Spatially Multi-conditional Image Generation

Nikola Popovic[1*] Ritika Chakraborty[1*] Danda Pani Paudel[1] Thomas Probst[1]
Luc Van Gool[1,2]
[1]Computer Vision Laboratory, ETH Zurich, Switzerland
[2]VISICS, ESAT/PSI, KU Leuven, Belgium
{nipopovic,critika, paudel, probstt, vangool}@vision.ee.ethz.ch

Abstract

In most scenarios, conditional image generation can be thought of as an inversion of the image understanding process. Since generic image understanding involves solving multiple tasks, it is natural to aim at generating images via multi-conditioning. However, multi-conditional image generation is a very challenging problem due to the heterogeneity and the sparsity of the (in practice) available conditioning labels. In this work, we propose a novel neural architecture to address the problem of heterogeneity and sparsity of the spatially multi-conditional labels. Our choice of spatial conditioning, such as by semantics and depth, is driven by the promise it holds for better control of the image generation process. The proposed method uses a transformer-like architecture operating pixel-wise, which receives the available labels as input tokens to merge them in a learned homogeneous space of labels. The merged labels are then used for image generation via conditional generative adversarial training. In this process, the sparsity of the labels is handled by simply dropping the input tokens corresponding to the missing labels at the desired locations, thanks to the proposed pixel-wise operating architecture. Our experiments on three benchmark datasets demonstrate the clear superiority of our method over the state-of-the-art and compared baselines. The source code can be found at `https://github.com/96ritika/TLAM`.

1. Introduction

In recent years, automated image generation under user control has become more and more of a reality. Such processes typically use so-called conditional image generation methods [32, 19]. One could see these as an intermediate between fully unconditional [14] and purely rendering based [6] generation, respectively. Methods belonging to these two extreme cases either offer no user control, or rely on all necessary information of image formation supplied by the user. In many cases, neither extreme is desirable. Therefore, several conditional image generation methods, that circumvent the rendering process altogether, have been proposed. These methods usually receive the image descriptions either in the form of text [32, 43] or as spatially localized semantic classes [19, 37, 45].

In this paper, we aim to condition the image generation on more than just the desired semantics. In this regard, we make a generic practical assumption that any semantic or geometric aspect of the desired image may be available as a label for conditioning (such as semantic segmentation, edges, depth, normals, etc). Moreover, these labels do not have to be defined at all spatial locations. For example, an augmented reality application may require to render a known object using its 3D model and specified pose, but with missing texture and lighting details. In such cases, geometric and semantic labels become instantly available at that object's location, while the labels of the other parts of the same image may also be available partially or completely. This offers us the multi-conditioning inputs for the image generation process. Incorporating all such information in a multi-conditional manner to generate the desired image, is the main challenge that we undertake. Another example is 3D graphic design where the mentioned labels are naturally a part of the designing and rendering process. The designer can now focus on constructing detailed semantic and geometric aspects for the important objects, while leaving the rest of the image to the partial-imagination of a deep network. In some sense, our approach bridges the gap between generative and rendering based image synthesis.

Two major challenges of multi-conditional image generation, in the context of this paper, are the heterogeneity and the sparsity of the available labels. The heterogeneity refers to differences in representations of different labels, for e.g. depth and semantics. On the other hand, the sparsity is either simply caused by the label definition (e.g. sky has no normal) or due to missing annotations [40]. It is important to note that some geometric aspects of images, such as an

*Equal contribution.

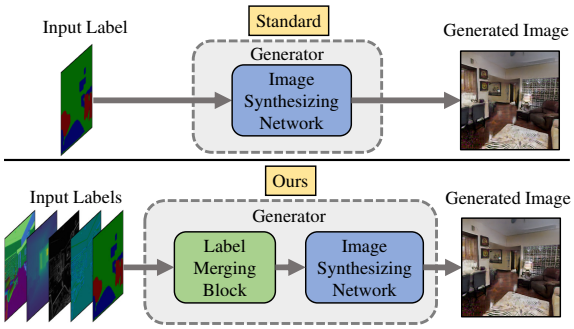


Figure 1. **Spatially multi-conditional image generation**. Our model uses multiple labels to generate an image, compared to standard approaches which use only the semantic segmentation label. Multiple input labels, coming from different sources, are handled by the proposed label merging block.

object's depth and orientation, can be introduced manually, without requiring any 3D model with its pose. This allows users to geometrically control images (beyond the semantics based control) both in the the presence or absence of 3D models. It goes without saying that the geometric manipulation of any image can be carried out by first inferring its geometric attributes using existing methods [40, 47, 56], followed by generation after manipulation.

To address the problems of both heterogeneity and diversity, we propose a label merging network that learns to merge the provided conditioning labels pixel-wise. To this end, we introduce a novel transformer-based architecture, that is designed to operate on each pixel individually. The provided labels are first processed by a label-specific multilayer perceptrons (MLPs) to generate a token for each label. The tokens are then processed by the transformer module. In contrast to standard vision transformers that perform spatial attention [8, 57], our transformer module applies self-attention across the label dimension, thus avoiding the high computational complexity. The pixel-wise interaction of available labels homogenizes the different labels to a common representation for the output tokens. The output tokens are then averaged to obtain the fused labels in the homogeneous space to form the local *concept*. This is performed efficiently for all pixels in parallel by sliding the pixel-wise transformer over the input label maps. Finally, the concepts are used for the image generation via conditional generative adversarial training, using a state-of-the-art method [45]. During the process of label merging, the spatial alignment is always preserved. The sparsity of the labels is handled by simply dropping the input tokens of the missing labels, at the corresponding pixel locations. This way, the transformer learns to construct the concept for each pixel, also in the case when not all labels are available. We study the influence of several spatially conditioning labels including semantics, depth, normal, curvature, edges, in three different benchmark datasets. The influence of the labels is studied both in the case of sparse and dense label availability. In both cases, the proposed method provides outstanding results by clearly demonstrating the benefit of conditioning labels beyond the commonly used image semantics. The major contribution of this paper can be summarized as follow:

1. We study the problem of spatially multi-conditional image generation, for the first time.

2. We propose a novel neural network architecture to fuse the heterogeneous multi-conditioning labels provided for the task at hand, while also handling the sparsity of the labels at the same time.

3. We analyse the utility of various conditioning types for image generation and present outstanding results obtained by the proposed method in benchmark datasets.

2. Related Work

Conditional Image Synthesis. A description of the desired image to be generated can be provided in various forms, from class conditions [33, 2], text [32, 43], spatially localized semantic classes [19, 37, 45], sketches [19], style information [12, 21, 22] to human poses [31]. Recently, different data structures (e.g. text sequences) have also received attention in the literature [59, 54]. The problem of spatially multi-conditional image generation is orthogonal to the problem of unifying different (non-spatial) modalities, as it seeks to fuse heterogeneous spatially localized labels into concepts, while preserving the spatial layout for image generation. Isola et al. [19] later introduced the *Pix2Pix* paradigm to convert sketches into photo-realistic images, leveraging the image-to-image translation *UNet* [44] backbone as a generator combined with a convolutional discriminator. This work was improved by Wang et al. to support high resolution image translation in *Pix2PixHD* [53] and video translation in *Vid2Vid* [52]. Recently, Shaham et al. introduced the *ASAP-Net* [45] which achieves a superior trade-off of inference time and performance on several image translation tasks. We employ the *ASAP-Net* as one component in our architecture.

Conditioning Mechanisms. The conditioning mechanism is at the core of semantically controllable neural networks, and is often realized in conjunction with normalization techniques [9, 17]. Perez et al. introduced a simple feature-wise linear modulation FiLM [39] for visual reasoning. In the context of neural style transfer [11], Huang et al. introduced Adaptive Instance Normalization AdaIN [18]. Park et al. extended AdaIN for spatial control in SPADE [37], where the normalization parameters are derived from a semantic segmentation. Zhu et al. [60] further extend SPADE to allow for independent application of global and local styles. Finally, generic normalisation schemes that make

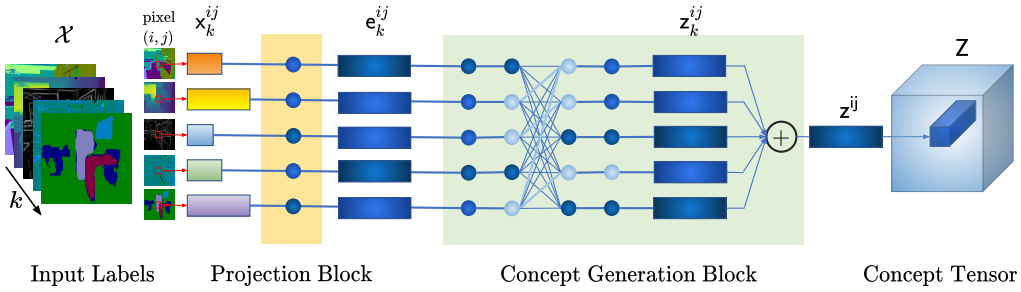


Figure 2. **Pixel-wise Transformer Label Merging block (TLAM)**. The heterogeneous labels of each pixel x_k^{ij} are first projected into the same dimensionality e_k^{ij}, and then passed to the concept generation block. A transformer module promotes the interaction between labels, before finally distilling them to a concept vector z^{ij} by averaging the homogeneous label representations z_k^{ij} at every pixel location.

use of kernel prediction networks to achieve arbitrarily global and local control include Dynamic Instance Normalization (DIN) [20] and Adaptive Convolutions (AdaConv) [4]. While being greatly flexible, they also increase the resulting inference time, unlike ASAP-Net that uses adaptive implicit functions [46] for efficiency.

Differentiable Rendering Methods. Since rendering is a complex and computationally expensive process, several approximations were proposed to facilitate its use for training neural networks. Methods starting from simple approximations of the rasterization function to produce silhouettes [24, 29, 26] to more complex approximations modelling indirect lighting effects [27, 35, 30] have been proposed. These algorithms have also been incorporated into popular deep learning frameworks [42, 27, 36]. Differentiable renderers have been successfully used in several neural networks including those used for face [50, 49] and human body [28, 38] reconstruction. We refer the interested reader to the excellent surveys of Tewari et al. [48] and Kato et al. [23] for more details. In contrast to rendering approaches which require setting numerous scene parameters, our method directly generates realistic images from only a sparse set of chosen labels.

3. Method

We start by introducing a few formal notations. As an input, we have a set of labels $\mathcal{X} = \{\mathsf{X}_1, \mathsf{X}_2, ..., \mathsf{X}_N\}$, where each label $\mathsf{X}_k \in \mathbb{R}^{H \times W \times C_k}$ has height H, width W and C_k channels. The element of X_k corresponding to the pixel location (i, j) is denoted as $\mathsf{x}_k^{ij} \in \mathbb{R}^{C_k}$. Hence, elements of the label set $\mathcal{X}$ corresponding to the pixel location (i, j) form a set $\mathcal{X}^{ij} = \{\mathsf{x}_1^{ij}, \mathsf{x}_2^{ij}, ..., \mathsf{x}_N^{ij}\}$. The model takes the set of labels $\mathcal{X}$ as an input and produces $\mathsf{I} = \phi(\mathcal{X})$, where $\mathsf{I} \in \mathbb{R}^{H \times W \times 3}$ is the generated image.

3.1. Label Merging

We describe the mechanism of the label merging component, as illustrated in Figure 2. This component processes different pixel locations of the input labels $\mathcal{X}^{ij}$ independently, and is efficiently executed on all pixels in parallel. We empirically found this to be sufficient. In this process, we are interested to merge all available heterogeneous labels $\{\mathsf{x}_k^{ij}\}$ into a latent vector z^{ij}, for each pixel location. Thus, we get a latent tensor Z that we can use to synthesize the image I, while avoiding the problems of label heterogeneity and sparsity. Input labels are heterogeneous because they come from different sources and can have different numbers of channels C_k, as well as different ranges of values (i.e. semantic segmentation is represented with discrete values, while surface normals are continuous). Input labels can be sparse because they do not have to be available for every pixel location [40]. Furthermore, label merging is performed by two blocks: the projection block and the concept generation block, which we described in the following.

3.1.1 Projection Block

This block projects every heterogeneous input label X_k into an embedding space $\mathsf{E}_k \in \mathbb{R}^{H \times W \times d}$, where d is the dimensionality of the embedding space. It does so by transforming every element x_k^{ij} with the projection function f_k, $f_k : \mathsf{x}_k^{ij} \mapsto \mathsf{e}_k^{ij}$, where $\mathsf{e}_k^{ij} \in \mathbb{R}^d$. All the elements of a given label X_k share the same projection function f_k, but different input labels k have different f_k. We use an affine transformation followed by the GeLU nonlinearity $a(\cdot)$ [15] to serve as the embedding function,

$$\mathsf{e}_k^{ij} = f_k(\mathsf{x}_k^{ij}) = a(\mathsf{A}_k \mathsf{x}_k^{ij} + \mathsf{b}_k), \tag{1}$$

where $\mathsf{A}_k \in \mathbb{R}^{d \times C_k}$ and $\mathsf{b}_k \in \mathbb{R}^d$. We implement f_k by using a different 1×1 convolution module for each input label X_k, followed by a GeLU activation function. When the value of a label X_k is missing at a certain spatial location (due to sparsity), its element x_k^{ij} is dropped by setting it to a zero vector. This will send a signal to the concept generation block that the label k is not present at location (i, j), and that it should extract the information from other labels. Also, the absence of a whole label X_k is handled similarly.

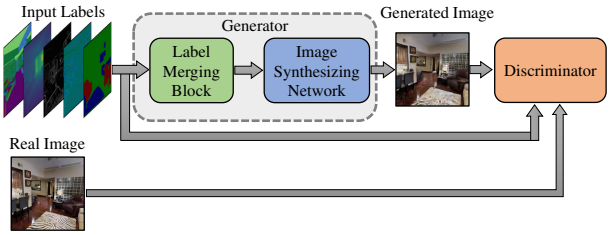


Figure 3. **Network overview.** Different input labels are embedded into a homogeneous space with the label merging block. The image synthesizer uses this embedding to generate an image. During training, the discriminator uses the labels, real and generated images, to optimize the label merger and the image synthesizer.

3.1.2 Concept Generation Block

The set of embedding vectors corresponding to different labels $\mathcal{E} = \{\mathsf{e}_1, \mathsf{e}_2, ..., \mathsf{e}_N\}$ serve as input tokens for our concept generation block (we drop the spatial index $\cdot^{ij}$ for simplicity). This block uses a novel attention module to model interaction across labels, and this module is shared across all pixel locations. It therefore does not require expensive spatial attention as is the case with standard vision transformers [51, 8]. In other words, we apply the same label-transformer on each pixel individually. Transformers naturally encourage interactions between different labels, thus labels share their label-specific information while forming the final homogeneous representation. Before feeding $\mathcal{E}$ to the transformer, we apply label specific encoding to obtain $\mathsf{z}_k^{(0)} = \mathsf{e}_k + \mathsf{p}_k$. Then we pass $\mathcal{Z}^{(0)} = \{\mathsf{z}_1^{(0)}, \mathsf{z}_2^{(0)}, ..., \mathsf{z}_N^{(0)}\}$ through l transformer blocks B_m. Each block B_m takes the output of the previous block $\mathcal{Z}^{(m-1)}$ and produces $\mathcal{Z}^{(m)} = \mathsf{B}_m(\mathcal{Z}^{(m-1)})$, where $\mathcal{Z}^{(m)} = \{\mathsf{z}_1^{(m)}, \mathsf{z}_2^{(m)}, \ldots, \mathsf{z}_N^{(m)}\}$.

Each transformer block B_m is composed of a Multi-head Self Attention block (MSA), followed by a Multilayer Perceptron block (MLP) [51, 8]. MSA is a global operation, where every label token interacts with every other token and thus information is shared across labels. The MSA block executes the following operations:

$$\hat{\mathcal{Z}}^{(m)} = \mathsf{MSA}(\mathsf{LN}(\mathcal{Z}^{(m-1)})) + \mathcal{Z}^{(m-1)}, \qquad (2)$$

where LN represents Layer Normalization [1]. The MSA block is followed by the MLP block, which processes each token separately using the same Multilayer Perceptron. This block further processes label tokens, after their global interactions in the MSA block, by sharing and refining the representations of each token across all their channels. The MLP block executes the following operations:

$$\mathcal{Z}^{(m)} = \mathsf{MLP}(\mathsf{LN}(\hat{\mathcal{Z}}^{(m)})) + \hat{\mathcal{Z}}^{(m)}. \qquad (3)$$

Finally, all elements of the output set $\mathcal{Z} = \mathcal{Z}^{(l)} = \{\mathsf{z}_1^{(l)}, ..., \mathsf{z}_N^{(l)}\}$ are averaged to obtain $\mathsf{z} = \frac{1}{N}\sum_{k=1:N} \mathsf{z}_k^{(l)}$. This gives us the *Concept Tensor* $\mathsf{Z} \in \mathbb{R}^{H \times W \times d}$.

Note that, for standard vision transformers [8] operating on M spatial elements (e.g. pixels/patches), the computational complexity of the self-attention is $\mathcal{O}(M^2)$. Our pixel-wise transformer, operating only on N label tokens, reduces the complexity of self-attention to $\mathcal{O}(N^2)$ (for each pixel) with the number of labels $N \ll M$.

3.2. Network Overview

The proposed generative model is divided into two components: the label merging block and the image synthesizing network. This is depicted in Figure 3.

Label Merging Block. This block takes a set of heterogeneous labels $\mathcal{X}$ as the input and merges them into a homogeneous space $\mathsf{Z} = \psi(\mathcal{X})$, where $\mathsf{Z} \in \mathbb{R}^{H \times W \times d}$ is the *Concept Tensor*. It is important to note that the label merging block does not require all input labels to be defined for each pixel location. The label merging block first translates all labels X_k into embeddings E_k with the same dimensionality, using the projection block. Then, it uses the concept generation block to translate embeddings E_k to the *Concept Tensor* Z. In short, the label merging block fuses the heterogeneous set of input labels $\mathcal{X}$ into a homogeneous representation Z. We name this block as the Transformer LAbel Merging (**TLAM**) block and it is depicted in Figure 2.

Image Synthesizing Network. The task of the image synthesizing network is to take the produced *Concept Tensor* Z and generate the image $\mathsf{I} = g(\mathsf{Z})$, as shown in Figure 3. For this purpose, we employ the state-of-the-art ASAP model [45]. This model synthesizes the high-resolution pixels using lightweight and highly parallelizable operators. ASAP performs most computationally expensive image analysis at a very coarse resolution. We modify the input to ASAP, by providing the *Concept Tensor* Z as an input, instead of giving only one specific input label X_k (i.e. semantics). The synthesizing network therefore exploits the merged information from all available input labels $\mathcal{X}$.

Adversarial Training for Multi-conditioning. We follow the optimization protocol of ASAP-Net [45]. We train our generator model adversarially with a multi-scale patch discriminator, as suggested by pix2pixHD [53]. To achieve this, we modify the input to the discriminator by using the *Concept Tensor* Z instead of a specific label X_k.

4. Experiments

Implementation details. We follow the optimization protocol of ASAP-Net [45]. We train our generator with a multi-scale patch discriminator, as suggested by pix2pixHD [53]. The training includes an adversarial hinge-loss, a perceptual loss and a discriminator feature matching loss. The learning rates for generator and discriminator are 0.0001 and 0.0004, respectively. We use ADAM [25] with $\beta_1 = 0$ and $\beta_2 = 0.999$, following [37, 45].

Figure 4. **Visual comparisons on the Taskonomy dataset.** Column 1 & 2: five different labels tiled horizontally for dense and 50% sparse cases. Column 5 & 6: images generated by our method using the labels from column 1 & 2. Column 3 & 4: images generated by the dense semantic methods. Our method generates more realistic images with fine geometric and visual details from both dense and sparse labels.

Backbone	Method	Label sparsity					FID
		S	E	C	D	N	
SPADE [37]	Regular	■					72.3
	Naïve baseline	■	■	■	■	■	66.1
ASAP [45]	Regular	■					73.8
	Naïve baseline	■	■	■	■	■	74.6
	CLAM baseline	◣	◣	◣	◣	◣	43.8
	TLAM (ours)	◣	◣	◣	◣	◣	37.9
	CLAM baseline	■	■	■	■	■	37.1
	TLAM (ours)	■	■	■	■	■	**30.6**

Table 1. **FID scores on the Taskonomy dataset**. Our TLAM method generates images with significantly better visual quality for both sparse and dense labels. Symbol ■ corresponds to dense labels, while ◣ corresponds to 50% label sparsity. S stands for semantics, E for edges, C for curvature, D for depth and N for normals. Please, refer Figure 4 for corresponding images.

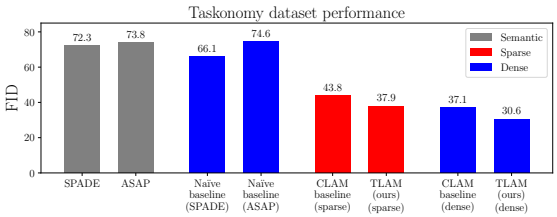


Figure 5. **Model comparison.** Our TLAM method preforms better compared to the established baselines as well as to the SotA models which use semantic segmentation labels only.

Label sparsity. Here we explain how inputs with $S\%$ label sparsity are generated. First, the pixel space is divided into distinct regions corresponding to semantic segmentation instances. Then, for each region we repeat the following process: for each available label independently, we drop all of its values in that region with $S\%$ probability. Please look at Fig. 1-4 in the supplementary for a visual demonstration.

4.1. Datasets

We experiment on three different datasets to demonstrate the versatility and effectiveness of our approach.

Taskonomy dataset [58] is a multi-label annotated dataset of indoor scenes. The entire dataset consists of over 4 Million images from around 500 different buildings with high resolution RGB images, segmentation masks and other labels. We chose this dataset for our main experiments because of its wide selection of both semantic and geometric labels. In our experiments, we use the following labels: semantic segmentation, depth, surface normals, edges and curvature. We selected two buildings from the large dataset resulting a total of 18,246 images, split into 14,630/3,619 training/validation images.

Cityscapes dataset [7] contains images of urban street scenes from 50 different cities and their dense pixel annotations for 30 classes. The training and validation split contains 3000 and 500 samples respectively. To obtain further labels, we use a state-of-the art depth estimation network [13] for depth. The estimated depth, along with the camera intrinsics were used to compute the local patch-wise surface normals. Additionally, we use canny filters for edge detection. This resulted four labels for Cityscape.

NYU depth v2 dataset [34] consists of 1449 densely labeled pairs of aligned RGB images and depth, normals, semantic segmentation and edges maps of indoor scenes. We split the data into 1200/249 training/validation sets.

The presented qualitative and quantitative results are generated on the respective hold-out test sets, with the exception of Figure 9, where we follow the protocol of [53] for comparison with the other methods.

4.2. Baselines and Metrics

Since this is the first work in spatially multi-conditional image generation, we construct our own baselines.

Naïve baseline takes all available labels and concatenates them along the channel dimension to create the input, which is fed to the ASAP-Net/SPADE backbone. This is equivalent to spatial multi-conditioning without explicit label merging, and therefore it also serves as an ablation to study the efficacy and necessity of the label merging block.

Convolutional Label Merging (CLAM) baseline stacks multiple consecutive blocks similar to the projection block from (1). The first block is exactly (1), while the following l blocks perform the same operation with $\mathsf{A}_k^l \in \mathbb{R}^{d\times d}$, $\mathsf{b}_\mathsf{k}^\mathsf{l} \in \mathbb{R}^d$, preserving the dimensionality d. After the final block, all output elements corresponding to the same spatial location are averaged, just like after the TLAM block. This baseline is a deep network with a very simple label merging mechanism. Thus, we use it to evaluate the efficacy of the more sophisticated label merging mechanism of TLAM.

SotA semantic-only methods. We also compare our method with state-of-the-art semantic image synthesis models, including SPADE [37], ASAP-Net [45], CRN [5], SIMS [41], Pix2Pix [19] and Pix2PixHD [53].

Performance Metrics. We follow the evaluation protocol of previous methods [45, 37]. We measure the quality of generated images using the Fréchet Inception Distance (FID) [16], which compares the distribution between generated data and real data. The FID summarizes how similar two groups of images are in terms of visual feature statistics. The second metric is the segmentation score, obtained by evaluating mean-Intersection-over-Union (mIoU) and pixel accuracy (accu) of a semantic segmentation model applied to the generated images. We use state-of-the-art semantic segmentation networks DRN-D-105 [55] for Cityscapes and DeepLabv3plus [3] for NYU depth v2 dataset.

4.3. Quantitative Results

Image Generation. Table 1 reports the FID scores obtained on the Taskonomy dataset. We compare our method with several baselines and under different label sparsity. Our TLAM significantly outpreforms SPADE and ASAP SotA methods, which use only semantic segmentation maps as inputs. This shows that using different spatial input labels can indeed improve the generation quality. Moreover, when SPADE and ASAP simply concatenate multiple labels as an input (naïve baseline), they do not preform significantly better than with just the semantics. This emphasizes the difficulty of merging multiple spatial labels, which are heterogeneous in nature. Some labels represent semantic image properties, while others represent geometric properties. Also, some labels are continuous, while others are discrete. Furthermore, our TLAM preforms better than the CLAM baseline, showing the value of having a better label merging block to deal with the heterogeneity present in the input labels. Finally, we compare TLAM and the CLAM baseline with dense and sparse labels. As expected, having dense labels achieves better image quality. Also, the visual quality when using 50% sparse labels is close to that of using dense labels. This is interesting and desirable, since in practical scenarios one often ends up having sparse labels [40].

Method	mIoU	Accuracy	FID
CRN [5]	52.4	77.1	104.7
SIMS [41]	47.2	75.5	**49.7**
Pix2Pix [19]	39.5	78.3	80.7
Pix2PixHD [53]	<u>58.3</u>	81.4	95.0
SPADE [37]	**62.3**	<u>81.9</u>	71.8
ASAP [45]	44.9	78.6	72.5
TLAM (ours)	45.5	**85.3**	<u>68.3</u>

Table 2. **Quantitative results on Cityscapes.** Our method uses ASAP as the image synthesizing network, which improves performance compared to a stand-alone ASAP by a significant margin.

Method	mIoU	Accuracy
SPADE [37]	33.1	47.4
ASAP [45]	<u>36.2</u>	<u>49.1</u>
TLAM (ours)	**38.3**	**53.1**

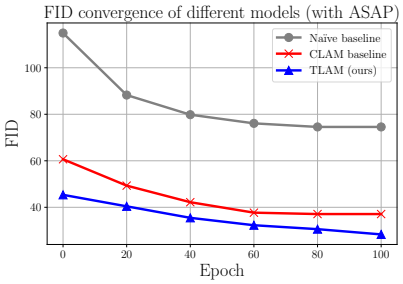
Table 3. **Quantitative results on NYU.** Our method demonstrates the benefit of spatial multi-conditioning over the baseline ASAP.

The results on Cityscapes and NYU are summarized in Tables 2 and 3. On the Cityscapes dataset, we compare our method with the SotA methods and report FID, and segmentation mIoU and accuracy. Our method achieves better accuracy compared to the other methods. As reported in [37] the SIMS model produces a lower FID, but has poor segmentation accuracy on the Cityscapes. This is because SIMS synthesizes an image by first stitching image patches from the training dataset. On the NYU dataset, our method achieves better mIoU and accuracy. Unfortunately, we are unable to compute the FID, due to the small size of only 249 images in the validation set.

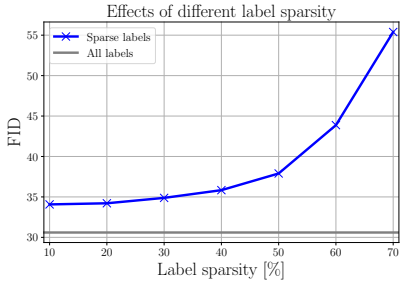
Training Convergence. Figure 6a shows the evolution of FID during training on Taskonomy. We evaluate the FID on the validation set every 20 epochs for TLAM, and the naïve and CLAM baselines. One can observe that TLAM quickly achieves a very good FID, even after 20 epochs. At this instance, TLAM has 2.5 and 1.3 times better FID than those of the naïve and CLAM baselines, respectively. We can also see that TLAM converges faster than the other models. One training epoch in Figure 6a takes about 1.7hrs for our method on one GeForce GTX TITAN X GPU.

4.4. Qualitative Results

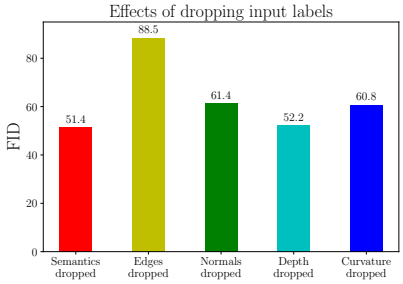
To visually compare our TLAM label merging, we present qualitative results for both dense and sparse labels on the Taskonomy dataset in Figure 4, together with the semantic-only baselines. TLAM with all-dense and all-sparse labels generates high-fidelity images. Our method generates fine structural details such as lights, decorations,



(a) **Convergence of different models.** We compare FID scores during training of our TLAM method to the naïve and CLAM baselines. Our method converges faster and achieves better FID compared to those of the baselines at every point during training.



(b) **Effects of different label sparsity.** Our model achieves good FID score even when a fraction of labels is presented during inference. With more labels present, our model achieves better performance, even though it was trained for 50% label sparsity.



(c) **Effects of dropping input labels.** We examine how dropping a specific label affects the image quality of our TLAM model. Dropping any label degrades FID which leads to the conclusion that all labels provide useful information for image synthesis.

Figure 6. **Training convergence and label sensitivity.** We analyse behavioural aspects of our models with regards to training and labels.

and even mirror reflections clearly better than SPADE and ASAP. The results on the Cityscapes dataset are shown in Figure 8. Notably, our method with sparse labels achieves similar visual quality as other methods. Figure 9 shows qualitative results on the NYU dataset, where Pix2PixHD also renders images with good quality, however it fails to capture the lighting conditions in contrast to our method. Notably, our method captures the rich geometric structures (such as on the ceiling), thanks to the geometric labels. Please, refer to our supplementary material for more visual results. Overall, the qualitative results demonstrate the effectiveness of TLAM, which exploits novel pixel-wise label transformers, even for sparse labels.

4.5. Label Sensitivity Study

We analyse the sensitivity of our method with regards to the provided labels on the Taskonomy dataset.

Label Sparsity. Figure 6b shows how the FID is affected by label sparsity. Experiments conducted on increasing sparsity from 10% to 70%, show a steady degradation of FID with less available labels. *Note that our model already achieves good FID using only 30% of available labels*. The experiments were conducted using a single model trained with 50% sparsity. It also shows the generalizability of our method across various levels of sparsity.

Removal of Labels. In Figure 6c, we plot the FID after removing each label from the input, using the TLAM model trained on Taskonomy with 50% sparsity. We observe that among the five labels, edges play the most significant role. On the other hand, the semantics and depth are the most dispensable. Nevertheless, the removal of any label results into a worsening of the FID at least by a factor of 1.3. This suggests that all labels provide different information crucial for image generation, while being mutually complimentary.

We conclude that the proposed label merging block can successfully deal with incomplete labels and is able to exploit information from all available labels.

4.6. Concept Visualization and Image Editing

Concept Visualization. To visualize the concept tensor $\mathsf{Z} \in \mathbb{R}^{H \times W \times 96}$, we project it to 3 channels, using Principal Component Analysis [10], and present it in Figure 7, along with the corresponding image and labels. One can observe that the visualized concept tensor indeed resembles different aspects of the input labels (e.g. edges and normals).

Geometric Image Editing with User Inputs. In order to demonstrate an intuitive application of our method, we perform image editing by inserting a new object into the scene. Figure 10 shows how our method can mimic rendering while allowing the geometric manipulation of an image. We render a TV in the given image, by simply augmenting different labels with user-provided labels for the TV.

5. Conclusion

In this work, we offer a new perspective on image generation as inverse of image understanding. In the same way as image understanding involves the solving of multiple different tasks, we desire the control over the generation process to include multiple input labels of various kinds. To this end, we design a neural network architecture that is capable of handling sparse and heterogeneous labels, by mapping them to a homogeneous concept space in a pixel-wise fashion. With our proposed module, we can equip spatially conditioned generators with the desired properties. From our experiments on challenging datasets, we conclude that the benefits and flexibility of this additional layer of control gives way to exciting results beyond the state-of-the-art.

Acknowledgments

This research was co-financed by Innosuisse under the project Know Where To Look, Grant No. 59189.1 IP-ICT.

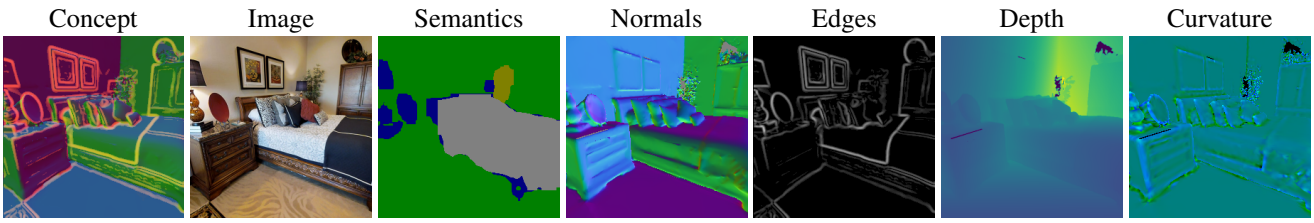


Figure 7. **Concept Tensor visualization.** From left to right: *Concept Tensor* projected to RGB; original image; five different input labels.

Figure 8. **Visual comparison on the Cityscapes.** Our approach achieves a visual quality on par with the compared methods.

Figure 9. **Visual comparison on NYU.** Our method generates images that better capture the lighting and geometry with more details.

Figure 10. **Object insertion.** From left to right: original image; mask of the inserted object; generated image using SPADE, ASAP and our method, respectively. This figure shows insertion of a TV, by inserting labels provided by the user at the location she/he chooses.

References

[1] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization, 2016.

[2] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*, 2018.

[3] Jinming Cao, Hanchao Leng, Dani Lischinski, Danny Cohen-Or, Changhe Tu, and Yangyan Li. Shapeconv: Shape-aware convolutional layer for indoor rgb-d semantic segmentation. *arXiv preprint arXiv:2108.10528*, 2021.

[4] Prashanth Chandran, Gaspard Zoss, Paulo Gotardo, Markus Gross, and Derek Bradley. Adaptive convolutions for structure-aware style transfer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7972–7981, June 2021.

[5] Qifeng Chen and Vladlen Koltun. Photographic image synthesis with cascaded refinement networks. pages 1520–1529, 10 2017.

[6] Robert L Cook, Loren Carpenter, and Edwin Catmull. The reyes image rendering architecture. *ACM SIGGRAPH Computer Graphics*, 21(4):95–102, 1987.

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

[8] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *ICLR*, 2021.

[9] Vincent Dumoulin, Ethan Perez, Nathan Schucher, Florian Strub, Harm de Vries, Aaron Courville, and Yoshua Bengio. Feature-wise transformations. *Distill*, 2018.

[10] Karl Pearson F.R.S. Liii. on lines and planes of closest fit to systems of points in space. *Philosophical Magazine Series 1*, 2:559–572.

[11] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. A neural algorithm of artistic style, 2015.

[12] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2414–2423, 2016.

[13] Clément Godard, Oisin Mac Aodha, and Gabriel J. Brostow. Unsupervised monocular depth estimation with left-right consistency. In *CVPR*, 2017.

[14] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

[15] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus), 2020.

[16] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *NIPS*, 2017.

[17] Lei Huang, Jie Qin, Yi Zhou, Fan Zhu, Li Liu, and Ling Shao. Normalization techniques in training dnns: Methodology, analysis and application, 2020.

[18] Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

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

[20] Yongcheng Jing, Xiao Liu, Yukang Ding, Xinchao Wang, Errui Ding, Mingli Song, and Shilei Wen. Dynamic instance normalization for arbitrary style transfer, 2019.

[21] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Computer Vision – ECCV 2016*, pages 694–711, Cham, 2016. Springer International Publishing.

[22] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks, 2019.

[23] Hiroharu Kato, Deniz Beker, Mihai Morariu, Takahiro Ando, Toru Matsuoka, Wadim Kehl, and Adrien Gaidon. Differentiable rendering: A survey, 2020.

[24] Hiroharu Kato, Yoshitaka Ushiku, and Tatsuya Harada. Neural 3d mesh renderer. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[25] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2015.

[26] Samuli Laine, Janne Hellsten, Tero Karras, Yeongho Seol, Jaakko Lehtinen, and Timo Aila. Modular primitives for high-performance differentiable rendering. *ACM Transactions on Graphics*, 39(6), 2020.

[27] Tzu-Mao Li, Miika Aittala, Frédo Durand, and Jaakko Lehtinen. Differentiable monte carlo ray tracing through edge sampling. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)*, 37(6):222:1–222:11, 2018.

[28] Kevin Lin, Lijuan Wang, and Zicheng Liu. End-to-end human pose and mesh reconstruction with transformers, 2021.

[29] Shichen Liu, Tianye Li, Weikai Chen, and Hao Li. Soft rasterizer: A differentiable renderer for image-based 3d reasoning, 2019.

[30] Guillaume Loubet, Nicolas Holzschuch, and Wenzel Jakob. Reparameterizing discontinuous integrands for differentiable rendering. *Transactions on Graphics (Proceedings of SIGGRAPH Asia)*, 38(6), Dec. 2019.

[31] Liqian Ma, Xu Jia, Qianru Sun, Bernt Schiele, Tinne Tuytelaars, and Luc Van Gool. Pose guided person image generation. In *NIPS*, 2017.

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

[33] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *ArXiv*, abs/1411.1784, 2014.

[34] Pushmeet Kohli Nathan Silberman, Derek Hoiem and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*, 2012.

[35] Merlin Nimier-David, Sébastien Speierer, Benoît Ruiz, and Wenzel Jakob. Radiative backpropagation: An adjoint method for lightning-fast differentiable rendering. *Transactions on Graphics (Proceedings of SIGGRAPH)*, 39(4), July 2020.

[36] Merlin Nimier-David, Delio Vicini, Tizian Zeltner, and Wenzel Jakob. Mitsuba 2: A retargetable forward and inverse renderer. *Transactions on Graphics (Proceedings of SIGGRAPH Asia)*, 38(6), Dec. 2019.

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

[38] Georgios Pavlakos, Vasileios Choutas, Nima Ghorbani, Timo Bolkart, Ahmed A. A. Osman, Dimitrios Tzionas, and Michael J. Black. Expressive body capture: 3d hands, face, and body from a single image, 2019.

[39] Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron Courville. Film: Visual reasoning with a general conditioning layer, 2017.

[40] Nikola Popovic, Danda Pani Paudel, Thomas Probst, Guolei Sun, and Luc Van Gool. Compositetasking: Understanding images by spatial composition of tasks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6870–6880, 2021.

[41] Xiaojuan Qi, Qifeng Chen, Jiaya Jia, and Vladlen Koltun. Semi-parametric image synthesis. pages 8808–8816, 06 2018.

[42] Nikhila Ravi, Jeremy Reizenstein, David Novotny, Taylor Gordon, Wan-Yen Lo, Justin Johnson, and Georgia Gkioxari. Accelerating 3d deep learning with pytorch3d. *arXiv:2007.08501*, 2020.

[43] Mr D Murahari Reddy, Mr Sk Masthan Basha, Mr M Chinnaiahgari Hari, and Mr N Penchalaiah. Dall-e: Creating images from text.

[44] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation, 2015.

[45] Tamar Rott Shaham, Michael Gharbi, Richard Zhang, Eli Shechtman, and Tomer Michaeli. Spatially-adaptive pixelwise networks for fast image translation. In *Computer Vision and Pattern Recognition (CVPR)*, 2021.

[46] Vincent Sitzmann, Michael Zollhöfer, and Gordon Wetzstein. Scene representation networks: Continuous 3d-structure-aware neural scene representations. *arXiv preprint arXiv:1906.01618*, 2019.

[47] Guolei Sun, Thomas Probst, Danda Pani Paudel, Nikola Popovic, Menelaos Kanakis, Jagruti Patel, Dengxin Dai, and Luc Van Gool. Task switching network for multi-task learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8291–8300, 2021.

[48] Ayush Tewari, Ohad Fried, Justus Thies, Vincent Sitzmann, Stephen Lombardi, Kalyan Sunkavalli, Ricardo Martin-Brualla, Tomas Simon, Jason Saragih, Matthias Nießner, Rohit Pandey, Sean Fanello, Gordon Wetzstein, Jun-Yan Zhu, Christian Theobalt, Maneesh Agrawala, Eli Shechtman, Dan B Goldman, and Michael Zollhöfer. State of the art on neural rendering, 2020.

[49] Ayush Tewari, Michael Zollöfer, Florian Bernard, Pablo Garrido, Hyeongwoo Kim, Patrick Perez, and Christian Theobalt. High-fidelity monocular face reconstruction based on an unsupervised model-based face autoencoder. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–1, 2018.

[50] Ayush Tewari, Michael Zollöfer, Hyeongwoo Kim, Pablo Garrido, Florian Bernard, Patrick Perez, and Theobalt Christian. MoFA: Model-based Deep Convolutional Face Autoencoder for Unsupervised Monocular Reconstruction. In *The IEEE International Conference on Computer Vision (ICCV)*, 2017.

[51] Ashish Vaswani, Noam M. Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *ArXiv*, abs/1706.03762, 2017.

[52] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Guilin Liu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. Video-to-video synthesis. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

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

[54] Weihao Xia, Yujiu Yang, Jing Xue, and Baoyuan Wu. Tedigan: Text-guided diverse face image generation and manipulation. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2256–2265, 2021.

[55] Fisher Yu, Vladlen Koltun, and Thomas Funkhouser. Dilated residual networks. In *Computer Vision and Pattern Recognition (CVPR)*, 2017.

[56] Ye Yu and William AP Smith. Inverserendernet: Learning single image inverse rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3155–3164, 2019.

[57] Li Yuan, Yunpeng Chen, Tao Wang, Weihao Yu, Yujun Shi, Francis E. H. Tay, Jiashi Feng, and Shuicheng Yan. Tokens-to-token vit: Training vision transformers from scratch on imagenet. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 538–547, 2021.

[58] Amir R. Zamir, Alexander Sax, William B. Shen, Leonidas J. Guibas, Jitendra Malik, and Silvio Savarese. Taskonomy: Disentangling task transfer learning. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2018.

[59] Zhu Zhang, Jianxin Ma, Chang Zhou, Rui Men, Zhikang Li, Ming Ding, Jie Tang, Jingren Zhou, and Hongxia Yang. M6-ufc: Unifying multi-modal controls for conditional image synthesis. *arXiv preprint arXiv:2105.14211*, 2021.

[60] Peihao Zhu, Rameen Abdal, Yipeng Qin, and Peter Wonka. Sean: Image synthesis with semantic region-adaptive normalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.