Supplementary Material: De-rendering 3D Objects in the Wild

Felix Wimbauer[1,2] Shangzhe Wu[2] Christian Rupprecht[2]
[1] Technical University Munich, [2] University of Oxford
wimbauer@in.tum.de {szwu, chrisr}@robots.ox.ac.uk

1. Additional Results

We show some additional qualitaive results. Figure 1 shows a large number of additional decomposition results. Figure 2 shows a large number of additional visual comparisons with other state-of-the-art methods. Tab. 1 analyses the effects of bias in the training data, which might get introduced as a result of our coarse shape, light, and material estimation.

2. Limitations

While our proposed method achieves good results on a wide variety of data, it also comes with some limitations.

First, when there is no coarse light supervision, the training can become unstable and might, after a long training, suddenly diverge to a state where it ignores specular effects. This usually shows in α taking maximum value and the a_{spec} taking minimum value.

Secondly, even though our image formation model is already fairly complex, it can sometimes still not capture all possible lighting effects that can appear in in-the-wild images. In extreme cases, the model tends to bake those shading effects into either the albedo or the shading map, as shown for example in Fig. 3. Additionally, hard shadows can sometimes have an influence on the normal map, even though the shape underneath is of course unaffected by the shadow of another object falling onto it (see second row of Fig. 3). We expect that an explicit modeling of shadows could alleviate this problem.

	Normal N		Albedo A		Specular I_{spec}	
Add. bias	**MSE** ↓	**DIA** ↓	**SIE** ↓	**SSIM** ↑	**MSE** ↓	**L1** ↓
None	0.173	37.8	0.075	0.760	0.112	0.059
Normal $+15°$	0.184	39.4	0.077	0.759	0.124	0.077
Brightn. -0.1	0.171	37.7	0.075	0.766	0.115	0.059
Brightn. $+0.1$	0.183	39.1	0.077	0.716	0.122	0.065

Table 1. **Effects of bias in training data** Artificial bias introduced for normal angles and the brightness of the coarse albedo estimates.

3. Broader Impact & Ethics

The goal of our method is to learn a model that lifts images of objects from 2D into a disentangled 3D representation. We expect this method to be useful for the increasing amount of XR and VFX application in consumer devices. Additionally, learning to de-render images is in itself a challenging computer vision task as the model needs to discover robust and general visual representations.

Alongside the paper, we are releasing a synthetic dataset of 3D objects for future benchmarking under an open, non-commerical license. We expect this to contribute to the field as it establishes a test set with ground truth annotations that was previously not available to researchers.

As the main focus of this work concerns general objects, potential negative societal implications are low. However, we include an evaluation on CelebA-HQ [1], which is a dataset of images of faces of celebrities. The copyright situation is unclear as the dataset was scraped from the internet and naturally contains person identifying information (faces). The dataset contains biases in many forms (celebrities are not representative of the general population, most images are professional photographs, general focus on celebrities from western countries etc.). Models trained on this data can reflect these biases and thus should only be used for research purposes. We include these results to show a comparison to methods that are specialized for human faces while our method can be trained on an object-centric dataset.

As described in the previous section, our method still has some limitations and thus should not be used in critical applications.

4. Technical Details

3D Geometry. The coordinate system used for representing the normal maps has its x-axis pointing left, y-axis pointing upwards, and z-axis pointing towards the viewer. We visualize normal maps by first scaling and translating the value range from $[-1, -1]$ to $[0, 1]$ and then directly converting the xyz coordinates to rgb colors.

The lighting direction is modeled by a vector pointing

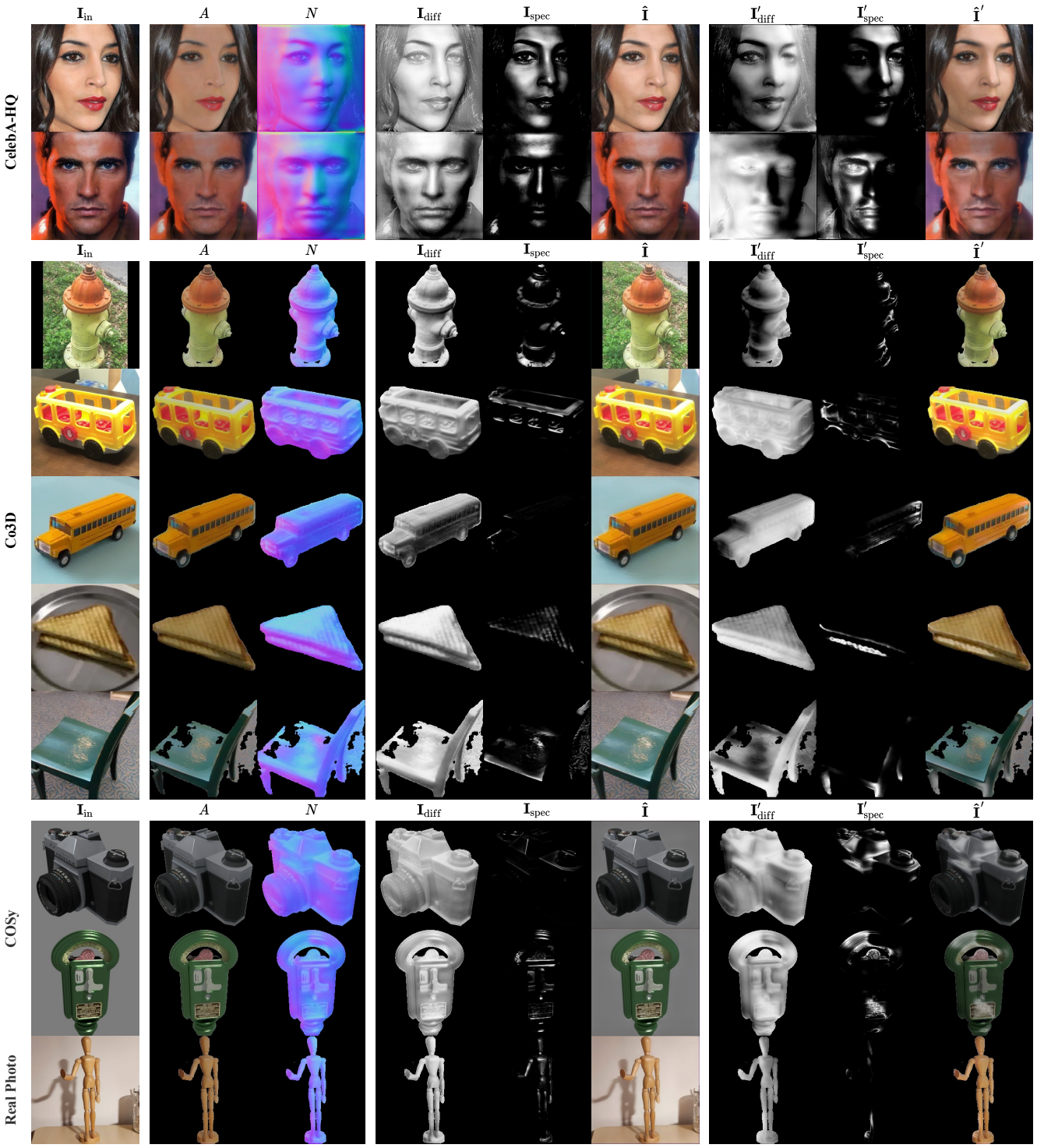

Figure 1. **Additional qualitative results.** Corresponding to Fig. 4 in the main paper.

from the model to the light source. Note, that the lighting is modeled to be relative to the camera position Therefore, the viewing direction, relevant for specular effects, is always constant at $v = (0, 0, 1)^T$.

To obtain the normal map N_{D} from the depth map D, which is used in Eq. 1, we compute the 3D tangent plane of a point using the 4-neighborhood of this point in image space. The normal of the tangent plant is then used as the

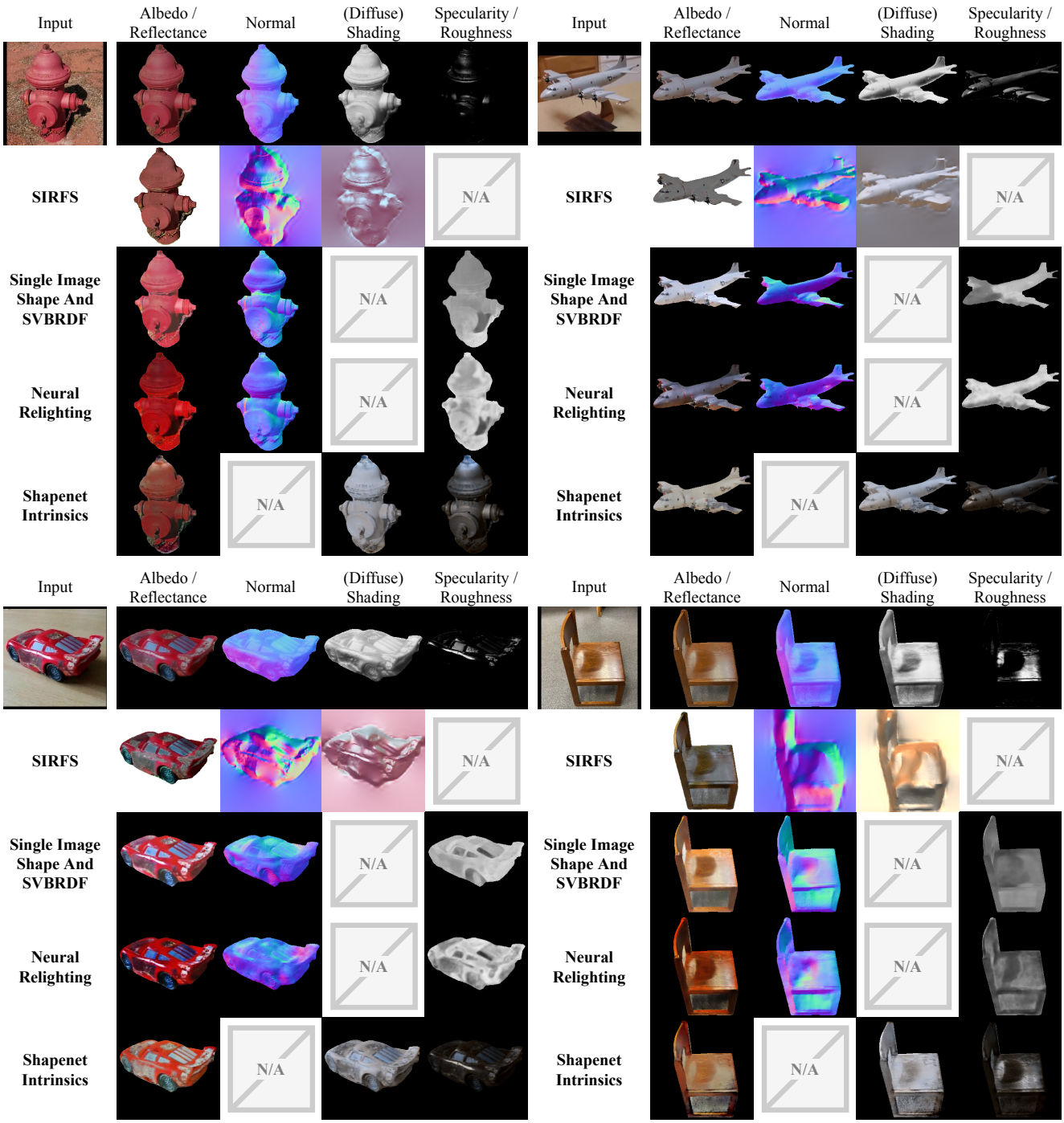

Figure 2. **Additional qualitative comparison with state of the art.** Corresponding to Fig. 6 in the main paper.

normal at this point.

The coarse normal map N_c is the a key component in the optimization to obtain the coarse albedo and light estimate. It is usually either given at a lower resolution than the input image, or it is sparse. In the case of lower resolution, we downsample the input image to match the resolution of the coarse normal map. When computing the coarse losses, we upsample the coarse normal and albedo map again.

In the case that N_c is sparse, we fill it in using nearest neighbor interpolation before computing the initial albedo estimate $\tilde{A}_c$. However, during albedo and light optimization, as well as during the computation of the coarse losses,

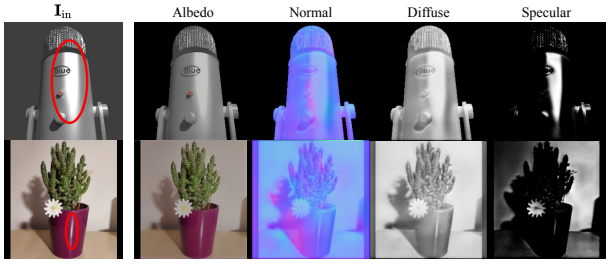

Figure 3. **Limitations.** Extreme lighting effects with out-of-distribution examples. Strong specular reflections can sometimes bleed into the albedo map. Strong shadows can have an effect on the normal map.

we only consider valid pixels.

Light Prediction and Sampling. When predicting the light direction, we fix the $z = 1$ (facing from the camera side to the object) and only predict the $x, y \in [-1, 1]$ components. Afterwards, we normalize the vector. The reasoning behind this is that there only exist meaningful lighting effects, when the light is not coming from behind the object. Further, we apply a custom scaling function $f(x) = (\frac{x+1}{2}(\alpha_{\max} - 1) + 1)^2$ to obtain α. We also limit the range of a_{spec} to $[a_{\text{spec,min}}, a_{\text{spec,max}}]$

The adversarial loss requires images that are relit under random lighting conditions. To also cover unusual light directions $l' = (x', y', z')$, we again fix $z' = 1$, sample $x', y' \sim \mathcal{N}(0, \sigma_l)$ and then normalize l'. As strength of ambient and directional lighting only really influences the overall brightness of the image, which is a very easy cue to pick up on for the discriminator, they have to remain within the distribution of the training data. Therefore, we compute the mean μ_{amb} of s_{amb} over the training batch and then sample $s'_{\text{amb}} \sim \mathcal{N}(\mu_{\text{amb}}, 0.1)$ (same for s'_{dir}).

Implementation. All models and data processing steps (except for the Point Cloud Library) are implemented in PyTorch [2]. The Image-to-image networks (shape, albedo, specular refinement) are implemented as auto-encoders with skip-connections, inspired by the U-Net [3]. The light network follows a classical encoder architecture. All of them use the tanh activation function and the respective outputs gets scaled to the corresponding value range (*e.g.* to $[0, 1]$ for colors). Sec. 4 and Sec. 4 give an overview over the different network configurations.

Training. For all training runs, we use a batch size of 12 on a GPU with 24GB VRAM. We train the **CelebA-HQ** model for 30 epochs (approx. 60k iterations) and the **Co3D** model for 10 epochs (approx. 20k iterations). We only use specular refinement for the CelebA-HQ model and train it in a second training stage for 5 more epochs. Here, we freeze all other network weights and only use the adversarial loss with a weight of $\lambda_{\text{gan}} = 0.1$. Tab. 2 shows the exact hyperparameters used for training. On Co3D, we use the objects masks obtained from SfM to mask the prediction during training and inference.

Model Configuration		
Param.	CelebA-HQ	CO3D
D	$[0.9, 1.1]$	$[0.9, 1.1]$
a_{spec}	$[0.0, 0.5]$	$[0.1, 0.5]$
$\alpha_{\max}$	64	64

Training Configuration			
Param.	Value	Param.	Value
n	12	λ_A	1
η	$1e^{-4}$	λ_L	$1_{\text{(CelebA-HQ)}}$ / $0_{\text{(Co3D)}}$
λ_D	0.5	λ_{rec}	0.5
λ_N	1	λ_{gan}	0.01

Optimization Configuration			
CelebA-HQ		**CO3D**	
Param.	Value	Param.	Value
i	100	i	100
η_{light}	0.01	η_{light}	0.01
η_{albedo}	0.04	η_{albedo}	0.01
λ_{TV}	5	λ_{TV}	20

Table 2. **Model Configuration and Hyperparameters.** n denotes batch size. η denotes the learning rate. i denotes the number of iterations for optimization.

Table 3. **Encoder Architecture.** Architecture of the shape f_S and pose network f_P. The network follows a convolutional encoder structure. n is the number of parameters predicted by each network.

Encoder	Output size
Conv(3, 64, 4, 2) + ReLU()	128×128
Conv(64, 128, 4, 2) + ReLU()	64×64
Conv(128, 256, 4, 2) + ReLU()	32×32
Conv(256, 512, 4, 2) + ReLU()	16×16
Conv(512, 512, 4, 2) + ReLU()	8×8
Conv(512, 512, 4, 2) + ReLU()	4×4
Conv(512, 512, 4, 2) + ReLU()	2×2
Conv(512, 512, 4, 1) + ReLU	1×1
Conv(512, c_{out}, 1, 1) $\rightarrow$ output	1×1

COSy Dataset. The **COSy** dataset is built from ten publicly available 3D scenes for the Blender 3D modeling software[1]. The 3D models can be downloaded from the following links and are available under variants of the Creative Commons license. Hot Dog[2], Accordion[3], Wall-Phone[4], Hy-

[1] https://www.blender.org/
[2] https://blendswap.com/blend/23962
[3] https://blendswap.com/blend/17099
[4] https://blendswap.com/blend/19579

Table 4. **Auto-Encoder Architecture.** Architecture of Φ_{shape}, Φ_{albedo}, and $\Phi_{\text{spec\_ref}}$. The network follows a U-Net structure [3]. All convolution operators zero-pad the input such that the output has the same resolution.

Encoder	Output size
Conv(3, 64, 3, 1) + LReLU(0.1) + Conv(64, 64, 3, 1) + LReLU(0.1) + MaxPool(2)	128×128
Conv(64, 128, 3, 1) + LReLU(0.1) + Conv(128, 128, 3, 1) + LReLU(0.1) + MaxPool(2)	64×64
Conv(128, 256, 3, 1) + LReLU(0.1) + Conv(256, 256, 3, 1) + LReLU(0.1) + MaxPool(2)	32×32
Conv(256, 512, 3, 1) + LReLU(0.1) + Conv(512, 512, 3, 1) + LReLU(0.1) + MaxPool(2)	16×16
Conv(512, 1024, 3, 1) + LReLU(0.1) + Conv(1024, 1024, 3, 1) + LReLU(0.1) + MaxPool(2)	8×8
Conv(1024, 1024, 3, 1) + LReLU(0.1) + Conv(1024, 1024, 3, 1) + LReLU(0.1) + MaxPool(2)	4×4
Conv(1024, 1024, 3, 1) + LReLU(0.1) + Conv(1024, 1024, 3, 1) + LReLU(0.1) + MaxPool(2)	2×2
Conv(1024, 1024, 3, 1) + LReLU(0.1) + Conv(1024, 1024, 3, 1) + LReLU(0.1)	2×2
Decoder	**Output size**
Conv(1024, 1024, 3, 1) + LReLU(0.1) + Conv(1024, 1024, 3, 1) + LReLU(0.1) + Upsample(2)	4×4
Conv(1024, 1024, 3, 1) + LReLU(0.1) + Conv(1024, 1024, 3, 1) + LReLU(0.1) + Upsample(2)	8×8
Conv(1024, 1024, 3, 1) + LReLU(0.1) + Conv(1024, 1024, 3, 1) + LReLU(0.1) + Upsample(2)	16×16
Conv(1024, 512, 3, 1) + LReLU(0.1) + Conv(512, 512, 3, 1) + LReLU(0.1) + Upsample(2)	32×32
Conv(512, 256, 3, 1) + LReLU(0.1) + Conv(256, 256, 3, 1) + LReLU(0.1) + Upsample(2)	64×64
Conv(256, 128, 3, 1) + LReLU(0.1) + Conv(128, 128, 3, 1) + LReLU(0.1) + Upsample(2)	128×128
Conv(128, 64, 3, 1) + LReLU(0.1) + Conv(64, 64, 3, 1) + LReLU(0.1) + Upsample(2)	256×256
Conv(64, c_{out}, 3, 1) + LReLU(0.1) + Conv(c_{out}, c_{out}, 3, 1)	256×256

drant[5], Wingback Chair[6], Camera[7], Toaster[8], Scooter[9], Microphone[10], Parkingmeter[11]. We adapt the models to fit our requirements, for example, define camera views and modify the material so that we can extract the diffuse albedo.

References

[1] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. In *ICLR*, 2018. 1

[2] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*, 2019. 4

[3] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, pages 234–241. Springer, 2015. 4, 5

[5] https://blendswap.com/blend/8443
[6] https://blendswap.com/blend/12555
[7] https://blendswap.com/blend/15833
[8] https://blendswap.com/blend/6231
[9] https://blendswap.com/blend/5256
[10] https://blendswap.com/blend/4145
[11] https://blendswap.com/blend/7714