Deep Value Networks Learn to Evaluate and Iteratively Refine Structured Outputs

Michael Gygli [1 *] Mohammad Norouzi [2] Anelia Angelova [2]

Abstract

We approach structured output prediction by optimizing a *deep value network* (DVN) to precisely estimate the task loss on different output configurations for a given input. Once the model is trained, we perform inference by gradient descent on the continuous relaxations of the output variables to find outputs with promising scores from the value network. When applied to image segmentation, the value network takes an image and a segmentation mask as inputs and predicts a scalar estimating the intersection over union between the input and ground truth masks. For multi-label classification, the DVN's objective is to correctly predict the F1 score for any potential label configuration. The DVN framework achieves the state-of-the-art results on multi-label prediction and image segmentation benchmarks.

1. Introduction

Structured output prediction is a fundamental problem in machine learning that entails learning a mapping from input objects to complex multivariate output structures. Because structured outputs live in a high-dimensional combinatorial space, one needs to design factored prediction models that are not only *expressive*, but also *computationally tractable* for both learning and inference. Due to computational considerations, a large body of previous work (*e.g.,* Lafferty et al. (2001); Tsochantaridis et al. (2004)) has focused on relatively weak graphical models with pairwise or small clique potentials. Such models are not capable of learning complex correlations among the random variables, making them not suitable for tasks requiring complicated high level reasoning to resolve ambiguity.

An expressive family of energy-based models studied by LeCun et al. (2006) and Belanger & McCallum (2016) exploits a neural network to score different joint configurations of inputs and outputs. Once the network is trained, one simply resorts to gradient-based inference as a mechanism to find low energy outputs. Despite recent developments, optimizing parameters of deep energy-based models remains challenging, limiting their applicability. Moving beyond large margin training used by previous work (Belanger & McCallum, 2016), this paper presents a simpler and more effective objective inspired by value based reinforcement learning for training energy-based models.

Our key intuition is that learning to *critique* different output configurations is easier than learning to directly come up with optimal predictions. Accordingly, we build a deep value network (DVN) that takes an input $\mathbf{x}$ and a corresponding output structure $\mathbf{y}$, both as inputs, and predicts a scalar score $v(\mathbf{x}, \mathbf{y})$ evaluating the quality of the configuration $\mathbf{y}$ and its correspondence with the input $\mathbf{x}$. We exploit a loss function $\ell(\mathbf{y}, \mathbf{y}^*)$ that compares an output $\mathbf{y}$ against a ground truth label $\mathbf{y}^*$ to teach a DVN to evaluate different output configurations. The goal is to distill the knowledge of the loss function into the weights of a value network so that during inference, in the absence of the labeled output $\mathbf{y}^*$, one can still rely on the value judgments of the neural net to compare outputs.

To enable effective iterative refinement of structured outputs via gradient ascent on the score of a DVN, similar to Belanger & McCallum (2016), we relax the discrete output variables to live in a continuous space. Moreover, we extend the domain of loss functions so the loss applies to continuous variable outputs. For example, for multi-label classification, instead of enforcing each output dimension y_i to be binary, we let $y_i \in [0, 1]$ and we generalize the notion of F_1 score to apply to continuous predictions. For image segmentation, we use a similar generalization of intersection over union. Then, we train a DVN on many output examples encouraging the network to predict precise (negative) loss scores for almost any output configuration. Figure 1 illustrates the gradient based inference process on a DVN optimized for image segmentation.

[*]Work done during an internship at Google Brain. [1]ETH Zürich & gifs.com [2]Google Brain, Mountain View, USA. Correspondence to: Michael Gygli <gygli@vision.ee.ethz.ch>, Mohammad Norouzi <mnorouzi@google.com>.

Proceedings of the 34^{th} International Conference on Machine Learning, Sydney, Australia, PMLR 70, 2017. Copyright 2017 by the author(s).

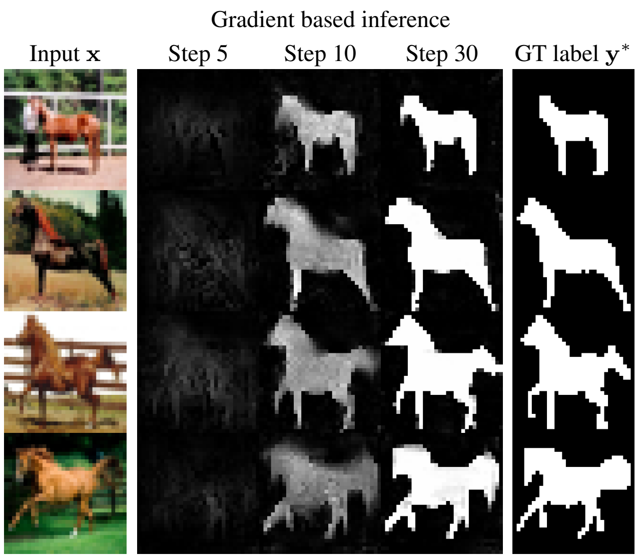

Figure 1. Segmentation results of DVN on Weizmann horses test samples. Our gradient based inference method iteratively refines segmentation masks to maximize the predicted scores of a deep value network. Starting from a black mask at step 0, the predictions converge within 30 steps yielding the output segmentation. See `https://goo.gl/8OLufh` for more & animated results.

This paper presents a novel training objective for deep structured output prediction, inspired by value-based reinforcement learning algorithms, to precisely evaluate the quality of any input-output pair. We assess the effectiveness of the proposed algorithm on multi-label classification based on text data and on image segmentation. We obtain state-of-the-art results in both cases, despite the differences of the domains and loss functions. Even given a small number of input-output pairs, we find that we are able to build powerful structure prediction models. For example, on the Weizmann horses dataset (Borenstein & Ullman, 2004), without any form of pre-training, we are able to optimize 2.5 million network parameters on only 200 training images with multiple crops. Our deep value network setup outperforms methods that are *pre-trained* on large datasets such as ImageNet (Deng et al., 2009) and methods that operate on $4\times$ larger inputs. Our source code based on TensorFlow (Abadi et al., 2015) is available at `https://github.com/gyglim/dvn`.

2. Background

Structured output prediction entails learning a mapping from input objects $\mathbf{x} \in \mathcal{X}$ (*e.g.,* $\mathcal{X} \equiv \mathbb{R}^M$) to multivariate discrete outputs $\mathbf{y} \in \mathcal{Y}$ (*e.g.,* $\mathcal{Y} \equiv \{0,1\}^N$). Given a training dataset of input-output pairs, $\mathcal{D} \equiv \{(\mathbf{x}^{(i)}, \mathbf{y}^{*(i)})\}_{i=1}^N$, we aim to learn a mapping $\widehat{\mathbf{y}}(\mathbf{x}) : \mathcal{X} \to \mathcal{Y}$ from inputs to ground truth outputs. Because finding the exact ground truth output structures in a high-dimensional space is often infeasible, one measures the quality of a mapping via a loss function $\ell(\mathbf{y}, \mathbf{y}') : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}^+$ that evaluates the distance between different output structures. Given such a loss function, the quality of a mapping is measured by empirical loss over a validation dataset $\mathcal{D}'$,

$$\sum_{(\mathbf{x},\mathbf{y}^*)\in\mathcal{D}'} \ell(\widehat{\mathbf{y}}(\mathbf{x}), \mathbf{y}^*) \tag{1}$$

This loss can take an arbitrary form and is often non-differentiable. For multi-label classification, a common loss is negative F_1 score and for image segmentation, a typical loss is negative intersection over union (IOU).

Some structured output prediction methods (Taskar et al., 2003; Tsochantaridis et al., 2004) learn a mapping from inputs to outputs via a score function $s(\mathbf{x}, \mathbf{y}; \boldsymbol{\theta})$, which evaluates different input-output configurations based on a linear function of some joint input-output features $\psi(\mathbf{x}, \mathbf{y})$,

$$s(\mathbf{x}, \mathbf{y}; \boldsymbol{\theta}) = \boldsymbol{\theta}^\mathsf{T} \psi(\mathbf{x}, \mathbf{y}) \,. \tag{2}$$

The goal of learning is to optimize a score function such that the model's predictions denoted $\widehat{\mathbf{y}}$,

$$\widehat{\mathbf{y}} \;=\; \underset{\mathbf{y}}{\operatorname{argmax}}\; s(\mathbf{x}, \mathbf{y}; \boldsymbol{\theta}) \,, \tag{3}$$

are closely aligned with ground-truth labels $\mathbf{y}^*$ as measured by empirical loss in (1) on the training set.

Empirical loss is not amenable to numerical optimization because the argmax in (3) is discontinuous. Structural SVM formulations (Taskar et al., 2003; Tsochantaridis et al., 2004) introduce a margin violation (slack) variable for each training pair, and define a continuous upper bound on the empirical loss. The upper bound on the loss for an example $(\mathbf{x}, \mathbf{y}^*)$ and the model's prediction $\widehat{\mathbf{y}}$ takes the form:

$$\begin{aligned} &\ell(\widehat{\mathbf{y}}, \mathbf{y}^*) \\ &\quad\le \max_{\mathbf{y}} \left[\ell(\mathbf{y}, \mathbf{y}^*) + s(\mathbf{x}, \mathbf{y}; \boldsymbol{\theta}) \right] - s(\mathbf{x}, \widehat{\mathbf{y}}; \boldsymbol{\theta}) && \text{(4a)} \\ &\quad\le \max_{\mathbf{y}} \left[\ell(\mathbf{y}, \mathbf{y}^*) + s(\mathbf{x}, \mathbf{y}; \boldsymbol{\theta}) \right] - s(\mathbf{x}, \mathbf{y}^*; \boldsymbol{\theta}) \,. && \text{(4b)} \end{aligned}$$

Previous work (Taskar et al., 2003; Tsochantaridis et al., 2004), defines a surrogate objective on the empirical loss, by summing over the bound in (4b) for different training examples, plus a regularizer. This surrogate objective is convex in $\boldsymbol{\theta}$, which makes optimization convenient.

This paper is inspired by the structural SVM formulation above, but we give up the convexity of the objective to obtain more expressive models using a multi-layer neural networks. Specifically, we generalize the formulation above in three ways: 1) use a non-linear score function denoted $v(\mathbf{x}, \mathbf{y}; \boldsymbol{\theta})$ that fuses $\psi(\cdot, \cdot)$ and $\boldsymbol{\theta}$ together and jointly learns the features. 2) use gradient descend in $\mathbf{y}$ for iterative refinement of outputs to approximately find the best $\widehat{\mathbf{y}}(\mathbf{x})$. 3) optimize the score function with a regression objective so that the predicted scores closely approximate the negative loss values,

$$\forall \mathbf{y} \in \mathcal{Y}, \quad v(\mathbf{x}, \mathbf{y}; \boldsymbol{\theta}) \approx -\ell(\mathbf{y}, \mathbf{y}^*) \; . \tag{5}$$

Our deep value network (DVN) is a non-linear function trying to evaluate the value of any output configuration $\mathbf{y} \in \mathcal{Y}$ accurately. In the structural SVM's objective, the score surface can vary as long as it does not violate margin constraints in (4b). By contrast, we restrict the score surface much more by penalizing it whenever it over- or underestimates the loss values. This seems to be beneficial as a neural network $v(\mathbf{x}, \mathbf{y}; \boldsymbol{\theta})$ has a lot of flexibility, and adding more suitable constraints can help regularization.

We call our model a *deep value network (DVN)* to emphasize the importance of the notion of *value* in shaping our ideas, but the DVN architecture can be thought as an example of structured prediction energy network (SPEN) (Belanger & McCallum, 2016) with similar inference strategy. Belanger & McCallum rely on the structural SVM surrogate objective to train their SPENs, whereas inspired by value based reinforcement learning, we learn an accurate estimate of the values as in (5). Empirically, we find that the DVN outperforms large margin SPENs on multi-label classification using a similar neural network architecture.

3. Learning a Deep Value Network

We propose a deep value network architecture, denoted $v(\mathbf{x}, \mathbf{y}; \boldsymbol{\theta})$, to evaluate a joint configuration of an input and a corresponding output via a neural network. More specifically, the deep value network takes as input both $\mathbf{x}$ and $\mathbf{y}$ jointly, and after several layers followed by non-linearities, predicts a scalar $v(\mathbf{x}, \mathbf{y}; \boldsymbol{\theta})$, which evaluates the quality of an output $\mathbf{y}$ and its compatibility with $\mathbf{x}$. We assume that during training, one has access to an oracle value function $v^*(\mathbf{y}, \mathbf{y}^*) = -\ell(\mathbf{y}, \mathbf{y}^*)$, which quantifies the quality of any $\mathbf{y}$. Such an oracle value function assigns optimal values to any input-output pairs given ground truth labels $\mathbf{y}^*$. During training, the goal is to optimize the parameters of a value network, denoted $\boldsymbol{\theta}$, to mimic the behavior of the oracle value function $v^*(\mathbf{y}, \mathbf{y}^*)$ as much as possible.

Example oracle value functions for image segmentation and multi-label classification include IOU and F_1 metrics, which are both defined on $(\mathbf{y}, \mathbf{y}^*) \in \{0,1\}^M \times \{0,1\}^M$,

$$v^*_{\text{IOU}}(\mathbf{y}, \mathbf{y}^*) = \frac{\mathbf{y} \cap \mathbf{y}^*}{\mathbf{y} \cup \mathbf{y}^*} \; , \tag{6}$$

$$v^*_{F_1}(\mathbf{y}, \mathbf{y}^*) = \frac{2\,(\mathbf{y} \cap \mathbf{y}^*)}{(\mathbf{y} \cap \mathbf{y}^*) + (\mathbf{y} \cup \mathbf{y}^*)} \; . \tag{7}$$

Here $\mathbf{y} \cap \mathbf{y}^*$ denotes the number of dimension i where both y_i and y_i^* are active and $\mathbf{y} \cup \mathbf{y}^*$ denotes the number of dimensions where at least one of y_i and y_i^* is active. Assuming that one has learned a suitable value network that attains $v(\mathbf{x}, \mathbf{y}; \boldsymbol{\theta}) \approx v^*(\mathbf{y}, \mathbf{y}^*)$ at every input-output pairs, in order to infer a prediction for an input $\mathbf{x}$, which is valued highly by the value network, one needs to find $\widehat{\mathbf{y}} = \text{argmax}_{\mathbf{y}}\, v(\mathbf{x}, \mathbf{y}; \boldsymbol{\theta})$ as described below.

3.1. Gradient based inference

Since $v(\mathbf{x}, \mathbf{y}; \boldsymbol{\theta})$ represents a complex non-linear function of $(\mathbf{x}, \mathbf{y})$ induced by a neural network, finding $\widehat{\mathbf{y}}$ is not straightforward, and approximate inference algorithms based on graph-cut (Boykov et al., 2001) or loopy belief propagation (Murphy et al., 1999) are not easily applicable. Instead, we advocate using a simple gradient descent optimizer for inference. To facilitate that, we relax the structured output variables to live in a real-valued space. For example, instead of using $\mathbf{y} \in \{0,1\}^M$, we use $\mathbf{y} \in [0,1]^M$. The key to make this inference algorithm work is that during training we make sure that our value estimates are optimized along the inference trajectory. Alternatively, one can make use of input convex neural networks (Amos et al., 2016) to guarantee convergence to optimal $\widehat{\mathbf{y}}$.

Given a continuous variable $\mathbf{y}$, to find a local optimum of $v(\mathbf{x}, \mathbf{y}; \boldsymbol{\theta})$ *w.r.t.* $\mathbf{y}$, we start from an initial prediction $\mathbf{y}^{(0)}$ (*i.e.,* $\mathbf{y}^{(0)} = [0]^M$ in all of our experiments), followed by gradient ascent for several steps,

$$\mathbf{y}^{(t+1)} = \mathcal{P}_{\mathcal{Y}}\left(\mathbf{y}^{(t)} + \eta \frac{\mathrm{d}}{\mathrm{d}\mathbf{y}}\, v(\mathbf{x}, \mathbf{y}^{(t)}; \boldsymbol{\theta})\right) , \tag{8}$$

where $\mathcal{P}_{\mathcal{Y}}$ denotes an operator that projects the predicted outputs back to the feasible set of solutions so that $\mathbf{y}^{(t+1)}$ remains in $\mathcal{Y}$. In the simplest case, where $\mathcal{Y} = [0,1]^M$, the $\mathcal{P}_{\mathcal{Y}}$ operator projects dimensions smaller than zero back to zero, and dimensions larger than one to one. After the final gradient step T, we simply round $\mathbf{y}^{(T)}$ to become discrete. Empirically, we find that for a trained DVN, the generated $\mathbf{y}^{(T)}$'s tend to become nearly binary themselves.

3.2. Optimization

To train a DVN using an oracle value function, first, one needs to extend the domain of $v^*(\mathbf{y}, \mathbf{y}^*)$ so it applies to continuous output $\mathbf{y}$'s. For our IOU and F_1 scores, we simply extend the notions of intersection and union by using element-wise min and max operators,

$$\mathbf{y} \cap \mathbf{y}^* \;=\; \sum\nolimits_{i=1}^{M} \min\left(y_i, y_i^*\right) \; , \tag{9}$$

$$\mathbf{y} \cup \mathbf{y}^* \;=\; \sum\nolimits_{i=1}^{M} \max\left(y_i, y_i^*\right) \; . \tag{10}$$

Substituting (9) and (10) into (6) and (7) provides a generalization of IOU and F_1 score to $[0,1]^M \times [0,1]^M$.

Our training objective aims at minimizing the discrepancy between $v(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})$ and $v^{*(i)}$ on a training set of triplets (input, output, value*) denoted $\mathcal{D} \equiv \{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)}, v^{*(i)}\}_{i=1}^{N}$. Very much like Q-learning (Watkins & Dayan, 1992), this training set evolves over time, and one can make use of an experience replay buffer. In Section 3.3, we discuss several strategies to generate training tuples and in our experiments we evaluate such strategies in terms of their empirical loss, once a gradient based optimizer is used to find $\widehat{\mathbf{y}}$.

Given a dataset of training tuples, one can use an appropriate loss to regress $v(\mathbf{x}, \mathbf{y})$ to v^* values. More specifically, since both IOU and F_1 scores lie between 0 and 1, we used a cross-entropy loss between oracle values *vs.* our DVN values. As such, our neural network $v(\mathbf{x}, \mathbf{y})$ has a sigmoid non-linearity at the top to predict a number between 0 and 1, and the loss takes the form,

$$\mathcal{L}_{\mathrm{CE}}(\boldsymbol{\theta}) = \sum_{(\mathbf{x},\mathbf{y},v^*)\in\mathcal{D}} - v^* \log v(\mathbf{x}, \mathbf{y}; \boldsymbol{\theta}) - (1 - v^*) \log(1 - v(\mathbf{x}, \mathbf{y}; \boldsymbol{\theta})) \tag{11}$$

The exact form of the loss does not have a significant impact on the performance and other loss functions can be used, e.g., L_2. A high level overview for training a DVN is shown in Algorithm 1. For simplicity, we show the case when not using a queue and batch size = 1.

Algorithm 1 Deep Value Network training

```
1: function TRAINEPOCH(training buffer $\mathcal{D}$, initial weights $\boldsymbol{\theta}$, learning rate $\lambda$)
2:     while not converged do
3:         $(\mathbf{x}, \mathbf{y}^*) \sim \mathcal{D}$                    ▷ Get a training example
4:         $\mathbf{y} \leftarrow$ GENERATEOUPUT($\mathbf{x}, \boldsymbol{\theta}$)          ▷ cf. Sec. 3.3
5:         $v^* \leftarrow v^*(\mathbf{y}, \mathbf{y}^*)$              ▷ Get oracle value for $\mathbf{y}$
6:         ▷ Compute loss based on estimation error cf. (11)
7:         $\mathcal{L} \leftarrow -v^* \log v(\mathbf{x}, \mathbf{y}; \boldsymbol{\theta})$
                   $-(1 - v^*) \log(1 - v(\mathbf{x}, \mathbf{y}; \boldsymbol{\theta}))$
8:         $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \lambda \frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\theta}} \mathcal{L}$                  ▷ Update DVN weights
9:     end while
10: end function
```

3.3. Generating training tuples

Each training tuple comprises an input, an output, and a corresponding oracle value, *i.e.,* $(\mathbf{x}, \mathbf{y}, v^*)$. The way training tuples are generated significantly impacts the performance of our structured prediction algorithm. In particular, it is important that the tuples are chosen such that they provide a good coverage of the space of possible outputs and result in a large learning signal. There exist several ways to generate training tuples including:

- running gradient based *inference* during training.
- generating *adversarial tuples* that have a large discrepancy between $v(\mathbf{x}, \mathbf{y}; \boldsymbol{\theta})$ and $v^*(\mathbf{y}, \mathbf{y}^*)$.
- *random samples* from $\mathcal{Y}$, maybe biased towards $\mathbf{y}^*$.

We elaborate on these methods below, and present a comparison of their performance in Section 5.4. Our ablation experiments suggest that combining examples from gradient based inference with adversarial tuples works best.

Ground truth. In this setup we simply add the ground truth outputs $\mathbf{y}^*$ into training with a $v^* = 1$ to provide some positive examples.

Inference. In this scenario, we generate samples by running a gradient based inference algorithm (Section 3.1) along our training. This procedure is useful because it helps learning a good value estimate on the output hypotheses that are generated along the inference trajectory at test time. To speed up training, we run a parallel inference job using slightly older neural network weights and accumulate the inferred examples in a queue.

Random samples. In this approach, we sample a solution $\mathbf{y}$ proportional to its exponentiated oracle value, *i.e.,* $\mathbf{y}$ is sampled with probability $p(\mathbf{y}) \propto \exp\{v^*(\mathbf{y}, \mathbf{y}^*)/\tau\}$, where $\tau > 0$ controls the concentration of samples in the vicinity of the ground truth. At $\tau = 0$ we recover the ground truth samples above. We follow (Norouzi et al., 2016) and sample from the exponentiated value distribution using stratified sampling, where we group $\mathbf{y}$'s according to their values. This approach provides a good coverage of the space of possible solutions.

Adversarial tuples. We maximize the cross-entropy loss used to train the value network (11) to generate adversarial tuples again using a gradient based optimizer (*e.g.,* see (Goodfellow et al., 2015; Szegedy et al., 2013). Such adversarial tuples are the outputs $\mathbf{y}$ for which the network over- or underestimates the oracle values the most. This strategy finds some difficult tuples that provide a useful learning signal, while ensuring that the value network has a minimum level of accuracy across all outputs $\mathbf{y}$.

4. Related work

There has been a surge of recent interest in using neural networks for structured prediction (Zheng et al., 2015; Chen et al., 2015; Song et al., 2016). The Structured Prediction Energy Network (SPEN) of (Belanger & McCallum, 2016) inspired in part by (LeCun et al., 2006) is identical to the DVN architecture. Importantly, the motivation and the learning objective for SPENs and DVNs are distinct – SPENs rely on a max-margin surrogate objective whereas we directly regress the energy of an input-output pair to its corresponding loss. Unlike SPENs that only consider multi-label classification problems, we also train a deep convolutional network to successfully address complex image segmentation problems.

Recent work has applied expressive neural networks to structured prediction to achieve impressive results on machine translation (Sutskever et al., 2014; Bahdanau et al., 2015) and image and audio synthesis (van den Oord et al., 2016b;a; Dahl et al., 2017). Such autoregressive models impose an order on the output variables and predict outputs one variable at a time by formulating a locally normalized probabilistic model. While training is often efficient, the key limitation of such models is inference complexity, which grows linearly in the number of output dimensions; this is not acceptable for high-dimensional output structures. By contrast, inference under our method is efficient as all of the output dimensions are updated in parallel.

Our approach is inspired in part by the success of previous work on value-based reinforcement learning (RL) such as Q-learning (Watkins, 1989; Watkins & Dayan, 1992) (see (Sutton & Barto, 1998) for an overview). The main idea is to learn an estimate of the future reward under the optimal behavior policy at any point in time. Recent RL algorithms use a neural network function approximator as the model to estimate the action values (Van Hasselt et al., 2016). We adopt similar ideas for structured output prediction, where we use the task loss as the optimal value estimate. Unlike RL, we use a gradient based inference algorithm to find optimal solutions at test time.

Gradient based inference, sometimes called *deep dreaming* has led to impressive artwork and has been influential in designing DVN (Gatys et al., 2015; Mordvintsev et al., 2015; Nguyen et al., 2016; Dumoulin et al., 2016). Deep dreaming and style transfer methods iteratively refine the input to a neural net to optimize a prespecified objective. Such methods often use a *pre-trained* network to define a notion of a perceptual loss (Johnson et al., 2016). By contrast, we train a task specific value network to learn the characteristics of a task specific loss function and we *learn* the network's weights from scratch.

Image segmentation (Arbelaez et al., 2012; Carreira et al., 2012; Girshick et al., 2014; Hariharan et al., 2015), is a key problem in computer vision and a canonical example of structured prediction. Many segmentation approaches based on Convolutional Neural Networks (CNN) have been proposed (Girshick et al., 2014; Chen et al., 2014; Eigen & Fergus, 2015; Long et al., 2015; Ronneberger et al., 2015; Noh et al., 2015). Most use a deep neural network to make a per-pixel prediction, thereby modeling pairs of pixels as being conditionally independent given the input.

To diminish the conditional independence problem, recent techniques propose to model dependencies among output labels to refine an initial CNN-based coarse segmentation. Different ways to incorporate pairwise dependencies within a segmentation mask to obtain more expressive models are proposed in (Chen et al., 2014; 2016; Ladickỳ et al., 2013; Zheng et al., 2015). Such methods perform joint inference of the segmentation mask dimensions via graph-cut (Li et al., 2015), message passing (Krähenbühl & Koltun, 2011) or loopy belief propagation (Murphy et al., 1999), to name a few variants. Some methods incorporate higher order potentials in CRFs (Kohli et al., 2009) or model global shape priors with Restricted Boltzmann Machines (Li et al., 2013; Kae et al., 2013; Yang et al., 2014; Eslami et al., 2014). Other methods learn to iteratively refine an initial prediction by CNNs, which may just be a coarse segmentation mask (Safar & Yang, 2015; Pinheiro et al., 2016; Li et al., 2016).

By contrast, this paper presents a new framework for training a score function by having a gradient based inference algorithm in mind during training. Our deep value network applies to generic structured prediction tasks, as opposed to some of the methods above, which exploit complex combinatorial structures and special constraints such as submodularity to design inference algorithms. Rather, we use expressive energy models and the simplest conceivable inference algorithm of all – gradient descent.

5. Experimental evaluation

We evaluate the proposed Deep Value Networks on 3 tasks: multi-label classification, binary image segmentation, and a 3-class face segmentation task. Section 5.4 investigates the sampling mechanisms for DVN training, and Section 5.5 visualizes the learned models.

5.1. Multi-label classification

We start by evaluating the method on the task of predicting tags from text inputs. We use standard benchmarks in multi-label classification, namely Bibtex and Bookmarks, introduced in (Katakis et al., 2008). In this task, multiple labels are possible per example, and the correct number is not known. Given the structure in the label space, methods modeling label correlations often outperform models with independent label predictions. We compare DVN to standard baselines including per-label logistic regression from (Lin et al., 2014), and a two-layer neural network with cross entropy loss (Belanger & McCallum, 2016), as well as SPENs (Belanger & McCallum, 2016) and PRLR (Lin et al., 2014), which is the state-of-the-art on these datasets. To allow direct comparison with SPENs, we adopt the same architecture in this paper. Such an architecture combines local predictions that are non-linear in $\mathbf{x}$, but linear in $\mathbf{y}$, with a so-called global network, which scores label configuration with a non-linear function of $\mathbf{y}$ independent of $\mathbf{x}$ (see Belanger & McCallum (2016), Eqs. (3) - (5)). Both local prediction and global networks have one or two hidden layers with Softplus non-linerarities. We follow the same experimental protocol and report F_1 scores on the same test split as (Belanger & McCallum, 2016).

Method	Bibtex	Bookmarks
Logistic regression (Lin et al., 2014)	37.2	30.7
NN baseline (Belanger & McCallum, 2016)	38.9	33.8
SPEN (Belanger & McCallum, 2016)	42.2	34.4
PRLR (Lin et al., 2014)	44.2	34.9
DVN (Ours)	44.7	37.1

Table 1. Tag prediction from text data. F_1 performance of Deep Value Networks compared to the state-of-the-art on multi-label classification. All prior results are taken from (Lin et al., 2014; Belanger & McCallum, 2016)

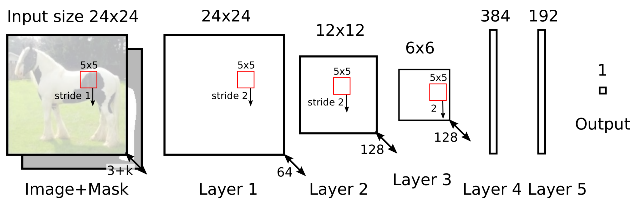

Figure 2. A deep value network with a feed-forward convolutional architecture, used for segmentation. The network takes an image and a segmentation mask as input and predicts a scalar evaluating the compatibility between the input pairs.

The results are summarized in Table 1. As can be seen from the table, our method outperforms the logistic regression baselines by a large margin. It also significantly improves over SPEN, despite not using any pre-training. SPEN, on the other hand, relies on pre-training of the feature network with a logistic loss to obtain good results. Our results even outperform (Lin et al., 2014). This is encouraging, as their method is specific to classification and encourages sparse and low-rank predictions, whereas our technique does not have such dataset specific regularizers.

5.2. Weizmann horses

The Weizmann horses dataset (Borenstein & Ullman, 2004) is a dataset commonly used for evaluating image segmentation algorithms (Li et al., 2013; Yang et al., 2014; Safar & Yang, 2015). The dataset consists of 328 images of left oriented horses and their binary segmentation masks. We follow (Li et al., 2013; Yang et al., 2014; Safar & Yang, 2015) and evaluate the segmentation results at 32×32 dimensions. Satisfactory segmentation of horses requires learning strong shape priors and complex high level reasoning, especially at a low resolution of 32×32 pixels, because small parts such as the legs are often barely visible in the RGB image. We follow the experimentation protocol of (Li et al., 2013) and report results on the same test split.

For the DVN we use a simple CNN architecture consisting of 3 convolutional and 2 fully connected layers (Figure 2). We use a learning rate of 0.01 and apply dropout on the first fully connected layer with the keeping probability 0.75

Input size	Method	Mean IOU %	Global IOU %
32×32	CHOPPS (Li et al., 2013)	69.9	-
	Fully conv (FCN) baseline	78.56	78.7
	DVN (Ours)	84.1	84.0
128×128	MMBM2 (Yang et al., 2014)	-	72.1
	MMBM2 + GC (Yang et al., 2014)	-	75.8
	Shape NN (Safar & Yang, 2015)	-	83.5

Table 2. Test IOU on Weizmann-32×32 dataset. DVN outperforms all previous methods, despite using a much lower input resolution than (Yang et al., 2014) and (Safar & Yang, 2015).

as determined on the validation set. We empirically found $\tau = 0.05$ to work best for stratified sampling. For training data augmentation purposes we randomly crop the image, similar to (Krizhevsky et al., 2012). At test time, various strategies are possible to obtain a full resolution segmentation, which we investigate in Section 5.4. For comparison we also implemented a Fully Convolutional Network (FCN) baseline (Long et al., 2015), by using the same convolutional layers as for the value network (*cf.* Figure 2). If not explicitly stated, masks are averaged over over 36 crops for our model and (Long et al., 2015) (see below).

We test and compare our model on the Weizmann horses segmentation task in Table 2. We tune the hyper-parameters of the model on a validation set and, once best hyper-parameters are found, fine-tune on the combination of training and validation sets. We report the mean image IOU, as well as the IOU over the whole test set, as commonly done in the literature. It is clear that our approach outperforms previous methods by a significant margin on both metrics. Our model shows strong segmentation results, without relying on externally trained CNN features as (*e.g.,* Safar & Yang (2015)). The weights of our value network are learned from scratch on crops of just 200 training images. Even though the number of examples is very small for this dataset, we did not observe overfitting during training, which we attribute to being able to generate a large set of segmentation masks for training.

In Figure 3 we show qualitative results for CHOPPS (Li et al., 2013), our implementation of fully convolutional networks (FCN) (Long et al., 2015), and our DVN model. When comparing our model to FCN, trained on the same data and resolution, we find that the FCN has challenges correctly segmenting legs and ensuring that the segmentation masks have a single connected component (*e.g.,* Figure 3, last two rows). Indeed, the masks produced by the DVN correspond to much more reasonable horse shapes as opposed to those of other methods – the DVN seem capable of learning complex shape models and effectively grounding them to visual evidence. We also note that in

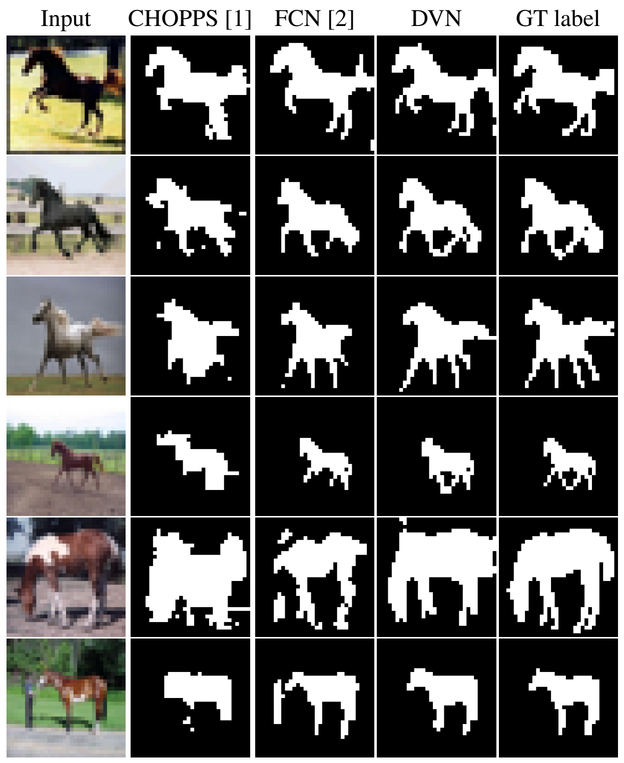

Figure 3. Qualitative results on the Weizmann 32 × 32 dataset. In comparison to previous works, DVN is able to learn a strong shape prior and thus correctly detect the horse shapes including legs. Previous methods are often misled by other objects or low contrast, thus generating inferior masks. References: [1] Li et al. (2013) [2] Our implementation of FCN (Long et al., 2015)

our comparison in Table 2, prior methods using larger inputs (*e.g.,* 128×128) are also outperformed by DVNs.

5.3. Labeled Faces in the Wild

The Labeled Faces in the Wild (LFW) dataset (Huang et al., 2007) was proposed for face recognition and contains more than 13000 images. A subset of 2927 faces was later annotated for segmentation by Kae et al. (2013). The labels are provided on a superpixel basis and consist of 3 classes: face, hair and background. We use this dataset to test the application of our approach to multiclass segmentation. We use the same train, validation, and test splits as (Kae et al., 2013; Tsogkas et al., 2015). As our method predicts labels for pixels, we follow (Tsogkas et al., 2015) and map pixel labels to superpixels by using the most frequent label in a superpixel as the class. To train the DVN, we use mean pixel accuracy as our oracle value function, instead of superpixel accuracy.

Table 3 shows quantitative results. DVN performs reasonably well, but is outperformed by state of the art methods on this dataset. We attribute this to three reasons. (i) the pre-training and more direct optimization of the per-pixel prediction methods of (Tsogkas et al., 2015; Long et al., 2015), (ii) the input resolution and (iii) the properties of the dataset. In contrast to horses, faces do not have thin parts and exhibit limited deformations. Thus, a feed forward method as used in (Long et al., 2015), which produces coarser and smooth predictions is sufficient to obtain good results. Indeed, this has also been observed in the negligible improvement of refining CNN predictions with Conditional Random Fields and Restricted Boltzmann machines (*cf.* Table 3 last three rows). Despite this, our model is able to learn a prior on the shape and align it with the image evidence in most cases. Some failure cases include failing to recognize subtle and more rare parts such as mustaches, given their small size, and difficulties in correctly labeling blond hair. Figure 4 shows qualitative results of our segmentation method on this dataset.

Input size	Method	SP Acc. %
32^2	Fully conv (FCN) baseline	95.36
	DVN (Ours)	92.44
250^2	CRF (as in Kae et al. (2013))	93.23
	GLOC (Kae et al., 2013)	94.95
	DNN (Tsogkas et al., 2015)	96.54
	DNN+CRF+SBM (Tsogkas et al., 2015)	96.97

Table 3. Superpixel accuracy (SP Acc.) on Labeled Faces in the Wild test set.

Configuration	Mean IOU %
Inference + Ground Truth	76.7
Inference + Stratified Sampling	80.8
Inference + Adversarial (DVN)	81.6
DVN + Mask averaging (9 crops)	81.3
DVN + Joint inference (9 crops)	81.6
DVN + Mask avg. non-binary (25 crops)	69.6
DVN + Joint inf. non-binary (25 crops)	80.3
DVN + Mask averaging (25 crops)	83.1
DVN + Joint inference (25 crops)	83.1

Table 4. Test performance of different configurations on the Weizmann 32x32 dataset.

5.4. Ablation experiments

In this section we analyze different configurations of our method. As already mentioned, generating appropriate training data for our method is key to learning good value networks. We compare 3 main approaches: 1) inference + ground truth, 2) inference + stratified sampling, and 3) inference + adversarial training. These experiments are conducted on the Weizmann dataset, described above. Table 4, top portion, reports IOU results for different approaches for training the dataset. As can be seen, including adversarial training works best, followed by stratified sampling. Both of these methods help explore the space of segmentation

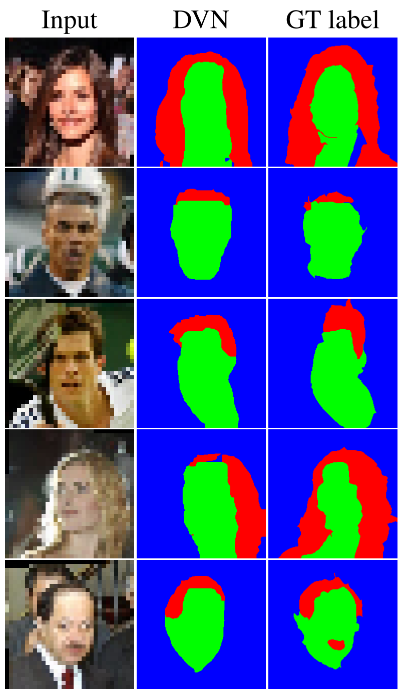

Figure 4. Qualitative results on 3-class segmentation on the LFW dataset. The last two rows show failure cases, where our model does not detect some of hair and moustache correctly.

masks in the vicinity of ground truth masks better, as opposed to just including the ground truth masks. Adding adversarial examples works better than stratified sampling, as the adversarial examples are the masks on which the model is least accurate. Thus, these masks provide useful gradient information as to help improve the model.

We also investigate ways to do model averaging (Table 4, bottom portion). Averaging the segmentation masks of multiple crops leads to improved performance. When the masks are averaged naïvely, the result becomes blurry, making it difficult to obtain a final segmentation. Instead, joint inference updates the complete segmentation mask in each step, using the gradients of the individual crops. This procedure leads to clean, near-binary segmentation masks. This is manifested in the performance when using the raw foreground confidence (Table 4, Mask averaging non-binary *vs.* Joint inference non-binary). Joint inference leads to somewhat improved segmentation results, even after binarization, in particular when using fewer crops.

5.5. Visualizing the learned correlations

To visualize what the model has learned, we run our inference algorithm on the mean image of the Weizmann dataset (training split). Optionally, we perturb the mean image by adding some Gaussian noise. The masks obtained through this procedure are shown in Figure 5. As one can see, the segmentation masks found by the value network on (noisy) mean images resemble a side-view of a horse with some uncertainty on the leg and head positions. These parts have the most amount of variation in the dataset. Even though noisy images do not contain horses, the value network hallucinates proper horse silhouettes, which is what our model is trained on.

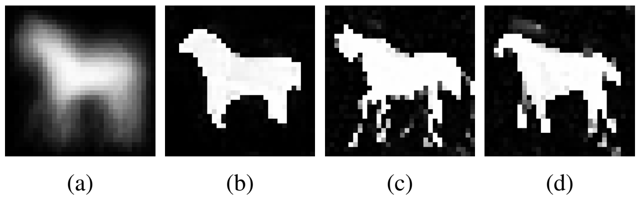

Figure 5. Visualization of the learned horse shapes on the Weizmann dataset. From left to right (a) The mean mask of the training set (b) mask generated when providing the mean horse image from the training set (c, d) Outputs generated by our model given mean horse image plus Gaussian noise ($\sigma = 10$) as the input.

6. Conclusion

This paper presents a framework for structured output prediction by learning a deep value network that predicts the quality of different output hypotheses for a given input. As the DVN learns to predict a value based on both, input and output, it implicitly learns a prior over output variables and takes advantage of the joint modelling of the inputs and outputs. By visualizing the prior for image segmentation, we indeed find that our model learns realistic shape priors. Furthermore, rather than learning a model by optimizing a surrogate loss, using DVNs allows to directly train a network to accurately predict the desired performance metric (*e.g.,* IOU), even if it is non-differentiable. We apply our method to several standard datasets in multi-label classification and image segmentation. Our experiments show that DVNs apply to different structured prediction problems, achieving state-of-the-art results with no pre-training.

As future work, we plan to improve the scalability and computational efficiency of our algorithm by inducing input features computed solely on $\mathbf{x}$, which is going to be computed only once. The gradient based inference can improve by injecting noise to the gradient estimate, similar to Hamiltonian Monte Carlo sampling. Finally, one can explore better ways to initialize the inference process.

7. Acknowledgment

We thank Kevin Murphy, Ryan & George Dahl, Vincent Vanhoucke, Zhifeng Chen, and the Google Brain team for insightful comments and discussions.

References

Abadi, Martín, Agarwal, Ashish, Barham, Paul, Brevdo, Eugene, Chen, Zhifeng, Citro, Craig, Corrado, Greg S., Davis, Andy, Dean, Jeffrey, Devin, Matthieu, Ghemawat, Sanjay, Goodfellow, Ian, Harp, Andrew, Irving, Geoffrey, Isard, Michael, Jia, Yangqing, Jozefowicz, Rafal, Kaiser, Lukasz, Kudlur, Manjunath, Levenberg, Josh, Mané, Dan, Monga, Rajat, Moore, Sherry, Murray, Derek, Olah, Chris, Schuster, Mike, Shlens, Jonathon, Steiner, Benoit, Sutskever, Ilya, Talwar, Kunal, Tucker, Paul, Vanhoucke, Vincent, Vasudevan, Vijay, Viégas, Fernanda, Vinyals, Oriol, Warden, Pete, Wattenberg, Martin, Wicke, Martin, Yu, Yuan, and Zheng, Xiaoqiang. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL `http://tensorflow.org/`. Software available from tensorflow.org.

Amos, Brandon, Xu, Lei, and Kolter, J Zico. Input convex neural networks. *arXiv:1609.07152*, 2016.

Arbelaez, Pablo, Hariharan, Bharath, Gu, Chunhui, Gupta, Saurabh, Bourdev, Lubomir, and Malik, Jitendra. Semantic segmentation using regions and parts. *CVPR*, 2012.

Bahdanau, Dzmitry, Cho, Kyunghyun, and Bengio, Yoshua. Neural machine translation by jointly learning to align and translate. *ICLR*, 2015.

Belanger, David and McCallum, Andrew. Structured prediction energy networks. *ICML*, 2016.

Borenstein, E. and Ullman, S. Learning to segment. *ECCV*, 2004.

Boykov, Yuri, Veksler, Olga, and Zabih, Ramin. Fast approximate energy minimization via graph cuts. *IEEE Trans. PAMI*, 2001.

Carreira, Joao, Caseiro, Rui, Batista, Jorge, and Sminchisescu, Cristian. Semantic segmentation with second-order pooling. *ECCV*, 2012.

Chen, Liang-Chieh, Papandreou, George, Kokkinos, Iasonas, Murphy, Kevin, and Yuille, Alan L. Semantic image segmentation with deep convolutional nets and fully connected crfs. *arXiv:1412.7062*, 2014.

Chen, Liang-Chieh, Schwing, Alexander, Yuille, Alan, and Urtasun, Raquel. Learning deep structured models. *ICML*, 2015.

Chen, Liang-Chieh, Papandreou, Iasonas, Murphy, Kevin, and Yuille, Alan L. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *arXiv:1606.00915*, 2016.

Dahl, Ryan, Norouzi, Mohammad, and Shlens, Jonathon. Pixel recursive super resolution. *arXiv:1702.00783*, 2017.

Deng, Jia, Dong, Wei, Socher, Richard, Li, Li-Jia, Li, Kai, and Fei-Fei, Li. ImageNet: A Large-Scale Hierarchical Image Database. *CVPR*, 2009.

Dumoulin, Vincent, Shlens, Jonathon, and Kudlur, Manjunath. A learned representation for artistic style. 2016.

Eigen, David and Fergus, Rob. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. *ICCV*, 2015.

Eslami, SM Ali, Heess, Nicolas, Williams, Christopher KI, and Winn, John. The shape boltzmann machine: a strong model of object shape. *IJCV*, 2014.

Gatys, Leon A, Ecker, Alexander S, and Bethge, Matthias. A neural algorithm of artistic style. *arXiv:1508.06576*, 2015.

Girshick, Ross, Donahue, Jeff, Darrell, Trevor, and Malik, Jitendra. Rich feature hierarchies for accurate object detection and semantic segmentation. *CVPR*, 2014.

Goodfellow, Ian J, Shlens, Jonathon, and Szegedy, Christian. Explaining and harnessing adversarial examples. *ICLR*, 2015.

Hariharan, Bharath, Arbelaez, Pablo, and Girshick, Ross. Hypercolumns for object segmentation and fine-grained localization. *CVPR*, 2015.

Huang, Gary B, Ramesh, Manu, Berg, Tamara, and Learned-Miller, Erik. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical report, Technical Report, University of Massachusetts, Amherst, 2007.

Johnson, Justin, Alahi, Alexandre, and Fei-Fei, Li. Perceptual losses for real-time style transfer and super-resolution. *ECCV*, 2016.

Kae, Andrew, Sohn, Kihyuk, Lee, Honglak, and Learned-Miller, Erik. Augmenting crfs with boltzmann machine shape priors for image labeling. *CVPR*, 2013.

Katakis, Ioannis, Tsoumakas, Grigorios, and Vlahavas, Ioannis. Multilabel text classification for automated tag suggestion. *ECML PKDD discovery challenge*, 2008.

Kohli, Pushmeet, Torr, Philip HS, et al. Robust higher order potentials for enforcing label consistency. *IJCV*, 2009.

Krähenbühl, Philipp and Koltun, Vladlen. Efficient inference in fully connected crfs with gaussian edge potentials. *NIPS*, 2011.

Krizhevsky, Alex, Sutskever, Ilya, and Hinton, Geoffrey E. Imagenet classification with deep convolutional neural networks. *NIPS*, 2012.

Ladický, L'ubor, Russell, Chris, Kohli, Pushmeet, and Torr, Philip HS. Inference methods for crfs with co-occurrence statistics. *IJCV*, 2013.

Lafferty, John, McCallum, Andrew, Pereira, Fernando, et al. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. *ICML*, 2001.

LeCun, Yann, Chopra, Sumit, Hadsell, Raia, Ranzato, M, and Huang, F. A tutorial on energy-based learning. *Predicting structured data*, 2006.

Li, Jianchao, Wang, Dan, Yan, Canxiang, and Shan, Shiguang. Object segmentation with deep regression. *ICIP*, 2015.

Li, Ke, Hariharan, Bharath, and Malik, Jitendra. Iterative instance segmentation. *CVPR*, 2016.

Li, Yujia, Tarlow, Daniel, and Zemel, Richard. Exploring compositional high order pattern potentials for structured output learning. *CVPR*, 2013.

Lin, Victoria (Xi), Singh, Sameer, He, Luheng, Taskar, Ben, and Zettlemoyer, Luke. Multi-label learning with posterior regularization. *NIPS Workshop on Modern Machine Learning and Natural Language Processing*, 2014.

Long, Jonathan, Shelhamer, Evan, and Darrell, Trevor. Fully convolutional networks for semantic segmentation. *CVPR*, 2015.

Mordvintsev, Alexander, Olah, Christopher, and Tyka, Mike. Inceptionism: Going deeper into neural networks. *Google Research Blog.*, 2015.

Murphy, Kevin P, Weiss, Yair, and Jordan, Michael I. Loopy belief propagation for approximate inference: An empirical study. *UAI*, 1999.

Nguyen, Anh, Dosovitskiy, Alexey, Yosinski, Jason, Brox, Thomas, and Clune, Jeff. Synthesizing the preferred inputs for neurons in neural networks via deep generator networks. *arXiv:1605.09304*, 2016.

Noh, Hyeonwoo, Hong, Seunghoon, and Han, Bohyung. Learning deconvolution network for semantic segmentation. *ICCV*, 2015.

Norouzi, Mohammad, Bengio, Samy, Chen, Zhifeng, Jaitly, Navdeep, Schuster, Mike, Wu, Yonghui, and Schuurmans, Dale. Reward augmented maximum likelihood for neural structured prediction. *NIPS*, 2016.

Pinheiro, P., Lin, T.-Y., Collobert, R., , and Dollar, P. Learning to refine object segments. *ECCV*, 2016.

Ronneberger, Olaf, Fischer, Philipp, and Brox, Thomas. U-net: Convolutional networks for biomedical image segmentation. *MICCAI*, 2015.

Safar, Simon and Yang, Ming-Hsuan. Learning shape priors for object segmentation via neural networks. *ICIP*, 2015.

Song, Yang, Schwing, Alexander, Zemel, Richard, and Urtasun, Raquel. Training deep neural networks via direct loss minimization. *ICML*, 2016.

Sutskever, Ilya, Vinyals, Oriol, and Le, Quoc V. Sequence to sequence learning with neural networks. *NIPS*, 2014.

Sutton, Richard and Barto, Andrew. Reinforcement learning: An introduction. *The MIT Press*, 1998.

Szegedy, Christian, Zaremba, Wojciech, Sutskever, Ilya, Bruna, Joan, Erhan, Dumitru, Goodfellow, Ian, and Fergus, Rob. Intriguing properties of neural networks. *ICLR*, 2013.

Taskar, B., Guestrin, C., and Koller, D. Max-margin Markov networks. *NIPS*, 2003.

Tsochantaridis, I., Hofmann, T., Joachims, T., and Altun, Y. Support vector machine learning for interdependent and structured output spaces. *ICML*, 2004.

Tsogkas, Stavros, Kokkinos, Iasonas, Papandreou, George, and Vedaldi, Andrea. Deep learning for semantic part segmentation with high-level guidance. *arXiv:1505.02438*, 2015.

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

van den Oord, Aaron, Kalchbrenner, Nal, Espeholt, Lasse, Kavukcuoglu, Koray, Vinyals, Oriol, and Graves, Alex. Conditional image generation with pixelcnn decoders. *NIPS*, 2016b.

Van Hasselt, Hado, Guez, Arthur, and Silver, David. Deep reinforcement learning with double q-learning. *AAAI*, 2016.

Watkins, Christopher J. C. H. and Dayan, Peter. Q-learning. *Machine Learning*, 1992.

Watkins, Christopher JCH. *Learning from delayed rewards*. PhD thesis, University of Cambridge England, 1989.

Yang, Jimei, Safar, Simon, and Yang, Ming-Hsuan. Max-margin boltzmann machines for object segmentation. *CVPR*, 2014.

Zheng, Shuai, Jayasumana, Sadeep, Romera-Paredes, Bernardino, Vineet, Vibhav, Su, Zhizhong, Du, Dalong, Huang, Chang, and Torr, Philip HS. Conditional random fields as recurrent neural networks. *CVPR*, 2015.