LATENT TRANSFORMATIONS FOR OBJECT VIEW POINTS SYNTHESIS

Anonymous authors
Paper under double-blind review

ABSTRACT

We propose a fully-convolutional conditional generative model, the latent transformation neural network (LTNN), capable of view synthesis using a light-weight neural network suited for real-time applications. In contrast to existing conditional generative models which incorporate conditioning information via concatenation, we introduce a dedicated network component, the conditional transformation unit (CTU), designed to learn the latent space transformations corresponding to specified target views. In addition, a consistency loss term is defined to guide the network toward learning the desired latent space mappings, a task-divided decoder is constructed to refine the quality of generated views, and an adaptive discriminator is introduced to improve the adversarial training process. The generality of the proposed methodology is demonstrated on a collection of three diverse tasks: multi-view reconstruction on real hand depth images, view synthesis of real and synthetic faces, and the rotation of rigid objects. The proposed model is shown to exceed state-of-the-art results in each category while simultaneously achieving a reduction in the computational demand required for inference by 30% on average.

1 INTRODUCTION

Generative models have been shown to provide effective frameworks for representing complex, structured datasets and generating realistic samples from underlying data distributions (Goodfellow, 2017). This concept has also been extended to form conditional models capable of sampling from conditional distributions in order to allow certain properties of the generated data to be controlled or selected (Mirza & Osindero, 2014). These generative models are designed to sample from broad classes of the data distribution, however, and are not suitable for inference tasks which require identity preservation of the input data. Models have also been proposed which incorporate encoding components to overcome this by learning to map input data to an associated *latent space* representation within a generative framework (Makhzani et al., 2015). The resulting inference models allow for the defining structure/features of inputs to be preserved while specified target properties are adjusted through conditioning (Yan et al., 2016). Conventional conditional models have largely relied on rather simple methods, such as concatenation, for implementing this conditioning process; however, Miyato & Koyama (2018) have shown that utilizing the conditioning information in a less trivial, more methodical manner has the potential to significantly improve the performance of conditional generative models. In this work, we provide a general framework for effectively performing inference with conditional generative models by strategically controlling the interaction between conditioning information and latent representations within a generative inference model.

In this framework, a conditional transformation unit (CTU), Φ, is introduced to provide a means for navigating the underlying manifold structure of the latent space. The CTU is realized in the form of a collection of convolutional layers which are designed to approximate the latent space operators defined by mapping encoded inputs to the encoded representations of specified targets (see Figure 1). This is enforced by introducing a *consistency loss* term to guide the CTU mappings during training. In addition, a conditional discriminator unit (CDU), Ψ, also realized as a collection of convolutional layers, is included in the network's discriminator. This CDU is designed to improve the network's ability to identify and eliminate transformation specific artifacts in the network's predictions.

The network has also been equipped with RGB balance parameters consisting of three values $\{\theta_R, \theta_G, \theta_B\}$ designed to give the network the ability to quickly adjust the global color balance of

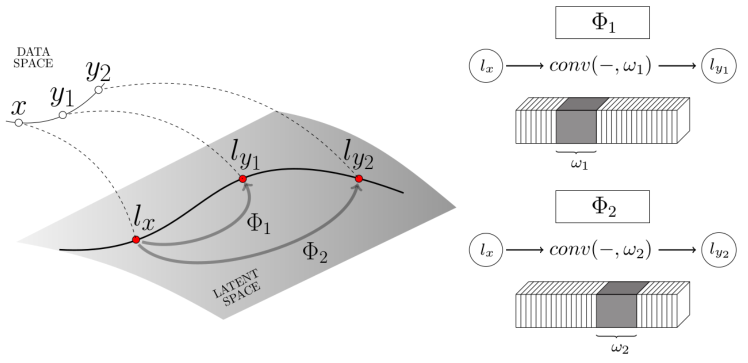
DATA SPACE
x
y_1
y_2
l_x
l_{y_1}
l_{y_2}
Φ_1
Φ_2
LATENT SPACE
Φ_1
$l_x \longrightarrow conv(-, \omega_1) \longrightarrow l_{y_1}$
ω_1
Φ_2
$l_x \longrightarrow conv(-, \omega_2) \longrightarrow l_{y_2}$
ω_2

Figure 1: The conditional transformation unit Φ constructs a collection of mappings $\{\Phi_k\}$ in the latent space which produce high-level attribute changes to the decoded outputs. Conditioning information is used to select the appropriate convolutional weights ω_k for the specified transformation; the encoding l_x of the original input image x is transformed to $\widehat{l}_{y_k} = \Phi_k(l_x) = \mathrm{conv}(l_x, \omega_k)$ and provides an approximation to the encoding l_{y_k} of the attribute-modified target image y_k.

the images it produces to better align with that of the true data distribution. In this way, the network is easily able to remove unnatural hues and focus on estimating local pixel values by adjusting the three RGB parameters rather than correcting each pixel individually. In addition, we introduce a novel estimation strategy for efficiently learning shape and color properties simultaneously; a *task-divided* decoder is designed to produce a coarse pixel-value map along with a refinement map in order to split the network's overall task into distinct, dedicated network components.

Summary of contributions:

1. We introduce the conditional transformation unit, with a family of modular filter weights, to learn high-level mappings within a low-dimensional latent space. In addition, we present a consistency loss term which is used to guide the transformations learned during training.
2. We propose a novel framework for color inference which separates the generative process into three distinct network components dedicated to learning i) coarse pixel value estimates, ii) pixel refinement scaling factors, and iii) the global RGB color balance of the dataset.
3. We introduce the conditional discriminator unit designed to improve adversarial training by identifying and eliminating transformation-specific artifacts present in generated images.

Each contribution proposed above has been shown to provide a significant improvement to the network's overall performance through a series of ablation studies. The resulting latent transformation neural network (LTNN) is placed through a series of comparative studies on a diverse range of experiments where it is seen to outperform existing state-of-the-art models for (i) simultaneous multi-view reconstruction of real hand depth images in real-time, (ii) view synthesis and attribute modification of real and synthetic faces, and (iii) the synthesis of rotated views of rigid objects. Moreover, the CTU conditioning framework allows for additional conditioning information, or target views, to be added to the training procedure *ad infinitum* without any increase to the network's inference speed.

2 Related work

Dosovitskiy et al. (2015) has proposed a supervised, conditional generative model trained to generate images of chairs, tables, and cars with specified attributes which are controlled by transformation and view parameters passed to the network. The range of objects which can be synthesized using the framework is strictly limited to the pre-defined models used for training; the network can generate different views of these models, but cannot generalize to unseen objects to perform inference tasks.

Conditional generative models have been widely used for geometric prediction (Park et al., 2017; Tatarchenko et al., 2016). These models are reliant on additional data, such as depth information or mesh models, to perform their target tasks, however, and cannot be trained using images alone. Other works have introduced a clamping strategy to enforce a specific organizational structure in the latent space (Reed et al., 2014; Kulkarni et al., 2015); these networks require extremely detailed labels for supervision, such as the graphics code parameters used to create each example, and are therefore very difficult to implement for more general tasks (e.g. training with real images). Zhou et al. (2016) have proposed the appearance flow network (AFN) designed specifically for the prediction of rotated viewpoints of objects from images. This framework also relies on geometric concepts unique to rotation and is not generalizable to other inference tasks.

The conditional variational autoencoder (CVAE) incorporates conditioning information into the standard variational autoencoder (VAE) framework (Kingma & Welling, 2013) and is capable of synthesizing specified attribute changes in an identity preserving manner (Sohn et al., 2015; Yan et al., 2016). CVAE-GAN (Bao et al., 2017) further adds adversarial training to the CVAE framework in order to improve the quality of generated predictions. Zhang et al. (2017) have introduced the conditional adversarial autoencoder (CAAE) designed to model age progression/regression in human faces. This is achieved by concatenating conditioning information (i.e. age) with the input's latent representation before proceeding to the decoding process. The framework also includes an adaptive discriminator with conditional information passed using a resize/concatenate procedure.

Isola et al. (2017) have proposed Pix2Pix as a general-purpose image-to-image translation network capable of synthesizing views from a single image. The IterGAN model introduced by Galama & Mensink (2018) is also designed to synthesize novel views from a single image, with a specific emphasis on the synthesis of rotated views of objects in small, iterative steps.

To the best of our knowledge, all existing conditional generative models designed for inference use fixed hidden layers and concatenate conditioning information directly with latent representations; in contrast to these existing methods, the proposed model incorporates conditioning information by defining dedicated, transformation-specific convolutional layers at the latent level. This conditioning framework allows the network to synthesize multiple transformed views from a single input, while retaining a fully-convolutional structure which avoids the dense connections used in existing inference-based conditional models. Most significantly, the proposed LTNN framework is shown to outperform state-of-the-art models in a diverse range of view synthesis tasks, while requiring substantially less FLOPs for inference than other conditional generative models (see Tables 1 & 2).

3 Latent Transformation Neural Network

In this section, we introduce the methods used to define the proposed LTNN model. We first give a brief overview of the LTNN network structure. We then detail how conditional transformation unit mappings are defined and trained to operate on the latent space, followed by a description of the conditional discriminator unit implementation and the network loss function used to guide the training process. Lastly, we describe the task-division framework used for the decoding process.

The basic workflow of the proposed model is as follows:

1. Encode the input image x to a latent representation $l_x = \mathrm{Encode}(x)$.
2. Use conditioning information k to select conditional, convolutional filter weights ω_k.
3. Map the latent representation l_x to $\widehat{l}_{y_k} = \Phi_k(l_x) = \mathrm{conv}(l_x, \omega_k)$, an approximation of the encoded latent representation l_{y_k} of the specified target image y_k.
4. Decode $\widehat{l}_{y_k}$ to obtain a coarse pixel value map and a refinement map.
5. Scale the channels of the pixel value map by the RGB balance parameters and take the Hadamard product with the refinement map to obtain the final prediction $\widehat{y}_k$.
6. Pass real images y_k as well as generated images $\widehat{y}_k$ to the discriminator, and use the conditioning information to select the discriminator's conditional filter weights $\overline{\omega}_k$.
7. Compute loss and update weights using ADAM optimization and backpropagation.

A detailed overview of the proposed network structure is provided in Section A.1 of the appendix.

LTNN Training Procedure

Provide: Labeled dataset $\big\{\big(x, \{y_k\}_{k\in\mathcal{T}}\big)\big\}$ with target transformations indexed by a fixed set $\mathcal{T}$, encoder weights θ_E, decoder weights θ_D, RGB balance parameters $\{\theta_R, \theta_G, \theta_B\}$, conditional transformation unit weights $\{\omega_k\}_{k\in\mathcal{T}}$, discriminator $\mathcal{D}$ with standard weights $\theta_{\mathcal{D}}$ and conditionally selected weights $\{\overline{\omega}_k\}_{k\in\mathcal{T}}$, and loss function hyperparameters $\gamma, \rho, \lambda, \kappa$ corresponding to the smoothness, reconstruction, adversarial, and consistency loss terms, respectively. The specific loss function components are defined in detail in Equations 1 - 5 in Section 3.2.

```
1: procedure TRAIN( )
2:     $x\,, \{y_k\}_{k\in\mathcal{T}} = \text{get\_train\_batch}()$        # Sample input and targets from training set
3:     $l_x = \text{Encode}[\,x\,]$        # Encoding of original input image
4:     for $k$ in $\mathcal{T}$ do
5:         $l_{y_k} = \text{Encode}[\,y_k\,]$        # True encoding of specified target image
6:         $\widehat{l}_{y_k} = \text{conv}(l_x, \omega_k)$        # Approximate encoding of target with CTU
7:         $\widehat{y}_k^{\,value}\,, \widehat{y}_k^{\,refine} = \text{Decode}[\,\widehat{l}_{y_k}\,]$        # Compute RGB value and refinement maps
8:         $\widehat{y}_k = \big[\,\theta_C \cdot \widehat{y}_{k,C}^{\,value} \odot \widehat{y}_{k,C}^{\,refine}\,\big]_{C\in\{R,G,B\}}$        # Assemble final network prediction for target
9:
10:        # Update encoder, decoder, RGB, and CTU weights
11:        $\mathcal{L}_{adv} = -\log(\mathcal{D}(\widehat{y}_k, \overline{\omega}_k))$
12:        $\mathcal{L}_{guide} = \gamma \cdot \mathcal{L}_{smooth}(\widehat{y}_k) + \rho \cdot \mathcal{L}_{recon}(\widehat{y}_k, y_k)$
13:        $\mathcal{L}_{consist} = \|\widehat{l}_{y_k} - l_{y_k}\|_1$
14:        $\mathcal{L} = \lambda \cdot \mathcal{L}_{adv} + \mathcal{L}_{guide} + \kappa \cdot \mathcal{L}_{consist}$
15:        for $\theta$ in $\{\theta_E, \theta_D, \theta_R, \theta_G, \theta_B, \omega_k\}$ do
16:            $\theta = \theta - \nabla_\theta \mathcal{L}$
17:
18:        # Update discriminator and CDU weights
19:        $\mathcal{L}_{adv}^{\mathcal{D}} = -\log(\mathcal{D}(y_k, \overline{\omega}_k)) - \log(1 - \mathcal{D}(\widehat{y}_k, \overline{\omega}_k))$
20:        for $\theta$ in $\{\theta_{\mathcal{D}}, \overline{\omega}_k\}$ do
21:            $\theta = \theta - \nabla_\theta \mathcal{L}_{adv}^{\mathcal{D}}$
```

3.1 Conditional transformation unit

Generative models have frequently been designed to explicitly disentangle the latent space in order to enable high-level attribute modification through linear, latent space interpolation. This linear latent structure is imposed by design decisions, however, and may not be the most natural way for a network to internalize features of the data distribution. Several approaches have been proposed which include nonlinear layers for processing conditioning information at the latent space level. In these conventional conditional generative frameworks, conditioning information is introduced by combining features extracted from the input with features extracted from the conditioning information (often using dense connection layers); these features are typically combined using standard vector concatenation, although some have opted to use channel concatenation (Zhang et al., 2017; Bao et al., 2017). Six of these conventional conditional network designs are illustrated in Figure 2 along with the proposed LTNN network design for incorporating conditioning information.

Rather than directly concatenating conditioning information, we propose using a conditional transformation unit (CTU), consisting of a collection of distinct convolutional mappings in the network's latent space; conditioning information is then used to select which collection of weights, i.e. which CTU mapping, should be used in the convolutional layer to perform a specified transformation. For view point estimation, there is an independent CTU per viewpoint. Each CTU mapping maintains its own collection of convolutional filter weights and uses Swish activations (Ramachandran et al., 2017). The filter weights and Swish parameters of each CTU mapping are selectively updated by controlling the gradient flow based on the conditioning information provided. The CTU mappings are trained to transform the encoded, latent space representation of the network's input in a manner which produces high-level view or attribute changes upon decoding. This is accomplished by introducing a *consistency* term into the loss function which is minimized precisely when the CTU mappings behave as depicted in Figure 1. In this way, different angles of view, light directions, and deformations, for example, can be synthesized from a single input image.

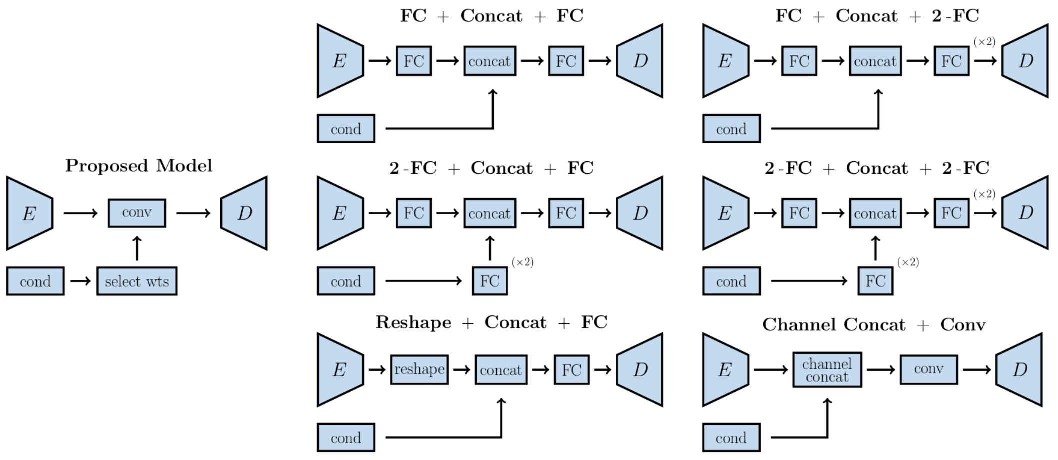

Figure 2: Selected methods for incorporating conditioning information; the proposed LTNN method is illustrated on the left, and six conventional alternatives are shown to the right.

3.2 Discriminator and loss function

The discriminator used in the adversarial training process is also passed conditioning information which specifies the transformation which the model has attempted to make. The conditional discriminator unit (CDU), consisting of convolutional layers with modular weights similar to the CTU, is trained to specifically identify unrealistic artifacts which are being produced by the corresponding conditional transformation unit mappings. For view point estimation, there is an independent CDU per viewpoint. The incorporation of this context-aware discriminator structure has significantly boosted the performance of the network (see Table 4 in the appendix).

The proposed model uses the adversarial loss as the primary loss component. The discriminator, $\mathcal{D}$, is trained using the adversarial loss term $\mathcal{L}_{adv}^{\mathcal{D}}$ defined below in Equation 1. Additional loss terms corresponding to structural reconstruction, smoothness (Jason et al., 2016), and a notion of consistency, are also used for training the encoder/decoder:

$$\begin{aligned}
\mathcal{L}_{adv}^{\mathcal{D}} &= -\log \mathcal{D}(y_k, \overline{\omega}_k) \,-\, \log\big(1 - \mathcal{D}(\widehat{y}_k, \overline{\omega}_k)\big) && (1)\\
\mathcal{L}_{adv} &= -\log \mathcal{D}(\widehat{y}_k, \overline{\omega}_k) && (2)\\
\mathcal{L}_{recon} &= \|\, \widehat{y}_k \,-\, y_k \,\|_2^2 && (3)\\
\mathcal{L}_{smooth} &= 1/8 \cdot \sum_{i\in\{0,\pm1\}} \sum_{j\in\{0,\pm1\}} \big\|\, \widehat{y}_k \,-\, \tau_{i,j}\widehat{y}_k \,\big\|_1 && (4)\\
\mathcal{L}_{consist} &= \big\|\, \Phi_k(\text{Encode}[x]) \,-\, \text{Encode}[y_k] \,\big\|_1 && (5)
\end{aligned}$$

where y_k is the modified target image corresponding to an input x, $\overline{\omega}_k$ are the weights of the CDU mapping corresponding to the k^{th} transformation, Φ_k is the CTU mapping for the k^{th} transformation, $\widehat{y}_k = \text{Decode}\big(\Phi_k\big(\text{Encode}[x]\big)\big)$ is the network prediction, and $\tau_{i,j}$ is the two-dimensional, discrete shift operator. The final loss function for the encoder and decoder components is given by:

$$\mathcal{L} \;=\; \lambda \cdot \mathcal{L}_{adv} \,+\, \rho \cdot \mathcal{L}_{recon} \,+\, \gamma \cdot \mathcal{L}_{smooth} \,+\, \kappa \cdot \mathcal{L}_{consist} \qquad (6)$$

with hyperparameters typically selected so that $\lambda, \rho \gg \gamma, \kappa$. The consistency loss is designed to guide the CTU mappings toward approximations of the latent space mappings which connect the latent representations of input images and target images as depicted in Figure 1. In particular, the consistency term enforces the condition that the transformed encoding, $\widehat{l}_{y_k} = \Phi_k(\text{Encode}[x])$, approximates the encoding of the k^{th} target image, $l_{y_k} = \text{Encode}[y_k]$, during the training process.

3.3 Task separation / Task-divided decoder

The decoding process has been divided into three tasks: estimating the refinement map, pixel-values, and RGB color balance of the dataset. We have found this decoupled framework for estimation helps the network converge to better minima to produce sharp, realistic outputs. The decoding process begins with a series of convolutional layers followed by bilinear interpolation to upsample the low resolution latent information. The last component of the decoder's upsampling process consists of two distinct transpose convolutional layers used for task separation; one layer is allocated for predicting the refinement map while the other is trained to predict pixel-values. The refinement map layer incorporates a sigmoidal activation function which outputs scaling factors intended to refine the coarse pixel value estimations. RGB balance parameters, consisting of three trainable variables, are used as weights for balancing the color channels of the pixel value map. The Hadamard product of the refinement map and the RGB-rescaled value map serves as the network's final output:

$$\widehat{y} = [\,\widehat{y}_R, \widehat{y}_G, \widehat{y}_B\,] \quad \text{where} \quad \widehat{y}_C = \theta_C \cdot \widehat{y}_C^{\,value} \odot \widehat{y}_C^{\,refine} \quad \text{for} \quad C \in \{R, G, B\} \tag{7}$$

In this way, the network has the capacity to mask values which lie outside of the target object (i.e. by setting refinement map values to zero) which allows the value map to focus on the object itself during the training process. Experimental results show that the refinement maps learn to produce masks which closely resemble the target objects' shapes and have sharp drop-offs along the boundaries.

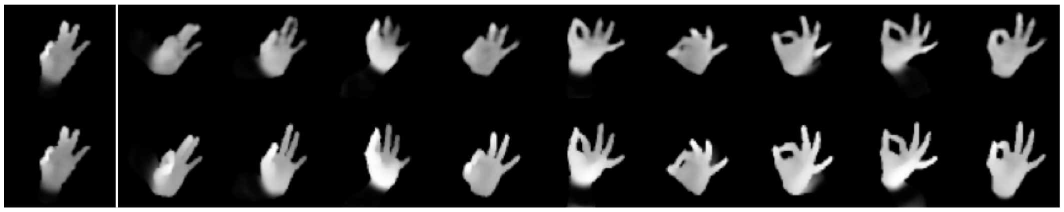

Figure 3: Comparison of CVAE-GAN (top) with proposed LTNN model (bottom) using the noisy NYU hand dataset (Tompson et al., 2014). The input depth-map hand pose image is shown to the far left, followed by the network predictions for 9 synthesized view points. The views synthesized using LTNN are seen to be sharper and also yield higher accuracy for pose estimation (see Figure 6).

4 Experiments and results

To show the generality of our method, we have conducted a series of diverse experiments: (i) hand pose estimation using a synthetic training set and real NYU hand depth image data (Tompson et al., 2014) for testing, (ii) synthesis of rotated views of rigid objects using the real ALOI dataset (Geusebroek et al., 2005) and synthetic 3D chair dataset (Chang et al., 2015a), (iii) synthesis of rotated views using a real face dataset (Fransens et al., 2005), and (iv) the modification of a diverse range of attributes on a synthetic face dataset (IEE, 2009). For each experiment, we have trained the models using 80% of the datasets. Since ground truth target depth images were not available for the real hand dataset, an indirect metric has been used to quantitatively evaluate the model as described in Section 4.1. Ground truth data was available for all other experiments, and models were evaluated directly using the L_1 mean pixel-wise error and the structural similarity index measure (SSIM) (Park et al., 2017; Mathieu et al., 2015) (the masked pixel-wise error L_1^M (Galama & Mensink, 2018) was used in place of the L_1 error for the ALOI experiment). More details regarding the precise training configurations and the creation of the synthetic datasets can be found in the appendix.

To evaluate the proposed framework with existing works, two comparison groups have been formed: conditional inference models (CVAE-GAN, CVAE, and CAAE) with comparable encoder/decoder structures for comparison on experiments with non-rigid objects, and view synthesis models (MV3D (Tatarchenko et al., 2016), IterGAN, Pix2Pix, AFN (Zhou et al., 2016), and TVSN (Park et al., 2017)) for comparison on experiments with rigid objects. Additional experiments have been performed to compare the proposed CTU conditioning method with other conventional concatenation methods (see Figure 2); results are shown in Figure 5. Qualitative results and comparisons for each experiment are provided in the appendix.

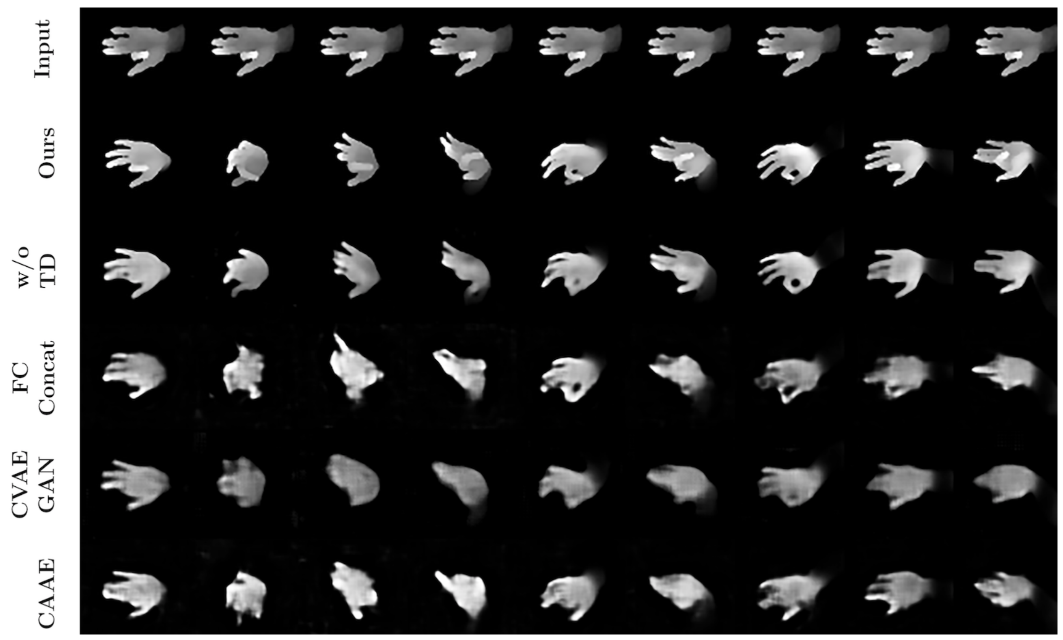

Figure 4: Qualitative evaluation for multi-view reconstruction of hand depth maps using the NYU dataset.

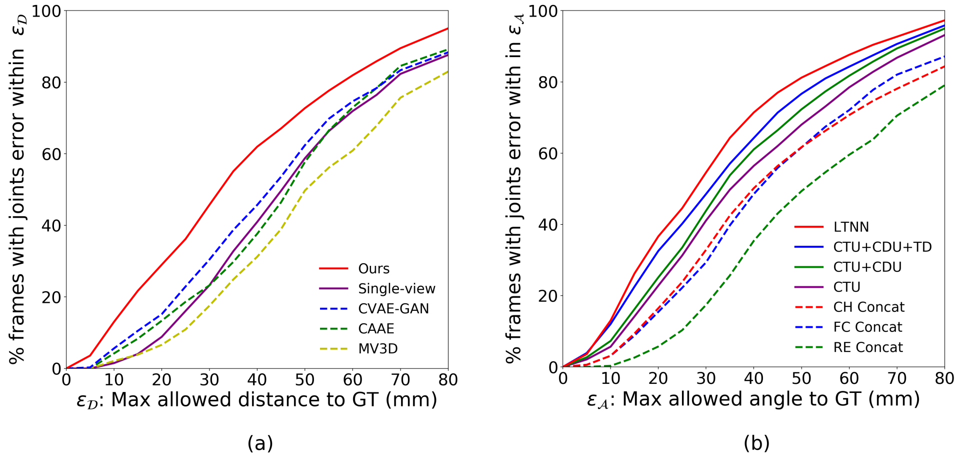

Figure 5: Quantitative evaluation for multi-view hand synthesis. (a) Evaluation with state-of-the-art methods using the real NYU dataset. (b) LTNN ablation results and comparison with alternative conditioning frameworks using synthetic hand dataset. Our models: conditional transformation unit (CTU), conditional discriminator unit (CDU), task-divide and RGB balance parameters (TD), and LTNN consisting of all previous components along with consistency loss. Alternative concatenation methods: channel-wise concatenation (CH Concat), fully connected concatenation (FC Concat), and reshape concatenation (RE Concat).

4.1 Experiment on non-rigid objects

Hand pose experiment: Since ground truth predictions for the real NYU hand dataset were not available, the LTNN model has been trained using a synthetic dataset generated using 3D mesh hand models. The NYU dataset does, however, provide ground truth coordinates for the input hand pose; using this we were able to indirectly evaluate the performance of the model by assessing the accuracy of a hand pose estimation method using the network's multi-view predictions as input.

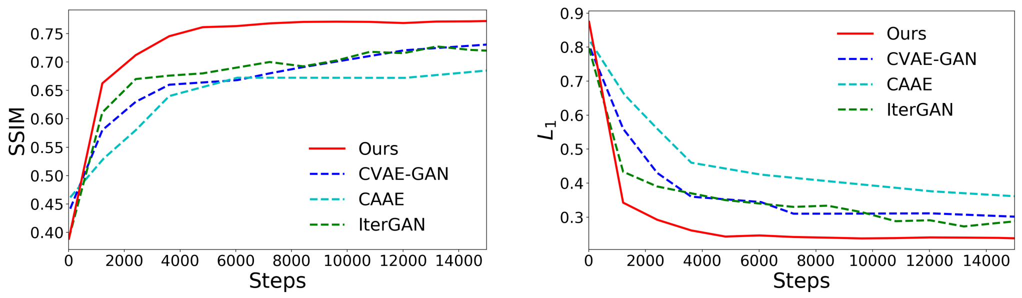

Figure 6: Quantitative comparison of model performances for experiment on the real face dataset.

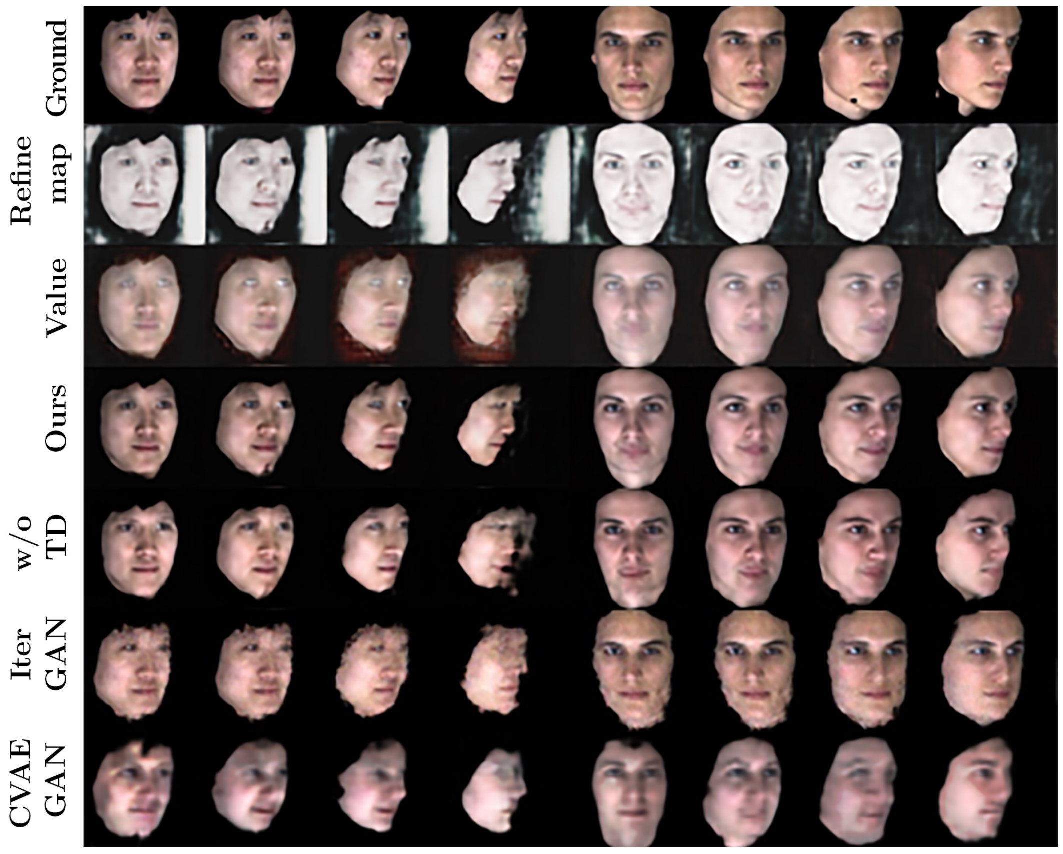

Figure 7: Qualitative evaluation for multi-view reconstruction of real face using the stereo face dataset (Fransens et al., 2005).

More specifically, the LTNN model was trained to generate 9 different views which were then fed into the pose estimation network from Choi et al. (2017) (also trained using the synthetic dataset).

A comparison of the quantitative hand pose estimation results is provided in Figure 5 where the proposed LTNN framework is seen to provide a substantial improvement over existing methods; qualitative results are also available in Figure 3. With regard to real-time applications, the proposed model runs at 114 fps without batching and at 1975 fps when applied to a mini-batch of size 128 (using a single TITAN Xp GPU and an Intel i7-6850K CPU).

Real face experiment: The stereo face database (Fransens et al., 2005), consisting of images of 100 individuals from 10 different viewpoints, was used for experiments with real faces; these faces were segmented using the method of Nirkin et al. (2018) and then cropped and centered to form the final dataset. The LTNN model was trained to synthesize images of input faces corresponding to three consecutive horizontal rotations. As shown in Figure 6, our method significantly outperforms the CVAE-GAN, CAAE, and IterGAN models in both the L_1 and SSIM metrics; qualitative results are also available in Figure 7 and Section A.6 of the appendix.

4.2 Experiment on rigid objects

Real object experiment: The ALOI dataset (Geusebroek et al., 2005), consisting of images of 1000 real objects viewed from 72 rotated angles (covering one full 360° rotation), has been used for experiments on real objects. As shown in Table 1 and in Figure 8, our method outperforms other state-of-the-art methods with respect to the L_1 metric and achieves comparable SSIM metric scores.

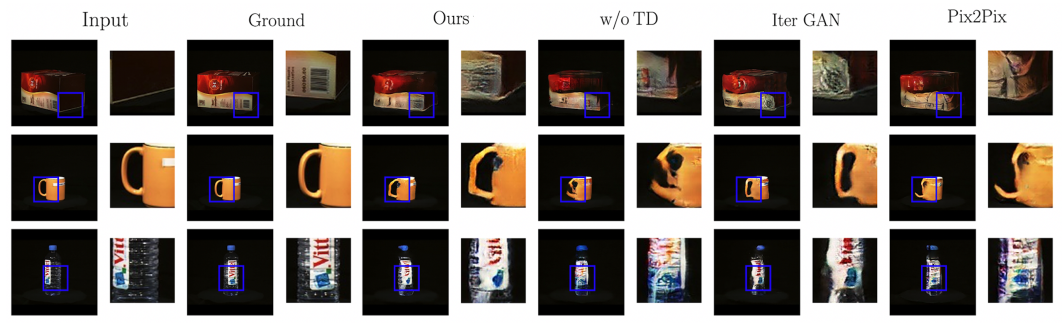

Figure 8: Unseen objects

Of note is the fact that the LTNN framework is capable of effectively performing the specified rigid-object transformations using only a single image as input, whereas most state-of-the-art view synthesis methods require additional information which is not practical to obtain for real datasets. For example, MV3D requires depth information and TVSN requires 3D models to render visibility maps for training which is not available in the ALOI dataset.

Model	L_1^M seen	L_1^M unseen	SSIM seen	SSIM unseen
Ours	**.138±.046**	**.221±.064**	**.927±.012**	.871±.031
IterGAN	.147±.055	.231±.094	.918±.019	**.875±.025**
Pix2Pix	.210±.092	.256±.100	.914±.021	.864±.041

Table 1: The result table for the experiment on the ALOI dataset.

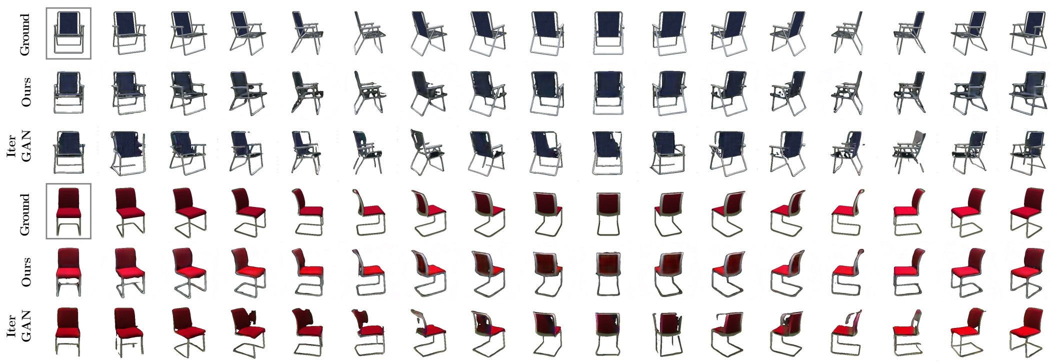

Figure 9: Generated 360° views for chair dataset. A single, gray-scale image of the chair at the far left (shown in box) is provided to the network as input.

3D chair experiment: We have tested our model's ability to perform 360° view estimation on the chairs and compared the results with the other state-of-the-art methods. The proposed model outperforms existing models specifically designed for the task of multi-view prediction and require the least FLOPs for inference compared with all other methods (see Table 2).

Model	SSIM	L_1	Parameters for train	Parameters for infer	GFLOPs / Image
Ours	**.912**	**.217**	65,438 K	**16,961 K**	**2.183**
IterGan	.865	.351	**59,951 K**	57,182 K	12.120
AFN	.891	.240	70,319 K	70,319 K	2.671
TVSN	.894	.230	60,224 K	57,327 K	2.860
MV3D	.895	.248	69,657 K	69,657 K	3.056

Table 2: Results for 3D chair 360° view synthesis. The proposed method uses significantly less parameters during inference, requires the least FLOPs, and yields the fastest inference times. FLOP calculations correspond to inference for a single image with resolution $256\times256\times3$.

4.3 Diverse attribute exploration with synthetic face data

To evaluate the proposed framework's performance on a more diverse range of attribute modification tasks, a synthetic face dataset and five conditional generative models with comparable encoder/decoder structures to the LTNN model have been selected for comparison. These models have been trained to synthesize discrete changes in elevation, azimuth, light direction, and age from a single grayscale image; results are shown in Table 3 and ablation results are available in Table 4. Near continuous attribute modification is also possible within the proposed framework, and distinct CTU mappings can be composed with one another to synthesize multiple modifications simultaneously; more details and related figures are provided in sections A.7.4 and A.7.5 of the appendix.

	Elevation		Azimuth		Light Direction		Age	
Model	SSIM	L_1	SSIM	L_1	SSIM	L_1	SSIM	L_1
Ours	**.923**	**.107**	**.923**	**.108**	**.941**	**.093**	**.925**	**.102**
CVAE-GAN	.864	.158	.863	.180	.824	.209	.848	.166
CVAE	.799	.166	.812	.157	.806	.209	.795	.173
CAAE	.777	.175	.521	.338	.856	.270	.751	.207
AAE	.748	.184	.520	.335	.850	.271	.737	.209

Table 3: Results for simultaneous colorization and attribute modification on synthetic face dataset.

5 Conclusion

In this work, we have introduced an effective, general framework for incorporating conditioning information into inference-based generative models. We have proposed a modular approach to incorporating conditioning information using CTUs and a consistency loss term, defined an efficient task-divided decoder setup for deconstructing the data generation process into managable subtasks, and shown that a context-aware discriminator can be used to improve the performance of the adversarial training process. The performance of this framework has been assessed on a diverse range of tasks and shown to outperform state-of-the-art methods.

REFERENCES

Jianmin Bao, Dong Chen, Fang Wen, Houqiang Li, and Gang Hua. Cvae-gan: Fine-grained image generation through asymmetric training. *arXiv preprint arXiv:1703.10155*, 2017.

Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. ShapeNet: An Information-Rich 3D Model Repository. Technical Report arXiv:1512.03012 [cs.GR], Stanford University — Princeton University — Toyota Technological Institute at Chicago, 2015a.

AX Chang, T Funkhouser, L Guibas, P Hanrahan, Q Huang, Z Li, S Savarese, M Savva, S Song, H Su, et al. An information-rich 3d model repository. arxiv preprint. *arXiv preprint arXiv:1512.03012*, 1(7):8, 2015b.

Chiho Choi, Sangpil Kim, and Karthik Ramani. Learning hand articulations by hallucinating heat distribution. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3104–3113, 2017.

Alexey Dosovitskiy, Jost Tobias Springenberg, and Thomas Brox. Learning to generate chairs with convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1538–1546, 2015.

Rik Fransens, Christoph Strecha, and Luc Van Gool. Parametric stereo for multi-pose face recognition and 3d-face modeling. In *International Workshop on Analysis and Modeling of Faces and Gestures*, pp. 109–124. Springer, 2005.

Ysbrand Galama and Thomas Mensink. Iterative gans for rotating visual objects. 2018.

Jan-Mark Geusebroek, Gertjan J Burghouts, and Arnold WM Smeulders. The amsterdam library of object images. *International Journal of Computer Vision*, 61(1):103–112, 2005.

Ian J. Goodfellow. NIPS 2016 tutorial: Generative adversarial networks. *CoRR*, abs/1701.00160, 2017. URL `http://arxiv.org/abs/1701.00160`.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034, 2015.

Gao Huang, Zhuang Liu, Kilian Q Weinberger, and Laurens van der Maaten. Densely connected convolutional networks. *arXiv preprint arXiv:1608.06993*, 2016.

A 3D Face Model for Pose and Illumination Invariant Face Recognition, Genova, Italy, 2009. IEEE.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pp. 448–456, 2015.

Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. *arXiv preprint*, 2017.

J Yu Jason, Adam W Harley, and Konstantinos G Derpanis. Back to basics: Unsupervised learning of optical flow via brightness constancy and motion smoothness. In *Computer Vision–ECCV 2016 Workshops*, pp. 3–10. Springer, 2016.

Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Tejas D Kulkarni, William F Whitney, Pushmeet Kohli, and Josh Tenenbaum. Deep convolutional inverse graphics network. In *Advances in Neural Information Processing Systems*, pp. 2539–2547, 2015.

Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, and Brendan Frey. Adversarial autoencoders. *arXiv preprint arXiv:1511.05644*, 2015.

Michael Mathieu, Camille Couprie, and Yann LeCun. Deep multi-scale video prediction beyond mean square error. *arXiv preprint arXiv:1511.05440*, 2015.

Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

Takeru Miyato and Masanori Koyama. cgans with projection discriminator. *arXiv preprint arXiv:1802.05637*, 2018.

Yuval Nirkin, Iacopo Masi, Anh Tran Tuan, Tal Hassner, and Gerard Medioni. On face segmentation, face swapping, and face perception. In *Automatic Face & Gesture Recognition (FG 2018), 2018 13th IEEE International Conference on*, pp. 98–105. IEEE, 2018.

Eunbyung Park, Jimei Yang, Ersin Yumer, Duygu Ceylan, and Alexander C Berg. Transformation-grounded image generation network for novel 3d view synthesis. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 702–711. IEEE, 2017.

Adam Paszke, Abhishek Chaurasia, Sangpil Kim, and Eugenio Culurciello. Enet: A deep neural network architecture for real-time semantic segmentation. *arXiv preprint arXiv:1606.02147*, 2016.

Pascal Paysan, Reinhard Knothe, Brian Amberg, Sami Romdhani, and Thomas Vetter. A 3d face model for pose and illumination invariant face recognition. In *Advanced Video and Signal Based Surveillance, 2009. AVSS'09. Sixth IEEE International Conference on*, pp. 296–301. Ieee, 2009.

Prajit Ramachandran, Barret Zoph, and Quoc V Le. Swish: a self-gated activation function. *arXiv preprint arXiv:1710.05941*, 2017.

Scott Reed, Kihyuk Sohn, Yuting Zhang, and Honglak Lee. Learning to disentangle factors of variation with manifold interaction. In *International Conference on Machine Learning*, pp. 1431–1439, 2014.

Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *Advances in Neural Information Processing Systems*, pp. 2234–2242, 2016.

Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. In *Advances in Neural Information Processing Systems*, pp. 3483–3491, 2015.

Xiao Sun, Yichen Wei, Shuang Liang, Xiaoou Tang, and Jian Sun. Cascaded hand pose regression. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 824–832, 2015.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9, 2015.

Maxim Tatarchenko, Alexey Dosovitskiy, and Thomas Brox. Multi-view 3d models from single images with a convolutional network. In *European Conference on Computer Vision*, pp. 322–337. Springer, 2016.

Jonathan Tompson, Murphy Stein, Yann Lecun, and Ken Perlin. Real-time continuous pose recovery of human hands using convolutional networks. *ACM Transactions on Graphics (ToG)*, 33(5):169, 2014.

Xinchen Yan, Jimei Yang, Kihyuk Sohn, and Honglak Lee. Attribute2image: Conditional image generation from visual attributes. In *European Conference on Computer Vision*, pp. 776–791. Springer, 2016.

Zhifei Zhang, Yang Song, and Hairong Qi. Age progression/regression by conditional adversarial autoencoder. *arXiv preprint arXiv:1702.08423*, 2017.

Tinghui Zhou, Shubham Tulsiani, Weilun Sun, Jitendra Malik, and Alexei A Efros. View synthesis by appearance flow. In *European Conference on Computer Vision*, pp. 286–301. Springer, 2016.

APPENDIX

A.1 ARCHITECTURE DETAILS

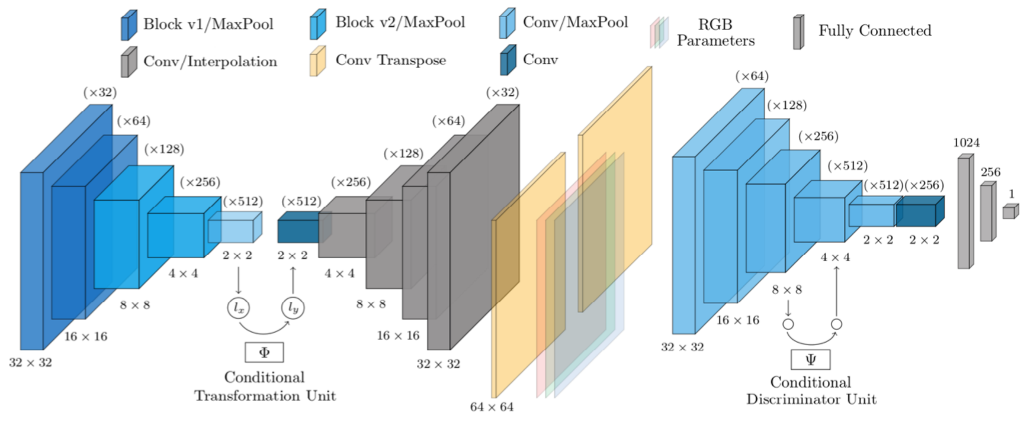

Figure A.1: The proposed network structure for the encoder/decoder (left) and discriminator (right). Features have been color-coded according to the type of layer which has produced them.

Input images of resolution 64×64 are passed through a Block v1 collaborative filter layer (see Figure A.2) along with a max pooling layer to produce the 32×32 features at the far left end of the figure. At the bottle-neck between the encoder and decoder, a conditional transformation unit (CTU) is applied to map the 2×2 latent features directly to the transformed 2×2 latent features on the right. This CTU is implemented as a convolutional layer with filter weights selected based on the conditioning information provided to the network. The noise vector $z \in \mathbb{R}^4$ from normal distribution $N(0, 1)$ is concatenated to the transformed 2×2 features and passed to the decoder for the face attributes task only. The 32×32 features near the end of the decoder component are processed by two independent convolution transpose layers: one corresponding to the value estimation map and the other corresponding to the refinement map. The channels of the value estimation map are rescaled by the RGB balance parameters, and the Hadamard product is taken with the refinement map to produce the final network output. For the ALOI data experiment, we have followed the IterGAN Galama & Mensink (2018) encoder and decoder structure, and for the stereo face dataset Fransens et al. (2005) experiment, we have added an additional Block v1 layer in the encoder and decoder to utilize the full 128×128×3 resolution images.

The encoder incorporates two main block layers, as defined in Figure A.2, which are designed to provide efficient feature extraction; these blocks follow a similar design to that proposed by Szegedy et al. (2015), but include dense connections between blocks, as introduced by Huang et al. (2016). We normalize the output of each network layer using the batch normalization method as described in Ioffe & Szegedy (2015). For the decoder, we have opted for a minimalist design, inspired by the work of Paszke et al. (2016). Standard convolutional layers with 3×3 filters and same padding are used through the penultimate decoding layer, and transpose convolutional layers with 5×5 filters and same padding are used to produce the value-estimation and refinement maps. All parameters have been initialized using the variance scaling initialization method described in He et al. (2015).

Our method has been implemented and developed using the TensorFlow framework. The models have been trained using stochastic gradient descent (SGD) and the ADAM optimizer Kingma & Ba (2014) with initial parameters: learning_rate = 0.005, β_1 = 0.9, and β_2 = 0.999 (as defined in the TensorFlow API r1.6 documentation for $\mathrm{tf.train.AdamOptimizer}$). , along with loss function hyperparameters: λ = 0.8, ρ = 0.2, γ = 0.0002, and κ = 0.00005 (as introduced in Equation 6). The discriminator is updated once every two encoder/decoder updates, and one-sided label smoothing (Salimans et al., 2016) has been used to improve stability of the discriminator training procedure. All datasets have also been normalized to the interval $[0, 1]$ for training.

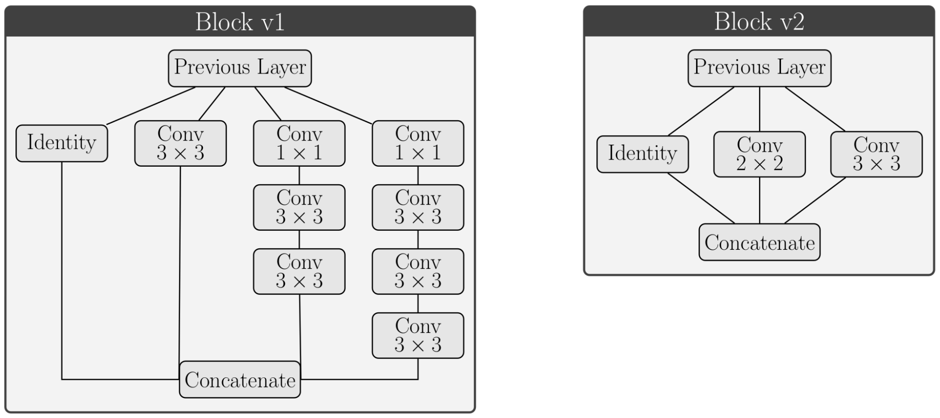

Figure A.2: Layer definitions for Block v1 and Block v2 collaborative filters. Once the total number of output channels, N_{out}, is specified, the remaining $N_{\text{out}} - N_{\text{in}}$ output channels are allocated to the non-identity filters (where N_{in} denotes the number of input channels). For the Block v1 layer at the start of the proposed LTNN model, for example, the input is a single grayscale image with $N_{\text{in}} = 1$ channel and the specified number of output channels is $N_{\text{out}} = 32$. One of the 32 channels is accounted for by the identity component, and the remaining 31 channels are the three non-identity filters. When the remaining channel count is not divisible by 3 we allocate the remainder of the output channels to the single 3×3 convolutional layer. Swish activation functions are used for each filter, however the filters with multiple convolutional layers (i.e. the right two filters in the Block v1 diagram) do not use activation functions for the intermediate 3×3 convolutional layers (i.e. those after the 1×1 layers and before the final 3×3 layers).

A.2 Hand pose experiment

A.2.1 Dataset

A kinematic hand model with 33 degrees of freedom has been used to generate 200,000 distinct hand poses with nine depth images from different viewpoints for each pose. We sampled hand pose uniformly from each of the 18 joint angle parameters, covering a full range of hand articulations. The nine viewpoints are centered around a designated input view and correspond to 30° changes in the spherical coordinates of the viewer (i.e. 30° up, 30° right, 30° up and 30° right, etc.). Testing was performed on the MSRA (Sun et al., 2015) and NYU (Tompson et al., 2014) hand datasets.

A.2.2 Hand pose estimation process

We follow the training procedure proposed by Choi et al. (2017) for the hand pose estimation network; after training, the LTNN multi-view predictions are fed into the network and the accuracy of the predicted angles is used to assess how well these network predictions approximate the unknown, true views. As seen in Figure A.3, the optimal results are obtained when all 9 synthesized views points are fed into the pose estimation network.

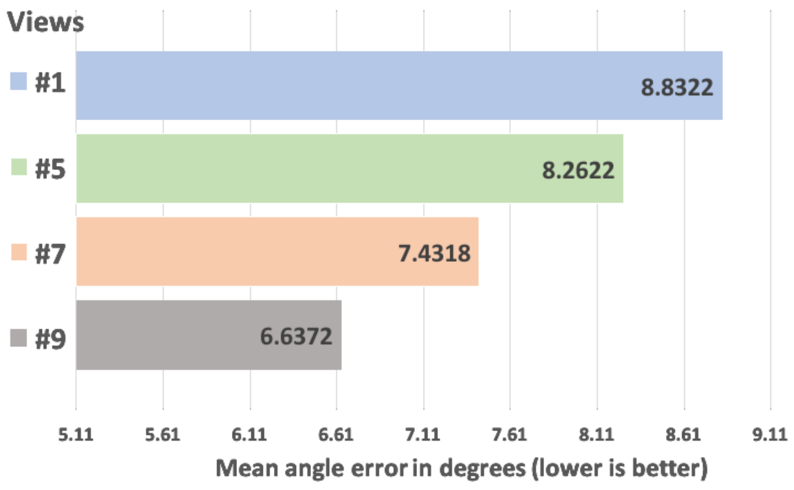

Figure A.3: Indirect quantitative evaluation for synthetic hand multi-view reconstruction. The mean angle error between the ground truth and predictions from pose estimation network from Choi et al. (2017) using LTNN multi-view predictions as input. The error is seen to clearly drop as the number of LTNN predicted views increases, indicating that the synthesized LTNN predictions accurately reflect the true target views.

A.2.3 Hand pose multi-view examples

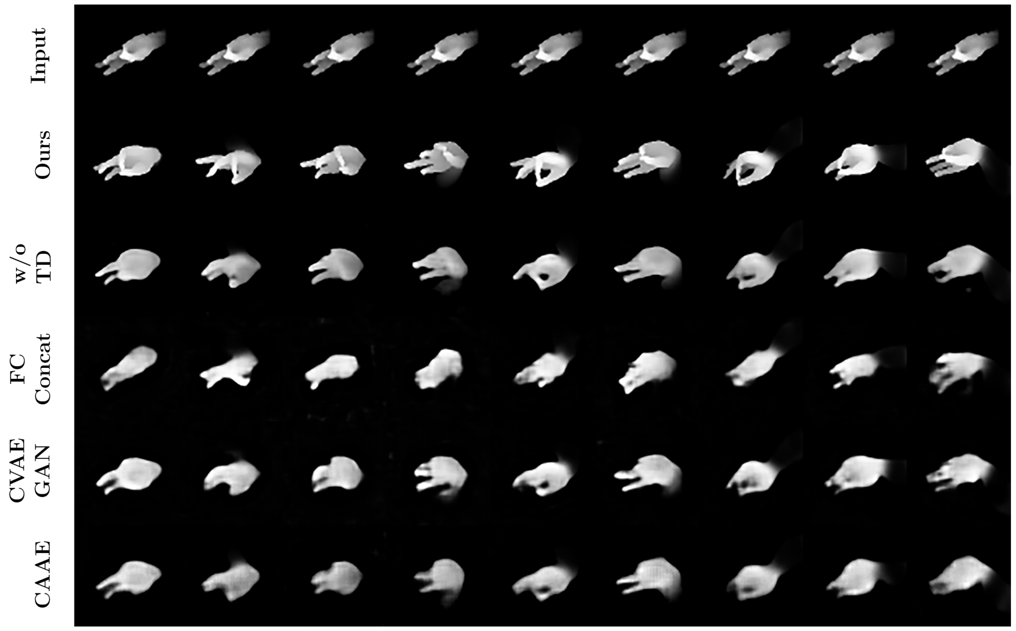

Figure A.4: Qualitative evaluation for multi-view reconstruction of hand depth maps using the NYU dataset.

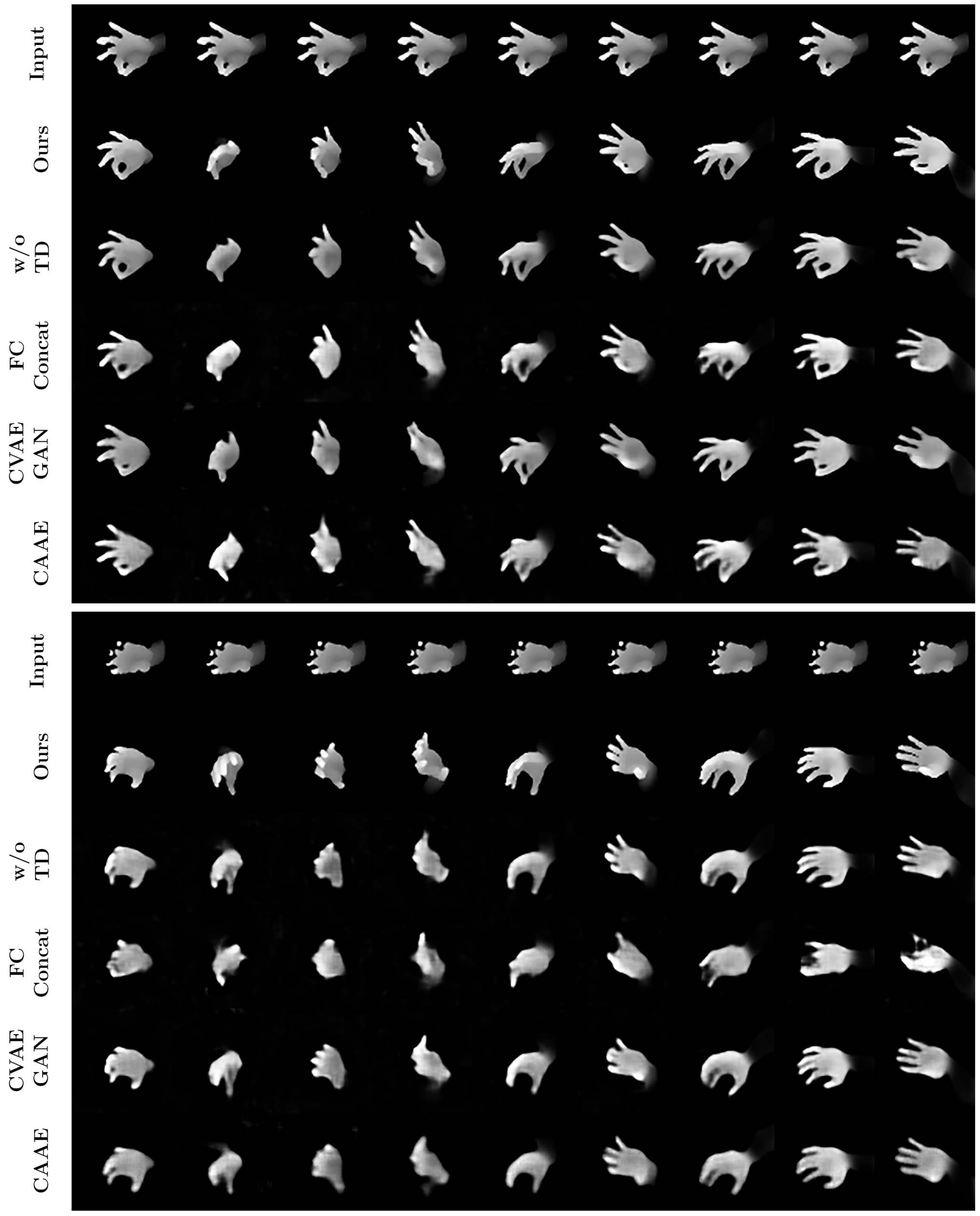

Figure A.5: Qualitative evaluation for multi-view reconstruction of hand depth maps using the NYU dataset.

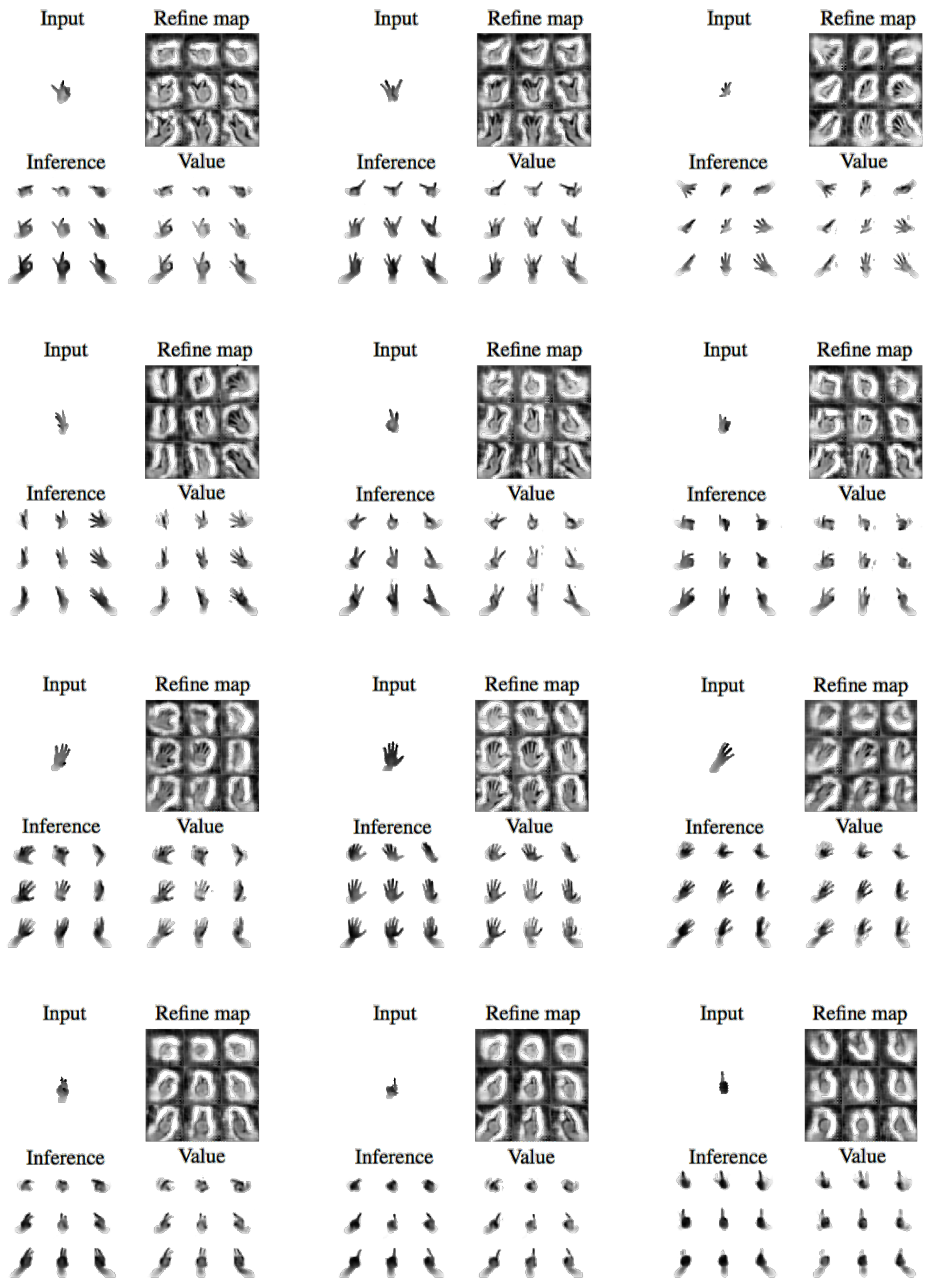

Figure A.6: Multi-view estimation applied to real hand depth images. First and second rows of results show proposed network predictions applied to the MSRA15 dataset. Third and fourth rows of results show proposed network predictions applied to the NYU dataset. The refine map works as a shape mask which is applied to the value-estimation map to produce the final, refined prediction.

A.3 Real object ALOI experiment

A.4 Dataset

Following the experimental design from Galama & Mensink (2018), we have used images of resolution $256 \times 256 \times 3$ from the ALOI database for training and testing. While the LTNN model is capable of making simultaneous predictions of multiple viewpoints, as illustrated in the hand and chair experiments, the Pix2Pix (Isola et al., 2017) and IterGAN (Galama & Mensink, 2018) networks are designed to produce a single synthesized view. To make fair comparisons between these existing networks and the proposed LTNN model, each model has been trained only to produce a single 30° rotated view of the ALOI objects. In particular, only two CTU mappings were trained: one corresponding to the identity, and one corresponding to the rotated view.

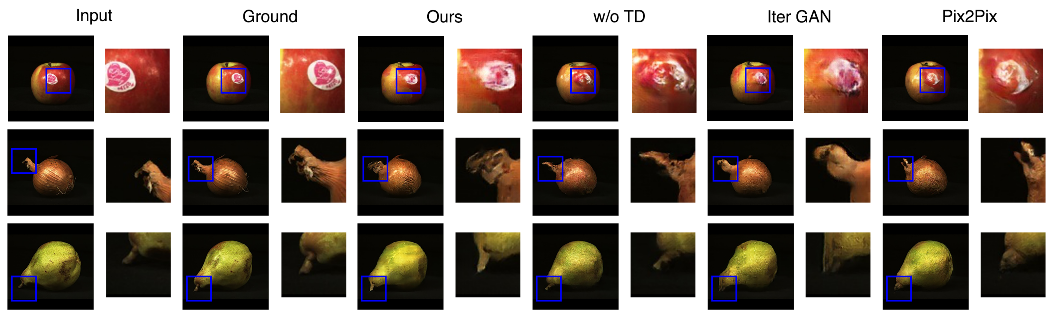

Figure A.7: Seen objects

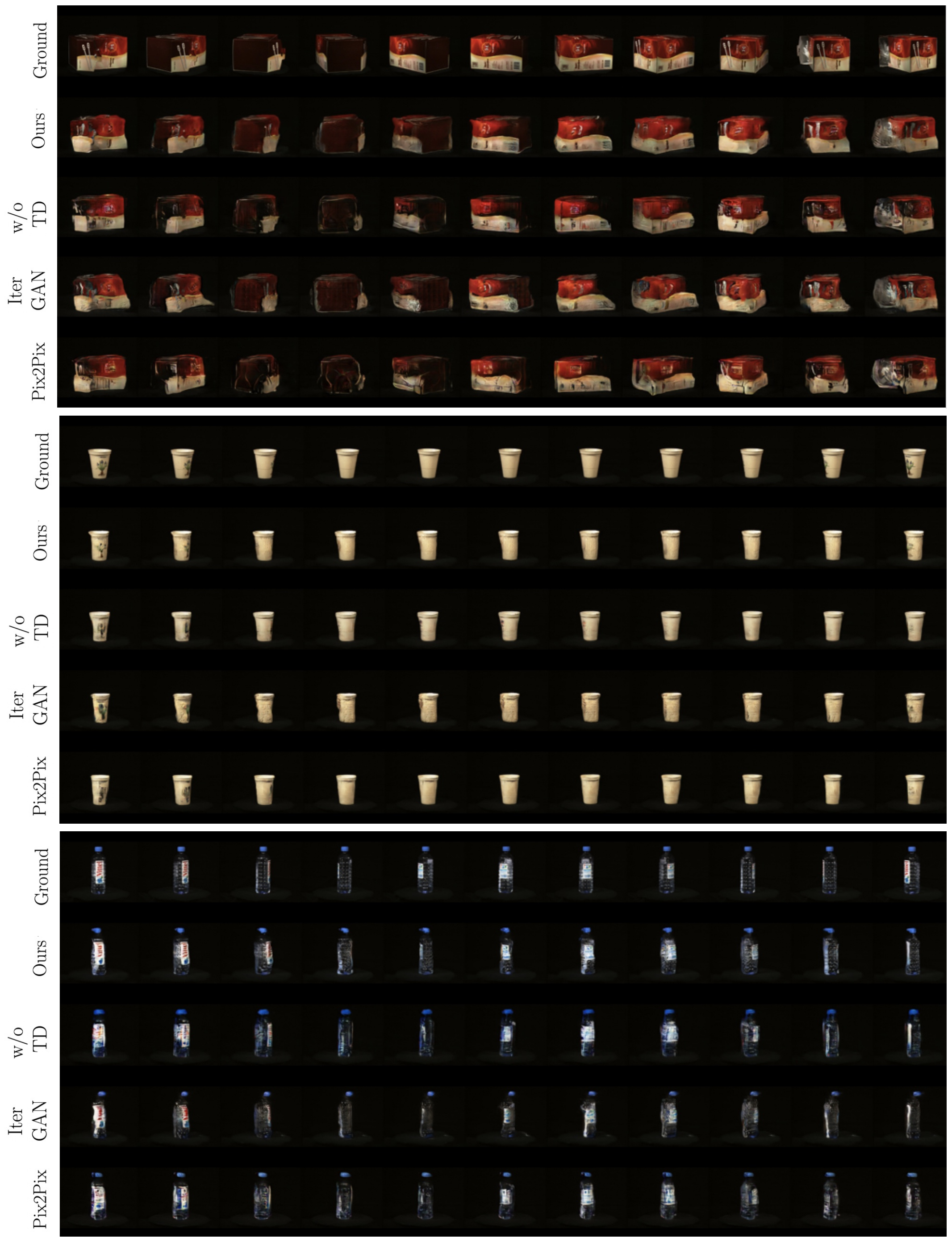

Figure A.8: Experiment on unseen objects from the ALOI real object dataset. First row of is ground truth, second row is ours, third row is ours without task-division, fourth row is IterGAN, and bottom row is Pix2Pix. As shown from the figure, our methods are sharper and realistic than other methods in the majority of generated views.

A.5 Synthetic 3D chair experiment

Chairs from the ShapeNet (Chang et al., 2015b) 3D model repository have been rotated horizontally 20° 17 times and vertically 10° 3 times; 6742 chairs have been selected following the data creation methodology from Park et al. (2017).

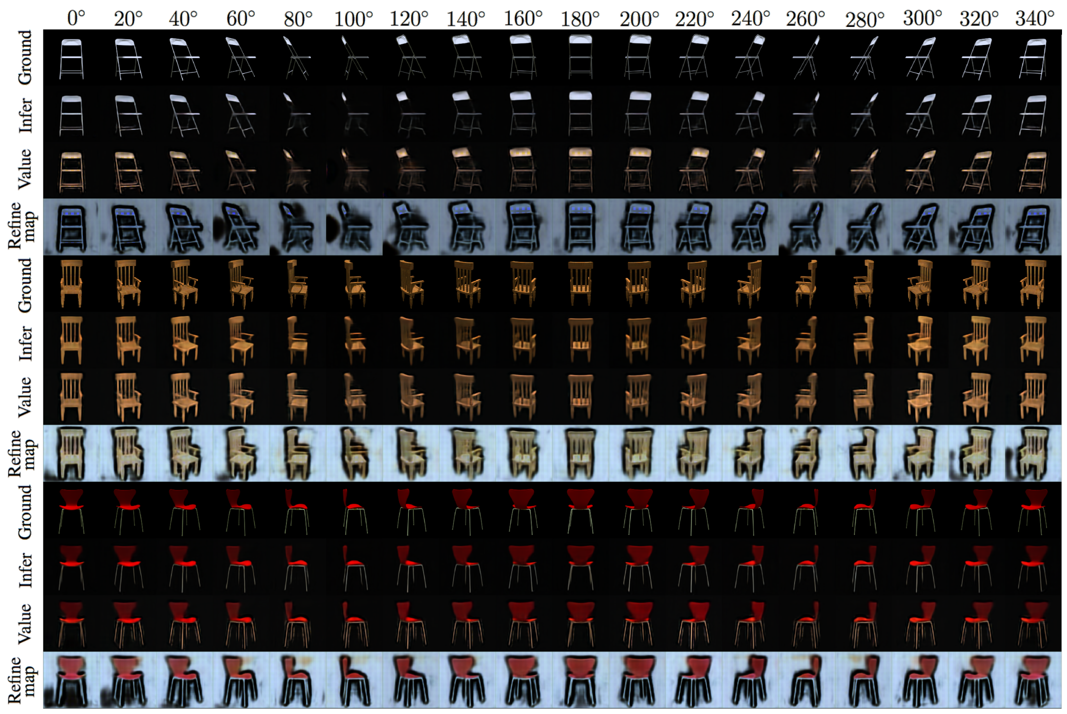

Figure A.9: Task division for chair dataset. First row shows ground truth, second row shows network prediction, third row shows value estimation map, and fourth row shows refinement map.

A.6 Real Face experiment

A.6.1 Dataset

The original dataset has background and 100 identification and 5 different views from 2 distinct cameras, which results 1000 face images in total. Since the dataset is not huge, we would like to reduce background noise and perform face segmentation with Nirkin et al. (2018) and manually filtered badly segmented faces from the background with the segmentation method and create $300 \times 300 \times 3$ face images. For training, we resize the original images into $128 \times 128 \times 3$ resolution.

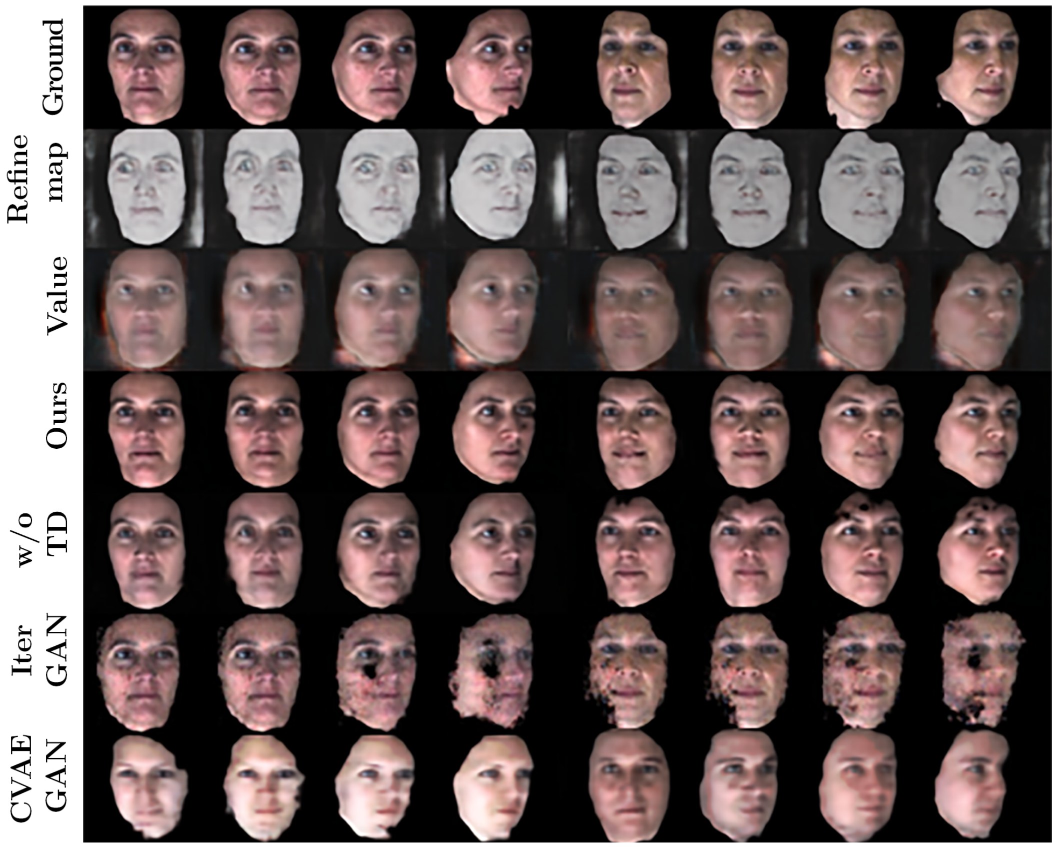

Figure A.10: Real face dataset qualitative evaluation.

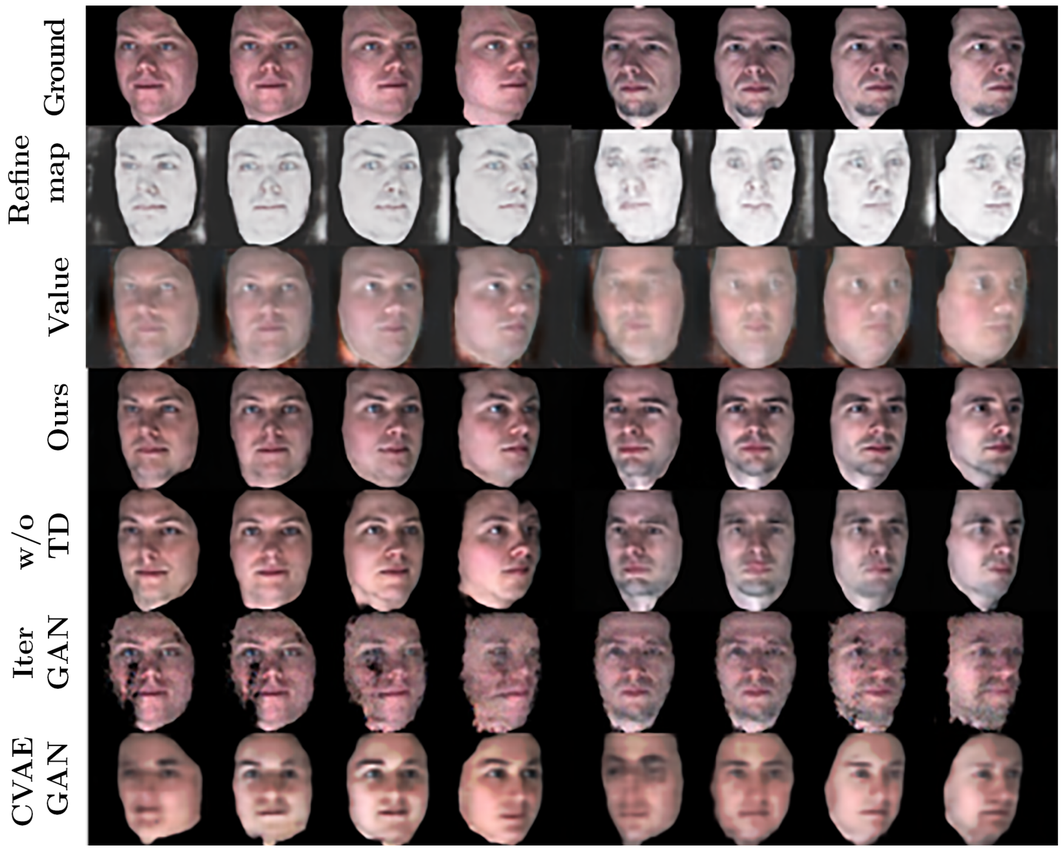

Figure A.11: Real face dataset qualitative evaluation.

A.7 Synthetic Face Experiment

A.7.1 Dataset

To generate realistic human faces, we have used the simulator created by Paysan et al. (2009) to create 4488 distinct faces.

Each face has been rendered at four distinct age ranges, and four different lighting directions have been used. The orientation of faces is allowed to vary in elevation from $-20°$ to $29°$ by increments of $7°$ and in azimuth from $10°$ to $150°$ by increments of $20°$. To demonstrate the model's colorization capabilities, the input images have been converted to gray-scale using the luminosity method.

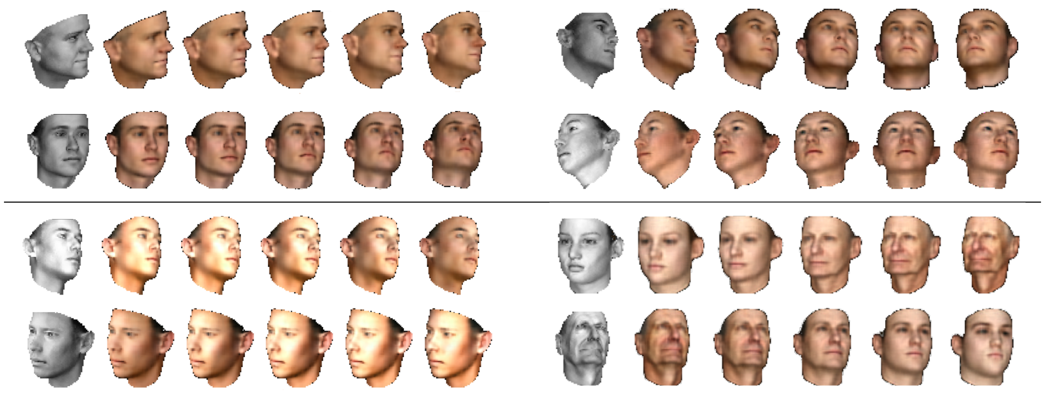

Figure A.12: Examples of attribute-modified faces generated by the LTNN model. A single gray-scale image is provided as input to the network (shown in gray) and changes in elevation (top-left), azimuth (top-right), light direction (bottom-left), and age (bottom-right) are produced.

A.7.2 Ablation study for proposed network components

Model	Elevation		Azimuth		Light Direction		Age	
	SSIM	L_1	SSIM	L_1	SSIM	L_1	SSIM	L_1
LTNN	**.923**	**.107**	**.923**	**.108**	**.941**	**.093**	**.925**	**.102**
CTU + CDU + TD	.917	.112	.922	.119	.938	.097	.913	.115
CTU + CDU	.901	.135	.908	.125	.921	.121	.868	.118
CTU	.889	.142	.878	.135	.901	.131	.831	.148
Channel Concat + Conv	.803	.179	.821	.173	.816	.182	.780	.188
2-FC + Concat + 2-FC	.674	.258	.499	.355	.779	.322	.686	.243
2-FC + Concat + FC	.691	.233	.506	.358	.787	.316	.687	.240
FC + Concat + 2-FC	.673	.261	.500	.360	.774	.346.	.683	.249
FC + Concat + FC	.681	.271	.497	.355	.785	.315.	.692	.246
Reshape + Concat + FC	.671	.276	.489	.357	.780	.318	.685	.251

Table 4: Ablation/comparison results using identical encoder, decoder, and training procedure.

Task-division: An overview of the task-division decoding procedure applied to the synthetic face dataset is provided in Figure A.13. As noted in Section 3.3, the refinement maps tend to learn to produce masks which closely resemble the target objects' shapes and have sharp drop-offs along the objects' boundaries. In addition to masking extraneous pixels, these refinement maps have been shown to apply local color balancing by, for example, filtering out the green and blue channels near lips when applied to human faces (i.e. the refinement maps for the green and blue channels show darker regions near the lips, thus allowing for the red channel to be expressed more distinctly).

The use of a task-divided decoder can also be seen to remove artifacts in the generated images in Figure A.13; e.g. removal of the blurred eyebrow (light), elimination of excess hair near the side of ear (azimuth), and reduction of the reddish vertical stripe on the forehead (age).

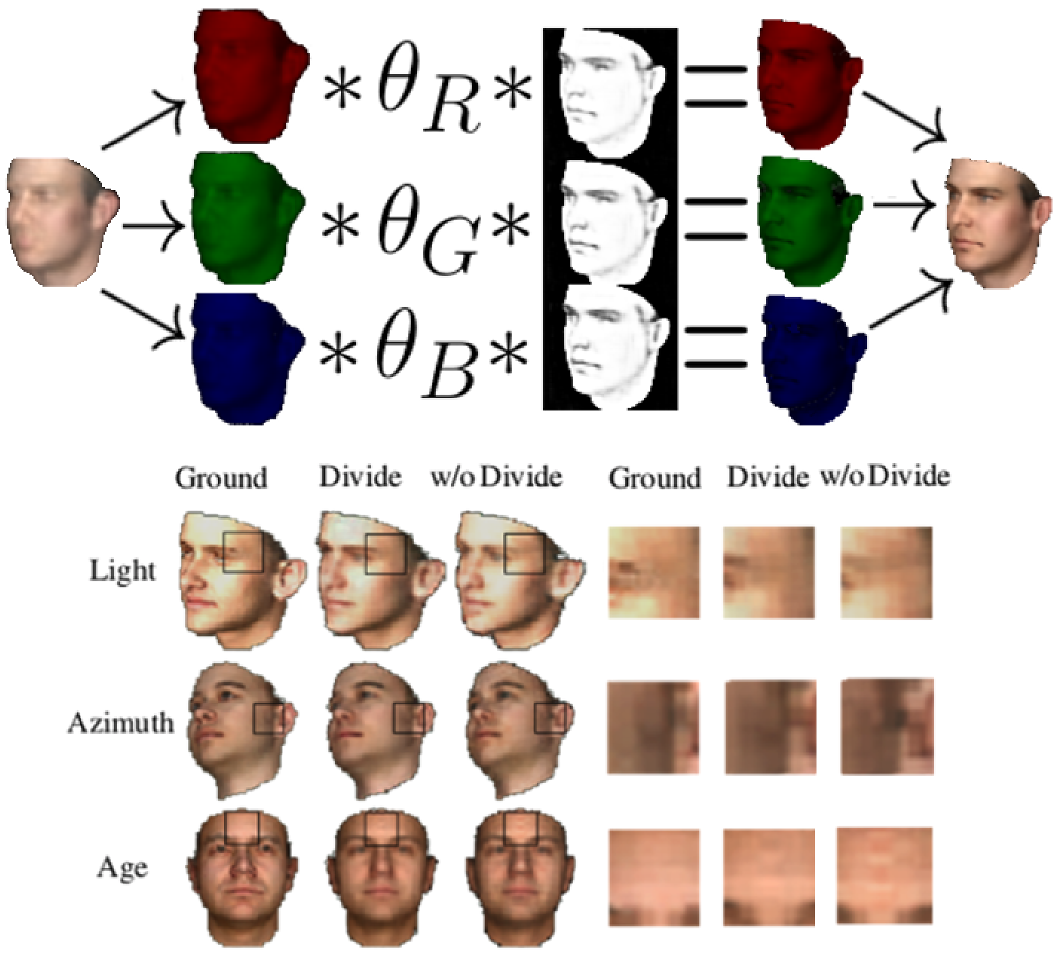

Figure A.13: Proposed task separation design for the LTNN decoder is shown at the top. The coarse pixel value estimation map is split into RGB channels, rescaled by the RGB balance parameters, and multiplied element-wise by the refinement map values to produce the final network prediction. On the bottom, qualitative comparisons between the ground truth and the proposed model's predictions with and without task separation are shown.

A.7.3 Qualitative Comparisons

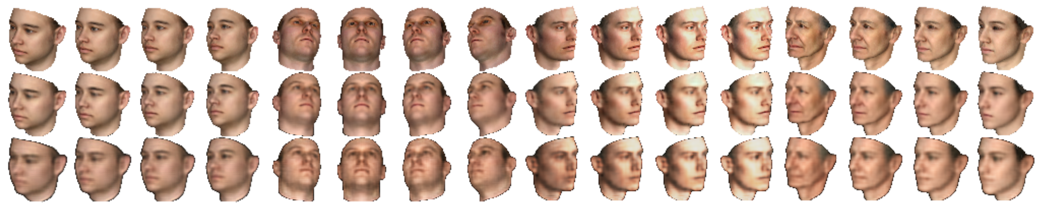

Figure A.14: Qualitative comparisons: top is ground truth, middle is ours, and bottom is CVAE-GAN.

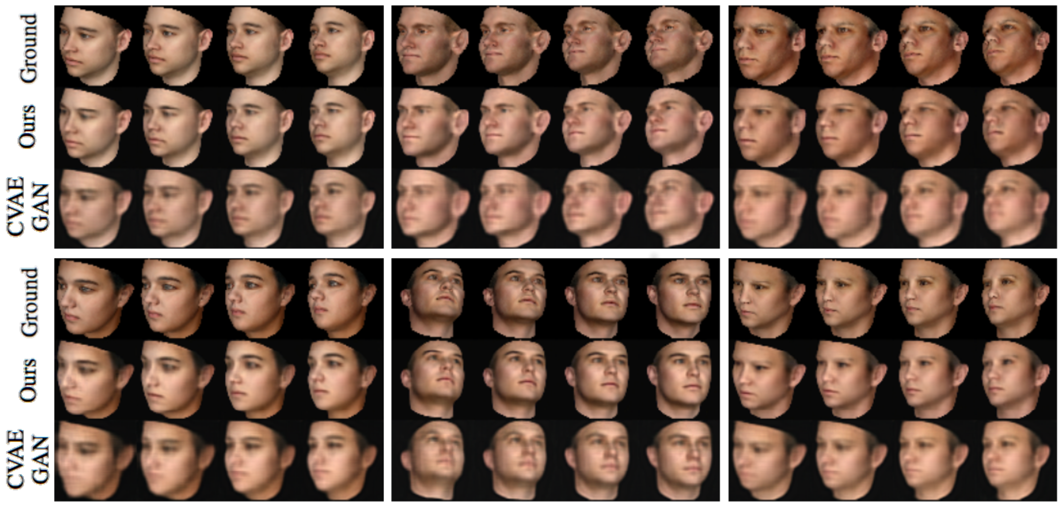

Figure A.15: Elevation changes: first row is the ground, second row is ours, third row is CVAE-GAN.

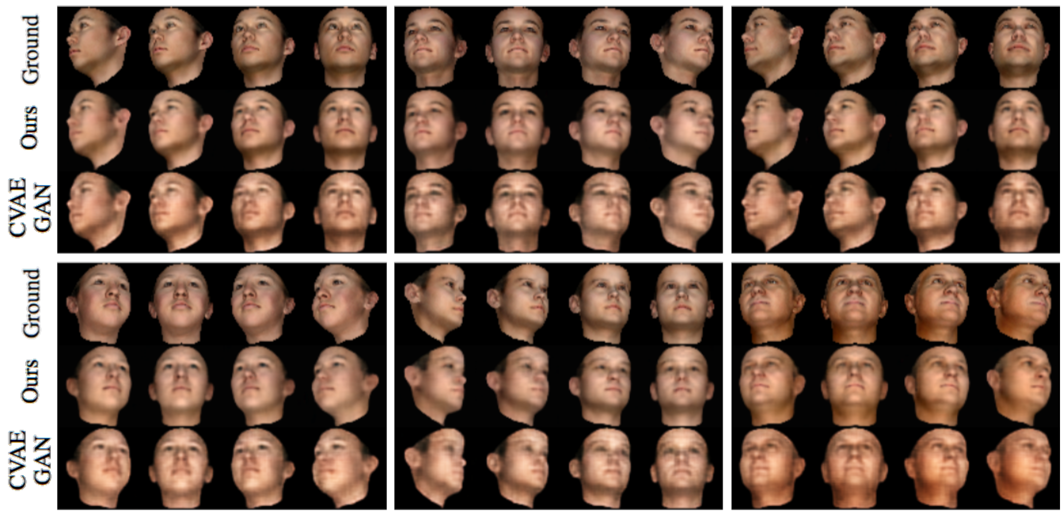

Figure A.16: Azimuth changes: first row is the ground, second row is ours, third row is CVAE-GAN.

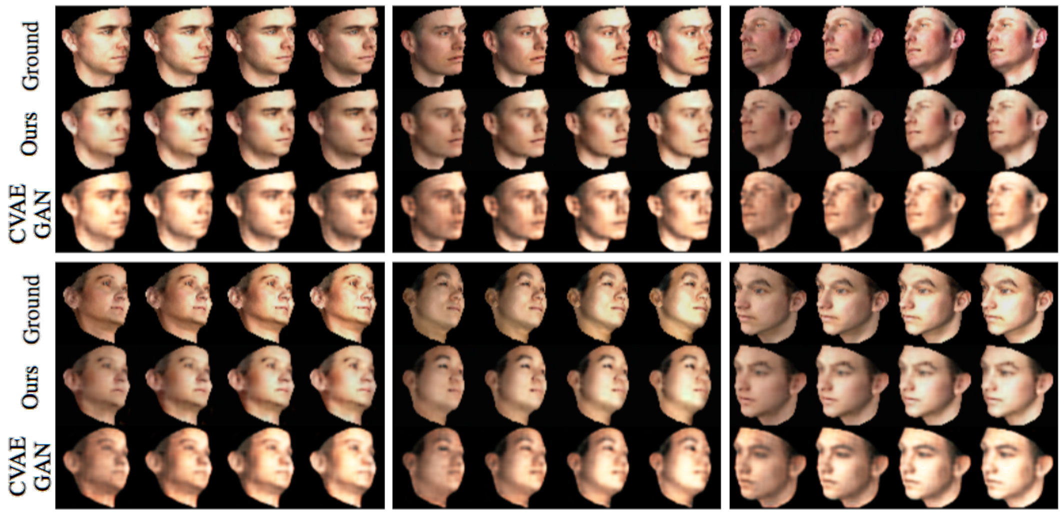

Figure A.17: Lighting changes: first row is the ground, second row is ours, third row is CVAE-GAN.

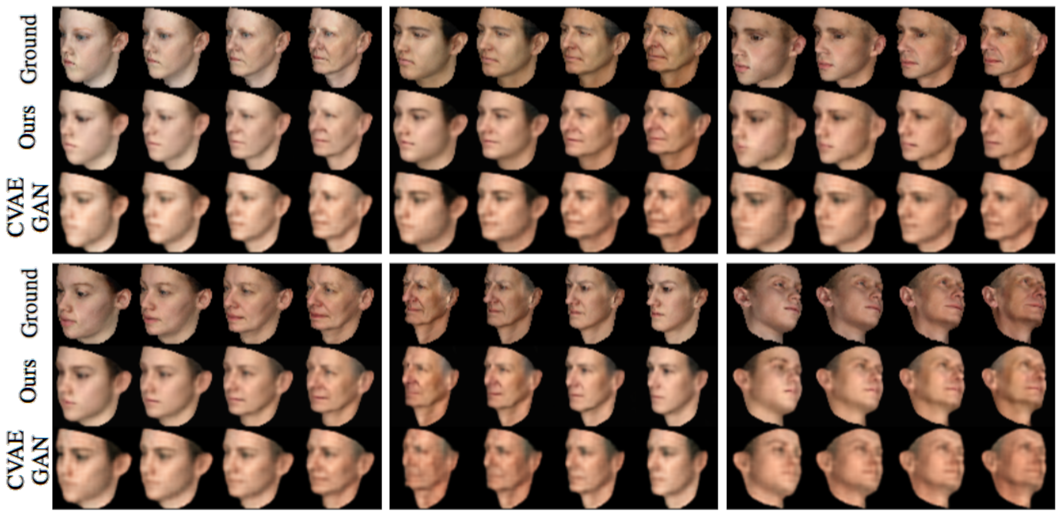

Figure A.18: Age changes: first row is the ground, second row is ours, third row is CVAE-GAN.

A.7.4 Near-Continuous Attribute Modification

As noted in Section 4.3, near-continuous attribute modification can be performed by piecewise-linear interpolation in the latent space. For example, we can train 9 CTU mappings $\{\Phi_k\}_{k=0}^{8}$ corresponding to discrete, incremental 7° changes in elevation $\{\theta_k\}$. In this setting, the network predictions for an elevation change of $\theta_0 = 0^\circ$ and $\theta_1 = 7^\circ$ are given by $\text{Decode}[\Phi_0(l_x)]$ and $\text{Decode}[\Phi_1(l_x)]$, respectively (where l_x denotes the encoding of the input image). To predict an elevation change of 3.5°, we can perform linear interpolation in the latent space between the representations $\Phi_0(l_x)$ and $\Phi_1(l_x)$; that is, we may take our network prediction for the intermediate change of 3.5° to be:

$$\widehat{y} = \text{Decode}[\,\widehat{l_y}\,] \quad \text{where} \quad \widehat{l_y} = 0.5 \cdot \Phi_0(l_x) + 0.5 \cdot \Phi_1(l_x)$$

Likewise, to approximate a change of 10.5° in elevation we may take $\text{Decode}[\widehat{l_y}]$, where $\widehat{l_y} = 0.5 \cdot \Phi_1(l_x) + 0.5 \cdot \Phi_2(l_x)$, as the network prediction. More generally, we can interpolate between the latent CTU map representations to predict a change θ via:

$$\widehat{y} = \text{Decode}[\,\widehat{l_y}\,] \quad \text{where} \quad \widehat{l_y} = \lambda \cdot \Phi_k(l_x) + (1-\lambda) \cdot \Phi_{k+1}(l_x)$$

with $k \in \{0, \dots, 7\}$ and $\lambda \in [0,1]$ chosen so that $\theta = \lambda \cdot \theta_k + (1-\lambda) \cdot \theta_{k+1}$. Accordingly, the proposed framework naturally allows for continuous attribute changes to be approximated by using this piecewise-linear latent space interpolation procedure.

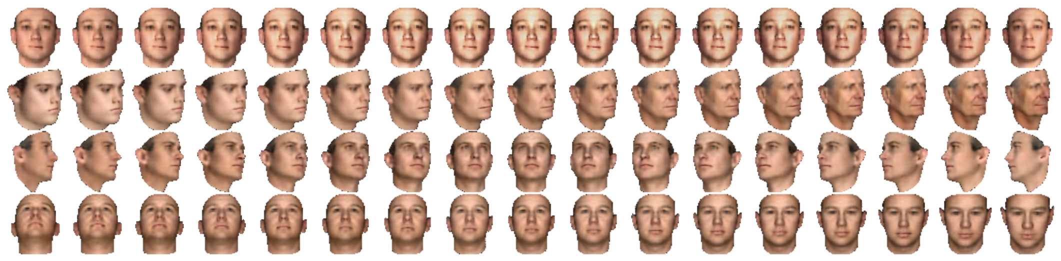

Figure A.19: Near continuous attribute modification is attainable using piecewise-linear interpolation in the latent space. Provided a gray-scale image (corresponding to the faces on the far left), modified images corresponding to changes in light direction (first), age (second), azimuth (third), and elevation (fourth) are produced with 17 degrees of variation. These attribute modified images have been produced using 9 CTU mappings, corresponding to varying degrees of modification, and linearly interpolating between the discrete transformation encodings in the latent space. Additional qualitative results for near-continuous attribute modification can be found in Section A.7.6.

A.7.5 Modification of Multiple Attributes

As noted in Section 4.3, multiple attributes can be modified simultaneously by composing CTU mappings. For example, we can train 4 CTU mappings $\{\Phi_k^{light}\}_{k=0}^3$ corresponding to incremental changes in lighting and 4 CTU mappings $\{\Phi_k^{azim}\}_{k=0}^3$ corresponding to incremental changes in azimuth. In this setting, the network predictions for lighting and azimuth changes are given by $\text{Decode}[\Phi_k^{light}(l_x)]$ and $\text{Decode}[\Phi_k^{azim}(l_x)]$, respectively (where $l_x = \text{Encode}[x]$ denotes the encoding of the original input image x).

To predict the effect of simultaneously changing both lighting and azimuth, we can compose the associated CTU mappings in the latent space; that is, we take our network prediction for the lighting change associated with Φ_i^{light} combined with the azimuth change associated with Φ_j^{azim} to be:

$$\widehat{y} = \text{Decode}[\,\widehat{l}_y\,] \quad \text{where} \quad \widehat{l}_y = \Phi_i^{light} \circ \Phi_j^{azim}(l_x) = \Phi_i^{light}\big[\Phi_j^{azim}(l_x)\big]$$

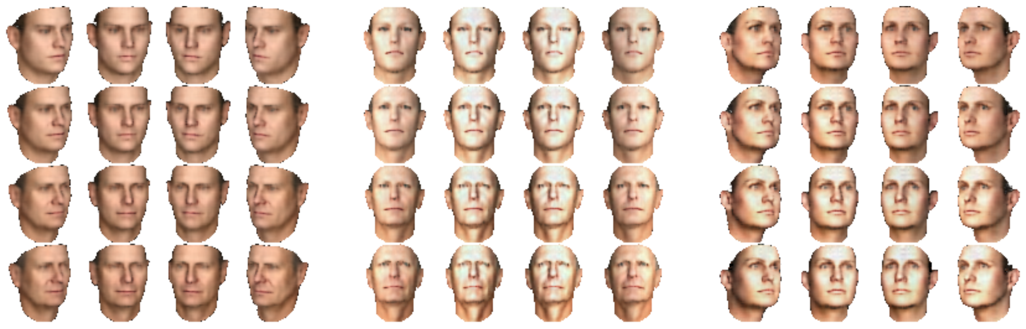

Figure A.20: Simultaneous learning of multiple attribute modifications. Azimuth and age (left), light and age (center), and light and azimuth (right) combined modifications are shown. The network has been trained using 4 CTU mappings per attribute (e.g. 4 azimuth mappings and 4 age mappings); results shown have been generated by composing CTU mappings in the latent space and decoding. Additional qualitative results multiple attribute modification can be found in Section A.7.7.

A.7.6 Near-Continuous Attribute Modification

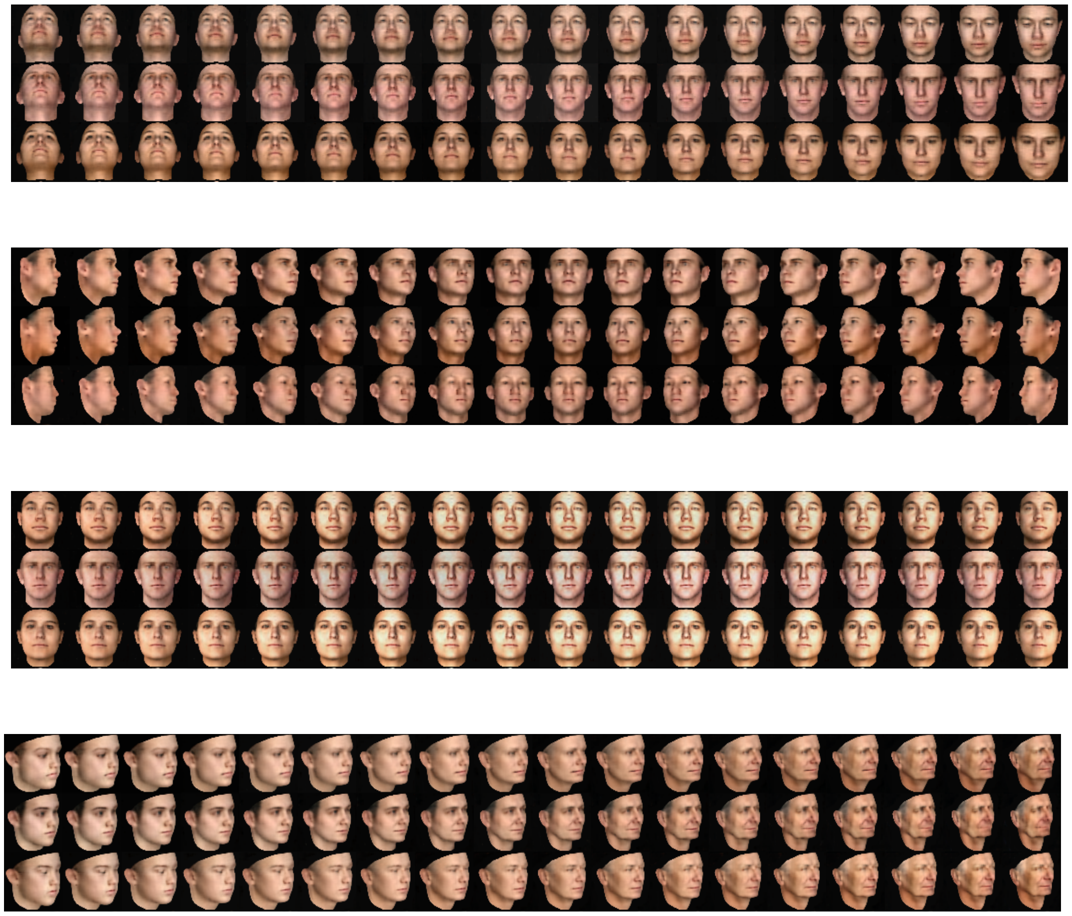

Figure A.21: Near-continuous elevation, azimuth, light, and age changes using LTNN model.

A.7.7 Modification of Multiple Attributes

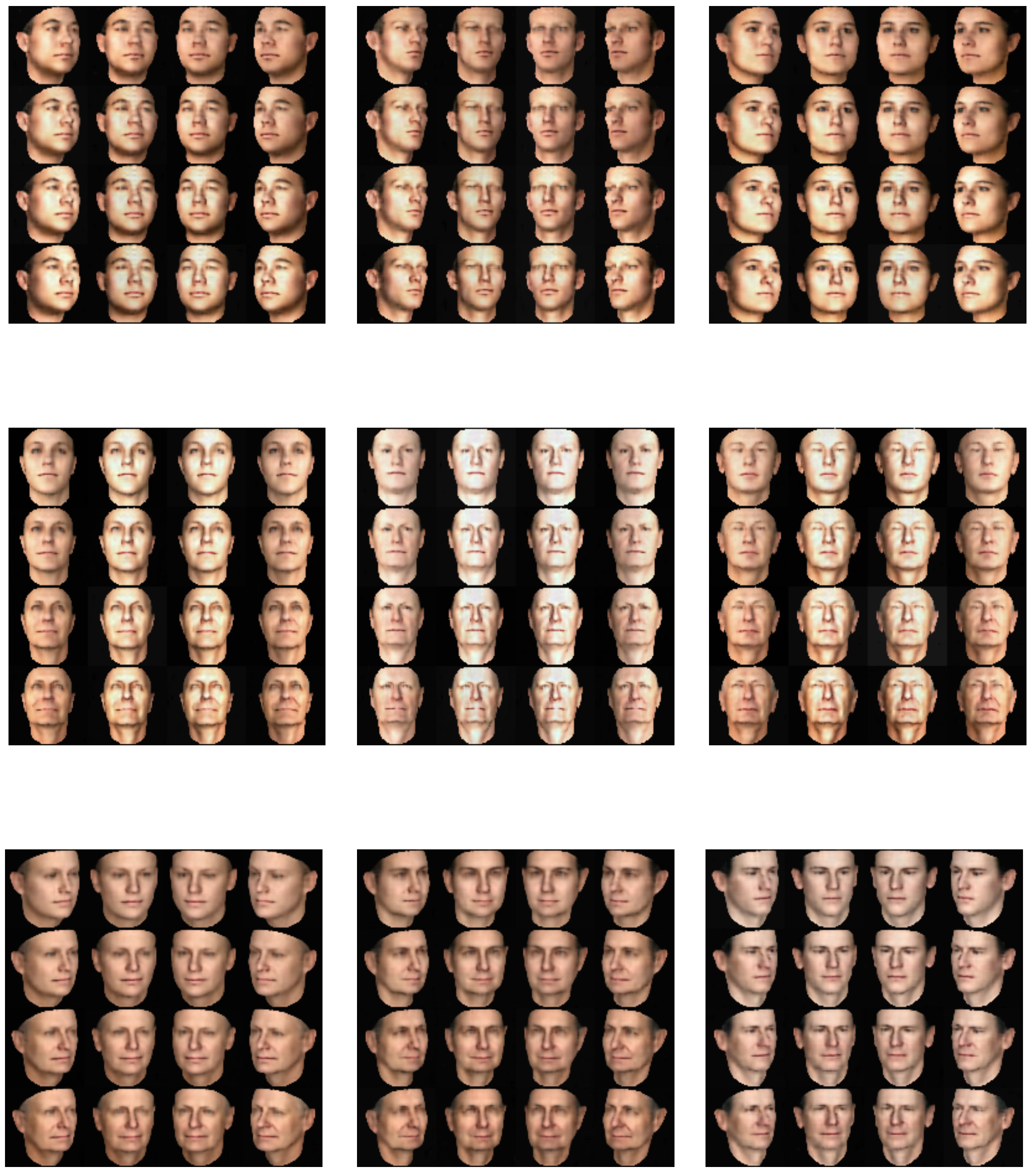

Figure A.22: Qualitative results for simultaneous modification of multiple attributes using the LTNN model. Top three figures show changes to ligh direction and azimuth, middle three figures show changes to light direction and age, and bottom three figures show changes to azimuth and age.

A.7.8 Value and Refinement Maps

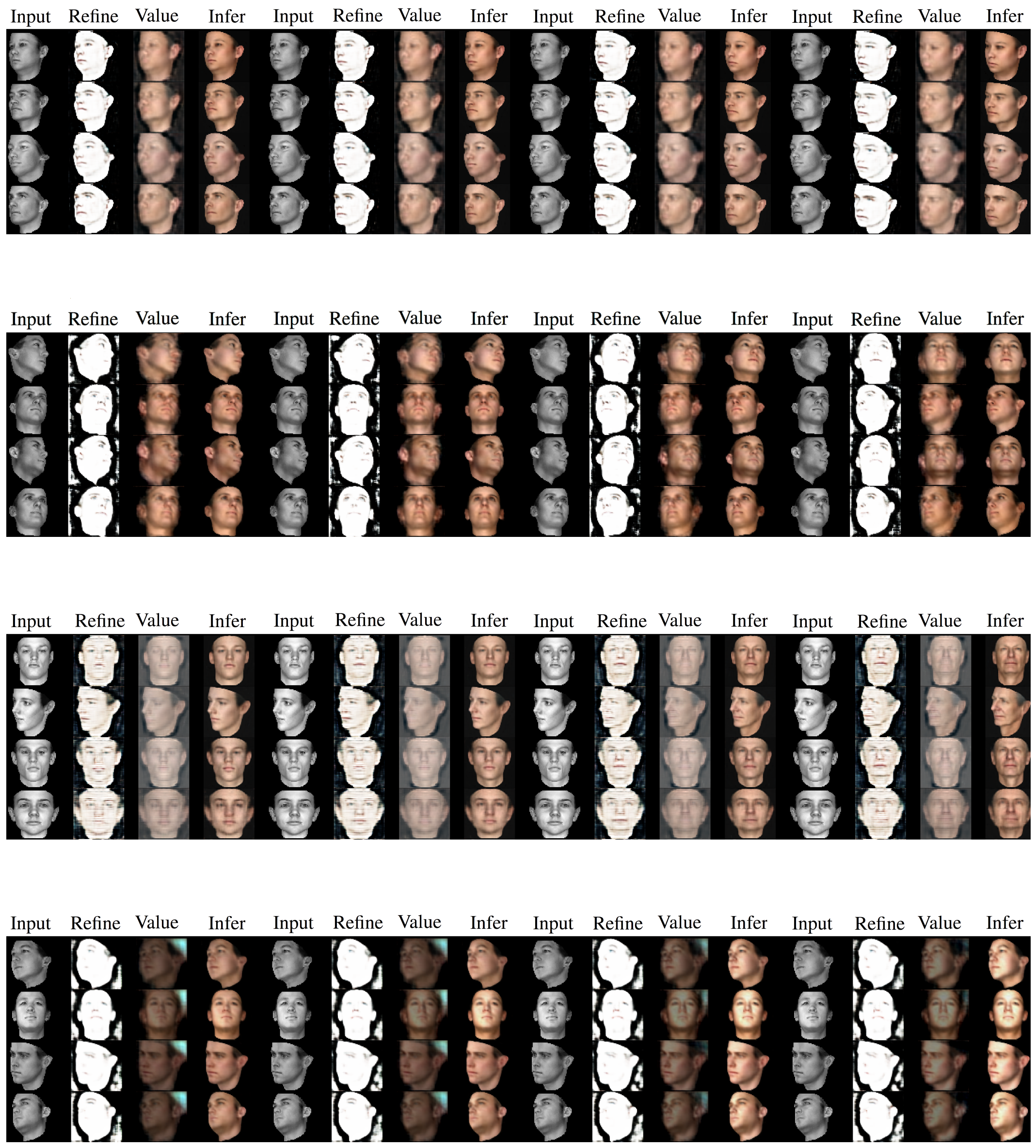
Input Refine Value Infer Input Refine Value Infer Input Refine Value Infer Input Refine Value Infer
Input Refine Value Infer Input Refine Value Infer Input Refine Value Infer Input Refine Value Infer
Input Refine Value Infer Input Refine Value Infer Input Refine Value Infer Input Refine Value Infer
Input Refine Value Infer Input Refine Value Infer Input Refine Value Infer Input Refine Value Infer

Figure A.23: Elevation, azimuth, age, and light refinement maps, value maps, and network predictions.