CSC 2547: Machine Learning for Vision as Inverse Graphics

Anthony Bonner
www.cs.toronto.edu/~bonner
Paper Presentations

• Each week will focus on one topic, as listed on the course web page (soon).
• You can vote for your choice of topic/week (soon).
• I will assign you to a week (soon).
• Papers on each topic will be listed on the course web page.
• If you have a particular paper you would like to add to the list, please let me know.
Paper Presentations

• Goal: high quality, accessible tutorials.
• 7 weeks and 44 students = 6 or 7 students per week and about 15 minutes per student.
• 2-week planning cycle:
  – 2 weeks before your presentation, meet me after class to discuss and assign papers.
  – The following week, meet the TA for a practice presentation (required).
  – Present in class under strict time constraints.
Team Presentations

• Papers may be presented in teams of two or more with longer presentations (15 minutes per team member).

• Unless a paper is particularly difficult or long, a team will be expected to cover more than one paper (one paper per team member).

• A team may cover one of the listed papers and one or more of its references (but see me first).
Tentative Topics

- Discriminative approaches.
- Generative approaches.
- Differentiable rendering.
- Capsule networks
- Group symmetries and equivariance
- Visual attention mechanisms
- Adversarial methods
Project Ideas

• Improve upon the work in a paper
  – Even a small improvement is OK
• For example,
  – Make a generative model conditional
  – Disentangle (some) latent variables
  – Adapt a method to new circumstances
    • Different kinds of data
    • Missing or noisy data
  – Make a supervised method semi-supervised
Project Ideas

• Examples (continued)
  – Modify the cost function
    • Introduce learnable parameters into a cost function
    • Use an adversarial cost
    • Try a variation on KL divergence
  – Modify the latent priors
    • Make the prior learnable
    • Do not assume Gaussianity
  – Modify the variational assumptions
    • Do not assume complete independence
    • Do not assume Gaussianity
Project Ideas

• Implement and compare different methods for the same problem (e.g., different methods for inferring 3D structure)
  – Clearly and succinctly describe each method
  – Clearly articulate their differences
  – Describe their strengths and weaknesses
  – Ideally, include experiments highlighting the differences between the methods on realistic problems.
Project Considerations

• Is your idea sensible?
• Can you download all the necessary data?
• Do you have the computational resources (GPUs)?
• Do you have time to complete it?
• Start by duplicating the results in the paper (if the paper gives enough details).
Project Dates

• Proposal due February 18
  – about 2 pages
  – include preliminary literature search

• Project presentations: March 24 and 31
  – about 5 minutes per student (like “spotlight presentations” at a conference)

• Project due: April 12
  – project report (4-8 pages) and code
Generative Approaches

• Given a scene, s, a graphics program, G, produces an image, G(s).
• Given an image, x, find s such that G(s) ≈ x
• More generally, find P(s|x),.
• P(s|x) is high when G(s) is close to x.
Variational Approximations

• Finding $P(s|x)$ is intractable in general.
• Use variational approximations.
• Variational auto-encoders work very well.
• $G$ can be a neural net that we learn (unsupervised).
• Computationally intensive.
We select we first conduct quantitative evaluations on the generated 3D volumes from the test set single-view trained the three models under two experimental settings: single category and multiple categories. Without ground-truth volumes (we denote the model trained with projection loss only, volume loss only, and combined loss as 4).

Table 1: Prediction IU using the models trained on volumes of all the instances in the test set. In addition, we provide a baseline method based on nearest neighbor (NN) search. Specifically, for each of the test image, we extract VGG feature from images. For each instance in the test set, we generate one volume per view image (24 volumes generated in total). Given a pair of ground-truth volume and our generated volume (threshold is 0.5), we used the ADAM [7] solver for stochastic optimization in all the experiments. During the fine-tuning stage (for volume decoder), we used mini-batch of size 6 and learning rate 3 and 2 for training the RNN-1, RNN-2, RNN-4, RNN-8, RNN-12 and RNN-16 as used in Yang et al. [23].

Variational Autoencoders

From Yan et al, Perspective Transformer Nets, arXiv 2017
Disentangled Representations

Disentangled Representations

From Reed et al, *Learning to Disentangle Factors of Variation*, ICML 2014
Learning 3D Shape

Figure 3: Single-class results. GT: ground truth, PR: PTN-Proj, CO: PTN-Comb, VO: CNN-V ol

(Best viewed in digital version. Zoom in for the 3D shape details). The angles are shown in the parenthesis. Please also see more examples and video animations on the project webpage.

As shown in Table 1, the model trained without volume supervision (projection loss) performs as good as model trained with volume supervision (volume loss) on the chair category (testing set). In addition to the comparisons of overall IU, we measured the view-dependent IU for each model. As shown in Figure 4, the average prediction error (mean IU) changes as we gradually move from the first view to the last view (15 to 360). For visual comparisons, we provide a side-by-side analysis for each of the three models we trained. As shown in Figure 3, each row shows an independent comparison. The first column is the 2D image we used as input of the model. The second and third column show the ground-truth 3D volume (same volume rendered from two views for better visualization purpose). Similarly, we list the model trained with projection loss only (PTN-Proj),

From Yan et al, Perspective Transformer Nets, arXiv 2017
Learning 3D Structure

From Niu et al, *Im2Struct: recovering 3D Shape Structure*, CVPR 2018
Scene Understanding

Abstract

We study the problem of holistic scene understanding. We would like to obtain a compact, expressive, and interpretable representation of scenes that encodes information such as the number of objects and their categories, poses, positions, etc. Such a representation would allow us to reason about and even reconstruct or manipulate elements of the scene. Previous works have used encoder-decoder based neural architectures to learn image representations; however, representations obtained in this way are typically uninterpretable, or only explain a single object in the scene.

In this work, we propose a new approach to learn an interpretable distributed representation of scenes. Our approach employs a deterministic rendering function as the decoder, mapping a naturally structured and disentangled scene description, which we named scene XML, to an image. By doing so, the encoder is forced to perform the inverse of the rendering operation (a.k.a. de-rendering) to transform an input image to the structured scene XML that the decoder used to produce the image. We use an object proposal based encoder that is trained by minimizing both the supervised prediction and the unsupervised reconstruction errors. Experiments demonstrate that our approach works well on scene de-rendering with two different graphics engines, and our learned representation can be easily adapted for a wide range of applications like image editing, inpainting, visual analogy-making, and image captioning.

1. Introduction

What properties are desirable in an image representation for visual understanding? We argue that the representation needs to be compact, expressive, and interpretable. Compactness makes it possible to store and exploit large amounts of data. Expressiveness allows it to capture the variations in the number, category, appearance, and pose of objects in an image. Lastly, an interpretable and disentangled representation enables us to reason about and even reconstruct or manipulate elements of an image.

Image representations learned by neural networks are often compact and expressive, but are hard to interpret. Recently, researchers studied how to obtain interpretable representations\[4,21,35\]. They mostly employed an encoding-decoding framework, using neural nets for both inference and approximate rendering. However, these methods typically assume each input image contains only a single, centered object in front of a clean background. Consequently, they are not robust and powerful enough for practical applications, where we often see images with an indefinite number of objects, heavy occlusions, and a cluttered background.

In contrast to neural decoders like the ones used in\[8,21\], the deterministic rendering functions used in graphics engines naturally take a structured and disentangled input to generate images. From this perspective, if we assume a given image is rendered by a generic graphics engine, we can aim to recover the structured representation required by the renderer to reconstruct the exact image (a.k.a. de-rendering). By learning an image representation this way, we achieve interpretability for free, and we will also be able to apply the representation to a range of applications like image editing.

This image de-rendering problem, however, is very challenging for multiple reasons. First, as we are no longer assuming a localized object, and the number of objects in an image is unknown, our representation should be extensible to an arbitrary number of objects in different positions. This cannot be achieved in a straightforward way with traditional convolutional networks that learn image representations of a fixed dimension. Previous works discussed the use of recurrent networks like LSTM\[14\] in these cases. However,
**Abstract**

We present a generative model of images based on layering, in which image layers are individually generated, then composited from front to back. We are thus able to factor the appearance of an image into the appearance of individual objects within the image — and additionally for each individual object, we can factor content from pose. Unlike prior work on layered models, we learn a shape prior for each object/layer, allowing the model to tease out which object is in front by looking for a consistent shape, without needing access to motion cues or any labeled data. We show that ordinary stochastic gradient variational bayes (SGVB), which optimizes our fully differentiable lower-bound on the log-likelihood, is sufficient to learn an interpretable representation of images. Finally we present experiments demonstrating the effectiveness of the model for inferring foreground and background objects in images.

**Introduction**

Recently computer vision has made great progress by training deep feedforward neural networks on large labeled datasets. However, acquiring labeled training data for all of the problems that we care about is expensive. Furthermore, some problems require top-down inference as well as bottom-up inference in order to handle ambiguity. For example, consider the problem of object detection and instance segmentation in the presence of clutter/occlusion, as illustrated in Figure 1. In this case, the foreground object may obscure almost all of the background object, yet people are still able to detect that there are two objects present, to correctly segment out both of them, and even to amodally complete the hidden parts of the occluded object (cf., Kar et al. (2015)).

One way to tackle this problem is to use generative models. In particular, we can imagine the following generative process for an image: (1) Choose an object (or texture) of interest, by sampling a "content vector" representing its class label, style, etc; (2) Choose where to place the object in the 2d image plane, by sampling a "pose vector", representing location, scale, etc. (3) Render an image of the object onto a hidden canvas or layer; (4) Repeat this process for \( N \) objects (we assume in this work that \( N \) is fixed); (5) Finally, generate the observed image by compositing the layers in order.

There have been several previous attempts to use layered generative models to perform scene parsing and object detection in clutter (see Section 2 for a review of related work). However, such methods usually run into computational bottlenecks, since inverting such generative models is intractable. In

---

From Huang et al, *Occlusion Aware Generative Models*, ICLR 2016
Conditional Image Generation

Figure 3: Visualization of generated samples with (left) 1 quadrant and (right) 2 quadrants for an input. We show in each row the input and the ground truth output overlaid with gray color (first), samples generated by the baseline NNs (second), and samples drawn from the CVAE (rest).

For qualitative analysis, we visualize the generated output samples in Figure 3. As we can see, the baseline NNs can only make a single deterministic prediction, and as a result the output looks blurry and doesn't look realistic in many cases. In contrast, the samples generated by the CVAE models are more realistic and diverse in shape; sometimes they can even change their identity (digit labels), such as from 3 to 5 or from 4 to 9, and vice versa.

We also provide a quantitative evidence by estimating the conditional log-likelihoods (CLLs) in Table 1. The CLLs of the proposed models are estimated in two ways as described in Section 4.1. For the MC estimation, we draw 10,000 samples per example to get an accurate estimate. For the importance sampling, however, 100 samples per example were enough to obtain an accurate estimation of the CLL. We observed that the estimated CLLs of the CVAE significantly outperforms the baseline NN. Moreover, as measured by the per pixel performance gap, the performance improvement becomes more significant as we use smaller number of quadrants for an input, which is expected as the input-output mapping becomes more diverse.

5.2 Visual Object Segmentation and Labeling

Caltech-UCSD Birds (CUB) database [36] includes 6,033 images of birds from 200 species with annotations such as a bounding box of birds and a segmentation mask. Later, Yang et al. [37] annotated these images with more fine-grained segmentation masks by cropping the bird patches using the bounding boxes and resized them into 128×128 pixels. The training/test split proposed in [36] was used in our experiment, and for validation purpose, we partition the training set into 10 folds and cross-validated with the mean intersection over union (IoU) score over the folds. The final prediction on the test set was made by averaging the posterior from ensemble of 10 networks that are trained on each of the 10 folds separately. We increase the number of training examples via “data augmentation” by horizontally flipping the input and output images.

We extensively evaluate the variations of our proposed methods, such as CVAE, GSNN, and the hybrid model, and provide a summary results on segmentation mask prediction task in Table 2. Specifically, we report the performance of the models with different network architectures and training methods (e.g., multi-scale prediction or noise-injection training).

First, we note that the baseline CNN already beat the previous state-of-the-art that is obtained by the max-margin Boltzmann machine (MMBM; pixel accuracy: 90.42, IoU: 75.92 with GraphCut for post-processing) [37] even without post-processing. On top of that, we observed significant performance improvement with our proposed deep CGMs. In terms of prediction accuracy, the GSNN performed the best among our proposed models, and performed even better when it is trained with hybrid objective function. In addition, the noise-injection training (Section 4.3) further improves the performance. Compared to the baseline CNN, the proposed deep CGMs significantly reduce the prediction error, e.g., 21% reduction in test pixel-level accuracy at the expense of 60% more time for inference.

Finally, the performance of our two winning entries (GSNN and hybrid) on the validation sets are both significantly better than their deterministic counterparts (GDNN) with p-values less than 0.05, which suggests the benefit of stochastic latent variables. As in the case of baseline CNNs, we found that using the multi-scale prediction was consistently better than the single-scale counterpart for all our models. So, we used the multi-scale prediction by default.

Mean inference time per image: 2.32 (ms) for CNN and 3.69 (ms) for deep CGMs, measured using GeForce GTX TITAN X card with MatConvNet; we provide more information in the supplementary material.

From Sohn et al, Deep Conditional Generative Models, NIPS 2015
Conditional Image Generation

From Ivanov et al, *Variational Autoencoder with Arbitrary Conditioning*, ICLR 2019
Attribute Conditioned Image Generation

Making Visual Analogies

• Given images A, B, C, generate image D so that D is to C as B is to A.