Structured Output Learning with Conditional Generative Flows

You Lu, Bert Huang

Virginia Tech
you.lu@vt.edu, bhuang@vt.edu

Introduction

In this paper, we develop conditional generative flows (c-Glow) for structured output learning. Our model has the following properties:

- It is a variant of Glow (Kingma et al. 2018), with additional networks to capture the relationship between input and structured output variables.
- C-Glow can directly model the conditional distribution $p(y|x)$ without restrictive assumptions (e.g., variables being fully connected).
- C-Glow uses invertible flows that allow exact computation of log-likelihood, removing the need for surrogates or inference.
- Compared to other methods using normalizing flows (e.g., Trippe & Turner, 2018), c-Glow's output y is both conditioned on complex input and a high-dimensional tensor rather than a one-dimensional scalar.

Structured Prediction

- In structured prediction, we collect a dataset $\mathcal{D} = \{(x_1, y_1), ..., (x_N, y_N)\}$, where x_i is the ith input vector and y_i is the corresponding output.
- Let x and y be random variables with unknown true distribution $p^*(y|x)$. We learn a model $p(y|x, \theta)$ by minimizing negative log-likelihood

$$\mathcal{L}(\mathcal{D}) = -\frac{1}{N}\sum_{i=1}^{N} \log p(y_i|x_i, \theta).$$

- The label y comes from a complex, high-dimensional output space $\mathcal{Y}$ with dependencies among output dimensions.
- Many structured output learning approaches use an energy-based model to define a conditional distribution:

$$p(y|x) = \frac{e^{E(y,x)}}{\int_{y' \in \mathcal{Y}} e^{E(y',x)}},$$

where $E(.,.) : \mathcal{X} \times \mathcal{Y} \to \mathcal{R}$ is the energy function. In deep structured prediction, $E(x, y)$ depends on x via a deep network.
- For high dimensional y, the partition function, i.e., $\int_{y' \in \mathcal{Y}} e^{E(y',x)}$, is intractable.
- Approximating the partition function with methods such as variational inference or surrogate objectives requires complicated training and sub-optimal results.

Conditional Normalizing Flows

- A normalizing flow is a composition of invertible functions $f = f_1 \circ f_2 \circ \cdots \circ f_M$, which transforms the target y to a latent code z drawn from a simple distribution.
- In conditional normalizing flows, we rewrite each function as $f_i = f_{x,\phi_i}$, making it parameterized by both x and its parameter ϕ_i.
- Thus, with the *change of variables* formula, we can rewrite the conditional likelihood as

$$\log p(y|x, \theta) = \log p_Z(z) + \sum_{i=1}^{M} \log \left| \det \left(\frac{\partial f_{x,\phi_i}}{\partial r_{i-1}} \right) \right|, \quad (1)$$

where $r_i = f_{\phi_i}(r_{i-1})$, $r_0 = x$, and $r_M = z$.
- In this paper, we address the structured output problem by using conditional normalizing flows, i.e., Equation 1, to calculate the conditional distribution.
- Our method is different from previous methods in that the labels in our problem are high-dimensional tensors rather than scalars.

Glow

- Glow is a flow-based generative model that extends other flow-based models: NICE and Real-NVP. The model mainly consists of three components.
  1. **Actnorm layers** each perform an affine transformation of activations using a scalar and bias parameters, i.e., s and b.
  2. **Invertible 1x1 convolutional layers** perform a generalization of a permutation operation. Each is parameterized by $c \times c$ matrix W.
  3. **Affine layers** capture the correlations among spatial dimensions. The affine coupling layer separates the v into two parts, i.e., v_1, v_2. It passes through the v_1 to a neural network and outputs the parameters, i.e., s_2 and b_2 for v_2.
- Glow uses a multi-scale architecture to combine layers, with "squeeze" layers for shuffling the variables and "split" layers for reducing the computation cost.

Conditional Generative Flows

Conditional Glow

- To modify Glow to be conditional, we augment its three components.
- We add a *conditioning network* (CN) to each component to generate the parameter weights for each layer.
  1. **Conditional Actnorm.** In conditional Glow, we use a conditioning network to generate the $c \times 1$ vectors, i.e., the scale s and the bias b, and then use them to transform the variable.

  $$s, b = \text{CN}(x), \quad u_{i,j} = s \odot v_{i,j} + b.$$

  2. **Conditional 1x1 Convolutional.** We use a conditioning network to generate the $c \times c$ weight matrix that permutes each dimension's variable.

  $$W = \text{CN}(x), \quad u_{i,j} = Wv_{i,j}.$$

  3. **Conditional Affine Coupling.** The affine coupling layer separates the input variable to two halves, i.e., v_1 and v_2. It uses v_1 as the input to a NN to generate scale and bias parameters for v_2. We use a CN to conditionally extract features from x, which we concatenate with v_1 to form the input of NN.

  $$\begin{aligned} &v_1, v_2 = \text{split}(v), \\ &x_r = \text{CN}(x), \quad s_2, b_2 = \text{NN}(v_1, x_r), \\ &u_2 = s_2 \odot v_2 + b_2, \quad u = \text{concat}(v_1, u_2). \end{aligned}$$

- We can still use the multi-scale architecture to combine these conditional components, so that can preserve the efficiency of computation.
- The conditioning networks do not need to be invertible when optimizing a conditional model.
- We can back-propagate to differentiate the exact conditional likelihood, and optimize all the c-Glow parameters using gradient methods.

Inference

- Given a model, we can perform efficient sampling with a single forward pass through the c-Glow.

$$z \sim p_Z(z), y = g_{x,\phi}(z), \quad (2)$$

where $g_{x,\phi} = f_{x,\phi}^{-1}$ is the inverse function.
- We use sample averages to estimate marginal expectations of output variables. Let $\{z_1, ..., z_M\}$ be samples drawn from $p_Z(z)$. We estimate marginals as

$$y^* \approx \frac{1}{M}\sum_{i=1}^{M} g_{x,\phi}(z_i). \quad (3)$$

- In some tasks like semantic segmentation, the space of y is discrete. We relax the discrete output space to a continuous space during training. When we do prediction, we simply round y to discrete values.

Image Segmentation

Experiment Settings

- We use the Weizmann Horse Images database, which contains 328 images of horses and their segmentation masks.
- The training and test sets contain 200 and 128 images, respectively. We resize the images and their masks to 64×64 pixels.
- We compare our method with non-linear transformations (NLStruct) and FCN-VGG
- We use pixel-wise accuracy and mean intersection-over-union (IOU) as metrics.

Segmentation Results

Table: Segmentation metrics comparing c-Glow with others.

	c-Glow	NLStruct	FCN-VGG
Accuracy	0.927	—	0.850
IOU	0.830	0.755	0.670

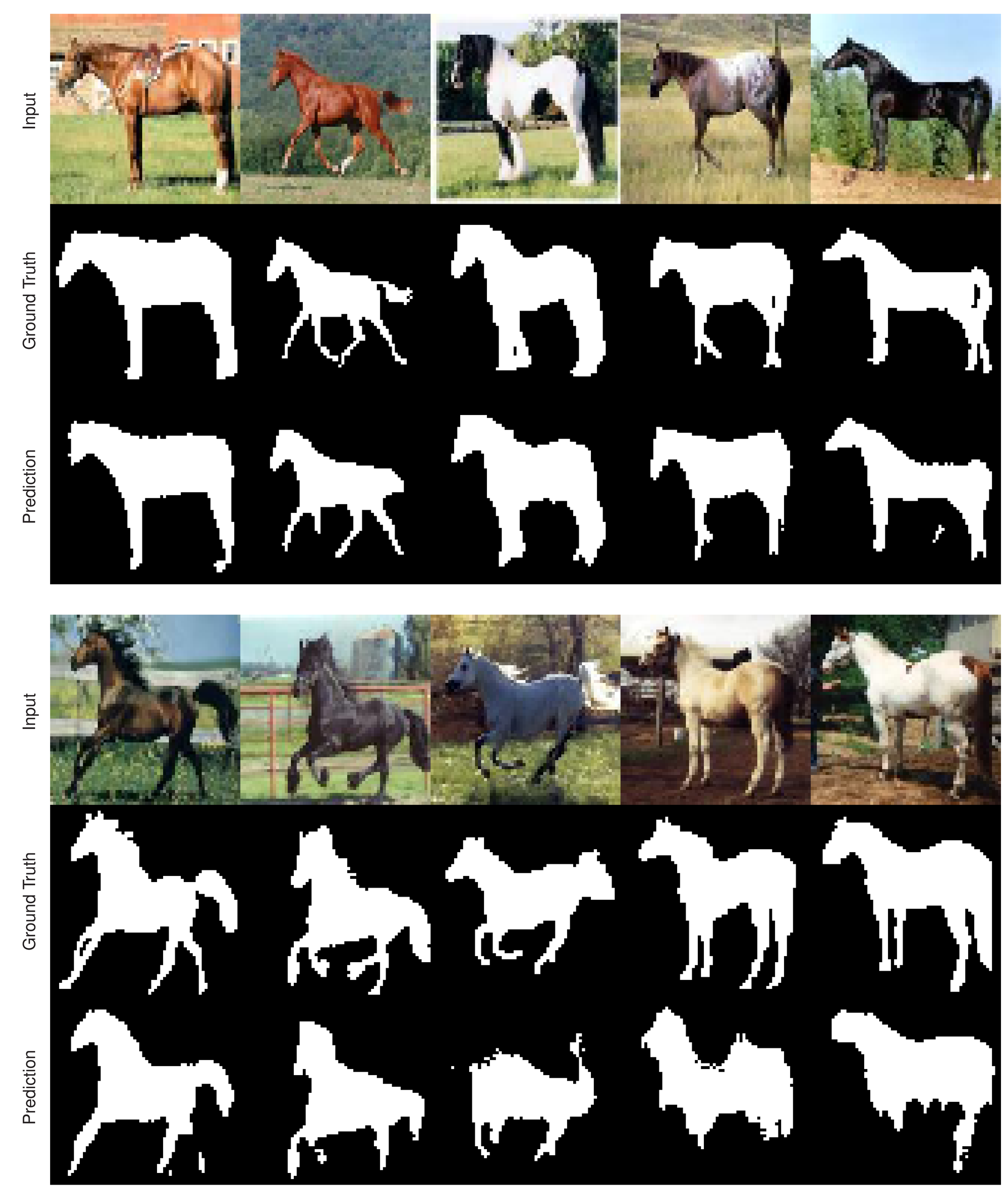

Figure: Sample segmentation results.

Conditional Sampling

- We can directly use the generative process, to generate conditional samples.

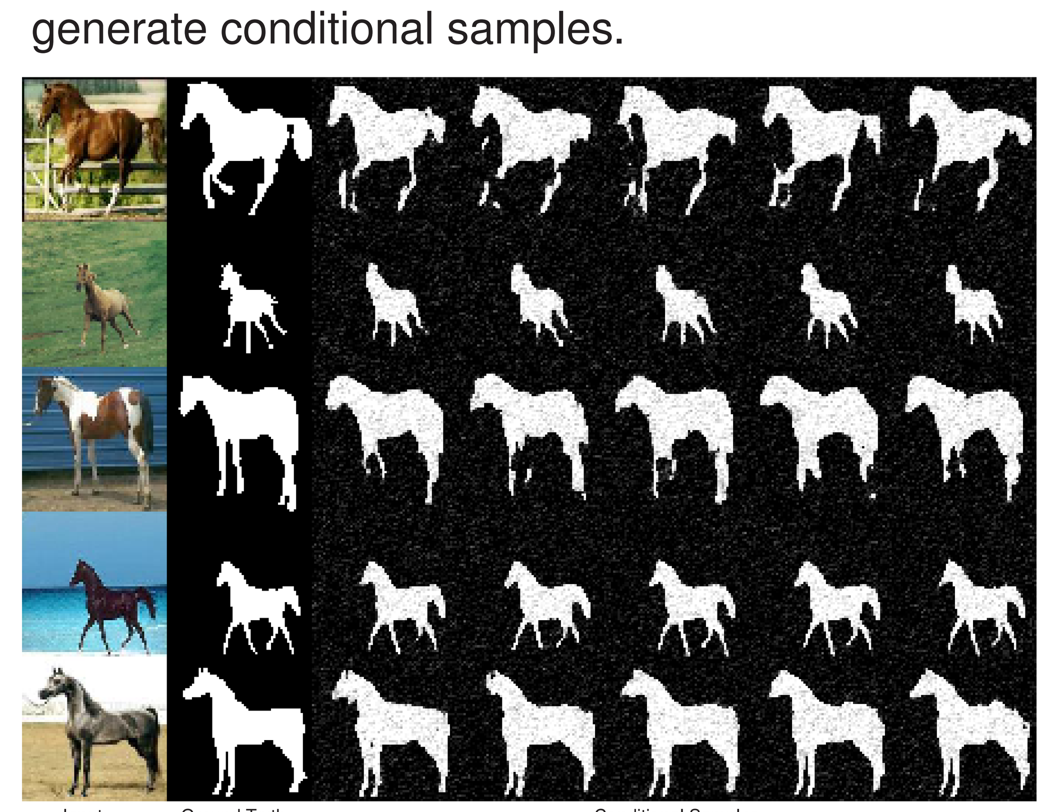

Figure: Conditional samples.

Conclusion

- We propose conditional generative flows (c-Glow), which are flow-based conditional generative models for structured output learning.
- We convert the Glow model to a conditional form by incorporating *conditioning networks*.
- In contrast with other deep structured output learning, we can maximize the exact likelihood, so we do not need surrogate objectives or approximate inference.
- In our experiments, we test c-Glow on image segmentation, finding that c-Glow is comparable to recent deep structured prediction approaches.