Variational Transformer Networks for Layout Generation

Diego Martin Arroyo\textsuperscript{1}
martinarroyo@google.com
\textsuperscript{1}Google, Inc

Janis Postels\textsuperscript{2}
jpostels@vision.ee.ethz.ch
\textsuperscript{2}ETH Zürich

Federico Tombari\textsuperscript{1,3}
tombari@google.com
\textsuperscript{3}Technische Universität München

Abstract

Generative models able to synthesize layouts of different kinds (e.g., documents, user interfaces or furniture arrangements) are a useful tool to aid design processes and as a first step in the generation of synthetic data, among other tasks. We exploit the properties of self-attention layers to capture high level relationships between elements in a layout, and use these as the building blocks of the well-known Variational Autoencoder (VAE) formulation. Our proposed Variational Transformer Network (VTN) is capable of learning margins, alignments and other global design rules without explicit supervision. Layouts sampled from our model have a high degree of resemblance to the training data, while demonstrating appealing diversity. In an extensive evaluation on publicly available benchmarks for different layout types VTNs achieve state-of-the-art diversity and perceptual quality. Additionally, we show the capabilities of this method as part of a document layout detection pipeline.

1. Introduction

Layouts, \textit{i.e.} the abstract positioning of elements in a scene or document, constitute an essential tool for various downstream tasks. Consequently, the ability to flexibly render novel, realistic layouts has the potential to yield significant improvements in many tasks, such as neural scene synthesis [36], graphic design or in data synthesis pipelines. Even though the task of synthesizing novel layouts has recently started to gain the attention of the deep learning community [23, 16, 22, 28], it is still a sparsely explored area and provides unique challenges to generative models based on neural networks, namely a non-sequential data structure consisting of varying length samples with discrete (classes) and continuous (coordinates) elements simultaneously.

Generative models based on neural networks have received a significant share of attention in recent years, as they proved capable of learning complex, high-dimensional distributions. Common formulations such as Generative Adversarial Networks (GANs) [8] and Variational Autoencoders (VAEs) [21] have shown impressive results in tasks such as image translation [43], image synthesis [17], and text generation [2]. A GAN is comprised of an arrangement of generator-discriminator neural networks in a zero-sum configuration, while a VAE learns a lower bound of the data distribution using an encoder-decoder neural network with a regularized bottleneck. Since these are general frameworks, they leave room for adapting the underlying neural architectures to exploit the properties of the data. For example, the weight sharing strategy of Convolutional Neural Networks (CNNs) renders them the most common building block for image processing, while for sequential data (e.g., text), Recurrent Neural Networks (RNNs) or attention modules are often the architecture of choice. In particular, the attention mechanism has recently demonstrated strong performance on a variety of tasks, such as language translation [35] and object detection [3], proving its superiority over RNNs regarding modeling long-term relationships.

Prior work has built the foundation by proving the ef-
fectiveness of deep learning to generate novel documents [22, 28, 9], natural scenes [16] and User Interface (UI) designs [22]. Mostly, the location and size of a given element depends not only on the particularities of its type (e.g., titles tend to be small and at the top of a document, while figures or tables usually occupy a significant amount of space), but also on their relationship to other elements. One way to incorporate this knowledge into modeling a layout distribution is to define handcrafted rules, (e.g., enforcing margins, alignment, the allowed number of elements in a document...). However, such rules are subjective, hard to define unambiguously and certainly do not generalize to arbitrary layout distributions. Consequently, we refrain from modeling any prior knowledge by i.e. enforcing heuristics, and instead equip the neural architecture itself with an inherent bias towards learning the relationship between elements in a layout. This makes the attention mechanism a suitable fundamental architectural component, since it naturally models many-to-many relationships and is, thus, particularly suitable for discovering relationships in a given layout distribution in an unsupervised manner.

By instantiating the VAE framework with an attention-based architecture, this work investigates an important gap in literature. We explore relevant design choices in great detail - e.g. autoregressive vs. non-autoregressive decoder, learned vs. non-learned prior. Furthermore, we tailor our novel approach to the yet under-explored task of layout generation, where we demonstrate state-of-the-art performance across various metrics on several publicly available datasets. To summarize, our main contributions are:

- A novel generative model specialized in layout generation that incorporates an inductive bias towards high-level relationships between a large number of elements in a layout without annotations.
- Exploration of strategies for creating a variational bottleneck on sequences with varying lengths.

2. Related work

Layout synthesis The task of layout synthesis has not yet been exhaustively covered by literature, but fueled increasing interest in the research community in recent years. LayoutGAN [23] is, to the best of our knowledge, the first paper to apply generative models (in particular GANs) to this task. The authors use a network to synthesize bounding box annotations. In order to use a CNN as discriminator, LayoutGAN applies a novel differential rendering module to turn a collection of bounding boxes into an image. Similarly to our approach, it uses self-attention to model many-to-many relationships. However, the authors only evaluate single-column documents with at most nine elements, which corresponds to much sparser layouts than provided by common publicly available datasets.

LayoutVAE [16] proposes an autoregressive model based on a conditional VAE with a conditional prior (conditioned on the number and type of elements in the layout). The authors use an LSTM to aggregate information over time. Additionally they propose using a second conditional VAE to model the distribution of category counts which is used as conditional information during layout generation. Their underlying neural architecture is comprised of fully-connected layers and LSTMs. Consequently, it is expected that LayoutVAE struggles to model layouts with a large number of elements, since LSTMs do not explicitly model the relationships of all components. Unlike LayoutVAE, our work explicitly biases the underlying neural network towards learning the relationships between elements in a layout, and only makes the decoder autoregressive (reducing the computational costs). Further, we only train a single VAE for learning the layout distribution instead of resorting to two separate VAEs.

In Neural Design Networks the authors [22] generate document layouts with an optimal set of design constraints. Initially, a complete graph for modeling the relationships between elements is built. The distribution of these relationships is learned using a VAE based on Graph Convolution Networks (GCNs), where the labels of the relationships are based on heuristics. The actual layout is subsequently generated by a separate GCN. The resulting raw layout is then polished by an additional refinement network. In contrast to Neural Design Networks, this work does not rely on labels extracted using heuristics on the training data for learning a layout distribution, which is prone to introduce ambiguities and unlikely to generalize across datasets. Moreover, our approach learns the layout distribution end-to-end without relying on training three separate neural networks.

Similarly, READ [28] also uses heuristics to determine the relationships between elements and then trains a VAE which is based on Recursive Neural Networks (RvNNs) [7] to learn the layout distribution.

Content-aware Generative Modeling of Graphic Design Layouts [39] trains a VAE conditioned on images, keywords, attributes of the layout and corresponding coordinates. However, the authors focus on learning the layout distribution conditioned on additional user input.

Layout Generation and Completion with Self-attention [9] is most relevant to this work. The authors perform self-supervised training (i.e. layout completion) using an autoregressive decoder motivated by Transformers [35]. Subsequently, novel layouts are synthesized using beam search [27]. While this generation approach can yield strong results, it requires optimizing additional hyperparameters (e.g. beam size) and, more importantly, it does not have any theoretical guarantees for learning the actual data distribution. The resulting distribution rather depends on finding the right level of regularization at training time.
Only if the model is regularized appropriately beam search will yield outcomes of sufficient diversity. Since this generation process lacks theoretical guarantees for capturing the full diversity of the layout distribution and heavily relies on heuristics, we directly approximate the distribution using an attention-based VAE instead.

Some works have been proposed with particular focus on furniture arrangement [36, 12]. In the method of Wang et al. [36], one CNN places elements in a room by estimating the likelihood of each possible location, while a second CNN determines when the scene is complete. [30] extends this to model orientations and room dimensions. Moreover, Henderson et al. [12] propose to learn a distribution for each element type and model high-order relationships between objects using a direct acyclical graph. Since all of these methods use the now unavailable SUNCG dataset [33] for training, establishing a comparison with them is difficult.

Additionally, Tab. 1 provides a high-level comparison between this work and the most relevant adjacent methods. We differentiate existing works along four important dimensions: 1) Are models equipped with inductive biases towards learning the relationships between elements? 2) Are these relationships learned without supervision or are there additional labels, using e.g. heuristics, necessary? 3) Can layouts contain an arbitrary number of elements? 4) Does the learning approach provide guarantees for learning the underlying distribution by applying probabilistic methods?

Attention-based VAEs are a recent development in the Natural Language Processing (NLP) literature. The common goal is to learn the distribution of real data more accurately than with deterministic self-supervised approaches [25, 26, 37]. To combine Transformers and VAEs [26] uses self-attention layers for the encoder and decoder components. The encoder turns a sentence into a collection of high-dimensional vectors of the same length as the input. These constitute the VAE bottleneck, and are passed after re-parameterization to the decoder to reconstruct the sentence. By feeding a set of vectors sampled from the prior, a sentence of the same length can be generated. Further, [25] implements a conditional VAE (conditioned on the context of a conversation) based on the Transformer to improve diversity on the task of response generation. [37] develops a Transformer-based VAE to enhance variability on the task of story completion. Their encoder and decoder share weights while the bottleneck of their VAE is fed into the penultimate layer of the decoder.

### 3. Variational Transformer Networks

This section illustrates the proposed Variational Transformer Networks. From a high-level perspective VTNs are an instance of the VAE framework tailored to the task of layout synthesis, where the main building blocks of the neural networks parameterizing the encoder and decoder are attention layers. Firstly, we briefly revisit the concept of VAEs. Subsequently, we explain how VTNs exploit the data format of layouts, their architecture and how to train them.

#### 3.1. Variational Autoencoders

VAEs are a family of latent variable models that approximate a data distribution \( P(X) \) by maximizing the evidence lower bound (ELBO) [21]

\[
\mathcal{L}(\theta, \phi) = \mathbb{E}_{z \sim q_\phi(z|x)} \log (p_\theta(x|z)) - KL (q_\theta(z|x) || p(z))
\]

where \( p_\theta(x|z) \) denotes a decoder parameterized by a neural network with parameter \( \phi \), \( q_\theta(z|x) \) is the approximate posterior distribution, similarly parameterized by a neural network with weights \( \theta \), and \( p(z) \) the prior distribution.

#### 3.2. Exploiting the Data Format of Layouts

The central aspect of layout generation is its unique underlying data format. Layouts are sets of elements of variable size, where each element can be described by both discrete and continuous features. More formally, each layout \( x \) in a given dataset \( X \) consists of a variable number \( l \) of bounding boxes. Further, each bounding box \( x_i \) with \( i \in [1, \ldots, l] \) contains information about its class (for documents, e.g. text, image...), location and dimension.

Another important characteristic of layout datasets is that there exists a high degree of correlation between the individual elements in a layout. For example, in case of document layouts, titles tend to be positioned at the top of a text. It is therefore essential to bias an approach for learning layout distributions towards exploiting the relationships between elements. While some methods introduce additional features, such as annotations for the relationships between elements [22, 28], our approach instead relies solely on bounding box annotations, since additional features are expensive.

<table>
<thead>
<tr>
<th>Inductive Bias</th>
<th>Unsupervised Relationship Learning</th>
<th>Arbitrary Size</th>
<th>Distribution Learning</th>
</tr>
</thead>
<tbody>
<tr>
<td>LayoutGAN [23]</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>LayoutVAE [16]</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>READ [28]</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>NDN [22]</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Gupta et al. [9]</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Ours</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

Table 1: Comparison with existing methods. We consider whether methods 1) equip their models with inductive biases towards learning the relationships between elements, 2) learn relationships unsupervised, 3) allow layouts of arbitrary size and 4) have guarantees for learning the underlying distribution by applying probabilistic methods.
to create, prone to ambiguity and fail to generalize across datasets. Therefore, we introduce an inductive bias to learn from the relationships by using an attention-based neural network. Notably, the attention mechanism is an ideal candidate for exploiting pairwise relationships, since it leverages the pairwise dot product as its fundamental computational block for guiding the information flow.

Moreover, the attention mechanism also helps modeling another aspect of the data - namely a varying and large number of elements. To mitigate this problem other works have restricted the maximum number of elements occurring in one layout [23, 22]. However, attention-based architectures are well suited for learning the relationships of a large number of elements, since this is one of the reasons for their success in the NLP literature [35]. Notably, RNNs, as used by LayoutVAE [16], are also capable of modeling a varying number of elements in a layout. However, they struggle with long-term dependencies, i.e. a large number of elements in a layout. This follows from results in the NLP literature and is also observed by us (see section 4.4).

3.3. Architecture of VTNs

The architecture of VTNs is based on Transformers [35], which are sequence models that consist of an encoder-decoder architecture, where both encoder and decoder use attention layers as their fundamental building blocks. We refer to fig. 2 for a schematic overview of our approach.

The encoder of the Transformer architecture parameterizes the posterior distribution \( q_\theta(z|x) \) in the VAE framework. In particular, \( q_\theta(z|x) \) is parameterized as a multivariate normal distribution with diagonal covariance matrix, whose parameters are determined by the output of the encoder network. To train the encoder using backpropagation, we apply the local re-parameterization trick [20]. The original Transformer is a highly specialized language model, which is usually trained on vast quantities of text data. Therefore, it is necessary to adjust the hyper-parameters. It is essential to keep the number of attention heads large (here \( n_{\text{heads}} = 8 \)) to average out outliers from individual attention heads [35]. Similarly, we keep a large model dimensionality \( d_{\text{model}} = 512 \) and size of the point-wise feed-forward layers \( d_{\text{ff}} = 2048 \). However, we find that the number of attention-blocks (see [35]) can be reduced to four without performance loss. This hints that relationships between elements in a layout are less complex than between words in language. We further omit the positional encodings used in the context of NLP since the features of bounding boxes already contain positional information.

The decoder \( p_\phi(x|z) \) in VTNs is a mirrored version of the encoder. Note that this breaks with [35] which adds additional attention-layers whose keys and queries are the output of the encoder. We empirically find that feeding the output of the encoder as an input to the first layer of the decoder yields better results. We further experiment with another major architectural choice: the autoregressive decoder, i.e. \( p_\phi(x|z) = \prod_{i=1}^l p_\phi(x_i|x_{i-1}, z) \), where \( l \) denotes the number of bounding boxes in a layout, and a non-autoregressive variant. While the former has more representational power, since theoretically any distribution can be modeled as an autoregressive one, it is also more prone to posterior collapse due to the expressive decoder [2] and requires more computational resources.

Furthermore, we consider two distinct prior distributions. First, we use the common choice of a fixed multivariate zero-mean normal distribution. However, this often proves too restrictive for learning the true posterior distribution [4]. In principle there are two avenues to mitigate this issue: use a more expressive parameterization of the posterior [19] or the prior distribution [4]. In this work we attempt to extend the expressiveness of the prior distribution by learning the parameters of the multivariate normal distribution with a diagonal covariance matrix. Since layouts consist of a varying number of bounding boxes, we parameterize the distribution with an LSTM [14].

Importantly, while an autoregressive decoder enables sampling of layouts with varying number of elements - e.g. by introducing symbols for start/end of the layout - the non-autoregressive decoder requires incorporating this into the prior distribution. Therefore, we model the prior in this case as \( p(z,s) = p(z|s)p(s) \) where \( s \) denotes the number of bounding boxes. We learn \( p(s) \) during training by counting the number of occurrences of each sequence length.

Finally, we note that in the case of the autoregressive decoder, we find empirically that aggregating the latent representations \( z \) across all elements in a layout yields better perceptual quality. This corresponds to parameterizing the posterior distribution with the output of the encoder aggregated along the dimension of the layout elements. To this end we follow BERT [6] where the final hidden state of the encoder for the first token is used to represent the entire sequence, and used as the first element in the the decoder input. In the case of the non-autoregressive decoder we do not aggregate the latent representations, but feed them directly with variable dimensionality to the decoder.

3.4. Optimizing VTNs

Since we are learning the layout distribution using a VAE, we optimize the ELBO defined in eq. (1). However, a practical optimization challenge of VAEs is the so-called posterior collapse [2, 10]. The decoder ignores the information in the latent representation and collapses onto modes of the data distribution. At the same time the posterior distribution parameterized by the encoder can perfectly match the prior distribution, since it does not need to transmit information to the decoder. Therefore, this work follows a common heuristic by optimizing the \( \beta \)-VAE objective in-
Figure 2: VTN. The encoder and decoder are parameterized by attention-based neural networks. This biases the network to learn relationships between the layout elements and enables processing layouts of arbitrary size. During training (black arrow) the reconstruction loss and the KL-divergence between the prior \( p(z) \) and the approximate posterior distribution are minimized. During inference (red arrow) we sample latent representations \( z \) from the prior and transform those into layouts using our self-attention-based decoder.

Instead of eq. (1)

\[
L(\theta, \phi) = \mathbb{E}_{z \sim q_{\phi}(z|x)} \left[ \log(p_{\phi}(x|z)) \right] - \beta KL(q_{\phi}(z|x) || p(z))
\]

(2)

To optimize eq. 2 we use Adam [18] and follow the learning rate schedule in [35]. To further reduce the risk of posterior collapse, it is common to increase \( \beta \) at the beginning of training from zero to the desired value. Specifically, we implement the exponential beta schedule proposed by [2, 26]. In all our experiments we use \( \beta = 1 \) with the autoregressive decoder and \( \beta = 0.5 \) with the non-autoregressive decoder.

Moreover, we follow [9] in discretizing the location, width and height of the bounding boxes. Thus each bounding box is represented by a feature vector containing a one-hot encoding of the class concatenated with the one-hot encodings representing the above discretization. We use categorical cross-entropy as a reconstruction loss.

**Implementation Details** We implement our method using Tensorflow 2 [1] and a NVIDIA V100 GPU for acceleration. We train using the Adam optimizer with a batch size of 64 for 30 epochs in the case of the autoregressive decoder and 50 epochs using the non-autoregressive version.

### 4. Experiments

#### 4.1. Datasets

We evaluate our method on the following publicly available datasets of layouts for documents, natural scenes, furniture arrangements and mobile phone UIs.

- **PubLayNet** [42] contains 330K samples of machine-annotated scientific documents crawled from the Internet. It has the categories `text`, `title`, `figure`, `list`, `table`.
- **RICO** [5] is a dataset of user interface designs for mobile applications. It contains 91K entries with 27 element categories (`button`, `toolbar`, `list item`...). Due to memory constraints we omit layouts with more than 100 elements\(^1\), in total removing 0.031% of the data.
- **COCO** [24] Contains \(~100K\) images of natural scenes. We use the `Stuff` variant, which contains 80 `thing` and 91 `stuff` categories, removing small bounding boxes (\(\leq 2\% \) image area), as well as instances that are tagged as "iscrowd".
- **SUN RGB-D** [32] is a scene understanding dataset with 10000 samples, including scenes from [31], [15] and [38]. The annotations comprise different household objects. We compute the 2D bounding boxes of the semantic regions from a top-down perspective.

#### 4.2. Evaluation methodology

It is important to evaluate layouts along two high-level dimensions - perceptual quality and diversity. Note that in the case of layouts perceptual quality is prone to subjectivity and different aspects must be considered from dataset to dataset. It is thus difficult to define a single metric that entirely covers both aspects. We therefore resort to a set of metrics where each aims at representing an individual aspect of either perceptual quality or diversity.

- **Alignment and overlap.** Some datasets, such as PubLayNet or RICO, consist of entries with strictly defined alignments and small overlaps between bounding boxes. Consequently, these properties are an indicator of the perceptual quality of synthesized layouts. We follow LayoutGAN [23] in measuring overlaps using the total overlapping area among any two bounding boxes inside the whole page (overlap index) and the average Intersection over Union (IoU) between elements. Additionally, we quantify alignment using the alignment loss proposed by [22].

- **Unique matches under DocSim metric.** We use the

\(^1\)Note that this restriction originates from memory constraints and does not imply that our approach is not capable of learning larger layouts given sufficient memory.
number of unique matches between real sets of layouts and synthesized layouts as a proxy for diversity. We use the DocSim metric [28] as a similarity metric. Note that, while the number of unique matches primarily analyzes diversity, it also partially reflects perceptual quality.

**Wasserstein distance.** A rigorous approach to evaluate diversity would be computing the Wasserstein distance between the real and learned data distributions. Unfortunately this is infeasible. However, we can approximate the Wasserstein distance between real and generated data for two marginal distributions - the class distribution (discrete) and the bounding box distribution (continuous, 4-d vectors \((x_{\text{center}}, y_{\text{center}}, \text{width}, \text{height})\)). In practice, we compute these Wasserstein distances from a finite set of samples.

### 4.3. Quantitative results

**Comparison to state of the art.** A comparison to any of the methods described in section 2 is difficult, since, to the best of our knowledge, none has a publicly available implementation. Similar to the authors of LayoutVAE [16], we were unable to reproduce the results of LayoutGAN [23] on documents. We reimplement LayoutVAE and the approach of Gupta et al. [9]. In the LayoutVAE case, we follow [9] and sample category counts from the test dataset. For Gupta et al., we use a mixture of nucleus sampling with \(p = 0.9\) and top-\(k\) sampling with \(k = 30\). As suggested by the authors, we found nucleus sampling to improve the diversity of the synthesized layouts. We, further, compare against NDN [28] on RICO using their proposed alignment metric.

In tab. 3 we ablate our model on the prior type and the decoding strategy. We observe that, while the autoregressive decoder slightly decreases diversity (Wasserstein distance and bounding box and number of unique matches), it yields large improvements regarding perceptual quality (IoU, overlap index and alignment). Moreover, in the case of the non-autoregressive decoder a learned prior yields improvements regarding perceptual quality. However, when using an autoregressive decoder the learned and non-learned priors yield similar results. Therefore, we apply an autoregressive decoder with a non-learned prior in the remaining experiments, since it strikes the optimal balance between diversity, perceptual quality and model simplicity.

We report quantitative results for the aforementioned metrics on different datasets. Unless explicitly stated, all metrics are computed on 1000 samples, and the value is averaged across 5 trainings with different random initialization. In tabs. 2 and 4 we show the results of our method in comparison to existing art. We show that our method produces a large number of distinct layouts that have similar alignment metrics as the real data. Furthermore, we clearly outperform LayoutVAE [16] across all metrics and demonstrate improved diversity at similar perceptual quality compared to [9], as expected since our method explicitly approximates the layout distribution. Given that both LayoutVAE and Gupta et al. generate layouts autoregressively and considering our ablation in tab. 3, we note that autoregressive modeling denotes an important element of learning layout distributions.

Furthermore, in tab. 5 we also compare our approach to Lee et al. [22] on the RICO dataset using their proposed alignment metric. We demonstrate superior results when no explicit design constraints are given (NDN-none), showing that our method is better at discovering relationships without supervision. Even in the NDN-all case, where all relationships are given to the network, we show similar performance despite not relying on this information.

### 4.4. Qualitative results

We show qualitative results for PubLayNet in fig. 3, as well as a qualitative comparison with existing methods in fig. 4. In alignment with the quantitative results in section 4.3, we observe that our approach and [9] yield similar perceptual quality. Furthermore, LayoutVAE [16] struggles to model layouts with a large number of elements. As previously discussed, this results from the application of RNNs, which are inferior at modeling the relationships between a large number of elements compared with the attention mechanism. In fig. 5 we show synthetic samples for RICO as well as the closest DocSim match in the real dataset. We show similar results for SUN RGB-D in fig. 7. In order to show the capabilities of our method on the task of natural scene generation, we train our model on the COCO-Stuff dataset. In fig. 6 we show samples from our network. For better understanding we feed our generations to a pretrained instance of LostGAN [34]. These results show that our method is capable of capturing relationships between elements regardless of their distance or position in the input sequence. This is observed by the strict margins modeled by our network, which resemble those of the real data. In the case of COCO or SUN RGB-D, we show how the network identifies joint occurrences of different elements (e.g. giraffe and tree, person and playingfield or table and chair).

### 4.5. Layout detection

This experiment demonstrates the benefit of our approach regarding data augmentation for a downstream task. Document understanding comprises multiple tasks that go beyond simple Optical Character Recognition (OCR). Understanding the arrangement of different pieces of text and images and their boundaries (the document layout) is also necessary for applications such as text extraction or to determine the reading order in a complex document. While

---

2Though the LayoutGAN authors recently released an implementation, they only did so for a toy example on MNIST: https://github.com/JiananLi2016/LayoutGAN-Tensorflow

3https://github.com/VMCL/LostGANs
we show several exam-
al dataset guided by the ground truth annotations and use
is labor-intensive, therefore, we crop these from the origi-
ing realistic text, images, tables or lists for a given domain
OCR-annotated data is quite abundant, this is not the case
for layout detection. Annotating documents is a tedious pro-
cess which is prone to ambiguity, as the rules that define e.g.
what a paragraph is are often subjective. This ambigu-
guity also makes automatic annotators based on heuristics
fail or be constrained to specific domains. Most works,
such as PubLayNet, LayoutLM or LayoutVAE are based
on object detection backbones using CNNs. Here, we use
our method to create a training dataset for a layout detector
on the PubLayNet dataset. We use the bounding boxes
generated by our method to guide the rendering. Obtain-
ing realistic text, images, tables or lists for a given domain
is labor-intensive, therefore, we crop these from the original
data set guided by the ground truth annotations and use
the most appropriate one for a particular box according to
its class and dimensionality. This approach ensures that the
aspect ratio is preserved. In fig. 3 we show several exam-

<table>
<thead>
<tr>
<th>Method</th>
<th>Alignment</th>
</tr>
</thead>
<tbody>
<tr>
<td>NDN-none [22]</td>
<td>0.91 ± 0.030</td>
</tr>
<tr>
<td>NDN-all [22]</td>
<td>0.92 ± 0.020</td>
</tr>
<tr>
<td>Ours</td>
<td>0.37 ± 0.009</td>
</tr>
<tr>
<td>Real data</td>
<td>0.0012</td>
</tr>
</tbody>
</table>

Table 5: Comparison between Neural Design Network [22] and our approach using their proposed alignment metric on RICO.

OCR-annotated data is quite abundant, this is not the case
for layout detection. Annotating documents is a tedious pro-
cess which is prone to ambiguity, as the rules that define e.g.
what a paragraph is are often subjective. This ambigu-
yuity also makes automatic annotators based on heuristics
fail or be constrained to specific domains. Most works,
such as PubLayNet, LayoutLM or LayoutVAE are based
on object detection backbones using CNNs. Here, we use
our method to create a training dataset for a layout detector
on the PubLayNet dataset. We use the bounding boxes
generated by our method to guide the rendering. Obtain-
ing realistic text, images, tables or lists for a given domain
is labor-intensive, therefore, we crop these from the original
data set guided by the ground truth annotations and use
the most appropriate one for a particular box according to
its class and dimensionality. This approach ensures that the
aspect ratio is preserved. In fig. 3 we show several exam-

<table>
<thead>
<tr>
<th>RICO</th>
<th>IoU</th>
<th>Overlap</th>
<th>Alignment</th>
<th>W class ↓</th>
<th>W bbox ↓</th>
<th># unique matches ↑</th>
</tr>
</thead>
<tbody>
<tr>
<td>[16]</td>
<td>0.193</td>
<td>0.400</td>
<td>0.416</td>
<td>-</td>
<td>0.045</td>
<td>496</td>
</tr>
<tr>
<td>[9]</td>
<td>0.086</td>
<td>0.145</td>
<td>0.366</td>
<td><strong>0.004</strong></td>
<td>0.023</td>
<td>604</td>
</tr>
<tr>
<td>Ours</td>
<td>0.115</td>
<td>0.165</td>
<td>0.373</td>
<td>0.007</td>
<td><strong>0.018</strong></td>
<td><strong>680</strong></td>
</tr>
<tr>
<td>Real</td>
<td>0.084</td>
<td>0.175</td>
<td>0.410</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>COCO</th>
<th>IoU</th>
<th>Overlap</th>
<th>Alignment</th>
<th>W class ↓</th>
<th>W bbox ↓</th>
<th># unique matches ↑</th>
</tr>
</thead>
<tbody>
<tr>
<td>[16]</td>
<td>0.325</td>
<td>2.819</td>
<td>0.246</td>
<td>-</td>
<td>0.062</td>
<td>700</td>
</tr>
<tr>
<td>[9]</td>
<td>0.194</td>
<td>1.709</td>
<td>0.334</td>
<td>0.001</td>
<td>0.016</td>
<td>601</td>
</tr>
<tr>
<td>Ours</td>
<td>0.197</td>
<td>2.384</td>
<td>0.330</td>
<td><strong>0.0005</strong></td>
<td><strong>0.013</strong></td>
<td><strong>776</strong></td>
</tr>
<tr>
<td>Real</td>
<td>0.192</td>
<td>1.724</td>
<td>0.347</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 4: Extension of Tab. 2 for RICO, COCO

Table 3: Quantitative ablation study on PubLayNet. We generate 1000 layouts and compare them in average IoU, overlap index, alignment, Wasserstein (W) distance of the classes and bounding boxes to the real data and the number of unique matches according to DocSim. Ours (autoregressive) denotes using an autoregressive decoder.

<table>
<thead>
<tr>
<th>Method</th>
<th>Alignment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ours (autoregressive)</td>
<td><strong>813 ± 51</strong></td>
</tr>
<tr>
<td>Learned prior</td>
<td>794 ± 34</td>
</tr>
<tr>
<td>Autoregressive decoder</td>
<td>677 ± 16</td>
</tr>
<tr>
<td></td>
<td>697 ± 13</td>
</tr>
</tbody>
</table>

Table 2: Quantitative evaluation on PubLayNet. We generate 1000 layouts with each method and compare them regarding average IoU, overlap index, alignment, Wasserstein (W) distance of the classes and bounding boxes to the real data and the number of unique matches according to the DocSim. Ours (autoregressive) denotes using an autoregressive decoder.
outs alone are capable of achieving a good accuracy score. Training images ("Real PubLayNet"). We compare the mean box annotations ("Real layouts"), as well as with actual training images ("Real layouts"). We compare the mean average precision (mAP) at 0.5 IoU. Our synthesized layouts alone are capable of achieving a good accuracy score.

Figures 4 and 5: Qualitative comparison between LayoutVAE, Gupta et al. and our method on PubLayNet. The RNN of LayoutVAE struggles with a large number of elements. Figure 4 shows the generated layouts for RICO and their associated DocSim matches with the corresponding image. Table 6: Detection accuracy scores for a layout detection model trained with synthetic and real data.

5. Conclusion and future work

This work proposes self-attention layers as fundamental building blocks of a VAE and develops a solution tailored to layout synthesis, evaluating it on a diverse set of public datasets. Our approach yields state-of-art quantitative performance across all our metrics (see section 4.3) and layout samples of appealing perceptual quality (see section 4.4). We observe that autoregressive decoding constitutes an important ingredient to obtain high quality layouts. We also demonstrate its applicability as a data synthesizer for the downstream task of layout detection (see section 4.5). However, we also note that our proposal can still be improved in promising future research directions. Namely, learning to generate additional properties (e.g., font or text size) or the dimensions of the layout, which could be useful for documents with varying size, (e.g., leaflets). Moreover, it could be interesting to incorporate an end-to-end approach for layout synthesis, such as ours, into a scene synthesis pipeline.
References


