READ: Recursive Autoencoders for Document Layout Generation

Akshay Gadi Patil[†]
Simon Fraser University

Omri Ben-Eliezer[†]
Tel-Aviv University

Or Perel
Amazon

Hadar Averbuch-Elor[§]
Cornell Tech, Cornell University

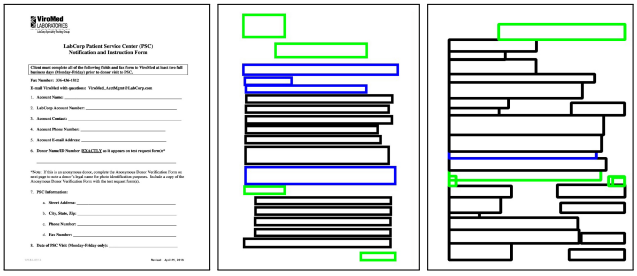
Abstract

Layout is a fundamental component of any graphic design. Creating large varieties of plausible document layouts can be a tedious task, requiring numerous constraints to be satisfied, including local ones relating different semantic elements and global constraints on the general appearance and spacing. In this paper, we present a novel framework, coined READ, for REcursive Autoencoders for Document layout generation, to generate plausible 2D layouts of documents in large quantities and varieties. First, we devise an exploratory recursive method to extract a structural decomposition of a single document. Leveraging a dataset of documents annotated with labeled bounding boxes, our recursive neural network learns to map the structural representation, given in the form of a simple hierarchy, to a compact code, the space of which is approximated by a Gaussian distribution. Novel hierarchies can be sampled from this space, obtaining new document layouts. Moreover, we introduce a combinatorial metric to measure structural similarity among document layouts. We deploy it to show that our method is able to generate highly variable and realistic layouts. We further demonstrate the utility of our generated layouts in the context of standard detection tasks on documents, showing that detection performance improves when the training data is augmented with generated documents whose layouts are produced by READ.

1. Introduction

"Do not read so much, look about you and think of what you see there." *-Richard Feynman*

Layouts are essential for effective communication and targeting one's visual attention. From newspapers articles, to magazines, academic manuscripts, websites and various other document forms, layout design spans a plethora of real world document categories and receives the foremost editorial consideration. However, while the last few years have



Real document | Real layout | Generated layout

Figure 1. Given a collection of training examples – annotated layouts (middle) of real-world documents (such as the fillable form on the left) – our method generates synthetic layouts (right) resembling those in the training data. Semantically labeled regions are marked in unique colors.

experienced growing interests among the research community in generating novel samples of images [8, 20], audio [19] and 3D content [11, 13, 29, 30], little attention has been devoted towards automatic generation of large varieties of plausible document layouts. To synthesize novel layouts, two fundamental questions must first be addressed. What is an appropriate representation for document layouts? And how to synthesize a new layout, given the aforementioned representation?

The first work to explicitly address these questions is the very recent LayoutGAN of Li et al. [12], which approaches layout generation using a generative adversarial network (GAN) [5]. They demonstrate impressive results in synthesizing plausible document layouts with up to nine elements, represented as bounding boxes in a document. However, various types of highly structured documents can have a substantially higher number of elements – up to tens or even hundreds.[1] Furthermore, their training data constitutes about 25k annotated documents, which may be difficult to obtain for various types of documents. Two natural questions therefore arise: Can one devise a generative method to synthesize highly structured layouts with a large number of entities? And is it possible to generate synthetic document layouts without requiring a lot of training data?

In this work, we answer both questions affirmatively.

[†] work done as an Intern at Amazon
[§] work done while working at Amazon

[1] As an example, consider the popular US tax form 1040; See https://www.irs.gov/pub/irs-pdf/f1040.pdf.

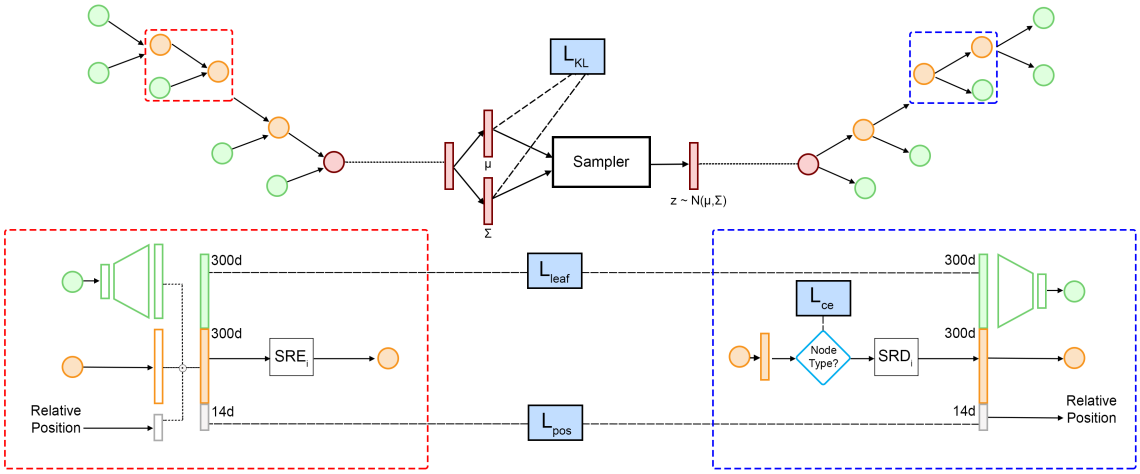


Figure 2. Overview of our RvNN-VAE framework. Training hierarchies are constructed for every document in the dataset. These hierarchies are mapped to a compact code (in a recursive fashion according to the encoder network marked in red), the space of which is approximated by a Gaussian distribution. Novel hierarchies can be sampled from this space (and decoded recursively according to the decoder network marked in blue), obtaining new document layouts.

Structured hierarchies are natural and coherent with human understanding of document layouts. We thus present *READ*: a *generative recursive* neural network (RvNN) that can appropriately model such structured data. Our method enables generating large quantities of plausible layouts containing dense and highly variable groups of entities, using just a few hundreds of annotated documents. With our approach, a new document layout can be generated from a random vector drawn from a Gaussian in a fraction of a second, following the pipeline shown in Figure 2.

Given a dataset of annotated documents, where a single document is composed of a set of labeled bounding boxes, we first construct document hierarchies, which are built upon connectivity and implicit symmetry of its semantic elements. These hierarchies, or trees, are mapped to a compact code representation, in a recursive bottom-up fashion. The resulting fixed length codes, encoding trees of different lengths, are constrained to roughly follow a Gaussian distribution by training a Variational Autoencoder (VAE). A novel document layout can be generated by a recursive decoder network that maps a randomly sampled code from the learned distribution, to a full document hierarchy. To evaluate our generated layouts, we introduce a new combinatorial metric (`DocSim`) for measuring layout similarity among structured multi-dimensional entities, with documents as a prime example. We use the proposed metric to show that our method is able to generate layouts that are representative of the latent distribution of documents which it was trained on. As one of the main motivations to study synthetic data generation methods stems from their usefulness as training data for deep neural networks, we also consider a standard document analysis task. We augment the available training data with synthetically generated documents whose layouts are produced by READ, and demonstrate that our augmentation boosts the performance of the network for the aforementioned document analysis task.

2. Related Work

Analysis of structural properties and relations between entities in documents is a fundamental challenge in the field of information retrieval. While local tasks, like optical character recognition (OCR) have been addressed with very high accuracy, the global and highly variable nature of document layouts has made their analysis somewhat more elusive. Earlier works on structural document analysis mostly relied on various types of specifically tailored methods and heuristics (e.g., [2, 3, 9, 18] Recent works have shown that deep learning based approaches significantly improve the quality of the analysis; e.g., see the work of Yang et al. [32], which uses a joint textual and visual representation, viewing the layout analysis as a pixel-wise segmentation task. Such modern deep learning based approaches typically require a large amount of high-quality training data, which call for suitable methods to synthetically generate documents with real-looking layout [12] and content [14]. Most recently, the LayoutVAE work of Jyothi et al. [7] uses variational autoencoders for generating stochastic scene layouts. Our work continues the line of research on synthetic layout generation, showing that our synthetic data can be useful to augment training data for document analysis tasks.

Maintaining reliable representation of layouts has shown to be useful in various graphical design contexts, which typically involve highly structured and content-rich objects. The most related work to ours is the very recent Layout-

GAN of Li et al. [12], which aims to generate realistic document layouts using a generative adversarial networks (GAN) with a wireframe rendering layer. Zheng et al. [33] also employ a GAN-based framework in generating documents, however, their work focuses mainly on content-aware generation, using the content of the document as an additional prior. Unlike Convolutional Neural Networks (CNNs) that operate on large dimensional vectors and involve multiple multi-channel transformations, in our work, we use recursive neural networks, which operate on low-dimensional vectors and employ two-layer perceptrons to merge any two vectors. Hence, they are computationally cheaper, plus can learn from just a few training samples.

Deka et al. [4] use an autoencoder to perform layout similarity search to simplify UI design for mobile applications. Ritchie et al. [23] present a design exploration tool for layout and content based retrieval of similarly looking web pages. O'Donovan et al. [17] present an interactive energy-based model that allows novice designers to improve their page layout design. Swearngin et al. [27] apply layout analysis to allow designers to manipulate layouts obtained from screenshots. More fundamentally, Talton et al. [28] leverage learned visual-structural and textual patterns learned from the data to obtain a formal grammar allowing to probabilistically generate new, similarly looking entities.

Recursive neural networks (RvNN) were first introduced by Socher et al. [25, 26] for parsing natural scenes and natural language sentences. Socher et al. [24] comprehensively present applications of RvNNs for various tasks in computer vision. However, RvNNs did not enjoy as much attention as CNNs, until recently, when RvNNs coupled with generative models were shown to work effectively on previously unexplored paradigms such as generating 3D shape structures [11, 34] and indoor 3D scenes [13]. Document layouts structurally resemble 3D indoor-scenes, in the sense that semantic entities are loosely related and not bound by geometric connectivity (like parts in a 3D shape). But unlike indoor scenes, where any permutation of valid *subscene* arrangements would synthesize plausible global scenes [15, 31], semantic entities in a document must be placed at the right positions for the generated layout to look realistic; e.g., *title* should always appear at the top. In other words, document layouts enforce more global constraints.

3. Method

Our RvNN-VAE framework of generating layouts is trained on a dataset of documents with semantic-based labels. That is, each document is composed of a set of labeled bounding boxes (ex., magazine-articles are labeled with *title*, *paragraph*, and so on). We use the set of labeled bounding boxes, which we call the *atomic units*, to build a training hierarchy for each document in our training set. These hierarchies are fed into our RvNN-VAE framework (see Figure 2) with a suitable training objective. Once trained, the RvNN-VAE network is used to generate a new layout by decoding a randomly sampled vector into a hierarchy of 2D bounding boxes with their corresponding semantic labels.

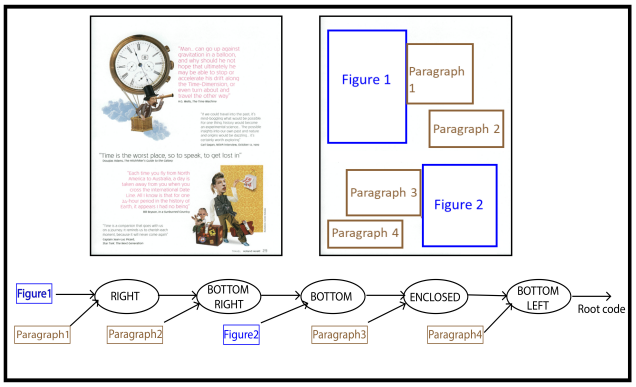


Figure 3. Exploratory layout extraction of a document from the ICDAR2015 [1] training set. The input document and the annotated boxes are shown on top. Note that when two boxes are merged, the merged bounding box is the union of the two boxes.

3.1. Building training hierarchies

Given labeled bounding box annotations, we first extract a structural decomposition for every document in the training set, based on connectivity and *implicit* symmetry of the atomic unit bounding boxes, by scanning the document from left-to-right and top-to-bottom. The results are stored as binary trees. We combine each pair of atomic elements, which we view as leaf nodes, into a union of boxes, viewed as an internal node, in a recursive manner, according to the relative position between the boxes. Internal nodes are also handled in a similar fashion. This exploratory process continues until all boxes are merged under a single root node. Figure 3 demonstrates the result of such an exploratory process on a single training sample. As the figure illustrates, we employ various types of spatial relationships (see Figure 4).

As documents are designed by humans, there is a weak symmetric structure between related atomic unit boxes; fields that are spatially-related usually have similar box geometry. Traversing left-to-right and top-to-bottom does not always guarantee that atomic units with similar geometry are grouped together, e.g., boxes that are placed one below the other with the same box geometry may not be grouped together. However, we demonstrate that our RvNN-VAE framework is able to effectively capture relationships among the boxes with our simple traversal strategy, without any complex hand-crafted heuristics.

3.2. Recursive model for document layouts

Every atomic unit in the extracted hierarchies, to be used for training, is initially represented using its bounding box dimensions ($[w, h]$ normalized in the range $[0, 1]$) concatenated with its semantic label, which is encoded as a one-hot

vector. To efficiently model document layouts using a recursive model, we first use a simple single-layer neural network to map the atomic unit bounding boxes to n-D vector representations (we empirically set $n = 300$). Our recursive autoencoder network is comprised of spatial-relationship encoders (SREs) and decoders (SRDs). Each encoder and decoder is a multi-layer perceptron (MLP), formulated as:

$$x_l = \tanh\left(W^{(l)} \cdot x_{l-1} + b^{(l)}\right).$$

We denote by $f_{W,b}(x)$ an MLP with weights $W = \{W^{(1)}, W^{(2)}, \dots\}$ and biases $b = \{b^{(1)}, b^{(2)}, \dots\}$ aggregated over all layers, operating on input x. Each MLP in our model has one hidden layer, and therefore, $l \in \{1, 2\}$.

Our SREs may operate over either (i) a pair of leaves, or (ii) an internal node and a leaf. Regardless, we denote both node representations as x_1, x_2. The merged parent code, y, is calculated according to x_1, x_2 and the relative position between the two bounding boxes, denoted by $r_{x_1x_2}$. The relative position is always calculated *w.r.t.* the left child (which is the internal node, when merging an internal node and a leaf node). The i-th SRE is formulated as:

$$y = f_{W_{e_i}, b_{e_i}}([x_1\ x_2\ r_{x_1x_2}]). \tag{1}$$

The corresponding SRD splits the parent code y back to its children x'_1 and x'_2 and the relative position between them $r'_{x'_1x'_2}$ (see Figure 2, bottom right). It uses a reverse mapping and is formulated as follows:

$$[x'_1\ x'_2\ r'_{x'_1x'_2}] = f_{W_{d_i}, b_{d_i}}(y). \tag{2}$$

Each node in the hierarchy represents a feature vector, which is encoded (or decoded) by one of c SREs (or SRDs). In particular, we note that since the network is recursive, the same encoder or decoder may be employed more than once for different nodes. As described in more detail below, the type of the encoder employed in each step depends on the spatial relationship between the elements in this step.

During decoding, we determine the spatial-relationship type i of a node so that the corresponding decoder can be used. To this end, we *jointly* train an auxiliary node classifier to determine which SRD to apply at each recursive decoding step. This classifier is a neural network with one hidden layer that takes as input the code of a node in the hierarchy, and outputs whether the node represents a leaf or an internal node. In the case of an internal node, the corresponding SRD is invoked, and if it is a leaf, the code is projected back onto a labeled bounding box representation (box dimensions concatenated with a one-hot vector corresponding to the semantic category) using a non-recursive single-layer neural network.

The types of spatial relationships we consider for encoding and decoding document layouts are: right, left, bottom, bottom-left, bottom-right, enclosed and wide-bottom

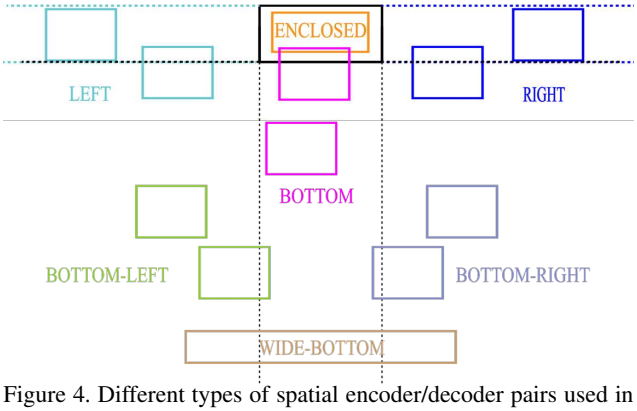


Figure 4. Different types of spatial encoder/decoder pairs used in learning document layouts. The left child (or the reference box) is shown with a thick black outline. Relative positions are calculated *w.r.t.* the left child.

($c = 7$), see Figure 4. Note that we traverse a document from left-to-right and top-to-bottom, and therefore, we do not have to consider any kind of *top* spatial relation. The structure of the binary tree (and the tree encoding thereof) is dictated by the position of the atomic units in the document. We show, in Section 5 that our model successfully learns to generate novel layouts, including plausible co-occurrences of atomic units.

3.3. Training details

The total training loss of our RvNN-VAE network is:

$$L_{total} = L_{leaf} + L_{pos} + L_{ce} + L_{KL} \tag{3}$$

where the first term is the leaf-level reconstruction loss:

$$L_{leaf} = \frac{1}{N}\sum_{k=1}^{N}(x'_k - x_k)^2. \tag{4}$$

Here, x'_k and x_k are the n-D leaf vectors at the decoder and the encoder, respectively, and N is the number of leaves.

The second term is the relative-position reconstruction loss between the bounding boxes (leaf-leaf or an internal node box and a leaf box):

$$L_{pos} = \frac{1}{N-1}\sum_{k=1}^{N-1}(r'_{x'_kx'_{k+1}} - r_{x_kx_{k+1}})^2 \tag{5}$$

where $r'_{x'_kx'_{k+1}}$ and $r_{x_kx_{k+1}}$ represent the relative position vectors at the decoder and encoder end, respectively.

The third term is a standard categorical cross-entropy loss:

$$L_{ce}(\mathbf{a}, i) = \log \sigma(\mathbf{a})_i, \tag{6}$$

where σ is the softmax function, $\mathbf{a}$ is a feature vector mapped from the output of an internal (or a root) node at which the node classifier is applied, and $i \in [0, c-1]$ corresponds to the ground truth spatial-relationship type at the node.

Finally, the last term in Eq. 3 is the KL-divergence loss for approximating the space of all root codes (encoder output of the RvNN-VAE):

$$L_{KL} = D_{KL}(q(z)||p(z)) \tag{7}$$

where $p(z)$ is the *latent space* and $q(z)$ is the standard normal distribution $\mathcal{N}(0,1)$.

To train our RvNN-VAE network, we randomly initialize the weights sampled from a Gaussian distribution. To output document layouts that are more spatially balanced, we developed a few (optional) post processing steps.

4. Evaluating Document Layouts

To evaluate how our method performs in terms of appearance and variability, we propose a new combinatorial layout similarity metric we call `DocSim`. Inspired by how the BLEU metric (bilingual evaluation understudy) for machine translation [21] measures sentences similarity, we aim to obtain a simple and easy-to-compute structural similarity measure between documents; one that resembles what humans perceive as similarity, yet is not too over-specified.[2] We introduce our metric through the following interpretation of BLEU: consider a bipartite graph between all words w in the first sentence S and all words w' in the second sentence S', where there is an edge between w and w' if both represent the same word (or, say, are synonyms). The BLEU score is then calculated by computing the number of edges in a *maximum matching* between these two sentences. Our metric, `DocSim`, similarly compares two given document layouts D, D' as follows: to any pair of bounding boxes $B \in D$ and $B' \in D'$, we assign a *weighted edge* that indicates how similar B and B' are in terms of shape, location, and "role" within the document. The final score is then calculated as the aggregated weight of the maximum (weighted) matching between the layouts D and D'.

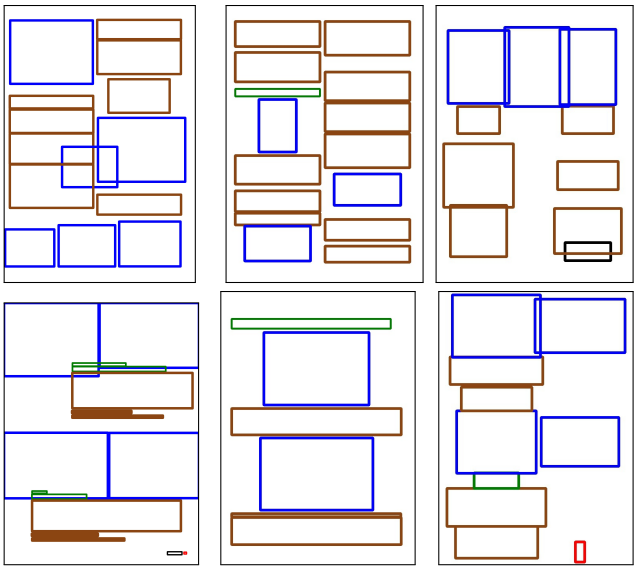
Formally, suppose we are given two documents, D1 and D2, each viewed as a set of bounding boxes of one or more "types" (examples of such types in real-world documents can be a paragraph, title, figure, and so on). Each bounding box is represented as a quadruple consisting of its minimum and maximum x and y coordinates within the document. The coordinates are *normalized* to fit in the unit 1×1 square. The similarity measure between two normalized documents D_1 and D_2 is calculated in two steps: weight assignment to box pairs, and maximum weight matching among boxes.

Assigning weights to box pairs. We would like to assign weights to pairs of boxes, so that similar pairs, that are roughly co-located and have approximately the same area, will have a higher weight. In the next step, we shall use these weights to assign a maximum weight matching between boxes of D_1 and boxes of D_2; the total similarity score would simply be the total weight of the matching. Let B_1 and B_2 be two normalized bounding boxes, where the x-coordinates of box B_i are denoted $a_i \leq b_i$ and its y-coordinated are $c_i \leq d_i$. If B_1 and B_2 have different types, then the weight between them is $W(B_1, B_2) = 0$ (this essentially means that boxes of different types cannot be matched). Otherwise, we calculate the weight as

$$W(B_1, B_2) = \alpha(B_1, B_2) 2^{-\Delta_C(B_1,B_2) - C_S \cdot \Delta_S(B_1,B_2)}$$

where the parameters $\alpha, \Delta_C, \Delta_S$ are defined as follows: The *location parameter* $\Delta_C(B_1, B_2)$ is the relative euclidean distance between the centers of B_1 and B_2 in the document. We wish to reduce the shared weight of B_1 and B_2 if they are far apart from each other. The *shape difference* is $\Delta_S(B_1, B_2) = |w_1 - w_2| + |h_1 - h_2|$ where w_i and h_i are the width and height of B_i, for $i = 1, 2$, respectively.

As larger bounding boxes have a more significant role in the "general appearance" of a document, we wish to assign larger weight to edges between larger boxes. Thus, we define the *area factor* as $\alpha(B_1, B_2) = min(w_1 h_1, w_2 h_2)^C$, where we choose $C = 1/2$. To explain this choice, observe that changing the constant to $C = 1$ would assign almost no weight to edges between small boxes, whereas $C = 0$ strongly favors this type of edges. Finally, we set the *shape constant* as $C_S = 2$. This means that the shape



Training sample | Probabilistic [32] | Ours

Figure 5. Given a document layout from ICDAR2015, we show the nearest neighbor obtained from the probabilistic approach described in [32] and the nearest neighbor using our approach. Color legend: *title*, *Paragraph*, *footer*, *page number*, *figure*.

[2] Generally speaking, there cannot exist a "one-size-fits-all" similarity metric ideal for all possible settings, as was discussed extensively regarding BLEU (see e.g. [16]). Thus, the quantitative evaluation of our paper combines `DocSim`-based comparisons with other evaluation methods, so as to try providing a complete picture of the efficacy of our approach.

difference between two boxes plays a slightly bigger role in their weight calculation than does the location parameter.

Maximum weight matching among boxes. Consider a bipartite graph where one part contains all boxes of D_1 while the other part consists of all boxes of D_2, and the edge weight $W(B_1, B_2)$ for $B_1 \in D_1$ and $B_2 \in D_2$ is as described above. We find a maximum weight matching $M(D_1, D_2)$ in this bipartite graph using the well-known Hungarian method [10]. The similarity score between D_1 and D_2 is defined as

$$\texttt{DocSim}(D_1, D_2) = \frac{1}{|M(D_1, D_2)|} \sum W(B_1, B_2),$$

where the sum is over all pairs $(B_1, B_2) \in M(D_1, D_2)$.

5. Results and Evaluation

To assess our layout generation method, we conducted several sets of experiments, aiming at understanding whether the generated layouts are highly variable and also *visually*-similar to the training documents. We also demonstrate their usefulness as training data for document analysis tasks. We evaluate our RvNN-VAE framework on the following two datasets.

ICDAR2015 Dataset. We use the publicly available ICDAR2015 [1] dataset, containing 478 documents that are themed along the lines of magazine-articles. For these documents, we consider the following semantic categories: *title*, *paragraph*, *footer*, *page number*, and *figure*.

User-Solicited (US) Dataset. We assembled a dataset of 2036 documents that solicit user-information (tax forms, banking applications, etc.). Such documents typically exhibit a highly complex structure and a large number of atomic elements. These characteristics present an interesting challenge for generative models producing document layouts. For these types of documents, we consider the following semantic categories: *key-value*, *title*, and *paragraph*. Key-value boxes are regions with a single question (key) that the user must answer/address (value). As the dataset we collected captures unfilled documents, the key-value box contains regions that should be filled out by the user. We semantically annotated all the categories using Amazon Mechanical Turk (AMT).

Training: We use the PyTorch framework [22], with a batch size of 128 and a learning rate of $3 * 10^{-4}$. On average, the number of semantically annotated bounding boxes is 27.73 (min=13, max=45) in the US *training* set and 17.61 (min=3, max=75) for ICDAR2015 *training* set. As is shown in the two rightmost columns of Table 4, the statistics on our generated data are similar. Training takes close to 24 hours on the US dataset and around 10 hours on the ICDAR2015 dataset, on an NVIDIA GTX 1080 Ti GPU.

5.1. Quantitative evaluation

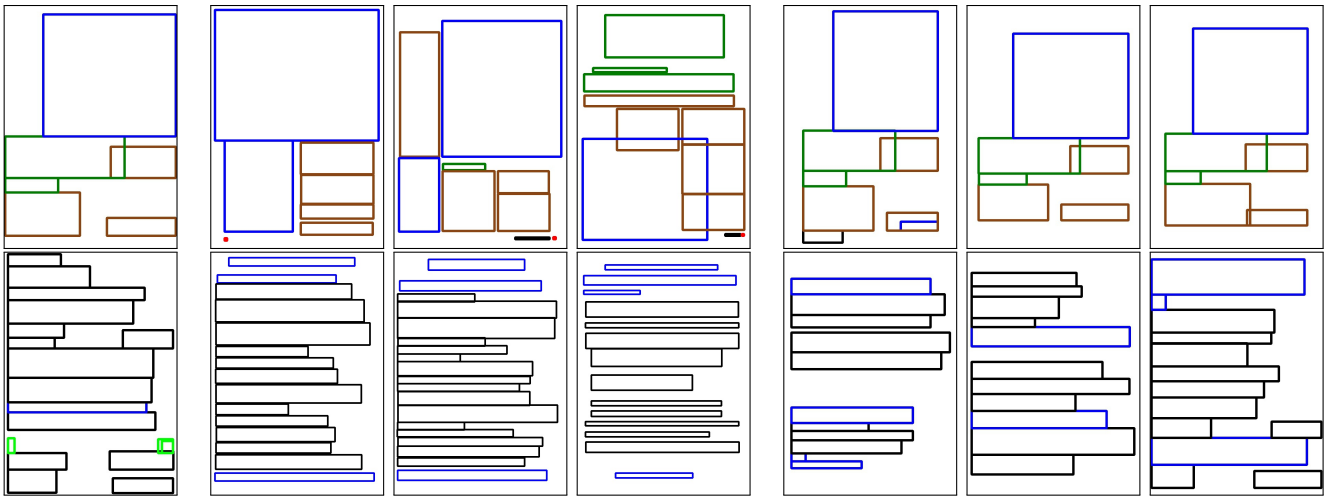
We use our proposed similarity metric, DocSim, to quantitatively evaluate our layout generation approach. To measure resemblance of our generated document layouts to the latent distribution of document layouts from which the training data is sampled from, we iterate over training-set and test-set, and for each document in these sets, we find the nearest neighbor in our generated layouts. To this end, the nearest neighbor of a document D is the document D' which *maximizes* the score $\texttt{DocSim}(D, D')$, and correspondingly, the similarity score that D has with respect to a dataset $\mathcal{D}$ is defined as $\max_{D' \in \mathcal{D}} \texttt{DocSim}(D, D')$. In our nearest neighbors experiments, we filter out documents D' whose number of boxes from any category is more than 3 higher or lower (before overlap removal) than that of D.

On the ICDAR2015 dataset. As a baseline, we obtain synthetic layouts using the probabilistic approach described in [32], using their publicly available implementation. Notably, the main focus of [32] is semantic segmentation of documents, and their probabilistic layout synthesis method (which outputs one-, two- and three-column documents) is developed as a helper for their main learning task.

In the probabilistic synthesis method of [32], labeled boxes are sampled according to a pre-defined distribution (e.g., a *paragraph* is selected with probability q). We obtain a collection $\mathcal{P}$ of 5k layouts using the probabilistic scheme of [32]; layouts are synthesized with the *title*, *paragraph* and *figure* classes, selected at probability 0.1, 0.7 and 0.2, respectively. Similarly, we obtain a collection $\mathcal{G}$ of 5k layouts generated by our RVNN-VAE framework, where we use a training set $\mathcal{T}$ of 400 documents from ICDAR2015. The collection $\mathcal{T}'$ of all remaining 78 documents from ICDAR2015 is considered our test set.

We experiment by comparing the baseline collection $\mathcal{P}$ with our collection $\mathcal{G}$ in terms of how well they capture the latent document layout space, where the evaluation uses our DocSim score. First, we run the following: for any training document $T \in \mathcal{T}$, we pick $G_T \in \mathcal{G}$ to be the generated document from our collection which maximizes $\texttt{DocSim}(T, G)$ among all $G \in \mathcal{G}$, and similarly $P_T \in \mathcal{P}$ as the document from the probabilistically synthesized collection which maximizes $\texttt{DocSim}(T, P)$ among all $P \in \mathcal{P}$. The similarity score between $\mathcal{T}$ and $\mathcal{G}$ is then calculated as the average of $\texttt{DocSim}(T, G_T)$ over all $T \in \mathcal{T}$; the similarity score between $\mathcal{T}$ and $\mathcal{P}$ is computed analogously using $\texttt{DocSim}(T, P_T)$ for all $T \in \mathcal{T}$. Finally, we repeat the above experiment, replacing the training set $\mathcal{T}$ with the test set $\mathcal{T}'$.

The scores, given in Table 2, demonstrate that our *learned* document layouts are more structurally-similar to samples in the ICDAR2015 dataset, suggesting that our network is able to meaningfully learn the latent distribution of



Generated sample | Nearest neighbors in train set | Nearest neighbors in generated set

Figure 6. Given a document layout generated by our approach, we retrieve three closest layouts from the training set (ICDAR2015 in the top row and US in the bottom row) and three closest from our generated set. Color legend (ICDAR2015): see Figure 5. Color legend (US): *title*, *paragraph*, *key-value*.

Measure	Real [1]	Probabilistic [32]	Generated (Ours)
Overlap (%)	2.3	0	1.9
Alignment (%)	17.8	9.6	18.2

Table 1. Spatial analysis of document layouts. Following [12], we use overlap index and alignment index of semantic entities as another measure to evaluate our layouts.

document layouts on which it was trained.

In addition, we perform a quantitative analysis using the overlap and alignment indices, following the evaluation in Li et al. [12]. Overlap index is the percentage of total overlapping area among any two bounding boxes inside the whole page. The second metric, alignment index, is calculated by finding the minimum standard deviation of either left or center coordinates of all bounding boxes. Table 1 shows the percentage of overlap index and alignment index for the real ICDAR2015 layouts [1], probabilistic layouts [32] and our generated layouts. As illustrated in the table, our results are very much comparable to those of the training data, demonstrating that our solution captures these metrics well (and does much better than the probabilistic layouts).

On the US dataset. As we are not aware of prior works that address these types of documents, we do not have a baseline method to compare to. We can, however, investigate the learning ability of our network on this dataset, which contains a relatively large number of documents (2036). Therefore, aside from training our network on the full dataset, we also use smaller subsets of training samples. As the entire US dataset is highly-variable, we compute our similarity score for every pair of document layouts in the entire US dataset and cluster the dataset into five groups (using spectral clustering). We then train our network on clusters that contain at least 500 documents, using a 80-20 train and test split, and generate $2K$ document layouts for each cluster.

We then compare the similarity scores obtained by training on the entire US dataset against the scores obtained on the US clusters (averaging over all cluster scores). Interestingly, the scores of the train/test sets are virtually almost identical (with a slight score advantage of 0.002 to 0.003 for the entire US dataset, which is a $2-3\%$ advantage). This suggests that our approach does not require a large amount of data to match the latent space of the training set reasonably well; Indeed, as indicated by the relatively similar scores, the models trained on the clusters capture the latent space of the training set roughly as good as the model that was trained on the full set. In Figure 6, we show the three closest document layouts from the training set to a randomly selected layout sample generated using our approach. As the middle three columns demonstrate, the three closest training samples bear some resemblance to our generated layouts, but they are not the same, further validating the novelty of the generated samples. The rightmost column, depicting the nearest neighbors in the generated set, illustrates the variations in the generated results.

5.2. Data augmentation for detection tasks

To demonstrate the utility of our generated layouts, we perform a standard detection task on documents and aug-

ICDAR [1]	[32]	Ours
Train (400)	0.123	**0.147**
Test (78)	0.118	**0.146**

Table 2. Comparing our approach to the probabilistic approach from [32], in terms of similarity to the latent distribution of the dataset (divided into train and test).

	Box IoU			Mask IoU		
Dataset	AP	AP_{50}	AP_{75}	AP	AP_{50}	AP_{75}
[1]	0.609	0.743	0.675	0.612	0.737	0.675
[1]+5k (aug.)	0.611	0.728	0.663	0.617	0.722	0.669
[1]+5k ([32])	0.605	0.753	0.676	0.612	0.750	0.665
[1]+5k (ours)	**0.634**	**0.770**	**0.702**	**0.644**	**0.769**	**0.700**

Table 3. Enhancing detection and segmentation performance on the ICDAR2015 [1] dataset using either data augmentations (second row), synthetic samples with probabilistic layouts (third row) or our learned layouts (bottom row).

ment the training data with generated documents whose layouts are produced by our method. We train Mask R-CNN [6], a popular object detection and segmentation network, on the ICDAR2015 dataset and evaluate the results obtained with and without performing data augmentation.

To generate training samples for Mask R-CNN, we inject content to our generated layouts (trained on 400 documents from the ICDAR2015 dataset). To do so, we scrape both text and images from Wikipedia. We also synthesize training samples using the probabilistic approach described in [32], and compare our results to the ones obtained by augmenting the dataset with their documents. The content in both cases is sampled from the same scraped data, thus the only difference is in the layouts. Furthermore, we compare our results to a standard augmentation technique, which uses photometric and geometric augmentations to enrich the ICDAR2015 dataset . In Table 3, we compare the bounding box detections and the segmentation results obtained by training on the different datasets. For both types of results (box/mask), we report the average precision (AP) scores averaged over IoU thresholds and at specific IoU values (AP_{50}, AP_{75}). The reported results are over the remaining 78 documents, which we do not train on. As the table demonstrates, our generated layouts consistently improve detection and segmentation IoU scores (by at least 3%). In comparison, scores obtained with documents synthesized using the probabilistic approach or using regular augmentation techniques are almost identical to the scores obtained on the dataset without any augmentations. The improved performance illustrates the vast importance of highly variable layout in generating meaningful synthetic data, validating that our technique successfully learns a layout distribution which is similar to the input dataset.

Method	#Training samples	#Semantic categories	#Boxes Avg.	#Boxes Max
[12]	25000	6	-	9
Ours (on [1])	400	5	17.06	74
Ours (on US)	560	3	28.27	45

Table 4. Comparison to previous work in terms of number of samples used for *training*, number of semantic categories in the *training* set, and average number of boxes per *generated* document.

5.3. Comparison to prior work

To the best of our knowledge, LayoutGAN [12] is the only prior work for our context. For the lack of publicly available code and dataset from [12], we perform a quantitative comparison on methodological statistics and present them in Table 4, and as was done in [12], we use the overlap and alignment metrics to compare between real layouts, our generated ones, and probabilistic layouts (see Table 1).

6. Conclusions

In this work, we present a new method for generating synthetic layouts for 2D documents, coupling a recursive neural network with a variational autoencoder. We introduce a metric for measuring document similarity, `DocSim`, and use it to show the novelty and diversity of our layouts.

There are several limitations to our approach. First, while our approach can generate highly variable layouts with dozens of elements, we are not yet able to generate highly complex layouts (e.g., the US tax form 1040), and it will be very interesting to understand how to reliably represent and generate such layouts. Second, our generated layouts may contain undesirable artifacts, such as misalignment and box overlaps. We addressed these artifacts using simple heuristics, but perhaps a more systematic solution would be to couple the current framework with a GAN, which will encourage the generated layouts to be more visually similar to the training samples.

In the future, it will be interesting to complement our layout generation approach with a suitable way to generate high quality semantic content that "makes sense" in view of the layout. Additionally, while our network does not require a huge amount of annotated data, it remains to be seen if there is a way to devise layout generation methods that require even less annotated training data, perhaps one-shot or few-shot approaches to generate plausible and "similarly looking" layouts. Finally, while recursive neural networks were shown (here and in previous works) useful for generating "human-made" hierarchical structures, like documents and indoor scenes, can they be used for generating highly structured *natural* scenes?

References

[1] Apostolos Antonacopoulos, Christian Clausner, Christos Papadopoulos, and Stefan Pletschacher. ICDAR2015 competition on recognition of documents with complex layouts-RDCL2015. In *2015 13th International Conference on Document Analysis and Recognition*, 2015. 3, 6, 7, 8

[2] Henry S. Baird, Horst Bunke, and Kazuhiko Yamamoto. *Structured Document Image Analysis*. Springer Berlin Heidelberg, 1992. 2

[3] Thomas M. Breuel. High performance document layout analysis, 2003. 2

[4] Biplab Deka, Zifeng Huang, Chad Franzen, Joshua Hibschman, Daniel Afergan, Yang Li, Jeffrey Nichols, and Ranjitha Kumar. Rico: A mobile app dataset for building data-driven design applications. In *Proceedings of the ACM Symposium on User Interface Software and Technology*, pages 845–854, 2017. 3

[5] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, 2014. 1

[6] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017. 8

[7] Akash Abdu Jyothi, Thibaut Durand, Jiawei He, Leonid Sigal, and Greg Mori. Layoutvae: Stochastic scene layout generation from a label set. In *International Conference on Computer Vision (ICCV)*, 2019. 2

[8] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. *arXiv:1812.04948*, 2018. 1

[9] Rangachar Kasturi, Lawrence O'Gorman, and Venu Govindaraju. Document image analysis: A primer. *Sadhana*, 27(1):3–22, 2002. 2

[10] Harold W. Kuhn. The hungarian method for the assignment problem. In *50 Years of Integer Programming 1958-2008*. Springer Berlin Heidelberg, 2010. 6

[11] Jun Li, Kai Xu, Siddhartha Chaudhuri, Ersin Yumer, Hao Zhang, and Leonidas Guibas. GRASS: Generative recursive autoencoders for shape structures. *ACM Transactions on Graphics*, 36(4):52, 2017. 1, 3

[12] Jianan Li, Tingfa Xu, Jianming Zhang, Aaron Hertzmann, and Jimei Yang. LayoutGAN: Generating graphic layouts with wireframe discriminator. In *International Conference on Learning Representations*, 2019. 1, 2, 3, 7, 8

[13] Manyi Li, Akshay Gadi Patil, Kai Xu, Siddhartha Chaudhuri, Owais Khan, Ariel Shamir, Changhe Tu, Baoquan Chen, Daniel Cohen-Or, and Hao Zhang. GRAINS: Generative recursive autoencoders for indoor scenes. *ACM Transactions on Graphics*, 38(2):1–16, 2019. 1, 3

[14] Thomas F. Liu, Mark Craft, Jason Situ, Ersin Yumer, Radomir Mech, and Ranjitha Kumar. Learning design semantics for mobile apps. In *Proceedings of the ACM Symposium on User Interface Software and Technology*, 2018. 2

[15] Rui Ma. *Sub-Scene Level Analysis and Synthesis of 3D Indoor Scenes*. PhD thesis, Simon Fraser University, 9 2017. 3

[16] Ehsan Montahaei, Danial Alihosseini, and Mahdieh Soleymani Baghshah. Jointly measuring diversity and quality in text generation models, 2019. NAACL 2019 workshop (NeuralGen 2019). 5

[17] Peter O'Donovan, Aseem Agarwala, and Aaron Hertzmann. Learning Layouts for Single-Page Graphic Designs. *IEEE Transactions on Visualization and Computer Graphics*, 20(8):1200–1213, 2014. 3

[18] Lawrence O'Gorman. The document spectrum for page layout analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(11):1162–1173, 1993. 2

[19] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv:1609.03499*, 2016. 1

[20] Aaron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. *arXiv:1601.06759*, 2016. 1

[21] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 311–318, 2002. 5

[22] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In *NIPS-W*, 2017. 6

[23] Daniel Ritchie, Ankita Arvind Kejriwal, and Scott R. Klemmer. D.Tour: Style-based exploration of design example galleries. In *ACM Symposium on User Interface Software and Technology*, 2011. 3

[24] Richard Socher. *Recursive deep learning for natural language processing and computer vision*. PhD thesis, Stanford University, 2014. 3

[25] Richard Socher, Cliff C Lin, Chris Manning, and Andrew Y Ng. Parsing natural scenes and natural language with recursive neural networks. In *Proceedings of the 28th international conference on machine learning*, pages 129–136, 2011. 3

[26] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, 2013. 3

[27] Amanda Swearngin, Mira Dontcheva, Wilmot Li, Joel Brandt, Morgan Dixon, and Andrew J. Ko. Rewire: Interface design assistance from examples. In *Proceedings, ACM Conference on Human Factors in Computing Systems*, 2018. 3

[28] Jerry Talton, Lingfeng Yang, Ranjitha Kumar, Maxine Lim, Noah Goodman, and Radomír Měch. Learning design patterns with bayesian grammar induction. In *Proceedings of ACM Symposium on User Interface Software and Technology*, 2012. 3

[29] Kai Wang, Manolis Savva, Angel X. Chang, and Daniel Ritchie. Deep convolutional priors for indoor scene synthesis. *ACM Transactions on Graphics*, 37(4):70, 2018. 1

[30] Jiajun Wu, Chengkai Zhang, Tianfan Xue, Bill Freeman, and Josh Tenenbaum. Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. In *Advances in neural information processing systems*, 2016. 1

[31] Kai Xu, Rui Ma, Hao Zhang, Chenyang Zhu, Ariel Shamir, Daniel Cohen-Or, and Hui Huang. Organizing heterogeneous scene collections through contextual focal points. *ACM Transactions on Graphics*, 33(4):35, 2014. 3

[32] Xiao Yang, Ersin Yumer, Paul Asente, Mike Kraley, Daniel Kifer, and C Lee Giles. Learning to extract semantic structure from documents using multimodal fully convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017. 2, 5, 6, 7, 8

[33] Xinru Zheng, Xiaotian Qiao, Ying Cao, and Rynson WH Lau. Content-aware generative modeling of graphic design layouts. *ACM Transactions on Graphics*, 38(4):133, 2019. 3

[34] Chenyang Zhu, Kai Xu, Siddhartha Chaudhuri, Renjiao Yi, and Hao Zhang. SCORES: Shape composition with recursive substructure priors. *ACM Transactions on Graphics*, 37(6):211, 2018. 3