Faster Attend-Infer-Repeat with Tractable Probabilistic Models

Karl Stelzner [1] **Robert Peharz** [2] **Kristian Kersting** [1 3]

Abstract

The recent Attend-Infer-Repeat (AIR) framework marks a milestone in structured probabilistic modeling, as it tackles the challenging problem of unsupervised scene understanding via Bayesian inference. AIR expresses the composition of visual scenes from individual objects, and uses variational autoencoders to model the appearance of those objects. However, inference in the overall model is highly intractable, which hampers its learning speed and makes it prone to suboptimal solutions. In this paper, we show that the speed and robustness of learning in AIR can be considerably improved by replacing the intractable object representations with tractable probabilistic models. In particular, we opt for sum-product networks (SPNs), expressive deep probabilistic models with a rich set of tractable inference routines. The resulting model, called SuPAIR, learns an order of magnitude faster than AIR, treats object occlusions in a consistent manner, and allows for the inclusion of a background noise model, improving the robustness of Bayesian scene understanding.

1. Introduction

Deriving meaningful representations from data with inherent structure is a key problem in machine learning and artificial intelligence. A natural approach to this problem is *generative modeling*, which postulates a latent data generating process, and reasons about this process conditioned on data. In the vision domain, for example, the idea of "vision as inverse graphics" has a particularly long history (Grenander, 1976) due to its natural appeal. Unfortunately, it has suffered from the highly intractable posterior of the graphical rendering process.

[1]CS Dept., TU Darmstadt, Darmstadt, Germany [2]Eng. Dept. (CBL), University of Cambridge, UK [3]Centre for Cognitive Science, TU Darmstadt, Darmstadt, Germany. Correspondence to: Karl Stelzner <stelzner@cs.tu-darmstadt.de>.

Proceedings of the 36^{th} International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019.

Recently, deep neural generative models such as variational autoencoders (VAEs) (Kingma & Welling, 2014) and generative adversarial networks (GANs) (Goodfellow et al., 2014) have shown remarkable success in generative image modeling. However, since their basic variants deliver rather unstructured latent representations, several structured latent variable models based on VAEs have been proposed. A particularly notable model is *Attend-Infer-Repeat* (AIR) (Eslami et al., 2016), which incorporates VAEs as object models within a scene generation process and learns a recurrent neural network (RNN) to dynamically detect multiple objects composed in a scene. Other examples of structured models are (Johnson et al., 2016), which incorporate VAEs into a latent switching linear dynamical system to infer behavioral patterns from mice depth videos, and SketchRNN (Ha & Eck, 2018), which uses an RNN to infer the trajectory of a pen from given sketches.

None of these models require supervision in the form of observed latent representations. Instead, the nature of these representations is specified through the model structure. To this end, the structure is imbued with available prior knowledge, such as the rules of object interaction, pen stroke rendering, or Markovian assumptions of biological behavior. Other parts such as the appearance of objects or typical pen trajectories are subject to learning. Exact inference is almost always intractable, either because the global model structure is already intractable itself or because intractable components such as VAEs are used.

Recent advances in variational inference, see e.g. (Zhang et al., 2017) for an overview, have enabled impressive results in such highly intractable models. Some of the key contributing factors in these advances are inference networks and amortization (Gershman & Goodman, 2014; Kingma & Welling, 2014), the reparameterization trick (Kingma & Welling, 2014; Titsias & Gredilla-Lázaro, 2014; Schulman et al., 2015), and techniques for reducing the variance in gradient estimates (Mnih & Gregor, 2014). Despite these improvements, however, inference in such large-scale structured models is far from solved, leading to slow learning or suboptimal inference results, expressed in e.g. poor local optima in the variational objective.

In this paper, we demonstrate that these issues can be effectively mitigated by replacing intractable components in

these systems with expressive tractable probabilistic models. Here we focus on AIR, but our insights generalize and are easily translated to other structured models. In particular, we propose a modification of AIR, which uses sum-product networks (SPNs) (Darwiche, 2003; Poon & Domingos, 2011) as object models, instead of VAEs. SPNs are a class of rich hierarchical latent variable models (Zhao et al., 2015; Peharz et al., 2017), which has been successfully applied in tasks like image recognition (Gens & Domingos, 2012), language modeling (Cheng et al., 2014), speech processing (Peharz et al., 2014), and robotics (Zheng et al., 2018). Compared to other deep probabilistic models, SPNs have a crucial advantage: any marginal probability can be efficiently and exactly computed in a simple forward pass. The availability of marginals plays a key role in our derivation of the variational lower bound for our model – called *Sum-Product Attend-Infer-Repeat* (SuPAIR) – and allows us to robustly handle noisy backgrounds and treat object occlusions in a principled manner.

Replacing intractable VAEs with SPNs leads to a dramatic reduction of the inference effort. In particular, our inference network does not need to predict latent object codes nor do we require object reconstructions—rather, we are able to directly assign well-calibrated likelihood scores to proposed scene descriptions. This approach can be understood as a form of *Rao-Blackwellization* and drastically reduces the variance in gradient estimates for the variational objective. As shown in our experiments, SuPAIR yields significant reductions in training time as well as increased robustness compared to the original AIR system. Our code is available online.[1]

We proceed with touching upon related work and reviewing the required background on AIR and SPNs. Based on this, we introduce the SuPAIR model and derive a learning objective for it. Before concluding, we present our experimental evaluation.

2. Related Work and Background

Let us start off by discussing related work and introducing the required background on SuPAIR.

2.1. Attend-Infer-Repeat (AIR)

Unlike previous work, e.g. (Lempitsky & Zisserman, 2010), the Attend-Infer-Repeat (AIR) framework approaches the problem of object counting and scene understanding in an unsupervised way, using a Bayesian approach. In particular, it is assumed that a given scene (image) x is generated according to some generative process $p(x, z) = p(x \mid z)\, p(z)$. Here, z denotes a latent scene description equipped with prior $p(z)$, and $p(x \mid z)$ represents the synthesis process (scene rendering) of the generative model. In this framework, scene analysis is cast into standard Bayesian inference, i.e. we condition on a scene x and infer the posterior $p(z \mid x) \propto p(x \mid z)\, p(z)$. The posterior might be used to derive a single scene description via a MAP solution $\arg\max_z p(z \mid x)$, or the whole posterior might be incorporated into a downstream decision making process. This interpretation of computer vision as inverse graphics has a long tradition (Grenander, 1976), but poses a notoriously hard inference problem.

Although general Bayesian scene understanding is hard, significant progress has been made by utilizing recent advances in variational inference. To this end, Eslami et al. (2016) introduced the following assumptions. The scene descriptor z is organized in N blocks, i.e. $z = (z^1, \ldots, z^N)$, corresponding to N objects in the scene. Each block $z^i = (z^i_{\text{where}}, z^i_{\text{what}})$ contains a description for its respective object, where z_{where} contains pose parameters (translation and scale) and z_{what} describes object appearance (object class, texture, etc). Since the number of objects in a scene varies, N is also a random variable, taking values between zero and some N_{max}.

Assuming a priori independence among the objects and the descriptor components, the prior of the entire scene description z takes the form $p(z) = p(N) \prod_{i=1}^{N} p(z^i_{\text{where}})\, p(z^i_{\text{what}})$. The number of objects N is straightforwardly modeled via e.g. a categorical or (truncated) geometric distribution. The distribution over pose parameters z^i_{where} can also take a simple form, such as a uniform distribution over a suitable range. In order to describe the appearance of objects, however, a more expressive model is required. The approach taken by Eslami et al. (2016) is to leverage a variational autoencoder (VAE) (Kingma & Welling, 2014) using the Gaussian distributed z^i_{what} as its latent code. z^i_{what} is processed by a neural net, generating an *object draft* y^i, the visual appearance of a single object. Each y^i is then transformed by its corresponding pose parameters z^i_{where} and inserted into a private canvas, denoted as $\tilde{y}^i$. Finally, the pixelwise means of scene x are generated as the sum of all $\tilde{y}^i$ which are present in the scene, i.e., for which $i \leq N$. The final distribution over x is then given by an isotropic Gaussian with these means and fixed variance.

Inference in AIR is addressed by recent variational inference techniques. First, it uses *amortization* (Gershman & Goodman, 2014; Kingma & Welling, 2014) by using an inference network to approximate the posterior $p(z \mid x)$. Following the compositional structure of the model, an RNN is employed as the inference network, at each step outputting a variational distribution $q(z^i_{\text{where}}, z^i_{\text{what}}, z^i_{\text{pres}})$, conditioned on the input x and the previously inferred object descriptors. Here, the binary variable z^i_{pres} indicates at each inference step whether the i^{th} object is present or if all objects have been found and the inference process should terminate. This

[1] github.com/stelzner/supair

effectively parameterizes $q(N)$ as a series of yes/no decisions, such that

$$q(N{=}n) = (1 - q(z^{n+1}_{\text{pres}})) \prod_{i=1}^{n} q(z^{i}_{\text{pres}}). \qquad (1)$$

Both the model and inference parameters are learned by stochastic optimization of the evidence lower bound (ELBO) (Hoffman et al., 2013). To yield gradient estimates of the ELBO, the reparameterization trick (Kingma & Welling, 2014; Titsias & Gredilla-Lázaro, 2014) is employed where possible. For discrete variables, in particular for the z^{i}_{pres}, score estimators with variance reduction techniques are used (Mnih & Gregor, 2014; Schulman et al., 2015).

While Eslami et al. (2016) delivered impressive results, learning and inference in AIR – and other structured probabilistic models – is far form solved. One issue is the non-trivial interplay between the generative model and the inference network, which frequently causes the generative model to adapt to a too weak inference network (Cremer et al., 2018). On the other hand, and somewhat paradoxically, trying to improve the quality of the variational approximation might also be detrimental for learning (Rainforth et al., 2018).

In this paper, we argue that these problems can be mitigated by switching to a model with significantly simplified inference. In particular, we propose to replace the biggest intractable part – the object VAEs – with tractable models, waiving the necessity to infer high-dimensional latent codes representing the objects. To this end, we propose to use models which are both tractable yet expressive, such as SPNs (Darwiche, 2003; Poon & Domingos, 2011), which are introduced next.

2.2. Sum-Product Networks (SPNs)

Let $x = (x_1, \ldots, x_D)$ denote a random vector. A sum-product network (SPN) over x is defined via an acyclic directed graph, containing leaf distributions, sum nodes, and product nodes. Each leaf of an SPN is a distribution function over some sub-vector y of x. The sub-vector y for which the leaf is a distribution, is denoted as the *scope* of the leaf. For internal nodes, we recursively define the scope as the union of the children's scopes. Internal nodes are either *mixtures* (sum nodes), i.e. they compute a convex combination of their children, or *factorized distributions* (product nodes), i.e. they compute a product of their children. An SPN needs to fulfill two structural requirements (Poon & Domingos, 2011), namely *completeness* (i.e. for each sum, all children have identical scope) and *decomposability* (i.e. for each product, the scopes of its children are non-overlapping). It follows by induction that every node in the SPN computes a proper distribution function over its scope. We assume that the SPN has a single root with scope x, which represents our model distribution over x.

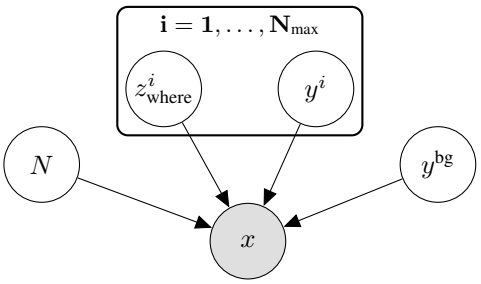


Figure 1. The generative model forming the basis for SuPAIR. The scene x is modeled to be composed of *background* y^{bg} and up to N_{max} objects drafts y^i. *Position* and *scale* of object y^i within the scene is encoded in the latent parameter vector z^i_{where}. The number of present objects is given by the discrete variable N.

A crucial advantage of SPNs is that they can compute *any sub-marginal* of the overall distribution. As shown in (Peharz et al., 2015), marginalization in SPNs reduces to the corresponding marginalization tasks at the leaves, and evaluating the rest of the SPN as usual, in a single feedforward pass. This marginalization is particularly easy when single-dimensional leaves are used, since in this case we simply need to set leaves of marginalized variables to 1.

The modeled distribution depends on both the SPN structure (the graph) and its parameters (sum weights and parameters of leaf distributions), which are both subject to learning. For structure learning, various approaches have been proposed, such as specifying the structure based on domain knowledge (Poon & Domingos, 2011), top-down co-clustering (Dennis & Ventura, 2012; Gens & Domingos, 2013; Rooshenas & Lowd, 2014; Vergari et al., 2015; Molina et al., 2018), and bottom-up greedy learning (Peharz et al., 2013). The parameters can be learned using gradient descent (Gens & Domingos, 2012), expectation-maximization (Poon & Domingos, 2011; Peharz et al., 2017), the convex-concave procedure (Zhao et al., 2016a), or Bayesian methods (Zhao et al., 2016b; Trapp et al., 2017; Vergari et al., 2019). Recently, *random tensorized SPNs* (RAT-SPNs) (Peharz et al., 2018) have been proposed, which use a random overparameterized structure, waiving the necessity of structure learning. In the setting of AIR, we do not know the dataset that each SPN will need to model a priori, complicating the application of structure learning. To demonstrate that our approach does not depend on domain knowledge informing the choice of SPN structure, we use RAT-SPNs in our proposed SuPAIR system. For details on their structure, we refer to the supplementary and Peharz et al. (2018).

3. Sum-Product Attend-Infer-Repeat

We now develop the Sum-Product AIR (SuPAIR) framework, following the generative model shown in Fig. 1. There are three main differences to the original AIR system:

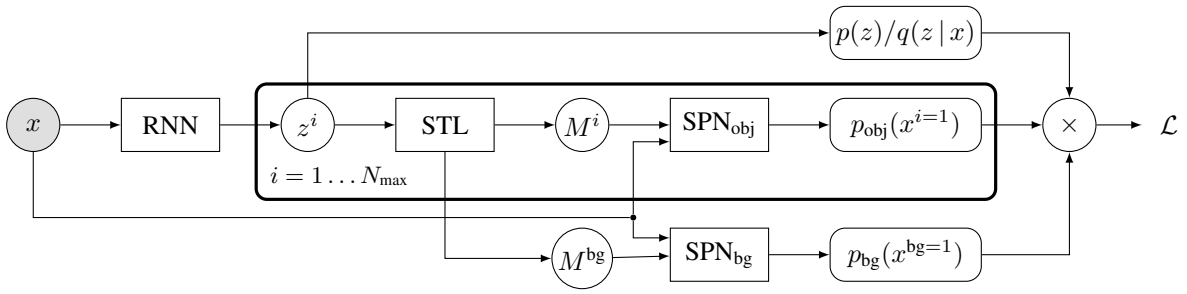


Figure 2. Computation graph for learning in SuPAIR. Given an image x, an RNN infers the number, position and size of objects, encoded in object descriptors z^i. The *spatial transformer layer* (STL) then computes object and background masks (M^i, M^{bg}) which indicate the marginalization domains for the object SPN and the background SPN. Due to our deterministic cut-and-paste interaction model and the tractable marginalization property in SPNs, a Rao-Blackwellized evidence lower bound is obtained from the SPNs' outputs.

1. We *directly* model the distribution over object drafts y^i with SPNs. Therefore, we do not need to infer latent object codes during learning, instead, we effectively marginalize both the latent SPN variables *and* y^i, which can be seen as a type of *Rao-Blackwellization*, speeding up the training process.

2. We incorporate a background model y^{bg} which allows for the capture of image noise and increases the model's robustness.

3. We use an alternative interaction (scene rendering) model, which plays well with efficient SPN inference.

Let us now devise the SuPAIR model in detail, touching in turn upon the priors, the interaction model, how to marginalize objects and the background, and finally how to perform variational inference. As a whole, this results in the SuPAIR computation graph shown in Fig. 2.

3.1. Objects, Background, and Scene Priors

The SuPAIR model generates a scene x of size $B \times B$. Each scene contains $0 \leq N \leq N_{\text{max}}$ objects, where the prior over N is modelled as a truncated geometric distribution. Each object i has latent pose parameters z^i_{where}, a 4-tuple representing the coordinates and size of the object's bounding box. The prior over each z^i_{where} is modeled as a uniform distribution with suitable bounds. To prevent highly or even fully occluded objects, we add an unnormalized penalty term modelled as a Gamma distribution over each object's *occlusion ratio*, the ratio of its pixels which are occluded in the scene.

The visual content y^i of object i is generated by a RAT-SPN over an $A \times A$ pixel array. To model the individual pixels, we employ univariate Gaussians at the leaf nodes. In this paper, we let the objects share the SPN's parameters, i.e. all objects have the same prior distribution, denoted $y^i \sim p_{\text{obj}}(\cdot)$. However, objects could also be easily equipped with private SPNs, and the SPNs could also be conditioned on some context, such as a class variable. While SuPAIR does not explicitly include latent object codes like AIR's z_{what}, if necessary such representations can still be obtained from the object SPN via the procedure proposed by Vergari et al. (2018). Furthermore, and differently from AIR, we assume a background model y^{bg} of the same size as the canvas, i.e. $B \times B$, also represented by a RAT-SPN with Gaussian leaves. We denote this density as $y^{\text{bg}} \sim p_{\text{bg}}(\cdot)$. Background and present objects are then combined into a scene according to an interaction (rendering) model, which is described next.

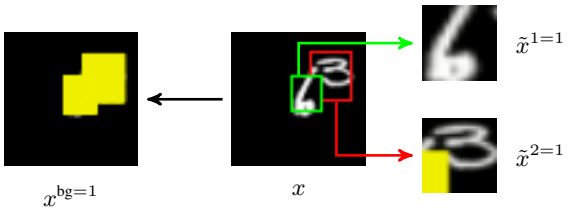


Figure 3. Cut-and-Paste Interaction Model: We assume that objects occlude the background and each other. Consequently, at inference time, we use the scene description z to decompose scenes x into their parts, the (rescaled) objects $\tilde{x}^{i=1}$ and the background $x^{\text{bg}=1}$. Occluded parts of background and objects are unobserved and drawn in yellow. The marginal likelihood of each component can then be evaluated using SPNs. (Best viewed in color)

3.2. Cut-and-Paste Interaction Model

To render a scene, we follow a natural *cut-and-paste* approach, as illustrated by Fig. 3. First, the background is inserted into the canvas, and then, one after the other, each object is inserted, occluding portions of the background and possibly other previously drawn objects. We assume that objects with smaller indices are in front of objects with larger indices, similar to how graphics engines employ z-buffers.

More formally, let the latent factors of SuPAIR N, y^{bg},

and all y^i, z^i_{where} be given. To avoid cluttered notation, we sometimes use $y^{1\dots N_{\text{max}}}$ for all objects, y for all objects *and* background, and z for all scene descriptors. Now, we define $B \times B$ *object indicator matrices* I^i which are 1 within the transformation window of the i^{th} object, and 0 elsewhere. Furthermore, we recursively define the *mask matrices* $M^i := I^i \prod_{j<i}(1-I^j)$ and $M^{\text{bg}} := \prod_{i=1}^{N}(1-I^j)$. Note that $\{M^i\}_{i=1}^N$ and M^{bg} encode a partition of the scene, i.e. they are all binary $B \times B$-matrices and sum up to the constant 1 matrix. In particular, they indicate visible pixels for the objects (if present) and the background. We define the scaled version of the objects $\tilde{y}^i = \tilde{y}^i(y^i, z^i_{\text{where}})$ as the result of transforming y^i according to z^i_{where}, and inserting it into an empty (black) $B \times B$-canvas. Now, the interaction model can be formalized as

$$p(x \mid y, z) = \delta_x \left[M^{\text{bg}} \times y^{\text{bg}} + \sum_{i=1}^{N} M^i \times \tilde{y}^i \right], \quad (2)$$

where $\times$ denotes element-wise multiplication, and $\delta_x[\cdot]$ the Dirac delta at x. Note that in this interaction model each pixel in x is deterministically given by either the background y^{bg} or one of the transformed objects $\tilde{y}^i$. This allows us to marginalize the scene components y, which will tremendously speed up the inference and learning process.

3.3. Marginalizing Objects and Background

Putting everything together, we arrive at the joint distribution of our model, given as $p(x, y, z) =$

$$p(x \mid y, z) p_{\text{bg}}(y^{\text{bg}}) p(N) \prod_{i=1}^{N_{\text{max}}} p_{\text{obj}}(y^i) p(z^i_{\text{where}}). \quad (3)$$

By conditioning on x, we can obtain a posterior over y and z. However, since we use SPNs as priors for the objects and the background, and since we use the *deterministic* interaction model (2), we are able to marginalize y:

$$p(x, z) = \int_{\dots} \int p(x, y^{\text{bg}}, y^{1\dots N_{\text{max}}}, z) \, \mathrm{d}y^{\text{bg}} \mathrm{d}y^{1\dots N_{\text{max}}}. \quad (4)$$

This is highly beneficial for learning, since we are ultimately interested in the posterior over the scene description z, and not in the latent scene components y. Note that each of the multi-dimensional integrals in (4) can be written as a series of single-dimensional integrals.

Marginalizing non-present objects (i.e. where $i > N$) is trivial, since the interaction model (2) does not depend on them. Thus, the integration over their appearances y^i can be switched with the product in (3). Since their priors $p(y^i)$ are normalized, the integrals over them evaluate to 1, i.e. they are effectively removed from the product.

Next, we discuss how to marginalize y^{bg}. To this end, let the entries in y^{bg} be split into two groups $y^{\text{bg}=0}$ and $y^{\text{bg}=1}$, indicated by $M^{\text{bg}} = 0$ and $M^{\text{bg}} = 1$, respectively. Since the interaction model $p(x \mid y, z)$ does not depend on $y^{\text{bg}=0}$, these pixels are just marginalized from the prior over y^{bg} in (3). On the other hand, for $y^{\text{bg}=1}$, we know that for all i, $M^i = 0$ at these pixels, so that (2) puts all probability mass on the event $y^{\text{bg}=1} = x^{\text{bg}=1}$, where, akin to the definition above, $x^{\text{bg}=1}$ are those pixels in x where $M^{\text{bg}} = 1$. Consequently, marginalizing over y^{bg} yields $p(x, z) =$

$$p_{\text{bg}}(x^{\text{bg}=1}) \int_{\dots} \int p(x^{\text{bg}=0}, y^{1\dots N}, z) \, \mathrm{d}y^{1\dots N}. \quad (5)$$

Here, we can readily draw on the remarkable property of SPNs, which allows us to evaluate the marginal $p_{\text{bg}}(x^{\text{bg}=1})$ using a *single network pass*. Moreover, by applying automatic differentiation, we can obtain the gradients required for learning without any extra effort.

We can, in principle, proceed in a similar way as with y^{bg} and eliminate one object y^i after the other, as there is no difference in the functional form for objects and background in (2). However, note that while the background y^{bg} enters *directly* in (2), each object y^i is incorporated via its *transformation* $\tilde{y}^i(y^i, z^i_{\text{where}})$. Following the same argument as above, the i^{th} object is marginalized from (5), by integrating y^i over the event $\tilde{y}^{i=1}(y^i, z^i_{\text{where}}) = x^{i=1}$, where $\tilde{y}^{i=1}$ and $x^{i=1}$ are the pixels indicated by $M^i = 1$. Unfortunately, when standard image transformations such as bilinear interpolation are employed, the mapping from y^i to $\tilde{y}^i$ is many-to-one, rendering this integral analytically intractable, even for SPNs.

One approach for addressing this problem is to choose a transformation which makes each pixel in y^i either independent of, or uniquely determined by, $\tilde{y}^i$, such as the probabilistic hard downsampling procedure described in the following. We assume that the size $A \times A$ of the objects y^i is chosen such that z^i_{where} only involves downsampling, i.e. the objects are modeled at their maximal resolution. Let $W \times H$ be the size of the transformed object, as determined by z^i_{where}. By interpreting the coefficients of the bilinear transform from size $W \times H$ to $A \times A$ as probabilities, we randomly select a pixel from y^i for each pixel in $x^{i=1}$. Now, given such a hard mapping, $x^{i=1}$ can be evaluated as a marginal of p_{obj} such that the joint (5) reduces to $p(x, z) =$

$$p(N) p_{\text{bg}}(x^{\text{bg}=1}) \prod_{i=1}^{N} p_{\text{obj}}(x^{i=1}) \prod_{i=1}^{N_{\text{max}}} p(z^i_{\text{where}}). \quad (6)$$

For a fixed downsampling scheme, only one random submarginal of the SPN is queried with $x^{i=1}$, and thus "informed" during learning. Ideally, we should marginalize over all possible downsampling schemes in order to get a smooth transformation in expectation. This marginalization can be efficiently incorporated into stochastic variational optimization, by simply drawing a new downsampling scheme

in each iteration. This introduces some additional noise into the learning process, but does not introduce bias into the SuPAIR model.

In practice, we choose a more pragmatic approach by using a simple but biased method to approximate probabilistic downsampling. Rather than evaluating the object SPN at a random sub-marginal, we scale the content of the object's bounding box $x^{i=1}$ to the SPN's native dimensions $A \times A$, approximating the inverse of $\tilde{y}^i$. We can then evaluate the SPN on this scaled version of $x^{i=1}$. We also scale the content of M^i to respect marginalized pixels stemming from occlusion. Since this scaling of the input introduces "artificial" dimensions in the model, we renormalize the result by $p \leftarrow p^{W_i H_i / A^2}$. In sum, the approximate version of probabilistic downsampling is computed as $p(x, z) =$

$$p(N) p_{\text{bg}}(x^{\text{bg}=1}) \prod_{i=1}^{N} p_{\text{obj}}(\tilde{x}^{i=1})^{W_i H_i / A^2} \prod_{i=1}^{N_{\text{max}}} p(z^i_{\text{where}}) \quad (7)$$

where $\tilde{x}^{i=1}$ denotes the scaled input.

3.4. Variational Inference

The joint, obtained from either (6) or (7), can now be used to perform posterior inference over the latent scene description z. To this end, we employ an RNN as an inference network, representing the variational distribution $q_\phi(z \mid x)$, which detects one object in each iteration. We process the input scene x using an LSTM layer with 256 hidden units, the output of which is fed into two fully connected layers with 50 and 9 units, respectively. The 9 outputs are the parameters for SuPAIR's scene descriptors for a single object, namely 8 parameters (means and standard deviations) for the 4-dimensional $q(z^i_{\text{where}})$, modeled as Gaussian, and one parameter for the Bernoulli $q(z^i_{\text{pres}})$. As in AIR, the latter is used to parameterize $q(N \mid x)$ as defined in (1). We summarize all parameters of the inference networks in ϕ and all SPN parameters in θ. The prior parameters of z are kept fixed (see supplementary).

We simultaneously learn the SuPAIR model and the inference network by optimizing the *evidence lower bound* (ELBO)

$$\mathcal{L} = \mathbb{E}_{q_\phi(z \mid x)} \left[\log p_\theta(x, z) - \log q_\phi(z \mid x) \right] \quad (8)$$

with respect to θ and ϕ. The ELBO can be estimated using Monte Carlo samples from the variational distribution $q_\phi(z \mid x)$, yielding an unbiased but potentially high-variance estimator. The reparameterization trick (Kingma & Welling, 2014; Schulman et al., 2015) is one of the most effective techniques to reduce the gradient's variance, and is straightforwardly applied to z^i_{where}. To ensure differentiability of the likelihood with respect to z^i_{where}, we use bilinear interpolation when computing the mask matrices M. This allows the masks to take non-integer values, slightly blending the scene components at their edges. The main invariant of the cut-and-paste model is still maintained, however, as the mask matrices continue to sum to one. Following AIR, we refer to this interpolation step as *spatial transformer layer* (STL), originally proposed as a subcomponent in (Jaderberg et al., 2015).

While the discrete variable N cannot be easily reparameterized, its expectation in (8) can be computed *exactly* via enumeration for only a low computational cost. Since our variational distribution factorizes over z, we can write the ELBO as $\mathcal{L} =$

$$\mathbb{E}_{q(z_{\text{where}})} \left[\sum_{n=0}^{N_{\text{max}}} q(n \mid x) \left(\log p(x, z) - \log q(z \mid x) \right) \right]. \quad (9)$$

As mentioned above, the outer expectation in (9) is handled by stochastic variational inference and the reparameterization of z_{where}. The expectation over N is treated as a sum over the possible numbers of objects. To compute it, we merely require $N_{\text{max}} + 1$ evaluations of the background network, to compute $p_{\text{bg}}(x^{\text{bg}=1})$ for each of the possible occlusion masks. Similarly, the object SPN only needs to be evaluated once for each of the N_{max} potential objects, since objects with higher indices are behind objects with lower indices. Thus, the object related terms in the joint are either not included (object is missing) or the same for each n.

4. Experiments

In this section we compare SuPAIR to the original AIR system (Eslami et al., 2016), and investigate the following two questions: **(Q1)** Do tractable appearance models lead to faster and more stable learning, i.e. with smaller variance? **(Q2)** Does an explicit background model make SuPAIR more robust to noise than AIR? To this end, we implemented SuPAIR in TensorFlow, making use of the RAT-SPN implementation by Peharz et al. (2018). We have also experimented with the SPN structure formulated by Poon & Domingos (2011) for the image domain, but have not found it to deliver significant improvements in learning speed or accuracy. We therefore report the results obtained using the more generally applicable random structures. All experiments were conducted using a single NVIDIA GeForce GTX 1080 Ti and a AMD Ryzen Threadripper 1950X CPU. Since the original code by Eslami et al. (2016) is not publically available, we made use of the well-documented AIR implementation in Pyro (Bingham et al., 2018) as our baseline, adopting the hyperparameter settings recommended by the authors.

4.1. Benchmark Datasets

We conducted experiments on two standard benchmarks for AIR, each with a different set of objects: *Multi-MNIST*,

Table 1. Number of parameters for SuPAIR and AIR. Note that AIR does not feature a (parameterized) background model, while SuPAIR does not require data-dependent baselines.

	SuPAIR	AIR
Inference network	2,836,477	2,879,488
Object model	286,560	344,884
Background model	90,108	0
Baselines	0	2,879,745
Total	3,213,145	6,104,117

using MNIST-digits as objects, and *Sprites*, a dataset using artificially generated geometric shapes. In both datasets, each scene is a 50×50 grayscale image containing zero, one or two objects, with their positions and scale varied according to uniform distributions. Scenes with excessively overlapping objects are discarded. 20% of each dataset was retained as a test set to evaluate the count accuracy achieved by the inference network.

4.2. Hyperparameters and Inductive Bias

Unlike AIR, our model does not make the hard assumption that the background will always be black. It is therefore necessary to provide SuPAIR with some inductive bias expressing what ought and ought not to be considered background. As is common for unsupervised models, we specify this bias by means of hyperparameters.

Since we expect the background to be less visually complex than the objects, we make the background-SPN shallower and narrower, giving it less room to model dependencies. In turn, we set a lower limit for the variance of its Gaussian leaf nodes, allowing it to achieve higher likelihood scores on low variance data, such as black background. We have found this to be a surprisingly subtle yet effective way of guiding the model. While stronger biases could of course be specified, for instance by constraining the means of the Gaussian leaves or even by pretraining the SPNs on manually labelled data, we have found this not to be necessary to achieve good performance. The total number of learnable parameters in SuPAIR is given by Table 1 and is comparable to AIR, with the main difference being the lack of a baseline inference network, a consequence of our choice to enumerate N.

4.3. (Q1) Counting Objects

Fig. 4 depicts the inference results obtained at various stages of training, illustrating that our model learns to correctly count and locate the objects. A comparison of the count accuracies achieved over the course of training is provided in Fig. 5, highlighting the main advantage of our approach: training on MNIST is close to an order of magnitude faster when compared to the original AIR system. We have found that executing a single training epoch for SuPAIR is about 40% faster than for AIR, likely a consequence of the lack of baseline networks. This also suggests that most of the overall speedup can be attributed to faster statistical convergence as opposed to computational speed. The final accuracies and ELBO values achieved are given by Table 2. We note that the convergence speed we observed for AIR compares favorably with the numbers reported in the literature: Eslami et al. (2016) state they achieved convergence after two days of training on a NVIDIA Quadro K4000, while Bingham et al. (2018) report convergence after about 15 minutes on the much more powerful NVIDIA K80. On the sprite dataset, SuPAIR converged even faster, reaching a count accuracy of over 95% in less than a minute.

Table 2. Count accuracies and ELBO values achieved after convergence. Note that the ELBO scores are not directly comparable between SuPAIR and AIR, due to the different formulations of the generative model.

	Count Accuracy		**ELBO**	
	SuPAIR	AIR	SuPAIR	AIR
Multi-MNIST	98.3%	94.7%	4923	615
Sprites	99.2%	95.9%	5022	685
Noisy-MNIST	93.8%	0.0%	1047	232
Grid-MNIST	97.5%	0.0%	3564	228

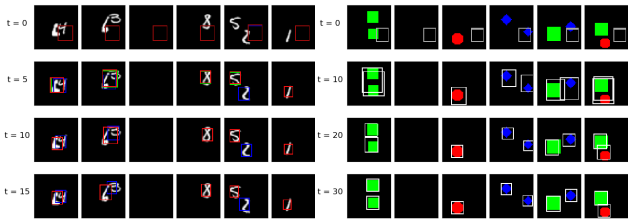
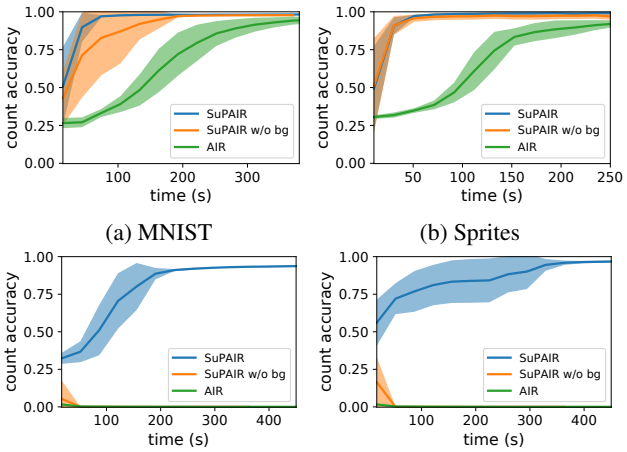


Figure 4. Inference results of SuPAIR on *Multi-MNIST* (left) and *Sprites* (right) after t training epochs. The outlines of the up to three predicted object positions are displayed as bounding boxes. The model reliably learns to count and locate the objects in both datasets.

4.4. (Q2) Robustness to Noise

In order to evaluate the robustness of our model, we also trained it on two variants of the Multi-MNIST dataset, each featuring a different type of background, one resembling pure noise, the other structured background. For the first case, we simply add Gaussian noise to the entire scene. For the second, we generate a regular grid by coloring every fifth row and column of pixels gray, starting at a randomly selected offset. The MNIST digits are then overlaid on top.

Fig. 6 depicts the inference results of both SuPAIR and AIR after training on these datasets. Our model is still able to locate the digits, albeit with a slightly decreased count accuracy of about 90%. The results of the ablation



(a) MNIST (b) Sprites

(c) Noisy MNIST (d) Grid MNIST

Figure 5. Learning progress of SuPAIR and AIR on the four datasets. As an ablation test, we also report results for a variant of SuPAIR without a learned background model ("w/o bg"). On *Multi-MNIST* and *Sprites*, SuPAIR achieves a high count accuracy almost an order of magnitude faster than AIR.

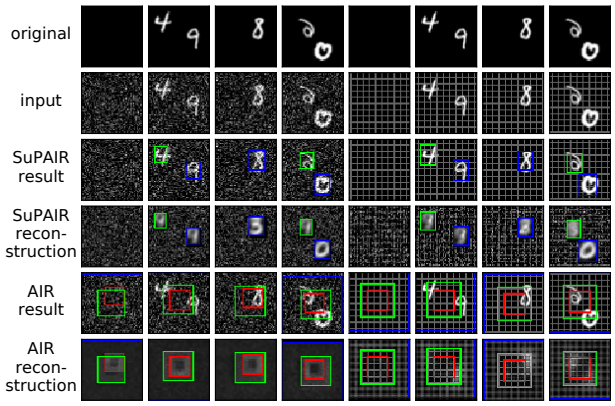


Figure 6. Comparison of the inference results and reconstructions of SuPAIR and AIR on two variants of Multi-MNIST featuring either Gaussian noise (left) or a regular grid background drawn at a random offset (right). Reconstructions for SuPAIR are obtained according to the procedure proposed by Vergari et al. (2018). Note that AIR always predicts $N_{\text{max}} = 3$ objects.

test reported in Fig. 5 indicate that the background model is indeed crucial for this. AIR on the other hand fails in this setting: unsurprisingly, due to its lack of background model, it is forced to allocate at least one object to the entire image in order to attempt to reconstruct the background. More severe, however, is the fact that AIR's variational autoencoder fails to properly capture the distribution over objects. Its reconstructions only render a vague blur in the place of detected objects. This effect significantly degrades the training signal for the RNN, which consequently fails to accurately detect and locate the digits.

5. Conclusion

Structured probabilistic models such as AIR have achieved impressive results in various applications, mainly fueled by recent advances in approximate inference. We cannot, however, expect approximate inference to be a silver bullet, in particular for models of increasing complexity. Meanwhile, advances in probabilistic deep learning have shown that tractable models, such as sum-product networks (SPNs), can also be used to faithfully capture high-dimensional distributions. Consequently, building structured probabilistic models which combine the best of both worlds appears to be a fruitful avenue.

We presented a modification of AIR called SuPAIR, which learns to count and locate scene elements using SPNs as object appearance models. This allows SuPAIR to marginalize object and background models, yielding a well-calibrated scene likelihood. As a result, the SuPAIR inference network does not need to predict latent object codes, drastically reducing the variance of gradient estimates for the variational objective. As shown in our experiments, this property yields a dramatic reduction in training time, higher object detection accuracy, and improved noise robustness compared to the original AIR system.

There are several interesting avenues for future work. One possible direction is to combine the insights in this paper with the various extensions of AIR which have been proposed since the original paper. SQAIR (Kosiorek et al., 2018) extends AIR to the sequential domain, SPAIR (Crawford & Pineau, 2019) proposes a more scalable inference network, and MONet (Burgess et al., 2019) features learned object masks. Since these models all use VAEs as object models, and since our contributions are independent of the inference network used, they should transfer seamlessly. More generally, other structured probabilistic models that do not address scene understanding (Lake et al., 2015; George et al., 2017) may also benefit from tractable components. To this end, the use of other explicit probabilistic models should also be explored, including autoregressive models such as NADE (Uria et al., 2014) or PixelCNNs (van den Oord et al., 2016) and normalizing flows (Rezende & Mohamed, 2015). While these models do not deliver tractable marginals, which formed the basis for our scene interaction model, this feature may not be required in other domains. Finally, one could apply the example set by SuPAIR to deep probabilistic programming in general (Tran et al., 2017; Bingham et al., 2018), striving for a framework which combines generic probabilistic modeling with ameliorated inference, by using tractable models as subcomponents and inference machines.

Acknowledgements

The authors thank the anonymous reviewers for their valuable feedback. The authors also want to thank Noah Goodman for very helpful discussions on the SuPAIR model. KK acknowledges the support of the Rhine-Main Universities Network for "Deep Continuous-Discrete Machine Learning" (DeCoDeML). RP acknowledges: This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie Grant Agreement No. 797223 — HYBSPN.

References

Bingham, E., Chen, J. P., Jankowiak, M., Obermeyer, F., Pradhan, N., Karaletsos, T., Singh, R., Szerlip, P., Horsfall, P., and Goodman, N. D. Pyro: Deep Universal Probabilistic Programming. *Journal of Machine Learning Research*, 2018.

Burgess, C. P., Matthey, L., Watters, N., Kabra, R., Higgins, I., Botvinick, M., and Lerchner, A. Monet: Unsupervised scene decomposition and representation. *CoRR*, abs/1901.11390, 2019. URL http://arxiv.org/abs/1901.11390.

Cheng, W.-C., Kok, S., Pham, H. V., Chieu, H. L., and Chai, K. M. A. Language modeling with sum-product networks. In *Interspeech*, 2014.

Crawford, E. and Pineau, J. Spatially invariant unsupervised object detection with convolutional neural networks. In *Proceedings of AAAI*, 2019.

Cremer, C., Li, X., and Duvenaud, D. Inference suboptimality in variational autoencoders. In *Proceedings of ICML*, volume 80, pp. 1078–1086, 2018.

Darwiche, A. A differential approach to inference in Bayesian networks. *Journal of the ACM*, 50(3):280–305, 2003.

Dennis, A. and Ventura, D. Learning the architecture of sum-product networks using clustering on variables. In *Proceedings of NIPS*, pp. 2042–2050, 2012.

Eslami, S. A., Heess, N., Weber, T., Tassa, Y., Szepesvari, D., Hinton, G. E., et al. Attend, infer, repeat: Fast scene understanding with generative models. In *Proceedings of NIPS*, pp. 3225–3233, 2016.

Gens, R. and Domingos, P. Discriminative learning of sum-product networks. In *Proceedings of NIPS*, pp. 3248–3256, 2012.

Gens, R. and Domingos, P. Learning the structure of sum-product networks. *Proceedings of ICML*, pp. 873–880, 2013.

George, D., Lehrach, W., Kansky, K., Lázaro-Gredilla, M., Laan, C., Marthi, B., Lou, X., Meng, Z., Liu, Y., Wang, H., et al. A generative vision model that trains with high data efficiency and breaks text-based captchas. *Science*, 358(6368):eaag2612, 2017.

Gershman, S. J. and Goodman, N. D. Amortized inference in probabilistic reasoning. In *Proceedings of CogSci*, 2014.

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial nets. In *Proceedings of NIPS*, pp. 2672–2680, 2014.

Grenander, U. *Lectures in Pattern Theory: Vol. 2 Pattern Analysis*. Springer-Verlag, 1976.

Ha, D. and Eck, D. A neural representation of sketch drawings. In *Proceedings of ICLR*, 2018.

Hoffman, M., Blei, D., Wang, C., and Paisley, J. Stochastic variational inference. *Journal of Machine Learning Research*, 14:1303–1347, 2013.

Jaderberg, M., Simonyan, K., Zisserman, A., et al. Spatial transformer networks. In *Proceedings of NIPS*, pp. 2017–2025, 2015.

Johnson, M., Duvenaud, D. K., Wiltschko, A., Adams, R. P., and Datta, S. R. Composing graphical models with neural networks for structured representations and fast inference. In *Proceedings of NIPS*, pp. 2946–2954, 2016.

Kingma, D. P. and Welling, M. Auto-encoding variational Bayes. In *Proceedings of ICLR*, 2014. arXiv:1312.6114.

Kosiorek, A. R., Kim, H., Posner, I., and Teh, Y. W. Sequential attend, infer, repeat: Generative modelling of moving objects. In *Proceedings of NIPS*, 2018. URL https://arxiv.org/abs/1806.01794.

Lake, B. M., Salakhutdinov, R., and Tenenbaum, J. B. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015.

Lempitsky, V. and Zisserman, A. Learning to count objects in images. In *Proceedings of NIPS*, pp. 1324–1332, 2010.

Mnih, A. and Gregor, K. Neural variational inference and learning in belief networks. In *ICML*, 2014.

Molina, A., Vergari, A., Di Mauro, N., Natarajan, S., Esposito, F., and Kersting, K. Mixed sum-product networks: A deep architecture for hybrid domains. In *Proceedings of AAAI*, 2018.

Peharz, R., Geiger, B., and Pernkopf, F. Greedy part-wise learning of sum-product networks. In *Proceedings of ECML/PKDD*, pp. 612–627. Springer Berlin, 2013.

Peharz, R., Kapeller, G., Mowlaee, P., and Pernkopf, F. Modeling speech with sum-product networks: Application to bandwidth extension. In *Proceedings of ICASSP*, pp. 3699–3703, 2014.

Peharz, R., Tschiatschek, S., Pernkopf, F., and Domingos, P. On theoretical properties of sum-product networks. In *Proceedings of AISTATS*, pp. 744–752, 2015.

Peharz, R., Gens, R., Pernkopf, F., and Domingos, P. On the latent variable interpretation in sum-product networks. *IEEE transactions on pattern analysis and machine intelligence*, 39(10):2030–2044, 2017.

Peharz, R., Vergari, A., Stelzner, K., Molina, A., Trapp, M., Kersting, K., and Ghahramani, Z. Probabilistic deep learning using random sum-product networks. *CoRR*, abs/1806.01910, 2018. URL http://arxiv.org/abs/1806.01910.

Poon, H. and Domingos, P. Sum-product networks: A new deep architecture. In *Proceedings of UAI*, pp. 337–346, 2011.

Rainforth, T., Kosiorek, A., Le, T. A., Maddison, C., Igl, M., Wood, F., and Teh, Y. W. Tighter variational bounds are not necessarily better. In *Proceedings of ICML*, volume 80, pp. 4277–4285, 2018.

Rezende, D. J. and Mohamed, S. Variational inference with normalizing flows. In *Proceedings of ICML*, 2015.

Rooshenas, A. and Lowd, D. Learning Sum-Product Networks with Direct and Indirect Variable Interactions. *ICML – JMLR W&CP*, 32:710–718, 2014.

Schulman, J., Heess, N., Weber, T., and Abbeel, P. Gradient estimation using stochastic computation graphs. In *Proceedings of NIPS*, pp. 3528–3536, 2015.

Titsias, M. K. and Gredilla-Lázaro, M. Doubly stochastic variational Bayes for non-conjugate inference. In *Proceedings of ICML*, pp. 1971–1979, 2014.

Tran, D., Hoffman, M. D., Saurous, R. A., Brevdo, E., Murphy, K., and Blei, D. M. Deep probabilistic programming. *CoRR*, abs/1701.03757, 2017. URL http://arxiv.org/abs/1701.03757.

Trapp, M., Madl, T., Peharz, R., Pernkopf, F., and Trappl, R. Safe semi-supervised learning of sum-product networks. In *Proceedings of UAI*, 2017.

Uria, B., Murray, I., and Larochelle, H. A deep and tractable density estimator. In *Proceedings of ICML*, pp. 467–475, 2014.

van den Oord, A., Kalchbrenner, N., Espeholt, L., Vinyals, O., Graves, A., et al. Conditional image generation with pixelcnn decoders. In *Proceedings of NIPS*, pp. 4790–4798, 2016.

Vergari, A., Di Mauro, N., and Esposito, F. Simplifying, regularizing and strengthening sum-product network structure learning. In *Proceedings of ECML/PKDD*, pp. 343–358. Springer, 2015.

Vergari, A., Peharz, R., Di Mauro, N., Molina, A., Kersting, K., and Esposito, F. Sum-product autoencoding: Encoding and decoding representations using sum-product networks. In *Proceedings of AAAI*, 2018.

Vergari, A., Molina, A., Peharz, R., Ghahramani, Z., Kersting, K., and Valera, I. Automatic Bayesian density analysis. In *Proceedings of AAAI*, 2019.

Zhang, C., Bütepage, J., Kjellström, H., and Mandt, S. Advances in variational inference. *CoRR*, abs/1711.05597, 2017. URL http://arxiv.org/abs/1711.05597.

Zhao, H., Melibari, M., and Poupart, P. On the relationship between sum-product networks and Bayesian networks. In *Proceedings of ICML*, pp. 116–124, 2015.

Zhao, H., Adel, T., Gordon, G., and Amos, B. Collapsed variational inference for sum-product networks. In *Proceedings of ICML*, 2016a.

Zhao, H., Poupart, P., and Gordon, G. J. A unified approach for learning the parameters of sum-product networks. In *Proceedings of NIPS*, 2016b.

Zheng, K., Pronobis, A., and R., R. Learning graph-structured sum-product networks for probabilistic semantic maps. In *Proceedings of AAAI*, 2018.