Structured Output Learning with Conditional Generative Flows

You Lu [1] **Bert Huang** [1]

Abstract

Traditional structured prediction models try to learn the conditional likelihood, i.e., $p(y|x)$, to capture the relationship between the structured output y and the input features x. For many models, computing the likelihood is intractable. These models are therefore hard to train, requiring the use of surrogate objectives or variational inference to approximate likelihood. In this paper, we propose conditional Glow (c-Glow), a conditional generative flow for structured output learning. C-Glow benefits from the ability of flow-based models to compute $p(y|x)$ exactly and efficiently. Learning with c-Glow does not require a surrogate objective or performing inference during training. Once trained, we can directly and efficiently generate conditional samples to do structured prediction. We evaluate this approach on image segmentation and find c-Glow's structured outputs comparable in quality with state-of-the-art deep structured prediction approaches.

1. Introduction

Structured prediction models are widely used in tasks like image segmentation (Nowozin & Lampert, 2011), sequential labeling (Lafferty et al., 2001), etc. For these types of problems, the goal is to model a mapping from the input x to the structured output y. In many such problems, it is also important to be able to make diverse predictions to capture the variability of plausible solutions to the structured output problem (Sohn et al., 2015).

Many existing methods for structured output learning use conditional random fields (CRFs, Wainwright & Jordan, 2008) to approximate the conditional distribution $p(y|x)$. Recently, deep structured prediction models (Chen et al., 2015; Zheng et al., 2015; Sohn et al., 2015; Wang et al., 2016; Belanger & McCallum, 2016; Graber et al., 2018) combine deep neural networks with graphical models, so that they can use the power of deep neural networks to extract high-quality features, and graphical models to model correlations and dependencies among variables. The main drawback of these approaches is that, due to the intractable likelihood, they are difficult to train. Training them requires the construction of surrogate objectives that approximate or bound the likelihood, often involving variational inference to infer latent variables. Moreover, sampling from CRFs requires expensive iterative procedures that require careful tuning (Koller et al., 2009).

In this paper, we develop conditional generative flows (c-Glow) for structured output learning. In contrast to other deep structured prediction models, our method can directly model the conditional distribution $p(y|x)$, without restrictive assumptions, e.g., variables are fully connected (Krähenbühl & Koltun, 2011). We can train c-Glow by optimizing the exactly log-likelihood, removing the need for surrogates or inference. Compared to other conditional flows (e.g., (Trippe & Turner, 2018; Kingma & Dhariwal, 2018)), c-Glow's output label y is both conditioned on complex input and a high-dimensional tensor rather than a one-dimensional scalar. We evaluate c-Glow on semantic segmentation, finding that c-Glow's structured outputs comparable in quality with state-of-the-art deep structured prediction approaches.

2. Related Work

There are two main branches of research related to our paper: structured prediction and normalizing flow. In this section, we briefly cover some of the most related literature.

2.1. Deep Structured Models

One emerging strategy to construct deep structured models is to combine deep neural networks with graphical models. However, this kind of model can be difficult to learn, since the likelihood of graphical models is usually intractable. Chen et al. (2015) proposed joint learning approaches that blend the learning and approximate inference to alleviate some of these computational challenges. Zheng et al. (2015) proposed CRF-RNN, a method that treats mean-field variational CRF inference as a recurrent neural network to allow gradient-based learning of model parameters. Wang et al. (2016) proposed proximal methods for inference. Sohn et al.

[1]Department of Computer Science, Virginia Tech. Correspondence to: You Lu <you.lu@vt.edu>.

First workshop on *Invertible Neural Networks and Normalizing Flows* (ICML 2019), Long Beach, CA, USA

(2015) use variational autoencoders (Kingma & Welling, 2013) to generate latent variables for predicting the output.

Another direction combining structured output learning with deep models is to construct energy functions with deep networks. Structured prediction energy networks (SPENs) (Belanger & McCallum, 2016) define energy functions for scoring structured outputs as differentiable deep networks. The likelihood of SPEN is intractable, so the authors use structured SVM loss to learn. SPENs can also be trained in an end-to-end learning framework (Belanger et al., 2017) based on unrolled optimization. Methods to alleviate the cost of SPEN inference include replacing the argmax inference with an inference network (Tu & Gimpel, 2018). Inspired by Q-learning, Gygli et al. (2017) use an oracle value function as the objective for energy-based deep network. Graber et al. (2018) generalize SPENs by adding non-linear transformations on top of the score function.

2.2. Normalizing Flows

Normalizing flows are neural networks constructed with fully invertible components. The invertibility of the resulting network provides various mathematical benefits. Normalizing flows have been successfully used to build likelihood-based deep generative models (Dinh et al., 2014; 2016; Kingma & Dhariwal, 2018) and to improve variational approximation (Rezende & Mohamed, 2015; Kingma et al., 2016). Autoregressive flows (Kingma et al., 2016; Papamakarios et al., 2017; Huang et al., 2018; Ziegler & Rush, 2019) condition each affine transformation on all previous variables, so that ensure an invertible transformation and triangular Jacobian matrix. Continuous normalizing flows (Chen et al., 2018; Grathwohl et al., 2018) define the transformation function using ordinary differential equations. Trippe & Turner (2018) developed radial flows to model univariate conditional probabilities.

Most related to our approach are flow-based generative models. Dinh et al. (2014) first proposed a flow-based model, NICE, for modeling complex high-dimensional densities. They later proposed Real-NVP (Dinh et al., 2016), which improves the expressiveness of NICE by adding more flexible coupling layers. The Glow model (Kingma & Dhariwal, 2018) improved the performance of such approaches further by incorporating new invertible layers. Most recently, Flow++ (Ho et al., 2019) improves generative flow by variational dequantization and architecture design, and Ma & Hovy (2019) proposed new invertible layers.

3. Background

In this section, we introduce notation and background knowledge directly related to our work.

3.1. Structured Output Learning

Let x and y be random variables with unknown true distribution $p^*(y|x)$. 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. To approximate $p^*(y|x)$, we develop a model $p(y|x, \theta)$ and then minimize the negative log-likelihood

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

In structured output learning, 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. Due to the high dimensionality of y, the partition function, i.e., $\int_{y' \in \mathcal{Y}} e^{E(y',x)}$, is intractable. To train the model, we need methods to approximate the partition function such as variational inference or surrogate objectives, resulting in complicated training and sub-optimal results.

3.2. 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 (Trippe & Turner, 2018), 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 normalizing flows. That is, we directly use the conditional normalizing flows, i.e., Equation 1, to calculate the conditional distribution. Thus, the model can be trained by locally optimizing the exact likelihood. Note that conditional normalizing flows have been used for conditional density estimation. Trippe & Turner (2018) use it to solve the one-dimensional regression problem. Our method is different from theirs in that the labels in our problem are high-dimensional tensors rather than scalars. We therefore will build on recently developed methods for (unconditional) flow-based generative models for high-dimensional data.

3.3. Glow

Built on NICE (Dinh et al., 2014) and Real-NVP (Dinh et al., 2016), Glow (Kingma & Dhariwal, 2018) achieves significant improvements in likelihood and sample quality for natural images. The model mainly consist of three components. Let u and v be input and output of a layer, whose shape is $[h \times w \times c]$, with spatial dimensions (h, w), and channel dimension c. The three components are as follows.

Actnorm layers. Each actnorm layer performs an affine transformation of activations using a scalar and bias parameters, i.e., s and b.

Invertible 1x1 convolutional layers. Each invertible 1x1 convolutional layer is a generalization of a permutation operation. Its weights is a $c \times c$ matrix W.

Affine layers. As in the NICE and Real-NVP models, Glow also has affine coupling layers to 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 (Dinh et al., 2016) to combine the layers. This architecture has a "squeeze" layer for shuffling the variables and a "split" layer for reducing the computation cost.

4. Conditional Generative Flows

In this section, we introduce our conditional generative flows, i.e., c-Glow, which is a flow-based generative model for structured prediction.

4.1. Conditional Glow

To modify Glow to be a conditional generative flow, we need to modify its three components: the actnorm layer, the 1x1 convolutional layer, and the affine coupling layer. The main idea is to use a neural network, which we refer to as a *conditioning network* (CN), to generate the parameter weights for each layer. The details are as follows.

Conditional Actnorm. The parameters of an actnorm layer are two $c \times 1$ vectors, i.e., the scale s and the bias b. In conditional Glow, we use a conditioning network to generate these two vectors and then use them to transform the variable.

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

Conditional 1x1 Convolutional. The 1x1 convolutional layer uses a $c \times c$ weight matrix to permute each dimension's variable. In conditional glow, we use a conditioning network to generate this matrix.

$$\begin{aligned} W &= \text{CN}(x), \\ u_{i,j} &= W v_{i,j}. \end{aligned}$$

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 an NN to generate scale and bias parameters for v_2. To build a conditional affine coupling layer, we use a CN to extract features from x, and then we concatenate it with v_a to form the input of NN.

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

We can still use the multi-scale architecture to combine these conditional components, so that can preserve the efficiency of computation.

Since the conditioning networks do not need to be invertible when optimizing a conditional model, we do not specify their architecture here. Any differentiable network suffices and preserves the ability of c-Glow to compute the exact conditional likelihood of each input-output pair.

To learn the model parameters, we can take advantage of the efficiently computable log-likelihood for flow-based models. Therefore, we can back-propagate to differentiate the exact conditional likelihood, i.e., Equation 1, and optimize all the c-Glow parameters using gradient methods.

4.2. Inference

Given a model, we can perform efficient sampling with a single forward pass through the c-Glow. We first calculate the transformation functions given x and then sample the latent code z from $p_Z(z)$. Finally, we propagate the sampled z to the model, and we get the corresponding sample y. The whole process can be summarized as

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

where $g_{x,\phi} = f_{x,\phi}^{-1}$ is the inverse function.

The core task in structured output learning is to predict the best output, i.e., y^*, for an input x. Many existing approaches solve the most-probable explanation (MPE) problem: $y^* = \arg\max_y p(y|x)$. However, MPE can be difficult for c-Glow because the likelihood function $p(y|x)$ is non-convex with a highly multi-modal surface. Optimization over y converges to a local optimum. In our experiments, we find the local optima to be only slightly better

than conditional samples. Therefore, we use sample averages to estimate marginal expectations of output variables. Let $\{z_1, ..., z_M\}$ be samples drawn from $p_Z(z)$. Estimated marginal expectations for each variable can be computed from the average

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

In the general form of c-Glow, the y variables are defined as continuous variables. In some tasks like semantic segmentation, the space of y is discrete. Following previous literature (Belanger & McCallum, 2016; Gygli et al., 2017), we relax the discrete output space to a continuous space during training. When we do prediction, we simply round y to discrete values.

5. Empirical Study

We test c-Glow on image segmentation. We use the Weizmann Horses database (Borenstein & Ullman, 2002), which contains 328 images of horses and their segmentation masks indicating whether pixels are part of horses or not. The training and test sets contain 200, and 128 images, respectively. We resize the images and their masks to 64×64 pixels. Due to the space limit, detailed experiment settings are in the appendix.

5.1. Semantic Segmentation

We compare our method with non-linear transformations (NLStruct) by Graber et al. (2018) and FCN-VGG[1] (Long et al., 2015). We use pixel-wise accuracy and mean intersection-over-union (IOU) as metrics. We reproduce NLStruct's performance reported by Graber et al. (2018). In their experiments, they set the input image size to be 224×224, and the mask size to be 64×64, which is slightly different from our setting. Since they did not report the accuracy of NLStruct, we leave that cell blank. Our c-Glow model generates higher quality segmentations than the baselines. Figure 1 shows some segmentation results.

Table 1. Segmentation metrics comparing c-Glow with others.

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

5.2. Conditional Sampling

Given an input x, we are interested in generating possible explanations for it. Doing so requires the model to have the ability to draw conditional samples, i.e., $y \sim p(y|x)$. We can directly use the generative process, i.e., Equation 2, to generate conditional samples. Figure 2 contains examples of these samples of segmentations for the horse image data.

[1]We use code from https://github.com/wkentaro/pytorch-fcn.

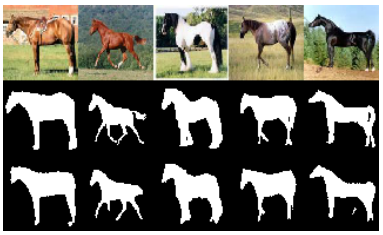


Figure 1. Sampled segmentation results. The top row contains the input images, the second row contains the ground truth masks, and the third row contains the c-Glow predictions.

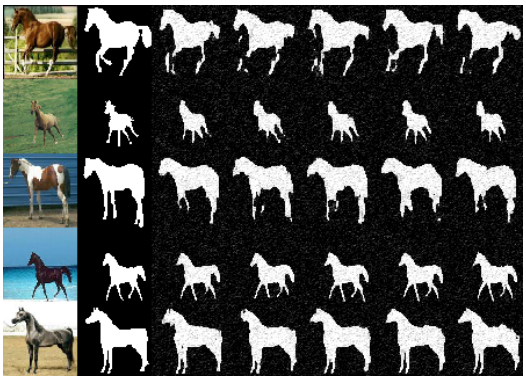


Figure 2. Conditional sampling from a learned C-Glow. The first column is the input images, the second column is the labels, and the last five columns are samples.

6. Conclusion

In this paper, we propose conditional generative flows (c-Glow), which are flow-based conditional generative models for structured output learning. The model is developed to allow the change-of-variables formula to transform conditional likelihood for high-dimensional variables. We show how to convert the Glow model to a conditional form by incorporating conditioning networks. In contrast with existing deep structured models, our model can be trained by directly minimizing the exact negative log-likelihood, so it does not need a surrogate objective or approximate inference. With a learned model, we can efficiently draw conditional samples from the exact learned distribution. In our experiments, we test c-Glow on image segmentation, finding that c-Glow can generate reasonable conditional samples and predictive abilities is comparable to recent deep structured prediction approaches.

Acknowledgments

We thank NVIDIA for their support through the GPU Grant Program and Amazon for their support via the AWS Cloud Credits for Research program.

References

Belanger, D. and McCallum, A. Structured prediction energy networks. In *International Conference on Machine Learning*, pp. 983–992, 2016.

Belanger, D., Yang, B., and McCallum, A. End-to-end learning for structured prediction energy networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 429–439, 2017.

Borenstein, E. and Ullman, S. Class-specific, top-down segmentation. In *European conference on computer vision*, pp. 109–122. Springer, 2002.

Chen, L.-C., Schwing, A., Yuille, A., and Urtasun, R. Learning deep structured models. In *International Conference on Machine Learning*, pp. 1785–1794, 2015.

Chen, T. Q., Rubanova, Y., Bettencourt, J., and Duvenaud, D. K. Neural ordinary differential equations. In *Advances in Neural Information Processing Systems*, pp. 6571–6583, 2018.

Dinh, L., Krueger, D., and Bengio, Y. NICE: Nonlinear independent components estimation. *arXiv preprint arXiv:1410.8516*, 2014.

Dinh, L., Sohl-Dickstein, J., and Bengio, S. Density estimation using real NVP. *arXiv preprint arXiv:1605.08803*, 2016.

Graber, C., Meshi, O., and Schwing, A. Deep structured prediction with nonlinear output transformations. In *Advances in Neural Information Processing Systems*, pp. 6320–6331, 2018.

Grathwohl, W., Chen, R. T., Betterncourt, J., Sutskever, I., and Duvenaud, D. Ffjord: Free-form continuous dynamics for scalable reversible generative models. *arXiv preprint arXiv:1810.01367*, 2018.

Gygli, M., Norouzi, M., and Angelova, A. Deep value networks learn to evaluate and iteratively refine structured outputs. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 1341–1351, 2017.

Ho, J., Chen, X., Srinivas, A., Duan, Y., and Abbeel, P. Flow++: Improving flow-based generative models with variational dequantization and architecture design. *arXiv preprint arXiv:1902.00275*, 2019.

Huang, C.-W., Krueger, D., Lacoste, A., and Courville, A. Neural autoregressive flows. *arXiv preprint arXiv:1804.00779*, 2018.

Kingma, D. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Kingma, D. P. and Dhariwal, P. Glow: Generative flow with invertible 1x1 convolutions. In *Advances in Neural Information Processing Systems*, pp. 10215–10224, 2018.

Kingma, D. P. and Welling, M. Auto-encoding variational Bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Kingma, D. P., Salimans, T., Jozefowicz, R., Chen, X., Sutskever, I., and Welling, M. Improved variational inference with inverse autoregressive flow. In *Advances in Neural Information Processing Systems*, pp. 4743–4751, 2016.

Koller, D., Friedman, N., and Bach, F. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

Krähenbühl, P. and Koltun, V. Efficient inference in fully connected CRFs with Gaussian edge potentials. In *Advances in Neural Information Processing Systems*, pp. 109–117, 2011.

Lafferty, J., McCallum, A., and Pereira, F. C. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *International Conference on Machine Learning*, 2001.

Long, J., Shelhamer, E., and Darrell, T. Fully convolutional networks for semantic segmentation. In *Computer Vision and Pattern Recognition*, pp. 3431–3440, 2015.

Ma, X. and Hovy, E. Macow: Masked convolutional generative flow. *arXiv preprint arXiv:1902.04208*, 2019.

Nowozin, S. and Lampert, C. H. Structured learning and prediction in computer vision. *Foundations and Trends in Computer Graphics and Vision*, 6:185–365, 2011.

Papamakarios, G., Pavlakou, T., and Murray, I. Masked autoregressive flow for density estimation. In *Advances in Neural Information Processing Systems*, pp. 2338–2347, 2017.

Rezende, D. J. and Mohamed, S. Variational inference with normalizing flows. *arXiv preprint arXiv:1505.05770*, 2015.

Sohn, K., Lee, H., and Yan, X. Learning structured output representation using deep conditional generative models. In *Advances in Neural Information Processing Systems*, pp. 3483–3491, 2015.

Trippe, B. L. and Turner, R. E. Conditional density estimation with Bayesian normalising flows. *arXiv preprint arXiv:1802.04908*, 2018.

Tu, L. and Gimpel, K. Learning approximate inference networks for structured prediction. *arXiv preprint arXiv:1803.03376*, 2018.

Wainwright, M. J. and Jordan, M. I. Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, 1:1–305, 2008.

Wang, S., Fidler, S., and Urtasun, R. Proximal deep structured models. In *Advances in Neural Information Processing Systems*, pp. 865–873, 2016.

Zheng, S., Jayasumana, S., Romera-Paredes, B., Vineet, V., Su, Z., Du, D., Huang, C., and Torr, P. H. Conditional random fields as recurrent neural networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1529–1537, 2015.

Ziegler, Z. M. and Rush, A. M. Latent normalizing flows for discrete sequences. *arXiv preprint arXiv:1901.10548*, 2019.

A. Experiment Settings

In this section, we introduce more details of our experiments.

A.1. Experiment Setup

For c-Glow, we use Adam (Kingma & Ba, 2014) to tune the learning rates, with $\alpha = 0.0002$, and default βs. We set the mini-batch size to be 2. Based on our empirical results, it is enough for the model to converge in reasonable amount of time. We follow (Kingma & Dhariwal, 2018) to preprocess the masks. That is, we copy each mask three times and tile them together, so y has three channels. We find that this transformation can improve the model performance. We run the program for 20000 iterations to guarantee it has fully converged.

A.2. Network Architectures

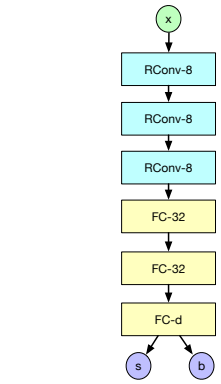
For c-Glow, we use the same multi-scale architecture as Glow to connect the layers, and we set $L = 3$, and $K = 8$. To specify a c-Glow architecture, we need to define conditioning networks that generate weights for the conditional actnorm, 1×1 convolutional, and affine layers.

For the conditional actnorm layer, we use a six-layer conditioning network. The first three layers are convolutional layers that downscale the input x to a reasonable size. The last three layers are then fully connected layers, which transform the resized x to the scale s and the bias b vectors. For the downscaling convolutional layers, we use a simple method to determine their kernel size and stride. Let H_i and H_o be the input and output sizes. Then we set the stride to H_i/H_o and the kernel size to $2 \times$ padding $+$ stride.

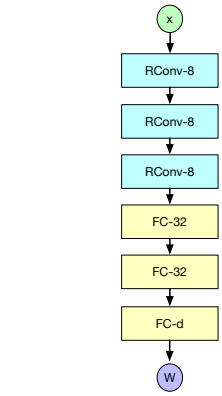
For the conditional 1×1 convolutional layer, we use a similar six-layer network to generate the weight matrix. The only difference is that the last fully connected layer will generate the weight matrix W. For the actnorm and 1×1 convolutional conditional networks, the number of channels of the convolutional layers, i.e., n_c, and the width of fully connected layers, i.e., n_w, will impact the model's performance. In our experiments, we set $n_c = 8$, and $n_w = 32$, which can get th best performance.

For the conditional affine layer, we use a three-layer conditional network to extract features from x, and we concatenate it with v_1. Among the three layers, the first and the last layers use 3×3 kernels. The middle layer is a downscaling convolutional layer. We vary the number of channels of this conditional network to be $\{8, 16, 32\}$, and we find that the model is not very sensitive to this variation. In our experiments, we fix it to have 16 channels. The affine layer itself is composed of three convolutional layers with 256 channels.

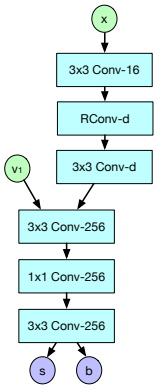
For all layers, we use the same zero initialization as (Kingma & Dhariwal, 2018), and for every layer except for the last one, we use ReLU to activate the output. The architectures of conditioning networks are in Figure 3.



(a) Conditional Actnorm



(b) Conditional 1x1 Convolutional

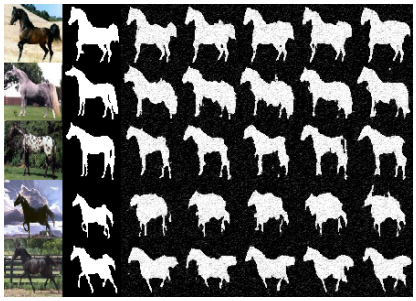


(c) Conditional Affine

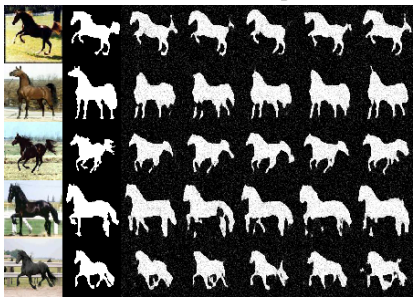
Figure 3. The networks we use to generate weights. The component "3x3 Conv-256" is a convolutional layer and the kernel size is 3×3, and the number of channels is 256. The component "FC-32" is a fully connected layer, and its width is 32. The parameter d depends on other variable sizes. In the conditional affine layer, d equals the number of channels of v_1. In the conditional actnorm layer, $d = 2c$, where c is the size of the scale. In the conditional 1x1 convolutional layer, W is a $c \times c$ matrix, so $d = c^2$. The "RConv" component is the convolutional layer for downscaling the input.

A.3. More Experiment Results

In this section, we show more results of conditional sampling, and semantic segmentation. The Figure 4 shows conditional samples, and the Figure 5 shows some sampled segmentation results. The images are from test set.



(a) Conditional samples.



(b) Conditional samples.

Figure 4. The conditional samples from test set. The first column is the input image, the second column is the true label, and the remaining columns are samples.

(a) Prediction results.

(b) Prediction results.

Figure 5. The sampled segmentation results. The first row is input images, the second row is ground-truth labels, and the last row is segmentation results.

A.4. Conditional Likelihoods

To the best of our knowledge, c-Glow is the first deep structured prediction model whose exact likelihood is tractable. Figure 6 plots the evolution of minibatch negative log likelihoods during training.

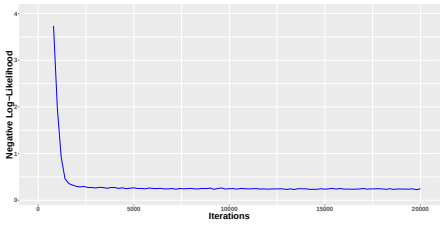


Figure 6. Evolution of negative log-likelihood during training.