish, Conv2D, Conv2D with skipped connections. In all branches, before every ResBlock2D, we also applied 2D bilinear upsampling. When employing the database, all input channel numbers must be increased by 3.

References

[1] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016. S3

[2] Kevin Cortacero, Tobias Fischer, and Yiannis Demiris. Rt-bene: A dataset and baselines for real-time blink estimation in natural environments. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*. S3

[3] Nikita Drobyshev, Jenya Chelishev, Taras Khakhulin, Aleksei Ivakhnenko, Victor Lempitsky, and Egor Zakharov. Megaportraits: One-shot megapixel neural head avatars. In *Proceedings of the 30th ACM International Conference on Multimedia*. Association for Computing Machinery. 2022. S3

[4] Y. Eldar, M. Lindenbaum, M. Porat, and Y.Y. Zeevi. The farthest point strategy for progressive image sampling. *IEEE Transactions on Image Processing*, 6(9):1305–1315. 1997. S1

[5] Tobias Fischer, Hyung Jin Chang, and Yiannis Demiris. RT-GENE: Real-Time Eye Gaze Estimation in Natural Environments. In *European Conference on Computer Vision*, pages 339–357. S3

[6] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *Computer Vision – ECCV 2016*, pages 694–711, Cham. Springer International Publishing. 2016. S2, S3

[7] Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 7(3):535–547. 2019. S2, S3

[8] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*. 2014. S2

[9] Ilya Loshchilov and Frank Hutter. SGDR: stochastic gradient descent with warm restarts. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net. 2017. S3

[10] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*. 2019. S3

[11] Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. Semantic image synthesis with spatially-adaptive normalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019. S1

[12] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc. 2019. S2

[13] Edgar Schönfeld, Vadim Sushko, Dan Zhang, Juergen Gall, Bernt Schiele, and Anna Khoreva. You only need adversarial supervision for semantic image synthesis. In *International Conference on Learning Representations*. 2021. S1

[14] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018. S3

[15] Z. Wang, E.P. Simoncelli, and A.C. Bovik. Multiscale structural similarity for image quality assessment. In *The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers*, volume 2, pages 1398–1402. 2003. S3

[16] Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018. S2

[17] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. 2017. S3