A. Additional Network and Training Details

Here, we present the architecture of our neural talking-head model. We also discuss the training details.

A.1. Network architectures

The implementation details of the networks in our model are shown in Fig. 1 and described below.

Appearance feature extractor F. The network extracts 3D appearance features from the source image. It consists of a number of downsampling blocks, followed by a convolution layer that projects the input 2D features to 3D features. We then apply a number of 3D residual blocks to compute the final 3D features f_s.

Canonical keypoint detector L. The network takes the source image and applies a U-Net style encoder-decoder to extract canonical keypoints. Since we need to extract 3D keypoints, we project the encoded features to 3D through a 1×1 convolution. The output of the 1×1 convolution is the bottleneck of the U-Net. The decoder part of the U-Net consists of 3D convolution and upsampling layers.

Head pose estimator H and expression deformation estimator Δ. We adopt the same architecture as in Ruiz *et al*. [10]. It consists of a series of ResNet bottleneck blocks, followed by a global pooling to remove the spatial dimension. Different linear layers are then used to estimate the rotation angles, the translation vector, and the expression deformations. The full angle range is divided into 66 bins for rotation angles, and the network predicts which bin the target angle is in. The estimated head pose and deformations are used to transform the canonical keypoints to obtain the source or driving keypoints.

Motion field estimator M. After the keypoints are predicted, they are used to estimate warping flow maps. We generate a warping flow map w_k based on the k-th keypoint using the first-order approximation [11]. Let p_d be a 3D coordinate in the feature volume of the driving image d. The k-th flow field maps p_d to a 3D coordinate in the 3D feature volume of the source image s, denoted by p_s, by:

$$w_k : R_s R_d^{-1}(p_d - x_{d,k}) + x_{s,k} \mapsto p_s. \tag{1}$$

This builds a correspondence between the source and driving.

Using the flow field w_k obtained from the k-th keypoint pair, we can warp the source feature f_s to construct a candidate warped volume, $w_k(f_s)$. After we obtain the warped source features $w_k(f_s)$ using all K flows, they are concatenated together and fed to a 3D U-Net to extract features. Then a softmax function is employed to obtain the flow composition mask m, which consists of K 3D masks, $\{m_1, m_2, ..., m_K\}$. These maps satisfy the constraints that $\sum_k m_k(p_d) = 1$ and $0 \leqslant m_k(p_d) \leqslant 1$ for all p_d. These K masks are then linearly combined with the K warping flow maps, w_k's, to construct the final warping map w by $\sum_{k=1}^{K} m_k(p_d) w_k(p_d)$. To handle occlusions caused by the warping, we also predict a 2D occlusion mask o, which will be inputted to the generator G.

Generator G. The generator takes the warped 3D appearance features $w(f_s)$ and projects them back to 2D. Then, the features are multiplied with the occlusion mask o obtained from the motion field estimator M. Finally, we apply a series of 2D residual blocks and upsamplings layers to obtain the final image.

A.2. Losses

We present details of the loss terms in the following.

Perceptual loss $\mathcal{L}_P$. We use the multi-scale implementation introduced by Siarohin *et al*. [11]. In particular, a pre-trained VGG network is used to extract features from both the ground truth and the output image, and the L_1 distance between the features is computed. Then both images are downsampled, and the same VGG network is used to extract features and compute the L_1 distance again. This process is repeated 3 times to compute losses at multiple image resolutions. We use layers `relu_1_1`, `relu_2_1`, `relu_3_1`, `relu_4_1`, `relu_5_1` of the VGG19 network with weights $0.03125, 0.0625, 0.125, 0.25, 1.0$, respectively. Moreover, since we are synthesizing face images, we also compute a single-scale perceptual loss using a pre-trained face VGG network [9]. These losses are then summed together to give the final perceptual loss.

GAN loss $\mathcal{L}_G$. We adopt the same patch GAN implementation as in [8, 14], and use the hinge loss. Feature matching [14] loss is also adopted to stabilize training. We use single-scale discriminators for training 256×256 images, and two-scale discriminators [14] for 512×512 images.

Equivariance loss $\mathcal{L}_E$. This loss ensures the consistency of estimated keypoints [11, 15]. In particular, let the original image be d and its detected keypoints be x_d. When a known spatial transformation $\mathbf{T}$ is applied on image d, the detected keypoints $x_{\mathbf{T}(d)}$ on this transformed image $\mathbf{T}(d)$ should be transformed in the same way. Based on this observation, we minimize the L_1 distance $\|x_d - \mathbf{T}^{-1}(x_{\mathbf{T}(d)})\|_1$. Affine transformations and randomly sampled thin plate splines are used to perform the transformation. Since all these are 2D transformations, we project our 3D keypoints to 2D by simply dropping the z values before computing the losses.

Keypoint prior loss $\mathcal{L}_L$. As described in the main paper, we penalize the keypoints if the distance between any pair of them is below some threshold D_t, or if the mean depth value deviates from a preset target value z_t. In other words,

$$\mathcal{L}_L = \sum_{i=1}^{K} \sum_{j=1}^{K} \max(0, D_t - \|x_{d,i} - x_{d,j}\|_2^2) + \|Z(x_d) - z_t\| \tag{2}$$

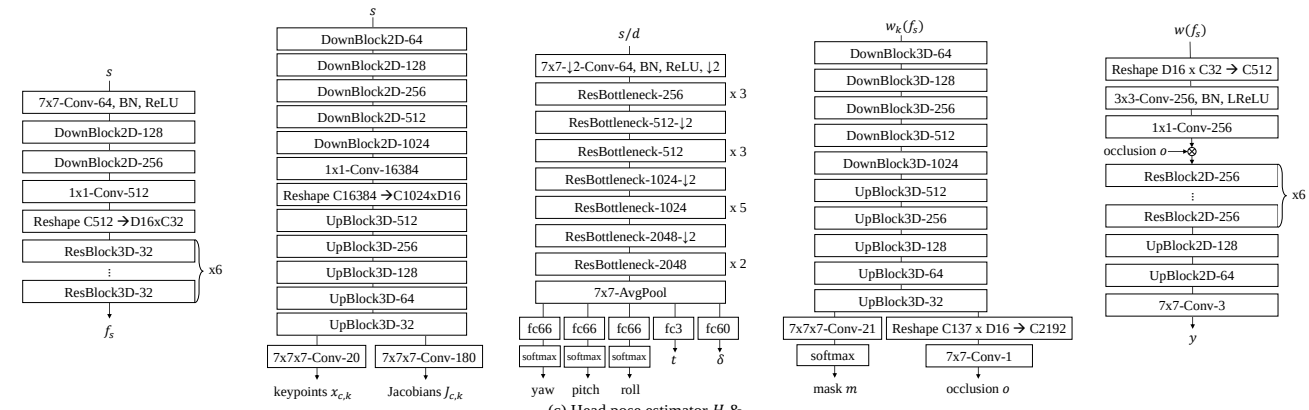

Figure 1: Architectures of individual components in our model. For the building blocks, please refer to Fig. 2

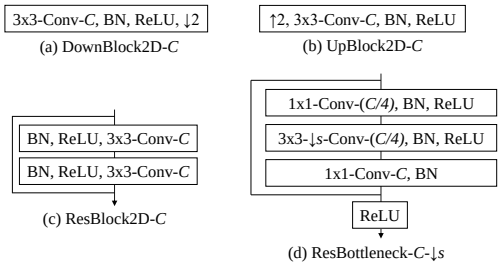

Figure 2: Building blocks of our model. For the 3D counterparts, we simply replace 2D convolutions with 3D convolutions in the blocks.

where $Z(\cdot)$ extracts the mean depth value of the keypoints. This ensures the keypoints are more spread out and used more effectively. We set D_t to 0.1 and z_t to 0.33 in our experiments.

Head pose loss $\mathcal{L}_H$. We compute the L_1 distance between the estimated head pose R_d and the one predicted by a pre-trained pose estimator $\bar{R}_d$, which we treat as ground truth. In other words, $\mathcal{L}_H = \|R_d - \bar{R}_d\|_1$, where the distance is computed as the sum of differences of the Euler angles.

Deformation prior loss $\mathcal{L}_\Delta$. Since the expression deformation Δ is the deviation from the canonical keypoints, their magnitude should not be too large. To ensure this, we put a loss on their $\mathcal{L}_1$ norm: $\mathcal{L}_\Delta = \|\delta_{d,k}\|_1$.

Final loss The final loss is given by:

$$\begin{aligned}\mathcal{L} =&\lambda_P \mathcal{L}_P(d, y) + \lambda_G \mathcal{L}_G(d, y) + \lambda_E \mathcal{L}_E(\{x_{d,k}\}) + \\ &\lambda_L \mathcal{L}_L(\{x_{d,k}\}) + \lambda_H \mathcal{L}_H(R_d, \bar{R}_d) + \lambda_\Delta \mathcal{L}_\Delta(\{\delta_{d,k}\})\end{aligned} \tag{3}$$

where λ's are the weights and are set to $10, 1, 20, 10, 20, 5$ respectively in our implementation.

A.3. Optimization

We adopt the ADAM optimizer [4] with $\beta_1 = 0.5$ and $\beta_2 = 0.999$. The learning rate is set to 0.0002. We apply Spectral Norm [7] to all the layers in both the generator and the discriminator. We use synchronized BatchNorm for the generator. Training is conducted on an NVIDIA DGX1 with 8 32GB V100 GPUs.

We adopt a coarse-to-fine approach for training. We first train our model on 256×256 images for 100 epochs. We then finetune on 512×512 images for another 10 epochs.

B. Additional Experiment Details

B.1. Datasets

We use the following datasets in our evaluations.

VoxCeleb2 [1]. The dataset contains about 1M talking-head videos of different celebrities. We follow the training and test split proposed in the original paper. where we use 280K videos with high bit-rates to train our model. We report our results on the validation set, which contains about 36k videos.

TalkingHead-1KH. We compose a dataset containing about 1000 hours of videos from various sources. A large portion of them is from the YouTube website with the creative common license. We also use videos from the Ryerson audio-visual dataset [6] as well as a set of videos that we recorded with the permission from the subject ourselves. We only use videos whose resolution and bit-rate are both high. We call this dataset *TalkingHead-1KH*. The videos in the TalkingHead-1KH are in general with higher resolutions and better image quality than those in the VoxCeleb2.

B.2. Metrics

We use a set of metrics to evaluate a talking-head synthesis method. We use L_1, PSNR, SSIM, and MS-SSIM for

Table 1: Cross-identity transfer using relative motion.

Method	VoxCeleb2 FID↓	VoxCeleb2 CSIM↑	TalkingHead-1KH FID↓	TalkingHead-1KH CSIM↑
fs-vid2vid [13]	48.48	0.928	44.83	0.955
FOMM [11]	48.91	0.954	42.26	0.961
Ours	**46.43**	**0.960**	**41.25**	**0.964**

quantifying the faithfulness of the recreated videos. We use FID to measure how close is the distribution of the recreated videos to that of the original videos. We use AKD to measure how close the facial landmarks extracted by an off-the-shelf landmark detector from the recreated video are to those in the original video. In the following, we discuss the implementation details of these metrics.

L_1**.** We compute the average L_1 distance between generated and real images.

PSNR measures the image reconstruction quality by computing the mean squared error (MSE) between the ground truth and the reconstructed image.

SSIM/MS-SSIM. SSIM measures the structural similarity between patches of the input images. Therefore, it is more robust to global illumination changes than PSNR, which is based absolute errors. MS-SSIM is a multi-scale variant of SSIM that works on multiple scales of the images and has been shown to correlate better with human perception.

FID [2] measures the distance between the distributions of synthesized and real images. We use the pre-trained InceptionV3 network to extract features from both sets of images and estimate the distance between them.

Average keypoint distance (AKD). We use a facial landmark detector [3] to detect landmarks of real and synthesized images and then compute the average distance between the corresponding landmarks in these two images.

B.3. Relative motion transfer

For cross-identity motion transfer results in our experiment section, we transfer absolute motions in the driving video. For completeness, we also report quantitative comparisons using relative motion proposed in [11] in Table 1. As can be seen, our method still performs the best.

B.4. Ablation study

We perform the following ablation studies to verify the effectiveness of our several important design choices.

Two-step vs. direct keypoint prediction. We estimate the keypoints in an image by first predicting the canonical keypoints and then applying the transformation and the deformations. To compare this approach with direct keypoint location prediction, we train another network that directly predicts the final source and driving keypoints in the image. In particular, the keypoint detector L directly predicts the final source and driving keypoints in the image instead of the canonical ones, and there is no pose estimator H and deformation estimator Δ. Since there is no pose estimator, we do not need any pose supervision (i.e., head pose loss) for this direct prediction network. Note that while this model is only slightly inferior to our final (two-step) model quantitatively, it has no pose control for the output video since the head pose is no longer estimated, so a major feature of our method would be lost.

Table 2: Ablation study. Compared with all the other alternatives, our model (the preferred setting) works the best.

Method	L1	PSNR	SSIM	MS-SSIM	FID	AKD
Direct pred.	10.84	24.00	0.80	0.83	58.55	4.26
Ours (20 kp)	**10.67**	**24.20**	**0.81**	**0.84**	**52.08**	**3.74**
2D Warp	11.64	23.38	0.79	0.82	58.75	4.20
Ours (20 kp)	**10.67**	**24.20**	**0.81**	**0.84**	**52.08**	**3.74**
10 kp	11.49	23.36	0.79	0.82	56.27	4.31
15 kp	11.35	23.53	0.79	0.82	54.36	4.50
Ours (20 kp)	**10.67**	**24.20**	**0.81**	**0.84**	**52.08**	**3.74**

Source image Driving frame Synthesized result

Figure 3: Example failure cases. Our method still struggles when there are occluders such as hands in the image.

3D vs. 2D warping. We generate 3D flow fields from the estimated keypoints to warp 3D features. Another option is to project the keypoints to 2D, estimate a 2D flow field, and extract 2D features from the source image. The estimated 2D flow filed is then used to warp 2D image features.

Number of keypoints. We show that our approach's output quality is positively correlated with the number of keypoints.

As can be seen in Table 2, our model works better than all the other alternatives on all of the performance metrics.

B.5. Failure cases

While our model is in general robust to different situations, it cannot handle large occlusions well. For example, when the face is occluded by the person's hands or other objects, the synthesis quality will degrade, as shown in Fig. 3

Table 3: Size of per-frame metadata in bytes for talking-head methods before and after arithmetic compression.

Method	Before Compression	After Compression			
		Mean	Min	Max	Median
fs-vid2vid [13]	504	231.42	158	599	238
FOMM [11]	240	171.09	159	210	169
Ours (20 kp)	132	84.44	78	104	84
Ours (adaptive)	81.16	53.03	25	102	45

B.6. Canonical keypoints for face recognition

Our canonical keypoints are formulated to be independent of the pose and expression change. They should only contain a person's geometry signature, such as the shapes of face, nose, and eyes. To verify this, we conduct an experiment using the canonical keypoints for face recognition.

We extract canonical keypoints from 384 identities in the VoxCeleb2 [1] dataset to form a training set. For each identity, we also pick a different video of the same identity to form the test set. The training and test videos of the same subject have different head poses and expressions. A face recognition algorithm would fail if it could not filter out pose and expression information. To prove our canonical keypoints are independent to poses and expressions, we apply a simple nearest neighbor classifier using our canonical keypoints for the face recognition task.

Overall, our canonical keypoints reaches an accuracy of 0.070, while a random guess has an accuracy of 0.0026 (Ours is 27× better than the random guess.). On the other hand, a classifier using the off-the-shelf dlib landmark detector only achieves an accuracy of 0.013, which means our keypoints are 5× more effective for face recognition.

C. Additional Video Conferencing Details

C.1. Entropy encoder

We represent each rotation angle, translation, and deformation value as an fp16 floating-point number. Each number consumes two bytes. Naively transmitting the $3K + 6$ floating numbers will result in transmitting $6K + 12$ bytes. We adopt arithmetic coding [5] to encode the $3K + 6$ numbers. Arithmetic coding is one kind of entropy coding. It assigns different codeword lengths to different symbols based on their frequencies. The symbol that appears more often will have a shorter code.

We first apply the driving image encoder to a validation set of 127 videos that are not included in the test set. Each frame will give us $6K + 12$ bytes. We treat each of the bytes separately and build a frequency table for each byte. This gives us $6K + 12$ frequency tables. When encoding the test set, we encode each byte using the associated frequency table learned from the validation set. This results in a varying-length representation that is much smaller than $6K + 12$ bytes on average.

Table 3 shows the sizes of the per-frame metadata in bytes that needs to be transmitted for various talking-head methods before and after performing the arithmetic compression for an image size of 512×512. Our adaptive scheme requires a per-frame metadata size of 53.03 B, which corresponds to $(53.03 \times 8/512^2) = 0.001618$ bits per pixel.

C.2. Adaptive number of keypoints

Our basic model uses a fixed number of keypoints during training and inference. However, on a video call, it is advantageous to adaptively change the number of keypoints used to accommodate varying bandwidth and internet connectivity. We devise a scheme where our synthesis model can dynamically use a smaller number of keypoints for reconstruction. This is based on the intuition that not all of the images are of the same complexity. Some just require fewer keypoints. Using fewer keypoints, we can reduce the bandwidth required for video conferencing because we just need to send a subset of $\delta_{d,k}$'s. To train a model that supports a varying number of keypoints, we randomly choose an index into the array of ordered keypoints, and dropout all values from that index till the end of the array. This dropout percentage ranges from 0% to 75%. This scheme is also helpful when the available bandwidth suddenly drops.

C.3. Binary encoding of the residuals

When the contents of the video being streamed change drastically, *e.g.* when new objects are introduced into the video or the person changes, it becomes necessary to update the source frame being used to perform the talking-head synthesis. This can be done by sending a new image to the receiver. We also devise a more efficient scheme to encode and send only the residual between the ground truth frame and the reconstructed frame, instead of an entirely new source image. To encode a residual image of size 512×512, we use a 3-layer network with convolutions of kernel size 3, stride 2, and 32 channels, similar to the network proposed by Tsai *et al*. [12]. We compute the sign of the latent code of size $32 \times 64 \times 64$ to obtain binary latent codes. The decoder also consists of 3 convolutional layers of 128 filters and uses the pixel shuffle layer to perform upsampling. After arithmetic coding, the binary latent code requires 13.40 KB on average. Note that we do not need to send the encoded binary residual every frame. We just need to send it when the current source image is not good enough to reconstruct the current driving image. In the receiver side, we will use the encoded residual to improve the quality of the reconstructed image. The reconstructed image will become the new source image for decoding future frames using the encoded rotation, translation, and deformations. Example improvements after adding the residual are shown in Fig. 4.

Figure 4: Fixing artifacts in compressed images using our binary residual encoder. We are able to fix artifacts caused due to the introduction of new objects, changes in background, as well as extreme poses by transmitting the residuals encoded as binary values. Each residual binary latent code requires only about 13.40 KB and can replace sending new source images.

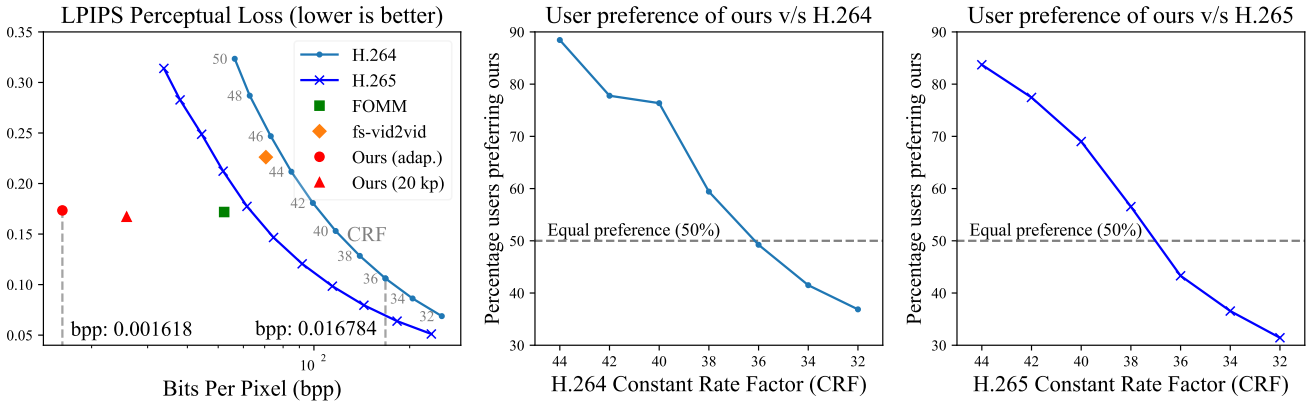

Figure 5: Automatic and human evaluations for video compression. Ours requires much lower bandwidth than the H.264 and H.265 codecs and other talking-head synthesis methods thanks to our keypoint decomposition and adaptive scheme.

C.4. Dataset

For testing, we collect a set of high-resolution talking-head videos from the web. We ensure that the head is of size at least 512×512 pixels and manually check each video to ensure its quality. This results in a total of 222 videos, with a mean of 608 frames, a median of 661 frames, and a min and max of 20 and 1024 frames, respectively.

C.5. Additional experiment results

In Fig. 5(a), we show the achieved LPIPS score by our approach under the adaptive setting (red circle), our approach under the 20 keypoint setting (red triangle), FOMM (green square), fs-vid2vid (orange diamond), H.264, and H.265 using different bpp rates. We observe that our method requires much lower bandwidth than the competing methods.

User study. Here, we describe the details of our user study. We use the Amazon Mechanical Turk (MTurk) platform for the user preference score. A worker needs to have a lift-time approval rate greater than 98 to be qualified for our study. This means that the requesters approve 98% of his/her task assignments. For comparing two competing methods, we generate 222 videos from each method. We show the corresponding pair of videos from two competing methods to three different MTurk workers and ask them to select which one has better visual quality. This gives 666 preference scores for each comparison. We report the average preference score achieved by our method. We compare our adaptive approach to both H.264 and H.265. The user preference scores of our approach when compared to H.264 and H.265 are shown in Fig. 5(b) and (c), respectively. We found that our approach renders comparable performance to H.264 with CRF value 36. For H.265, our approach is comparable to CRF value 37. Our approach was able to achieve the same visual quality using a much lower bit-rate.

References

[1] Joon Son Chung, Arsha Nagrani, and Andrew Zisserman. VoxCeleb2: Deep speaker recognition. In *INTERSPEECH*, 2018. 2, 4

[2] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. GANs trained by a two time-scale update rule converge to a local Nash equilibrium. In *NeurIPS*, 2017. 3

[3] Davis E. King. Dlib-ml: A machine learning toolkit. *JMLR*, 2009. 3

[4] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. 2

[5] Glen G Langdon. An introduction to arithmetic coding. *IBM Journal of Research and Development*, 1984. 4

[6] Steven R Livingstone and Frank A Russo. The ryerson audio-visual database of emotional speech and song (ravdess): A dynamic, multimodal set of facial and vocal expressions in north american english. *PloS one*, 13(5):e0196391, 2018. 2

[7] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. In *ICLR*, 2018. 2

[8] Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. Semantic image synthesis with spatially-adaptive normalization. In *CVPR*, 2019. 1

[9] Omkar M Parkhi, Andrea Vedaldi, and Andrew Zisserman. Deep face recognition. In *BMVC*, 2015. 1

[10] Nataniel Ruiz, Eunji Chong, and James M. Rehg. Fine-grained head pose estimation without keypoints. In *CVPR Workshop*, 2018. 1

[11] Aliaksandr Siarohin, Stéphane Lathuilière, Sergey Tulyakov, Elisa Ricci, and Nicu Sebe. First order motion model for image animation. In *NeurIPS*, 2019. 1, 3, 4

[12] Yi-Hsuan Tsai, Ming-Yu Liu, Deqing Sun, Ming-Hsuan Yang, and Jan Kautz. Learning binary residual representations for domain-specific video streaming. In *AAAI*, 2018. 4

[13] Ting-Chun Wang, Ming-Yu Liu, Andrew Tao, Guilin Liu, Jan Kautz, and Bryan Catanzaro. Few-shot video-to-video synthesis. In *NeurIPS*, 2019. 3, 4

[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 *CVPR*, 2018. 1

[15] Yuting Zhang, Yijie Guo, Yixin Jin, Yijun Luo, Zhiyuan He, and Honglak Lee. Unsupervised discovery of object landmarks as structural representations. In *CVPR*, 2018. 1