Noname manuscript No.
(will be inserted by the editor)

Latent Transformations Neural Network for Object View Synthesis

Sangpil Kim, · **Nick Winovich,** · **Hyun-Gun,** · **Guang Lin,** · **Karthik Ramani**

Received: date / Accepted: date

Abstract We propose a generative model, the latent transformation neural network(LTNN), capable of rigid and non-rigid object view synthesis using a light-weight neural network suited for real-time applications. In contrast to existing object view synthesis methods 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 the object 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 of objects, 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 be comparable with state-of-the-art methods in SSIM and L_1 metrics while simultaneously achieving a reduction in the computational demand and memory consumption for inference.

Keywords Object view synthesis · Latent transformation · Fully-convolutional · Conditional generative model

1 Introduction

Generative models have been shown to provide effective frameworks for representing complex, structured datasets and generating realistic samples from underlying data distributions [7]. 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 [18]. 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 [17]. The resulting inference models allow for the defining structure/features of inputs to be preserved while specified target properties are adjusted through conditioning [32]. Conventional conditional models have largely relied on rather simple methods, such as concatenation, for implementing this conditioning process; however, [19] 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 im-

Sangpil Kim
E-mail: kim2030@purdue.edu
Purdue University
West Lafayette, IN 47906, USA

prove 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 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

The early work [4] have proposed a supervised, conditional generative model trained to generate images of chairs, tables, and cars with specified attributes which are controlled

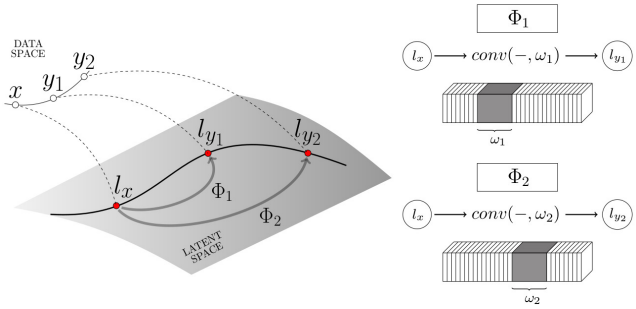


Fig. 1: The conditional transformation unit Φ constructs a collection of mappings $\{\Phi_k\}$ in the latent space which produce high-level attribute changes to the decoded outputs. Conditioning information is used to select the appropriate convolutional weights ω_k for the specified transformation;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

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. The appearance flow network (AFN) [34] proposed methods for the prediction of rotated viewpoints of objects by predicting apperance flows, the visual appearance of different views, and predict views with the input view. The M2N from [28] proposed recurrent network base view prediction and utilize self-learned confidence map, which is incremental version of AFN. These models are reliant on additional data, such as depth information, camera poses, or mesh models, to train their models, however, and cannot be trained using images alone. Conditional generative models have been widely used for geometric prediction [21,30]. Other works have introduced a clamping strategy to enforce a specific organizational structure in the latent space [24,16]; 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). The DFN by [13] proposed dynamic filter which conditioned on sequence of previous frames. This work is fundamentally different since the filter is applied in original input sources not latent embeddings. It use temporal information and, this network would not be applicable given single image. This work only introduce on the scene view synthesis only and not showed on the object view synthesis. This framework also relies on geometric concepts unique to rotation and is not generalizable to other inference tasks. The IterGAN model introduced by [6] 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. The conditional variational autoencoder (CVAE) incorporates conditioning information into the standard variational autoencoder (VAE) framework [15] and is capable of synthesizing specified attribute changes in an identity preserving manner [27,32]. CVAE-GAN [1] further adds adversarial training to the CVAE framework in order to improve the quality of generated predictions. The work from Zhang et al. [33] 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. 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 2).

LTNN Training Procedure

Provide: Labeled dataset $\{(x,\{y_k\}_{k\in\mathscr{T}})\}$ with target transformations indexed by a fixed set $\mathscr{T}$, encoder weights θ_E, decoder weights θ_D, RGB balance parameters $\{\theta_R,\theta_G,\theta_B\}$, conditional transformation unit weights $\{\omega_k\}_{k\in\mathscr{T}}$, discriminator $\mathscr{D}$ with standard weights $\theta_{\mathscr{D}}$ and conditionally selected weights $\{\overline{\omega}_k\}_{k\in\mathscr{T}}$, 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 2 - 1 in Section 3.2.

procedure TRAIN( )
  $x,\{y_k\}_{k\in\mathscr{T}}$ = get_train_batch()
    # Sample input and targets from training set
  l_x = Encode$[x]$
    # Encoding of original input image
  for k in $\mathscr{T}$ **do**
    l_{y_k} = Encode$[y_k]$
      # True encoding of specified target image
    $\widehat{l}_{y_k} = \text{conv}(l_x,\omega_k)$
      # Approximate encoding of target with CTU
    $\widehat{y}_k^{value}, \widehat{y}_k^{refine}$ = Decode$[\widehat{l}_{y_k}]$
      # Compute RGB value and refinement maps
    $\widehat{y}_k = \left[\theta_C \cdot \widehat{y}_{k,C}^{value} \odot \widehat{y}_{k,C}^{refine}\right]_{C\in\{R,G,B\}}$
  # Assemble final network prediction for target

    # Update encoder, decoder, RGB, and CTU weights
    $\mathscr{L}_{adv} = -\log(\mathscr{D}(\widehat{y}_k,\overline{\omega}_k))$
    $\mathscr{L}_{guide} = \gamma\cdot\mathscr{L}_{smooth}(\widehat{y}_k)+\rho\cdot\mathscr{L}_{recon}(\widehat{y}_k,y_k)$
    $\mathscr{L}_{consist} = \|\widehat{l}_{y_k}-l_{y_k}\|_1$
    $\mathscr{L} = \lambda\cdot\mathscr{L}_{adv}+\mathscr{L}_{guide}+\kappa\cdot\mathscr{L}_{consist}$
    for θ in $\{\theta_E,\theta_D,\theta_R,\theta_G,\theta_B,\omega_k\}$ **do**
      $\theta = \theta-\nabla_\theta\mathscr{L}$

    # Update discriminator and CDU weights
    $\mathscr{L}_{adv}^{\mathscr{D}} = -\log(\mathscr{D}(y_k,\overline{\omega}_k))-\log(1-\mathscr{D}(\widehat{y}_k,\overline{\omega}_k))$
    for θ in $\{\theta_{\mathscr{D}},\overline{\omega}_k\}$ **do**
      $\theta = \theta-\nabla_\theta\mathscr{L}_{adv}^{\mathscr{D}}$

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 $l_x = \text{Encode}(x)$.
2. Use conditioning information k to select conditional, convolutional filter weights ω_k.
3. Map the latent representation l_x to $\widehat{l}_{y_k} = \Phi_k(l_x) = \text{conv}(l_x,\omega_k)$, 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.

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.

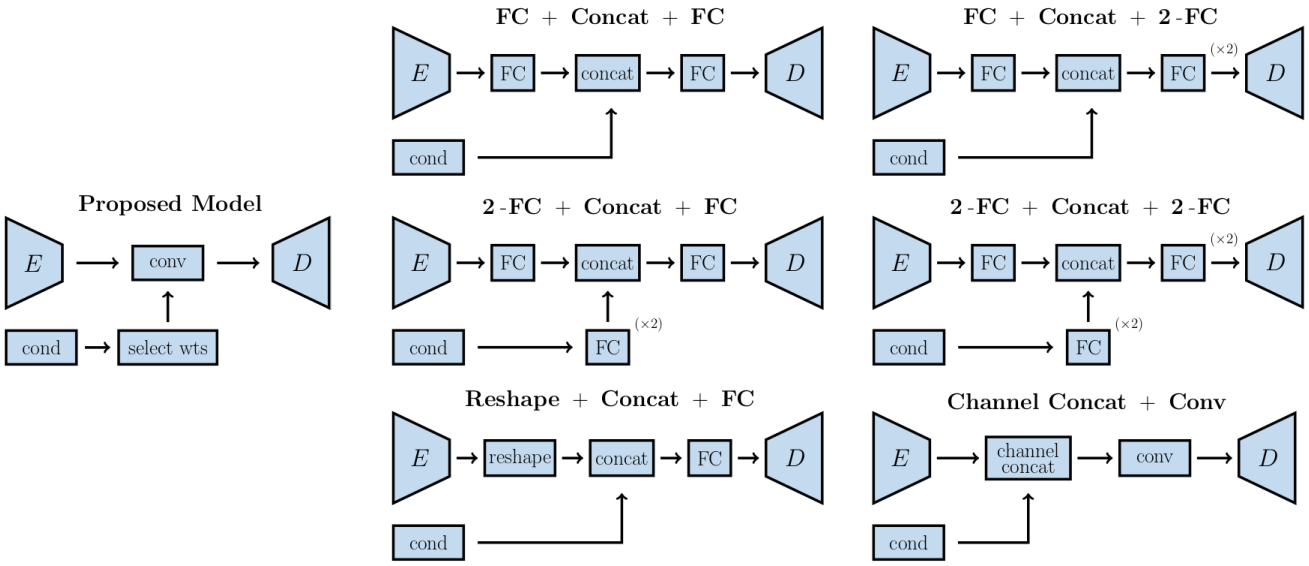


Fig. 2: Selected methods for incorporating conditioning information; the proposed LTNN method is illustrated on the left, and six conventional alternatives are shown to the right.

Several approaches have been proposed which include non-linear 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 [33, 1]. 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. There is an independent CTU per viewpoint and these CTU mapping process is ensure with the consist loss bellow in Equation 1. 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 [23]. 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. I_x denote given single image and y_k indicate a k^{th} transformation target image.

$$\mathscr{L}_{consist} = \left\| \Phi_k(\text{Encode}[I_x]) - \text{Encode}[y_k] \right\|_1 \tag{1}$$

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 1). The discriminator, $\mathscr{D}$, is trained using the adversarial loss term $\mathscr{L}^{\mathscr{D}}_{adv}$ defined below in Equation 2. The proposed model uses the adversarial loss in Equation 3 to ensure capturing multi modal distribution, which give sharper output.

$$\mathscr{L}^{\mathscr{D}}_{adv} = -\log \mathscr{D}(y_k, \overline{\omega}_k) - \log\left(1 - \mathscr{D}(\widehat{y}_k, \overline{\omega}_k)\right) \tag{2}$$

$$\mathscr{L}_{adv} = -\log \mathscr{D}(\widehat{y}_k, \overline{\omega}_k) \tag{3}$$

Table 1: Ablation/comparison results using identical encoder, decoder, and training procedure.

	Elevation		Azimuth		Light Direction		Age	
Model	SSIM	L_1	SSIM	L_1	SSIM	L_1	SSIM	L_1
LTNN + CDU + TD + CON	**.923**	**.107**	**.923**	**.108**	**.941**	**.093**	**.925**	**.102**
LTNN + CDU + TD	.917	.112	.922	.119	.938	.097	.913	.115
LTNN + CDU	.901	.135	.908	.125	.921	.121	.868	.118
LTNN	.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

Additional loss terms corresponding to structural reconstruction with reconstruction loss in Equation 4, and remove discrepancy between near by pixels with smoothness [12] with smooth loss in Equation 5, are also used for training the encoder/decoder:

$$\mathscr{L}_{recon} = \|\widehat{y}_k - y_k\|_2^2 \tag{4}$$

$$\mathscr{L}_{smooth} = 1/8 \cdot \sum_{i\in\{0,\pm1\}} \sum_{j\in\{0,\pm1\}} \left\|\widehat{y}_k - \tau_{i,j}\widehat{y}_k\right\|_1 \tag{5}$$

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, $\widehat{y}_k = \text{Decode}\left(\Phi_k\left(\text{Encode}[x]\right)\right)$ 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:

$$\mathscr{L} = \lambda \cdot \mathscr{L}_{adv} + \rho \cdot \mathscr{L}_{recon} + \gamma \cdot \mathscr{L}_{smooth} + \kappa \cdot \mathscr{L}_{consist} \tag{6}$$

with hyper parameters 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-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 without additional loss terms. As seen on the Figure 3, refine map produce shape mask and penalized errors in each pixels. The decoding process begins with a series of convolutional layers followed by bilinear interpolation to up sample the low resolution latent information. The last component of the decoder's

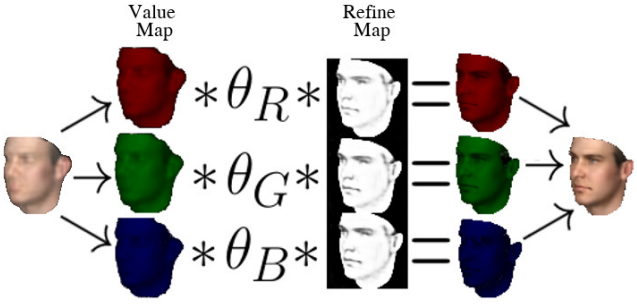


Fig. 3: Proposed task-divided design for the LTNN decoder. The coarse pixel value estimation map is split into RGB channels, rescaled by the RGB balance parameters, and multiplied element-wise by the refinement map values to produce the final network prediction.

up sampling process consists of two distinct transpose convolutional layers used for task divided; 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:

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

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.

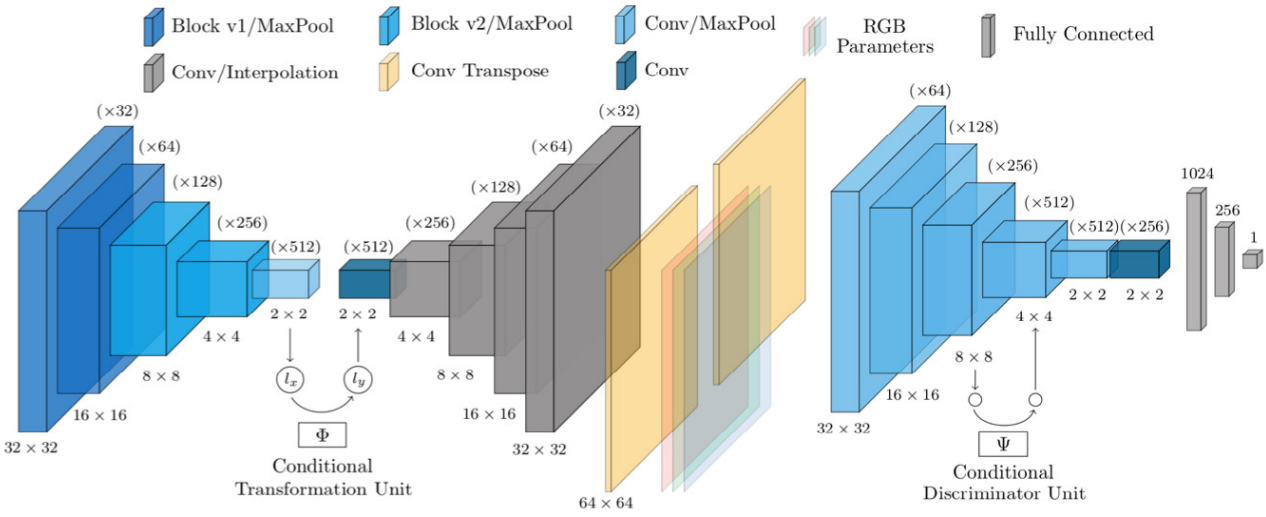


Fig. 4: The proposed network structure for the encoder/decoder (left) and discriminator (right). Features have been color-coded according to the type of layer which has produced them.

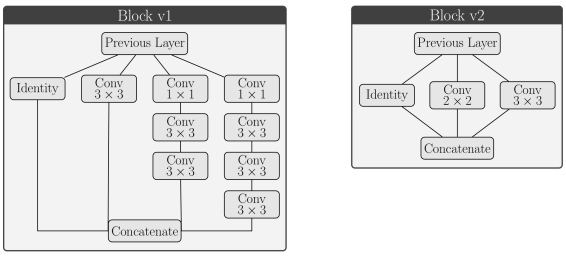


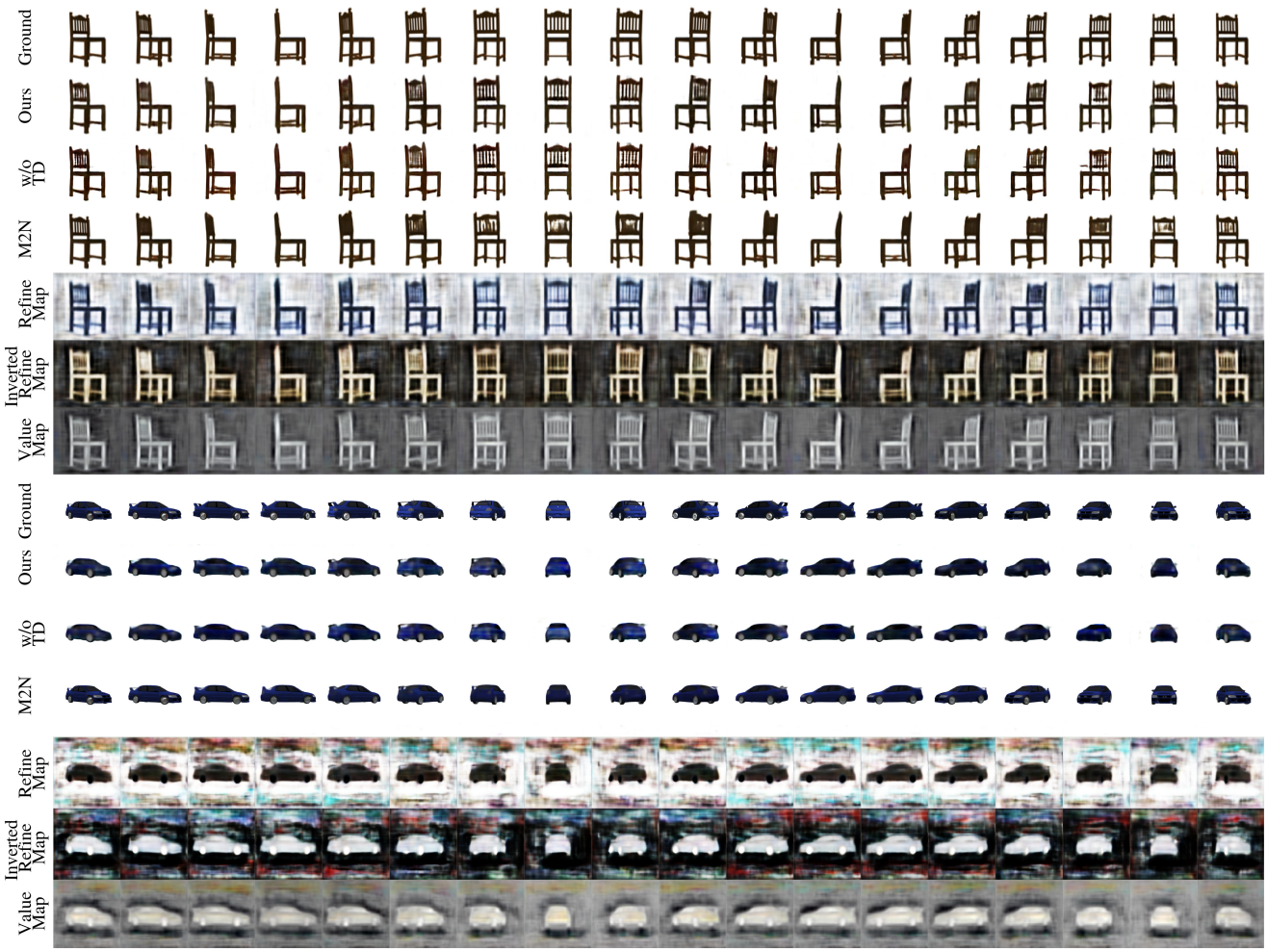
Fig. 5: Layer definitions for Block v1 and Block v2 collaborative filters. Once the total number of output channels, N_{out}, is specified, the remaining $N_{\text{out}} - N_{\text{in}}$ output channels are allocated to the non-identity filters (where N_{in} denotes the number of input channels). For 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 do not use activation functions for the intermediate 3×3 convolutional layers.

4 Architecture details

Input images of resolution 64×64 are passed through a Block v1 collaborative filter layer (see Figure 5) 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 stereo face dataset [5] experiment, we have added an additional Block v1 layer in the encoder and decoder to utilize the full $128 \times 128 \times 3$ resolution images.

The encoder incorporates two main block layers, as defined in Figure 5, which are designed to provide efficient feature extraction; these blocks follow a similar design to that proposed by [29], but include dense connections between blocks, as introduced by [9]. We normalize the output of each network layer using the batch normalization method as described in [11]. For the decoder, we have opted for a minimalist design, inspired by the work of [22]. Standard convolutional layers with 3×3 filters and same padding are used through the penultimate decoding layer, and transpose convolutional layers with 1×1 filters for 3D non-rigid objects and 5×5 for other experiments. We have used same padding

Fig. 6: Qualitative comparison of 360° view prediction of rigid-objects. A single image, first column in ground row images, is used as input to the network. Inverted refine map represent inverted color version of original refine map, which make easier to examine the refine map since target images have white background. Refine map create sharp boundary to mask out noise from value map.



to produce the value-estimation and refinement maps. All parameters have been initialized using the variance scaling initialization method described in [8]. 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 [14] with initial parameters: learning_rate = 0.005, $\beta_1 = 0.9$, and $\beta_2 = 0.999$ (as defined in the TensorFlow API r1.6 documentation for tf.train.AdamOptimizer). , along with loss function hyper parameters: $\lambda = 0.8$, $\rho = 0.2$, $\gamma = 0.0002$, and $\kappa = 0.00005$ (as introduced in Equation 6). The discriminator is updated once every two encoder/decoder updates, and one-sided label smoothing [26] has been used to improve stability of the discriminator training procedure.

5 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 [31] for testing, (ii) synthesis of rotated views of rigid objects using the 3D object dataset [2], (iii) synthesis of rotated views using a real face dataset [5], and (iv) the modification of a diverse range of attributes on a synthetic face dataset [10]. 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 5.2. 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) and the L_1 for evaluation metrics as used in other works [21,28]. To evaluate the proposed framework with existing works, two comparison groups have been formed: conditional inference methods, CVAE-GAN [1] and CAAE [33], with comparable encoder/decoder structures for comparison on experiments with non-rigid objects, and view synthesis methods, MV3D [30], M2N [28], AFN [34], and TVSN [21], 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 8 and Table 1.

5.1 Experiment on rigid objects

Rigid object experiment: We have tested our model's ability to perform 360° view estimation on the 3D objects and compared the results with the other state-of-the-art methods. The models are trained on the same dataset from author of M2N. We found that the results are better for car categories when the number of the operation increases, when the networks has the connections between encoder layers and decoder layers, proposed on U-net [25], which make the network preserve the information effectively. However, these connection significantly enhance operation [21,28]. This is because the connection increase the number of channels of the decoder. From this insight, we removed the connection between encoder and decoder and create CTU and DTU to enlarge information capacity, which will reduce information loss during training. The proposed model is comparable with existing models specifically designed for the task of multi-view prediction and require the least FLOPs for inference compared with all other methods as shown in Table 2, 3.

Table 2: Quantitative comparison table for 3D chair and car 360° view synthesis. Smaller numbers are better for L_1 and higher numbers are better for SSIM. We performed ablation study with and with out Task-divided Decoder (TD) and other methods.

	Car		Chair	
Model	SSIM	L_1	SSIM	L_1
Ours	.902	.121	**.897**	**.178**
Ours (w/o TD)	.861	.187	.871	.261
M2N	**.923**	**.098**	.895	.181
TVSN	.913	.119	.894	.230
AFN	.877	.148	.891	.240
MV3D	.875	.139	.895	.248

Table 3: FLOPs and parameter calculations correspond to inference for a single image with resolution 256×256×3. We calculated the FLOPs with the code from authors and definitions on the papers. Smaller numbers are better for parameters and GFLOPs/Image.

Model	Parameters (Million)	GFLOPs / Image
Ours	**17.0 M**	**2,183**
M2N	127.1 M	341,404
TVSN	57.3 M	2.860
AFN	70.3 M	2.671
MV3D	69.7 M	3.056

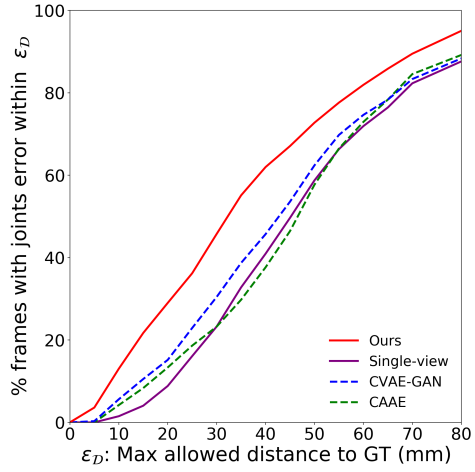


Fig. 7: Quantitative evaluation for multi-view hand synthesis. Evaluation with other methods using the real NYU dataset.

5.2 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. More specifically, the LTNN model was trained to generate 9 different views which were then fed into the pose estimation network from [3] (also trained using the synthetic dataset).

A comparison of the quantitative hand pose estimation results is provided in Figure 7 where the proposed LTNN framework is seen to provide a substantial improvement over existing methods; qualitative results are also available in Figure 9. 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), which could be run for real time applications.

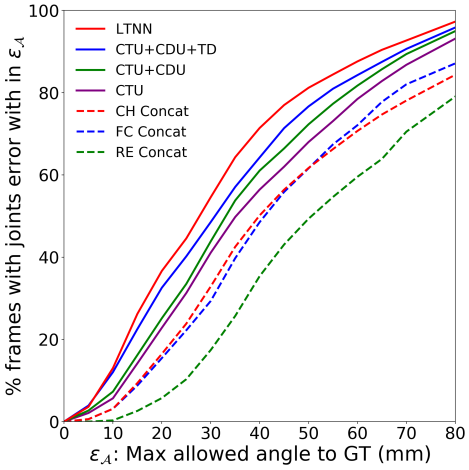


Fig. 8: Quantitative evaluation for multi-view hand synthesis. LTNN ablation results and comparison with alternative conditioning frameworks using synthetic hand dataset. Our models: conditional transformation unit (CTU), conditional discriminator unit (CDU), task-divide and RGB balance parameters (TD), 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).

Fig. 9: Comparison of CVAE-GAN (top) with proposed LTNN model (bottom) using the noisy NYU hand dataset [31]. 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 Figure10).

Real face experiment: We experiment on real face to show generalizability of LTNN. The stereo face database [5], consisting of images of 100 individuals from 10 different viewpoints, was used for experiments with real faces; these faces were first segmented using the method of [20] and then we manually cleaned up the failure cases. The cleaned

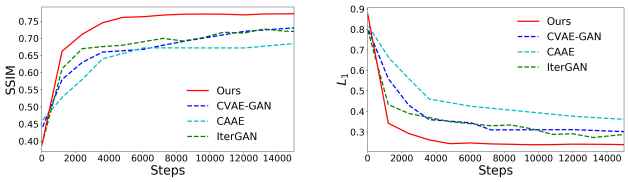


Fig. 10: Quantitative comparison of model performances for experiment on the real face dataset.

faces have been 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 10, our method significantly outperforms the CVAE-GAN, CAAE, and IterGAN models in both the L_1 and SSIM metrics.

Fig. 11: Qualitative evaluation for multi-view reconstruction of real face using the stereo face dataset [5].

5.3 Diverse attribute exploration

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 image; results are shown in Table 4 and ablation results are available in Table 1. Near continuous attribute modification is also possible within the proposed framework, and distinct CTU mappings can be

Table 4: Results for attribute modification on synthetic face dataset.

	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
CAAE	.777	.175	.521	.338	.856	.270	.751	.207

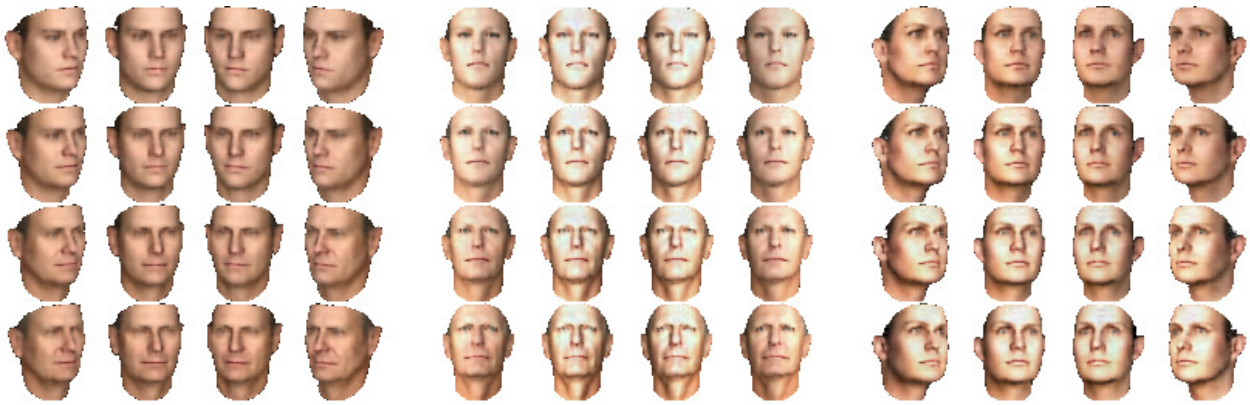


Fig. 12: Simultaneous learning of multiple attribute modifications. Azimuth and age (left), light and age (center), and light and azimuth (right) combined modifications are shown. The network has been trained using 4 CTU mappings per attribute (e.g. 4 azimuth mappings and 4 age mappings); results shown have been generated by composing CTU mappings in the latent space and decoding.

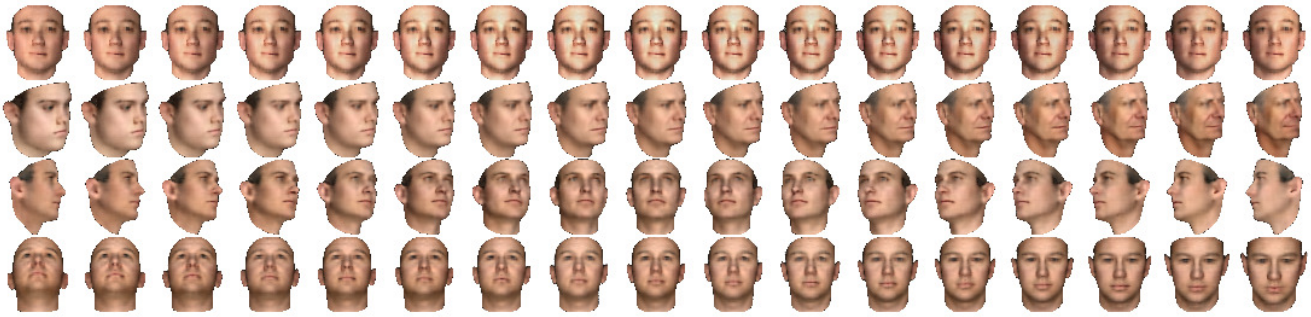


Fig. 13: Near continuous attribute modification is attainable using piecewise-linear interpolation in the latent space. Provided a gray-scale image (corresponding to the faces on the far left), modified images corresponding to changes in light direction (first), age (second), azimuth (third), and elevation (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.

composed with one another to synthesize multiple modifications simultaneously.

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 an lighting and azimuth changes are given by Decode$[\Phi_k^{light}(l_x)]$ and Decode$[\Phi_k^{azim}(l_x)]$, respectively (where l_x denotes the encoding of the input image).

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 may 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:

$$\begin{aligned} \widehat{y} &= \text{Decode}[\widehat{l_y}] \quad \text{where} \\ \widehat{l_y} &= \Phi_i^{light} \circ \Phi_j^{azim}(l_x) = \Phi_i^{light}\left[\Phi_j^{azim}(l_x)\right] \end{aligned} \tag{8}$$

5.4 Near-Continuous Attribute Modification

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 $k \in \{0, \ldots, 7\}$ 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.

6 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 deconstructions the data generation process into manageable 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

1. Bao, J., Chen, D., Wen, F., Li, H., Hua, G.: Cvae-gan: Fine-grained image generation through asymmetric training. arXiv preprint arXiv:1703.10155 (2017)
2. Chang, A.X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., Xiao, J., Yi, L., Yu, F.: ShapeNet: An Information-Rich 3D Model Repository. Tech. Rep. arXiv:1512.03012 [cs.GR], Stanford University — Princeton University — Toyota Technological Institute at Chicago (2015)
3. Choi, C., Kim, S., Ramani, K.: Learning hand articulations by hallucinating heat distribution. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3104–3113 (2017)
4. Dosovitskiy, A., Tobias Springenberg, J., Brox, T.: Learning to generate chairs with convolutional neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1538–1546 (2015)
5. Fransens, R., Strecha, C., Van Gool, L.: 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)
6. Galama, Y., Mensink, T.: Iterative gans for rotating visual objects (2018)
7. Goodfellow, I.J.: NIPS 2016 tutorial: Generative adversarial networks. CoRR **abs/1701.00160** (2017). URL `http://arxiv.org/abs/1701.00160`
8. He, K., Zhang, X., Ren, S., Sun, J.: 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)
9. Huang, G., Liu, Z., Weinberger, K.Q., van der Maaten, L.: Densely connected convolutional networks. arXiv preprint arXiv:1608.06993 (2016)
10. IEEE: A 3D Face Model for Pose and Illumination Invariant Face Recognition (2009)
11. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: International conference on machine learning, pp. 448–456 (2015)
12. Jason, J.Y., Harley, A.W., Derpanis, K.G.: 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)
13. Jia, X., De Brabandere, B., Tuytelaars, T., Gool, L.V.: Dynamic filter networks. In: Advances in Neural Information Processing Systems, pp. 667–675 (2016)
14. Kingma, D., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
15. Kingma, D.P., Welling, M.: Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114 (2013)
16. Kulkarni, T.D., Whitney, W.F., Kohli, P., Tenenbaum, J.: Deep convolutional inverse graphics network. In: Advances in Neural Information Processing Systems, pp. 2539–2547 (2015)
17. Makhzani, A., Shlens, J., Jaitly, N., Goodfellow, I., Frey, B.: Adversarial autoencoders. arXiv preprint arXiv:1511.05644 (2015)
18. Mirza, M., Osindero, S.: Conditional generative adversarial nets. arXiv preprint arXiv:1411.1784 (2014)
19. Miyato, T., Koyama, M.: cgans with projection discriminator. arXiv preprint arXiv:1802.05637 (2018)
20. Nirkin, Y., Masi, I., Tuan, A.T., Hassner, T., Medioni, G.: 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)
21. Park, E., Yang, J., Yumer, E., Ceylan, D., Berg, A.C.: 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)

22. Paszke, A., Chaurasia, A., Kim, S., Culurciello, E.: Enet: A deep neural network architecture for real-time semantic segmentation. arXiv preprint arXiv:1606.02147 (2016)
23. Ramachandran, P., Zoph, B., Le, Q.V.: Swish: a self-gated activation function. arXiv preprint arXiv:1710.05941 (2017)
24. Reed, S., Sohn, K., Zhang, Y., Lee, H.: Learning to disentangle factors of variation with manifold interaction. In: International Conference on Machine Learning, pp. 1431–1439 (2014)
25. Ronneberger, O., Fischer, P., Brox, T.: U-net: Convolutional networks for biomedical image segmentation. In: International Conference on Medical image computing and computer-assisted intervention, pp. 234–241. Springer (2015)
26. Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., Chen, X.: Improved techniques for training gans. In: Advances in Neural Information Processing Systems, pp. 2234–2242 (2016)
27. Sohn, K., Lee, H., Yan, X.: Learning structured output representation using deep conditional generative models. In: Advances in Neural Information Processing Systems, pp. 3483–3491 (2015)
28. Sun, S.H., Huh, M., Liao, Y.H., Zhang, N., Lim, J.J.: Multi-view to novel view: Synthesizing novel views with self-learned confidence. In: Proceedings of the European Conference on Computer Vision (ECCV), pp. 155–171 (2018)
29. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 1–9 (2015)
30. Tatarchenko, M., Dosovitskiy, A., Brox, T.: Multi-view 3d models from single images with a convolutional network. In: European Conference on Computer Vision, pp. 322–337. Springer (2016)
31. Tompson, J., Stein, M., Lecun, Y., Perlin, K.: Real-time continuous pose recovery of human hands using convolutional networks. ACM Transactions on Graphics (ToG) **33**(5), 169 (2014)
32. Yan, X., Yang, J., Sohn, K., Lee, H.: Attribute2image: Conditional image generation from visual attributes. In: European Conference on Computer Vision, pp. 776–791. Springer (2016)
33. Zhang, Z., Song, Y., Qi, H.: Age progression/regression by conditional adversarial autoencoder. arXiv preprint arXiv:1702.08423 (2017)
34. Zhou, T., Tulsiani, S., Sun, W., Malik, J., Efros, A.A.: View synthesis by appearance flow. In: European Conference on Computer Vision, pp. 286–301. Springer (2016)