Learning Neural Parsers with Deterministic Differentiable Imitation Learning

Tanmay Shankar
Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213
tanmayshankar@cmu.edu

Nicholas Rhinehart
Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213
nrhineha@cs.cmu.edu

Katharina Muelling
Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213
kmuelling@nrec.ri.cmu.edu

Kris M. Kitani
Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213
kkitani@cs.cmu.edu

Abstract:
We explore the problem of learning to decompose spatial tasks into segments, as exemplified by the problem of a painting robot covering a large object. Inspired by the ability of classical decision tree algorithms to construct structured partitions of their input spaces, we formulate the problem of decomposing objects into segments as a parsing approach. We make the insight that the derivation of a parse-tree that decomposes the object into segments closely resembles a decision tree constructed by ID3, which can be done when the ground-truth available. We learn to imitate an expert parsing oracle, such that our neural parser can generalize to parse natural images without ground truth. We introduce a novel deterministic policy gradient update, DRAG (i.e., DeteRministically AGgrevate) in the form of a deterministic actor-critic variant of AggreVaTeD [1], to train our neural parser. From another perspective, our approach is a variant of the Deterministic Policy Gradient [2, 3] suitable for the imitation learning setting. The deterministic policy representation offered by training our neural parser with DRAG allows it to outperform state of the art imitation and reinforcement learning approaches.

Keywords: Imitation Learning, Reinforcement Learning, Parsing

1 Introduction

Consider the task of a robot painting an object or an aerial robot surveying a large field. These spatial tasks represent a coverage problem that the robot may not be able to address in a single shot. For instance, a robot may not be able to paint the entirety of a large object with a single stroke, being limited by the footprint of its paint brush. Instead, the robot must decompose the spatial task of painting objects into smaller segments that it can cover in single stroke. However, discovering or learning an appropriate decomposition of such tasks into segments is challenging. In the object-painting problem, there may be several constraints upon the resultant segments, such as the overall paint coverage. Further, there exist multiple ways to decompose an object into constituent segments - for example, an object may be decomposed into length or breadth-wise segments.

A few well studied algorithms are able to somewhat circumvent these challenges. In particular, when applied to the task of classifying a set of points, ID3 [4] and C4.5 [5] recursively partition the input space in order to achieve an accurate classification of the points. As we demonstrate in Section 3, an ID3-like Information Gain maximizing algorithm, IGM, builds a similar partitioning of an object as in Fig. 1. However, this IGM algorithm (like the ID3 and C4.5 algorithms it is derived from) requires ground truth classification labels, and cannot be applied to decompose novel objects for which the ground truth labels are unavailable, or are expensive to obtain.

2nd Conference on Robot Learning (CoRL 2018), Zurich, Switzerland.

To bypass this issue, we approach the problem of decomposing objects into segments as a *parsing* problem, based on the insight that the derivation of a parse-tree (Fig. 1) that decomposes a given object into segments closely resembles a decision tree constructed by IGM (Fig. 1). Rather than learn to parse objects by reinforcement learning (RL) as in [6], we propose to learn how to parse objects into such structured decompositions via *imitation learning* (IL), treating the IGM algorithm as a *parsing oracle*. Our neural parser is trained to parse objects by imitating the IGM oracle *observing only raw object images* as input, while the IGM oracle exploits access to ground truth information to demonstrate how to parse a particular object. This allows our neural parser to construct structured decompositions of novel unseen objects despite lacking ground truth information. As expected, our imitation learning approach significantly outperforms reinforcement learning baselines in practice.

We further introduce a novel deterministic policy gradient update, DRAG, suitable for training deterministic policies in the hybrid imitation-reinforcement learning setting. The DRAG policy gradient update serves to train the deterministic policy component of our neural parser, eliminating the complexity of maintaining probability distributions specific to the parsing setting. By rephrasing the AggreVaTeD [1] objective in the deterministic policy case, we retrieve a gradient update to a deterministic policy that relies on a differentiable approximation to the oracle's cost-to-go. This policy gradient update may be viewed as a *deterministic actor-critic variant of AggreVaTeD*, which we refer to as DRAG (i.e., DeteRministically AGgrevate). DRAG may also be viewed as a variant of the Deterministic Policy Gradient [3] suitable for imitation learning, and replaces an approximation to the true gradient in the original Deterministic Policy Gradient [3], with the true policy gradient.

Training our parser via DRAG allows our parser to outperform several baselines on the task of parsing novel objects, showcasing its potential to achieve performance closer to that of the oracle than several other existing imitation and reinforcement learning approaches.

2 Related Work

Facade Parsing: Facade parsing attempts to identify the topology of a building facade, by parsing an image of the facade into its various components [7, 6, 8, 9, 10]. [7, 6] learn to apply production rules of a grammar to reduce a shape into its constituent segments in the RL setting. We build on [6], addressing the problem of decomposing objects. In contrast to other facade parsing approaches that use labels of the *resulting parse* [8, 9, 10], we seek to imitate the *decisions* of an expert parser.

Policy Gradient Reinforcement Learning: Stochastic policy gradient approaches [11, 12, 13, 14] have been used to learn control policies in the reinforcement learning (RL) setting. Silver et al. [2] introduced the Deterministic Policy Gradient (DPG), a deterministic counterpart to [11], and later extended DPG to the function approximator case [3]. We introduce a variant of DPG [2, 3] suitable for imitation learning, that removes the approximation of the true gradient used in DPG [2, 3].

Imitation Learning: Recent imitation learning algorithms [15, 16, 17] address the setting when one has access to an expert policy that may be queried. Ross et al. [17] demonstrated an *interactive* imitation learning paradigm, DAgger, is preferable over a naive behavioural cloning approach. Ross and Bagnell [18] further introduced AggreVaTe, using estimates of the cost-to-go of the expert to better learn control policies. Sun et al. [1] subsequently derived a stochastic policy gradient update of the AggreVaTe [18] objective, enabling its use on complex neural network policies. [19, 20] further explore the hybrid imitation and reinforcement learning, by reward shaping using an oracle and switching to policy gradient RL after imitation respectively. We follow [18, 1], by training agents with partial information to imitate oracles with full information at train time, as in [21, 22].

Semantic Segmentation: The problem of semantic segmentation addresses assigning semantic labels to pixels in a given image. [23] employed a graph partitioning approach for image segmentation, to construct image superpixels. Several recent works [24, 25, 23, 26] train end-to-end models on large scale datasets for semantic segmentation, as more thoroughly reviewed in [27]. While the notion of a set of segments with assigned labels is common to our parsing approach and semantic segmentation, our approach builds a hierarchical decomposition of the object image as the end result. Our problem further differs in that constraints may be imposed that exclude arbitrary results such as the aforementioned stroke coverage constraints of a painting robot.

3 Method

We seek to learn how to parse objects directly using object images as input, by learning to imitate an expert parsing oracle. The connection between object parsing and the decision trees constructed by IGM Fig. 1 afford us such a parsing oracle, that makes the imitation setting preferable over the

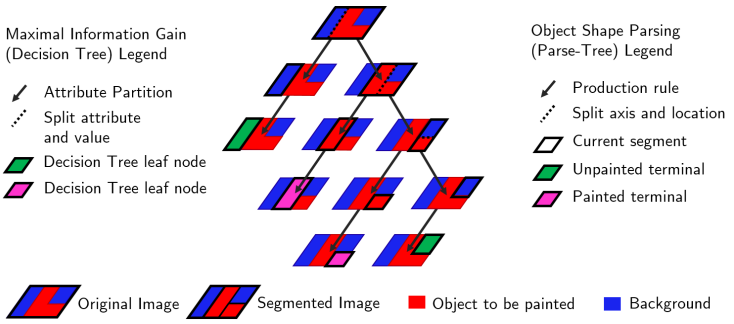

Figure 1: Constructing an equivalent hierarchical decomposition of an object image by two methods. The Information Gain Maximization algorithm creates a decision tree (legend on the left), while the Shape Parsing approach to the right constructs a parse tree (legend to the right). The equivalent hierarchical decomposition (center), shows correspondence of each node in the decision tree and parse tree to an image segment.

RL setting followed in [6]. To successfully learn a policy capable of imitating a parsing oracle, we introduce DRAG, a deterministic actor-critic variant of AggreVaTeD [1]. We explain our approach to learning this parsing policy by first describing the problem setting of parsing objects, then describing the IGM algorithm that serves as a ground truth parsing oracle. We then demonstrate how the object parsing problem may be framed as an MDP, finally highlighting how we use DRAG to train a neural parser to decompose objects.

Consider the following problem setting. Given an object image to be painted, our objective is to decompose the object image into a set of segments, maximizing the paint coverage of the object while minimizing the "paint wasted". We may view this as assigning labels of whether or not to paint each pixel, where painting object pixels increases coverage, while not painting the surroundings decreases "wasted paint". Under this label assignment problem, our objective translates to constructing an object decomposition (a set of segments) constrained to mutually compose the object image, while *accurately* assigning labels of whether or not to paint each of the resultant segments.

3.1 Parsing Objects by Imitating Maximal Information Gain

Representing the painting process as a labeling problem allows us to employ decision tree algorithms such as ID3 [4] and C4.5 [5], that naturally address labeling tasks by partitioning the space (i.e., the object image) into regions that each contain pixels of a single class. ID3 achieves this by using ground truth information to select the partitioning with the maximum information gain over the resultant segments. By modifying ID3 to allow multiple splits (i.e., partitions) along a particular axis (or attributes in ID3), we construct a ground-truth oracle that is able to perfectly label any object image allowed sufficient partitions. We refer to this oracle as the Information Gain Maximization algorithm (IGM), or π_{IGM}. Fig. 1 depicts such a "decision tree" constructed by IGM on a toy image.

The IGM oracle uses ground truth labeling of object images, which are unavailable for novel objects. To address this issue, we draw inspiration from facade parsing literature [7, 6], where images of facades were decomposed into various components via *shape-parsing*. We observe that parse-trees that decompose an image into segments in [6] resemble the decision trees constructed by IGM (Fig. 1). Motivated by this insight, we decompose objects into segments via a *parsing* approach.

3.2 Shape Parsing Objects

To learn how to decompose object images into their constituent segments, we require an appropriate representation of the recursive object decompositions that arise in the divide-and-conquer paradigm. Shape parsing [6] provides us a compact representation of such potentially complex decompositions of an object. We adopt a *shape parsing* approach similar to [6], using a binary split grammar to represent the hierarchical object decomposition. Formally, we use a probabilistic context-free grammar $\mathcal{G}$, defined as a tuple $\mathcal{G} = (\mathcal{V}, \mathcal{T}, \mathcal{R}, V_0, \mathcal{P})$, where $\mathcal{V}$ is a set of non-terminal symbols, $\mathcal{T}$ is a set of terminal symbols, V_0 is a starting symbol, $\mathcal{R}$ is a set of production rules, and $\mathcal{P}$ defines a set of probabilities of applying the production rules on a given non-terminal V.

Set of non-terminal symbols $\mathcal{V}$: A non-terminal symbol $V \in \mathcal{V}$ is defined as an axis-aligned rectangle over the input image. Each non-terminal is specified with a set of attributes (x, y, w, h), where (x, y) defines the origin of the rectangle and (w, h) define the spatial extents of the rectangle along the horizontal and vertical directions respectively. The starting symbol $V_0 \in \mathcal{V}$ is a rectangular region encompassing the entire object image.

Set of terminal symbols $\mathcal{T}$: A terminal symbol $T \in \mathcal{T}$ is an axis-aligned rectangle image segment, with an additional attribute b denoting whether a region is to be painted or not.

Set of production rules $\mathcal{R}$: We consider binary split rules, which split a non-terminal V along its axes into two constituent non-terminals. A split rule is specified by a horizontal or vertical split axis (h or v), and split location l, as $V \xrightarrow{h:l} V\ V$, or $V \xrightarrow{v:l} V\ V$. An instance of a horizontal split rule is visualized in the top row of Fig. 1, resulting in two non-terminals (V_{left}, V_{right}). Our grammar also includes assignment rules that assign a precedant non-terminal symbol to one of the two terminal symbols, a region b_{p} to be painted, or a region not to be painted, b_{np}. The full set $\mathcal{R}$ is as follows:

$$\mathcal{R} = \{V_0 \rightarrow V, V \xrightarrow{x:l} (V_{\text{left}}, V_{\text{right}}), V \xrightarrow{y:l} (V_{\text{top}}, V_{\text{bottom}}), V \rightarrow b_{\text{p}}, V \rightarrow b_{\text{np}}\}$$

Rule probabilities $\mathcal{P}$: Production rules r have an associated probability p_r of applying rule r on the current non-terminal (i.e., an image segment). In our problem, we seek to learn these probabilities p_r of applying the production rules, along with the associated split location attribute l of the rules.

Recursively applying production rules $\mathcal{R}$ of the grammar $\mathcal{G}$ described above on an object image I (and the resultant segments) decomposes the image into its constituent segments, a process known as shape-parsing. Parsing an object image results in a hierarchical decomposition of the object, or an object parse-tree, as depicted for a toy object image in Fig. 1.

Starting with the entire image (represented by starting symbol V_0), the object parse-tree is constructed by expanding each node in the tree in a top-down and depth-first manner. While expanding a node N of the parse-tree, we sample a rule $r \in \mathcal{R}$ with probability p_r, from the set of rules applicable on N. We add the antecedants of this rule to the parse tree as children nodes of the expanded node. We then continue to derive the tree in a depth-first fashion, moving on to the next unexpanded node in the tree. The labels assigned to these leaf nodes of a fully expanded image parse tree yield a segmentation of the image. We present a sample object image, the corresponding image parse tree derived via shape parsing, and the final segmented image in Fig. 1.

3.3 Shape Parsing as a Markov Decision Process

The shape parsing process may be seen as a sequential decision making process, where a parsing agent finds a sequence of partitions and label assignments that maximizes the paint coverage of the object, while affording us a decomposition of the object. We can formally describe the sequential process of shape parsing as a Markov Decision Process (MDP) $\mathcal{M}$.

Here, the current image segment ρ_t corresponds to node N in the parse tree as the current state $s \in S$ of $\mathcal{M}$. Actions $a \in \mathcal{A}$ correspond to applying production rules $r \in \mathcal{R}$ with a particular split location $l \in [0, 1]$. Upon taking an action a from state s, we "transition" to the next unexpanded node s' in the parse tree. Here, s' is specified by the deterministic transition dynamics $p(s_{t+1}|a_t, s_t)$ enforced by the top-down, depth-first expansion of the tree τ.

The sequence of nodes expanded during the expansion of the image parse-tree corresponds to the sequence of states observed by our agent. This may be incorporated elegantly in the definition of both the one-step reward and the cumulative discounted reward (returns) G of the agent. The one-step reward function encodes the coverage objective of our object parsing problem. For every terminal symbol T, we evaluate the image correlation between the predicted label assignments $\mathbb{P}$ over each of the terminal segments, and the ground truth paint labels of the objects $\mathbb{L}$. For any non-terminal node V in the parse tree, the return $G(V)$ is defined *recursively* in Eq. (1) as the discounted sum of the returns of all child nodes of V. This recursion propagates rewards up the tree in a bottom-up manner, starting from the terminal leaf-nodes T, where the return is the one-step reward.

$$G(N) = \begin{cases} \sum_{(x,y)\in N} \mathbb{L}(x,y)\mathbb{P}(x,y) & \text{if } N \in \mathcal{T} \\ \sum_{c\in \text{Children}(N)} G(c) & \text{if } N \in \mathcal{V} \end{cases} \tag{1}$$

Here, x and y represent pixel locations in the image, Children(N) is the set of children nodes of N in the tree, C indexes these child nodes.

We seek to learn a policy $\pi : s \rightarrow a$ mapping the current state s to one of the possible actions a available in the current state. In our setting, the policy must select (1) which production rule to apply, and (2) a corresponding split location. To predict these facets of an action a directly from visual input of the current image segment, we represent our policy as a deep convolutional network. Our policy network thus takes in as input an image region $\rho_t(x, y, w, h)$ defined by the current non-terminal V that is being expanded.

Algorithm 1 Train Parser via DRAG

Input: $\mathcal{D}, \pi^*, \beta, N_{\text{iterations}}$ ▷ Require a dataset, expert parser, mixing parameter, iterations
Output: π_θ ▷ Output the learned policy
$\theta \leftarrow \mathbf{0}, \mathcal{M} \leftarrow \{\}$ ▷ Initialize Policy Parameters, Initialize Memory
for $i \in [1, 2, ..., N_{\text{iterations}}]$ **do**
  $\pi_i \leftarrow \beta\pi^* + (1-\beta)\pi_\theta$
  for $j \in [1, 2, ..., N_{\text{images}} = |\mathcal{D}|]$ **do**
    $t \sim \mathbb{U}[1, H]$ ▷ Sample a switching index
    $\tau_j = \text{Parse}(\mathcal{D}_j)$ ▷ Parse the image, following π_i till step t, and π^* thereafter.
    $G_t = G(\tau_j^t)$ ▷ Evaluate returns at node τ_j^t via expert's cost to go in Eq. (1).
    $\mathcal{M} \leftarrow \mathcal{M} \cup \{(s_t, r_t, l_t, t, G_t)\}$ ▷ Store the transition at index t in memory
  $\mathcal{B} \sim \mathbb{U}[\mathcal{M}], |\mathcal{B}| = B$ ▷ Sample a minibatch from memory
  $\theta \leftarrow \theta + \alpha\nabla_\theta|_{\mathcal{B}}$ ▷ Update θ via Eq. (18) approximated at $\mathcal{B}$
  $\omega \leftarrow \omega - \alpha\nabla_\omega|_{\mathcal{B}}$ ▷ Update ω by gradient of objective in Eq. (6) approximated at $\mathcal{B}$

The policy then predicts (1) a categorical probability distribution over the valid production rules, $\pi(r|s_t, \theta)$, and (2) a split-location $l = \mu(s_t|\theta)$ within the current image segment in case of applying a split rule. Since the size of the image segment varies at every step, the valid split locations *vary* with the current state. While maintaining a valid probability distribution over such a varying range of values is possible, it is notably challenging, due to the normalizing a distribution across changing scale and limits of the distribution at every step. Instead, we employ a *deterministic* representation of the split location policy, thus $\mu(s_t|\theta)$ is *deterministically* predicted as a scaled logistic function of the deep network features. The two components of our policy network represent the *mixed* deterministic and stochastic nature of the our policy.

3.4 Learning the Shape Parser via Imitation Learning

While Teboul et al. [6] learn a shape-parser via reinforcement learning, it is known that the imitation learning paradigm is preferable to reinforcement learning if an expert agent may be easily obtained [18]. In our case, the connection object parsing and the decision trees constructed by IGM afford us such an expert. We hence consider learning this mixed deterministic-stochastic policy in the imitation learning setting. The stochastic component of the policy $\pi(r|s, \theta)$ may be learned via *off-policy* Monte-Carlo [12] or actor-critic [13] policy gradient algorithms (we point the reader towards [2] for a review of these algorithms). However, existing algorithms for learning the deterministic component of the policy $l = \mu(s|\theta)$ (notably the Deterministic Policy Gradient introduced in [2]), have only been developed in the reinforcement learning setting, not the imitation learning setting.

To learn the deterministic component of the policy $\mu(s|\theta)$, we introduce a deterministic policy gradient update suitable for training deterministic policies in the cost sensitive imitation learning setting. DRAG (DeteRministically AGgrevate) may be viewed as a deterministic actor-critic variant of AggreVaTeD [1], or alternatively, a variant of the Deterministic Policy Gradient [3] suitable for imitation learning. DRAG replaces an approximation to the true gradient in the original Deterministic Policy Gradient [3], with the correct gradient.

We present DRAG by first describing the AggreVaTe / AggreVaTeD setting - an ideal starting point given we have an oracle (IGM) that we may *query* for the optimal action to execute from any state. AggreVaTe [18] and AggreVaTeD [1] approach the problem of learning a policy π_θ by training the policy π_{θ_n} at training iteration n to minimize the cost-to-go Q^* of the oracle π^*, over the aggregated distribution of states $d^t_{\pi_n}$ induced by the current learner's policy, π_{θ_n}. To do so, they roll-out a trajectory with a mixture policy $\pi_n(s) = \beta\pi^*(s) + (1-\beta)\pi_\theta(s)$ till time step $t \in [1, ..., H]$, and subsequently follow the expert π^* then onwards. β simply represents the mixing coefficient, and H is the horizon length of the MDP, and the aggregated distribution of states $d^t_{\pi_n}$ is defined as $\sum_{\{s_i, a_i\}_{i \leq t-1}} \rho_0(s_1)\Pi_{i=1}^{t-1}\pi_n(a_t|s_{t-1})p(s_t|s_{t-1}, a_{t-1})$, and ρ_0 is the initial state distribution. The AggreVaTe [18] objective to be optimized may be represented as:

$$J_n(\theta) = \mathbb{E}_{\,t\sim U(1,...,H), s_t \sim d^t_{\pi_{1:n}}, a_t \sim \pi_n(a|s)} \Big[Q^*_t(s_t, a_t) \Big]. \tag{2}$$

Sun et al. [1] assume a stochastic policy $\pi(a|s, \theta)$ to derive a stochastic policy gradient update to the parameters of the policy θ. However, as mentioned in section 3.3, maintaining a valid probability distribution over split locations in the stochastic policy case is challenging. We hence employ a *deterministic* policy $\mu(s|\theta)$ for split locations — making the stochsatic policy gradient

update derived in [1] unsuitable for learning $\mu(s|\theta)$. The AggreVaTe objective [18] of minimizing the cost-to-go of the oracle in the deterministic policy setting may be expressed as:

$$J_n(\theta) = \mathbb{E}_{\ t\sim U(1,\ldots,H), s_t\sim d^t_{\mu_{1:n}}}\Big[Q^*_t(s_t, \mu(s|\theta))\Big]. \tag{3}$$

Rather than sampling a split location l_t from a stochastic policy, we retrieve the split location deterministically from the policy $l_t = \mu(s|\theta)$. As in [1], we may improve the policy by updating its parameters θ in the direction of improvement of $J_n(\theta)$, given by the gradient of equation 3:

$$\nabla_\theta J_n(\theta) = \mathbb{E}_{\ t\sim U(1,\ldots,H), s_t\sim d^t_{\mu_{1:n}}}\Big[\nabla_\theta Q^*_t(s_t, \mu(s|\theta))\Big]. \tag{4}$$

The Deterministic Policy Gradient [2] allows us to evaluate this gradient, applying the chain rule:

$$\nabla_\theta J_n(\theta) = \mathbb{E}_{\ t\sim U(1,\ldots,H), s_t\sim d^t_{\mu_{1:n}}}\Big[\nabla_a\ Q^*_t(s_t, a)|_{a=\mu(s|\theta)}\ \nabla_\theta \mu(s|\theta)\Big]. \tag{5}$$

The AggreVaTeD framework [1] uses Monte Carlo samples G_t of the oracle's cost to go, directly estimating $Q^*_t(s_t, a)$ by sampling. While this provides an unbiased estimate of Q^*_t, we cannot compute the gradient $\nabla_a Q^*_t(s_t, a)$ using non-differentiable samples of the oracle's cost-to-go. Instead, we construct a *differentiable* approximation of the oracle's cost to go, in the form of a critic network $Q(s, a|\omega)$, parametrized by ω. While the notion of the critic network is similar to that present in DPG [2] and DDPG [3], note that our critic network approximates the cost-of-go of the *oracle*, rather than the *learner's* policy μ, i.e., the critic is trained to optimize:

$$\min_\omega \mathbb{E}_{s_t\sim d^t_{\mu_{1:n}}, a=\mu_n(s_t|\theta)}\Big[\big(Q(s_t, a|\omega) - G_t\big)^2\Big]. \tag{6}$$

Using a critic network $Q_t(s_t, a|\omega)$ to approximate the oracle's cost-to-go allows us to perform an update to the policy $\mu(s|\theta)$ by replacing $Q^*_t(s_t, a)$ in equations 4 and 5 with critic network's estimate, $Q_t(s_t, a|\omega)$, leading to the following deterministic policy gradient update:

$$\nabla_\theta J_n(\theta) = \mathbb{E}_{\ t\sim U(1,\ldots,H), s_t\sim d^t_{\mu_{1:n}}}\Big[\nabla_a\ Q_t(s_t, a|\omega)|_{a=\mu(s|\theta)}\ \nabla_\theta \mu(s|\theta)\Big]. \tag{7}$$

Employing a differentiable critic network to estimate the oracle's cost-to-go thus introduces an *deterministic actor critic variant of AggreVaTeD*, which we refer to as DRAG. DRAG hence serves as a deterministic policy gradient update that we use for training our deterministic split location policy $\mu(s|\theta)$ in the cost sensitive imitation learning setting.

We further note that applying the Deterministic Policy Gradient Theorem [2] typically requires an approximation of the true gradient $\nabla_\theta J_n(\theta)$ from [13], due to the implicit dependence of $Q(s_t, a_t|\omega)$ on the parameters of the policy θ. However, in DRAG, the learned estimates of $Q(s_t, a_t|\omega)$ estimate the cost-to-go of the *oracle* π^*, and not the learner's policy $\pi(a_t|s_t, \theta)$. The true cost-to-go of the oracle $Q^*(s_t, a_t)$ (and any estimate $Q(s_t, a_t|\omega)$ of this cost) are both *independent* of the parameters θ of the learner's policy. DRAG hence removes the dependence of $Q(s_t, a_t|\omega)$ on the learner's policy θ by virtue of following the oracle after time t, hence the gradient update we present in equation 7 is no longer an approximation to the true gradient based on [13].

DRAG allows us to compute a gradient update to deterministic split-location component of our neural parser, while we employ a standard stochastic actor-critic policy gradient update to for the rule policy $\pi(a|s, \theta)$. This leads to a *mixed* stochastic-deterministic policy gradient update in Eq. (18):

$$\begin{aligned}\nabla_\theta J_n(\theta) = \mathbb{E}_{\ t\sim U(1,\ldots,H), s_t\sim d^t_{\pi_{1:n},\mu_{1:n}}, r_t\sim\pi_n(r|s_t,\theta)}\Big[&\nabla_\theta \log \pi(r_t|s_t, \theta)\cdot Q_t(s_t, r_t, \mu(s_t|\theta)|\omega) \\ &+ \nabla_l\ Q_t(s_t, r_t, l|\omega)|_{l=\mu(s_t|\theta)}\cdot\nabla_\theta\mu(s_t|\theta)\Big].\end{aligned} \tag{8}$$

The full derivation of this mixed policy gradient update is provided in the supplementary material. The resultant *mixed* stochastic-deterministic policy gradient update is based on DRAG, and is applicable to any policy with both a deterministic and stochastic component. We utilize this mixed update to train our neural parser, as described in Algorithm 1.

4 Experimental Results and Analysis

We evaluate our idea of learning to parse objects by imitation, and quantify how well our proposed DRAG approach is able to parse *novel* objects relative to baseline imitation learning (IL), reinforcement learning (RL) and hybrid IL+RL approaches, and the IGM oracle. This also provides us insight

Table 1: Parsing Accuracies of Proposed model and various baselines.

Model	Train Accuracy	Test Accuracy
IGM Oracle with GT Access (Depth 7)	98.50%	—
Monte-Carlo Policy Gradient (RL)	53.54%	51.23%
DDPG (RL)	51.94%	48.78%
Behavior Cloning (IL)	75.11%	75.10%
DAgger (IL)	84.01%	84.03%
Stochastic AggreVaTeD (IL+RL)	81.85%	81.07%
Actor-Critic AggreVaTeD (IL+RL)	82.18%	81.31%
Off-Policy Monte-Carlo Policy Gradient (IL+RL)	84.85%	83.65%
Off-Policy Actor-Critic Policy Gradient (IL+RL)	80.91%	80.94%
DRAG (IL+RL) **(Ours)**	**88.05%**	**86.86%**

into the relative benefits of IL, RL, and IL+RL approaches in the context of parsing. The IL and IL+RL baselines are provided access to the IGM Oracle, with a maximum allowed parse tree depth of 7 enforced for computational reasons. The considered baselines are as follows (with details in the supplementary).

RL baselines: We consider two RL baselines where the learner must maximize its own cumulative reward without IGM oracle access: 1) An on-policy stochastic *Monte-Carlo Policy Gradient (MCPG)* as in [12], 2) A *Deterministic Policy Gradient (DPG)* as in [3].

IL baselines: We consider two IL baselines, where the learner imitates the IGM oracle, with no reward function access. 1) *Behavior Cloning (BC)*, where the agent imitates a fixed set of demonstrated parses from the oracle, 2) the interactive IL paradigm *DAgger* [17].

Hybrid IL+RL baselines: We consider four hybrid IL+RL baselines, where the learner has access to both the IGM oracle and the reward function during training. 1) The original stochastic *AggreVaTeD* policy gradient [1], 2) An *Actor-Critic variant of AggreVaTeD (AC-AggreVaTeD)*, 3) A stochastic *Off-Policy Monte-Carlo Policy Gradient (Off-MCPG)*, where the oracle acts as a behavioral policy, and it's actor-critic variant, 4) An *Off-Policy Actor-Critic Policy Gradient (Off-ACPG)*. Note that all stochastic baselines are evaluated selecting the most likely action at test time.

Experimental Setup: To evaluate our proposed DRAG approach against the above baselines, we collect a set of 362 RGB object images of size 256×256 pixels, each annotated with a per-pixel binary label of 1 (to be painted), or 1 (not to be painted), serving as ground truth object labels. We evaluate our models with 3-fold cross validation, training on 3 sets of 300 randomly sampled images, measuring performance on 3 corresponding sets of 62 test images. We measure performance as the pixel accuracy between the predicted assignment of labels of the object image against these ground truth object labels, presented for each of the above baselines and our approach in Table 1. An ideal parse assigns contiguous paint labels to all portions of the object to be painted, ensuring that parts of the object do not go unpainted. We present the parses created for 3 sample images in Fig. 2.

Analysis: We make the following observations based on the results in Table 1 and Figure 2:
1) Reinforcement learning applied to our task is unsuccessful. The recursive nature of the parsing process requires good rules and split locations to be selected *consistently* for a good parse, which is unlikely via random exploration. The MCPG and DPG baselines rely on random exploration of rules and splits, thus only achieving random performance and failing to parse object images at all.

2) Imitation learning applied to our parsing task is successful. The structural similarity between IGM decision-trees and parsing allows the IGM oracle to *guide sampling* towards good rules and split locations rather than the naive exploration of RL. As seen in Table 1, even our most naive IL baseline, Behavior Cloning, significantly outperforms the RL baselines, achieving 75.10% test accuracy. DAgger reconciles with the state distribution mismatch [17], boosting the parser to 84.03% test accuracy. The IL paradigm is thus able to learn reasonable parses of object images, overcoming the issues of learning a parser present in the RL setting.

3) DRAG outperforms state-of-the-art IL and RL baselines on the task of parsing novel objects. Quantitatively, our proposed DRAG approach achieves notably higher train and test accuracies (88.05 % and 86.86% respectively) than other baseline approaches. We note the following:

(a) While some improvement of performance may be attributed to the use of a lower variance return estimate offered by a critic network, comparing the performance achieved by the actor-critic

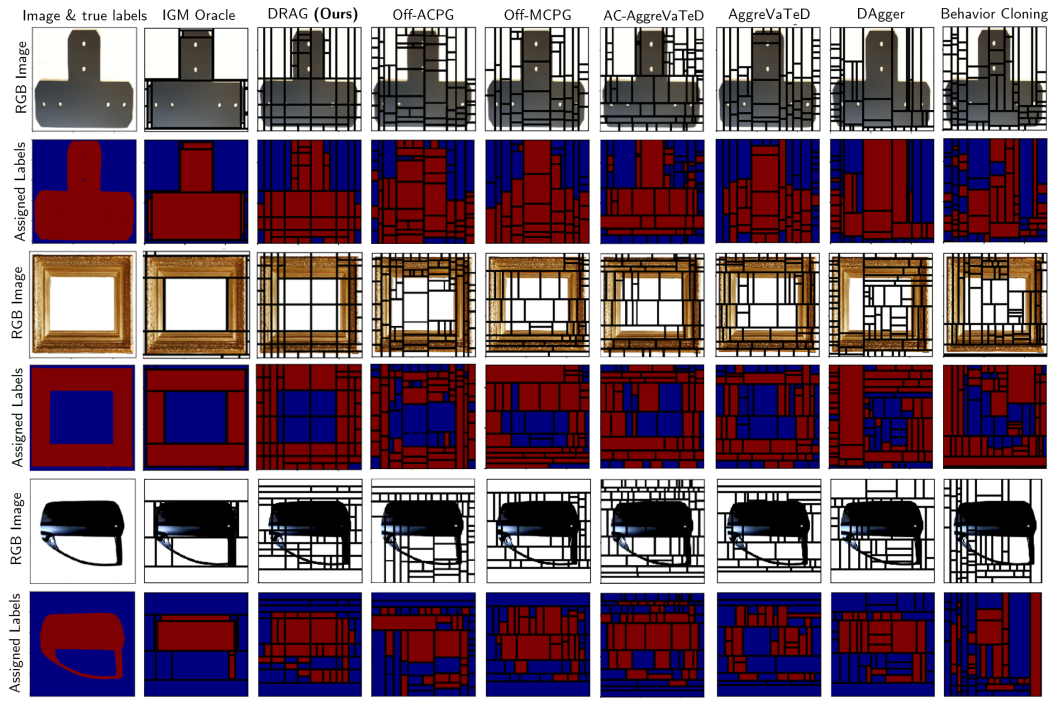

Figure 2: Depiction of constructed sample parses for a metal plate (rows 1 & 2), a window frame (rows 3 & 4), and a car door (rows 5 & 6), from the oracle (column 2), the proposed DRAG (column 3), and the various baselines (columns 4-9). The first column shows the original image (odd rows) with ground truth labels (even rows, red object pixels are to be painted, blue are not to be painted). Each column shows the segmented object image along with the predicted label assignment.

baselines against their actor-only variants shows there is little benefit to using a critic network with a stochastic policy representation. The deterministic policy representation used in DRAG allows for better learning of split locations, contributing towards the notable improvement in performance observed.

(b) The deterministic policy representation of split locations used in DRAG enables it to construct more regular parses of the object images by selecting splits that align with object boundaries, as compared to baseline approaches. This is particularly suitable for axis-aligned images, or images with small aberrations (such as rows 3 & 4 in Fig. 2). While the stochastic hybrid IL+RL baselines are able to capture coarse object structure, they fail to capture splits aligned with prominent image gradients. In contrast, DRAG is able to assign labels that correlate strongly with that of the ground truth (and the IGM oracle).

(c) DRAG better optimizes for the underlying cost compared to other hybrid IL+RL baselines, generalizing past demonstrated expert actions. This is exemplified in the case of irregular images such as the car door in rows 5 & 6 of Fig. 2. Here, the IGM oracle misses out on the correctly labeling the rim of the door due to a restricted parse tree depth. Despite this lack of supervisory actions to imitate, DRAG learns to correctly label this portion of the door, and is capable of performing cost-sensitive imitation in a superior manner compared to other baseline approaches.

5 Conclusion

In this paper, we address the problem of learning to parse objects into hierarchical decompositions via imitation learning. By treating an Information Gain Maximizing algorithm as an expert parsing oracle, our neural parser learns to parse objects by imitating this IGM oracle, observing only raw object images as input. We further introduce a novel deterministic policy gradient update, DRAG , suitable for generic imitation learning tasks with a deterministic policy representation. The proposed DRAG may be seen as both as an deterministic actor-critic variant of AggreVaTeD [1] and a variant of DDPG [3] suitable for imitation learning. Training our neural parser to parse objects using DRAG outperforms existing RL, IL and IL+RL baselines, leading to more accurate and coherent parses. Our experimental results demonstrate the capability of our approach to successfully parse objects, and potentially address more generic spatially decomposable tasks.

Acknowledgments

The authors would like to thank Wen Sun, Anirudh Vemula, and Arjun Sharma for technical discussions, and Marinus Analytics for providing us access to computing resources for our experiments. We would also like to thank the anonymous reviewers for their comments on our paper.

References

[1] W. Sun, A. Venkatraman, G. J. Gordon, B. Boots, and J. A. Bagnell. Deeply AggreVaTeD: Differentiable Imitation Learning for Sequential Prediction. 2017. doi:arXiv:1504.01391. URL http://arxiv.org/abs/1703.01030.

[2] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller. Deterministic Policy Gradient Algorithms. *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 387–395, 2014. ISSN 1938-7228.

[3] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning. 2015. ISSN 1935-8237. doi:10.1561/2200000006. URL http://arxiv.org/abs/1509.02971.

[4] J. R. Quinlan. Induction of decision trees. *MACH. LEARN*, 1:81–106, 1986.

[5] S. L. Salzberg. C4.5: Programs for machine learning by j. ross quinlan. morgan kaufmann publishers, inc., 1993. *Machine Learning*, 16(3):235–240, Sep 1994. ISSN 1573-0565. doi:10.1007/BF00993309. URL https://doi.org/10.1007/BF00993309.

[6] O. Teboul, I. Kokkinos, L. Simon, P. Koutsourakis, and N. Paragios. Shape grammar parsing via reinforcement learning. In *CVPR*, pages 2273–2280. IEEE Computer Society, 2011. ISBN 978-1-4577-0394-2. URL http://dblp.uni-trier.de/db/conf/cvpr/cvpr2011.html#TeboulKSKP11.

[7] O. Teboul, L. Simon, P. Koutsourakis, and N. Paragios. Segmentation of building facades using procedural shape priors. In *CVPR*, pages 3105–3112. IEEE Computer Society, 2010. ISBN 978-1-4244-6984-0. URL http://dblp.uni-trier.de/db/conf/cvpr/cvpr2010.html#TeboulSKP10.

[8] A. Martinović, M. Mathias, J. Weissenberg, and L. Van Gool. A three-layered approach to facade parsing. In *European conference on computer vision*, pages 416–429. Springer, 2012.

[9] H. Zhang, K. Xu, W. Jiang, J. Lin, D. Cohen-Or, and B. Chen. Layered analysis of irregular facades via symmetry maximization. *ACM Trans. Graph.*, 32(4):121–1, 2013.

[10] M. Kozinski, R. Gadde, S. Zagoruyko, G. Obozinski, and R. Marlet. A mrf shape prior for facade parsing with occlusions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2820–2828, 2015.

[11] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, pages 1057–1063, 2000.

[12] R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3):229–256, May 1992. ISSN 1573-0565. doi:10.1007/BF00992696. URL https://doi.org/10.1007/BF00992696.

[13] T. Degris, M. White, and R. S. Sutton. Off-policy actor-critic. *CoRR*, abs/1205.4839, 2012. URL http://arxiv.org/abs/1205.4839.

[14] T. Jie and P. Abbeel. On a connection between importance sampling and the likelihood ratio policy gradient. In *Advances in Neural Information Processing Systems*, pages 1000–1008, 2010.

[15] H. Daumé, J. Langford, and D. Marcu. Search-based structured prediction. *Machine learning*, 75(3):297–325, 2009.

[16] S. Ross, G. Gordon, and D. Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635, 2011.

[17] S. Ross, G. J. Gordon, and J. A. Bagnell. A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning. *Proceedings of AISTATS*, 15:627–635, 2011. ISSN <null>. URL http://arxiv.org/abs/1011.0686.

[18] S. Ross and J. A. Bagnell. Reinforcement and Imitation Learning via Interactive No-Regret Learning. pages 1–14, 2014. URL http://arxiv.org/abs/1406.5979.

[19] W. Sun, J. A. Bagnell, and B. Boots. TRUNCATED HORIZON POLICY SEARCH: DEEP COMBINATION OF REINFORCEMENT AND IMITATION. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=ryUlhzWCZ.

[20] C.-A. Cheng, X. Yan, N. Wagener, and B. Boots. Fast Policy Learning through Imitation and Reinforcement. *ArXiv e-prints*, May 2018.

[21] S. Choudhury, A. Kapoor, G. Ranade, S. Scherer, and D. Dey. Adaptive Information Gathering via Imitation Learning. 2017. URL http://arxiv.org/abs/1705.07834.

[22] M. Bhardwaj, S. Choudhury, and S. Scherer. Learning heuristic search via imitation. *arXiv preprint arXiv:1707.03034*, 2017.

[23] Z. Li, X.-M. Wu, and S.-F. Chang. Segmentation using superpixels: A bipartite graph partitioning approach. In *CVPR*, pages 789–796. IEEE Computer Society, 2012. ISBN 978-1-4673-1226-4. URL http://dblp.uni-trier.de/db/conf/cvpr/cvpr2012.html#LiWC12.

[24] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. *CoRR*, abs/1411.4038, 2014. URL http://arxiv.org/abs/1411.4038.

[25] V. Badrinarayanan, A. Handa, and R. Cipolla. Segnet: A deep convolutional encoder-decoder architecture for robust semantic pixel-wise labelling. *CoRR*, abs/1505.07293, 2015. URL http://arxiv.org/abs/1505.07293.

[26] A. Sharma, O. Tuzel, and D. W. Jacobs. Deep hierarchical parsing for semantic segmentation. In *Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on*, pages 530–538. IEEE, 2015.

[27] A. Garcia-Garcia, S. Orts-Escolano, S. Oprea, V. Villena-Martinez, and J. G. Rodríguez. A review on deep learning techniques applied to semantic segmentation. *CoRR*, abs/1704.06857, 2017. URL http://arxiv.org/abs/1704.06857.

[28] P. Marbach and J. N. Tsitsiklis. Simulation-based optimization of markov reward processes. *IEEE Transactions on Automatic Control*, 46(2):191–209, Feb 2001. ISSN 0018-9286. doi: 10.1109/9.905687.

[29] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL https://www.tensorflow.org/. Software available from tensorflow.org.

Supplementary Material

Mixed Policy Gradient:

In section 3.4 of the main paper, we derived a deterministic policy gradient update to deterministic policy trained in the AggreVaTeD [1] setting.

The specific policy representation that we employ in our neural parser has a stochastic component predicting which rules to apply, $\pi(r|s_t, \theta)$, as well as a deterministic component to predict split locations, $\mu(s_t|\theta)$. Given such a *mixed* stochastic-deterministic policy representation, we derive a corresponding *mixed* stochastic-deterministic policy gradient update that we employ to train our neural parser.

Consider that at any time step t, rules r_t are sampled from $\pi(r|s_t, \theta)$, and split locations l_t are given by $\mu(s_t|\theta)$. Following the AggreVaTeD [1] training paradigm, we seek to train the components of our policy $\pi(r|s_t, \theta)$ and $\mu(s_t|\theta)$, to maximize the cost to go of the expert, $Q^*(s_t, r_t, l_t)$. Formally, we seek to maximize:

$$J_n(\theta) = \mathbb{E}_{\ t\sim U(1,\ldots,H), s_t\sim d^t_{\pi_{1:n},\mu_{1:n}}, r_t\sim\pi_n(r|s_t,\theta)} \Big[Q^*_t(s_t, r_t, \mu(s_t|\theta)) \Big] \tag{9}$$

As described in section 3.4 of the main paper, the actor critic variant of AggreVaTeD uses a learnt *estimate* $Q(s_t, r_t, l_t|\omega)$ of the cost-to-go of the expert $Q^*(s_t, r_t, l_t)$. The objective equation 9 thus becomes:

$$J_n(\theta) = \mathbb{E}_{\ t\sim U(1,\ldots,H), s_t\sim d^t_{\pi_{1:n},\mu_{1:n}}, r_t\sim\pi_n(r|s_t,\theta)} \Big[Q_t(s_t, r_t, \mu(s_t|\theta)|\omega) \Big] \tag{10}$$

The expectation of $r_t \sim \pi(r|s_t, \theta)$ may be represented as follows:

$$J_n(\theta) = \mathbb{E}_{\ t\sim U(1,\ldots,H), s_t\sim d^t_{\pi_{1:n},\mu_{1:n}}} \Big[\sum_{r\in\mathcal{R}} \pi(r_t|s_t, \theta)\ Q_t(s_t, r_t, \mu(s_t|\theta)|\omega) \Big] \tag{11}$$

To compute an update to the policy, we may compute the gradient of this objective $J_n(\theta)$ with respect to the parameters of the policy θ:

$$\nabla_\theta J_n(\theta) = \nabla_\theta\ \mathbb{E}_{\ t\sim U(1,\ldots,H), s_t\sim d^t_{\pi_{1:n},\mu_{1:n}}} \Big[\sum_{r\in\mathcal{R}} \pi(r_t|s_t, \theta)\ Q_t(s_t, r_t, \mu(s_t|\theta)|\omega) \Big] \tag{12}$$

Considering the linearity of expectations, and taking the gradient ∇_θ inside the sum $\sum_{r\in\mathcal{R}}$, this gives us:

$$\nabla_\theta J_n(\theta) = \mathbb{E}_{\ t\sim U(1,\ldots,H), s_t\sim d^t_{\pi_{1:n},\mu_{1:n}}} \Big[\sum_{r\in\mathcal{R}} \nabla_\theta \Big\{ \pi(r_t|s_t, \theta)\ Q_t(s_t, r_t, \mu(s_t|\theta)|\omega) \Big\} \Big] \tag{13}$$

Note that Proposition (1) from Marbach and Tsitsiklis [28] shows that the gradient of a cummulative reward objective, $\nabla_\theta J_n(\theta)$, is independent of the gradient of the state distribution $d^t_{\pi_{1:n},\mu_{1:n}}$. Applying the product rule, we have:

$$\nabla_\theta J_n(\theta) = \mathbb{E}_{\ t\sim U(1,\ldots,H), s_t\sim d^t_{\pi_{1:n},\mu_{1:n}}} \Big[\sum_{r\in\mathcal{R}} \Big\{ \nabla_\theta \pi(r_t|s_t, \theta)\ .\ Q_t(s_t, r_t, \mu(s_t|\theta)|\omega) + \pi(r_t|s_t, \theta)\ .\nabla_\theta Q_t(s_t, r_t, \mu(s_t|\theta)|\omega) \Big\} \Big] \tag{14}$$

The first term in the expectation, $\nabla_\theta \pi(r_t|s_t, \theta)\ .\ Q_t(s_t, r_t, \mu(s_t|\theta)|\omega)$ may be simplified by applying the importance sampling trick:

$$\nabla_\theta J_n(\theta) = \mathbb{E}_{\ t\sim U(1,\ldots,H), s_t\sim d^t_{\pi_{1:n},\mu_{1:n}}} \Big[\sum_{r\in\mathcal{R}} \Big\{ \pi(r_t|s_t, \theta)\ .\frac{\nabla_\theta \pi(r_t|s_t, \theta)}{\pi(r_t|s_t, \theta)}\ .\ Q_t(s_t, r_t, \mu(s_t|\theta)|\omega) + \pi(r_t|s_t, \theta)\ .\nabla_\theta Q_t(s_t, r_t, \mu(s_t|\theta)|\omega) \Big\} \Big] \tag{15}$$

This becomes the gradient of the *log-probability* of the policy $\pi(r|s_t, \theta)$:

$$\nabla_\theta J_n(\theta) = \mathbb{E}_{\ t\sim U(1,\ldots,H), s_t\sim d^t_{\pi_{1:n},\mu_{1:n}}} \Big[\sum_{r\in\mathcal{R}} \Big\{ \pi(r_t|s_t, \theta)\ .\nabla_\theta \log \pi(r_t|s_t, \theta)\ .\ Q_t(s_t, r_t, \mu(s_t|\theta)|\omega) + \pi(r_t|s_t, \theta)\ .\nabla_\theta Q_t(s_t, r_t, \mu(s_t|\theta)|\omega) \Big\} \Big] \tag{16}$$

The second term in the expectation, $\pi(r_t|s_t,\theta)\ .\nabla_\theta Q_t(s_t,r_t,\mu(s_t|\theta)|\omega)$, may be computed using the Deterministic Policy Gradient Theorem [2] (i.e. essentially applying the chain rule):

$$\nabla_\theta J_n(\theta) = \mathbb{E}_{\ t\sim U(1,\ldots,H), s_t\sim d^t_{\pi_{1:n},\mu_{1:n}}}\Big[\sum_{r\in\mathcal{R}}\Big\{\pi(r_t|s_t,\theta)\ .\nabla_\theta \log\pi(r_t|s_t,\theta)\ .\ Q_t(s_t,r_t,\mu(s_t|\theta)|\omega) + \pi(r_t|s_t,\theta)\ .\nabla_l\ Q_t(s_t,r_t,\mu(s_t|\theta)|\omega)|_{l=\mu(s_t|\theta)}\ .\nabla_\theta\mu(s_t|\theta)\Big\}\Big] \quad (17)$$

We may now convert the expression $\sum_{r\in\mathcal{R}}\pi(r|s_t,\theta)$ back to an expectation of $r_t\ \sim\ \pi(r|s_t,\theta)$, leading to the following policy gradient update to our neural parser:

$$\nabla_\theta J_n(\theta) = \mathbb{E}_{\ t\sim U(1,\ldots,H), s_t\sim d^t_{\pi_{1:n},\mu_{1:n}}, r_t\sim\pi_n(r|s_t,\theta)}\Big[\nabla_\theta \log\pi(r_t|s_t,\theta)\ .\ Q_t(s_t,r_t,\mu(s_t|\theta)|\omega) + \nabla_l\ Q_t(s_t,r_t,\mu(s_t|\theta)|\omega)|_{l=\mu(s_t|\theta)}\ .\nabla_\theta\mu(s_t|\theta)\Big] \quad (18)$$

Note that applying the Deterministic Policy Gradient Theorem typically requires an approximation from [13], due to the implicit dependence of $Q(s_t,a_t|\omega)$ on the parameters of the policy θ. However, in the deterministic variant of AggreVaTeD, the learnt estimates of $Q(s_t,a_t|\omega)$ estimate the cost-to-go of the *expert policy* π^*, and not the learner's policy $\pi(a_t|s_t,\theta)$.

The true cost-to-go of the expert $Q^*(s_t,a_t)$ (and any estimate $Q(s_t,a_t|\omega)$ of this cost) are both *independent* of the parameters of the policy θ. The deterministic actor-critic variant of AggreVaTeD removes the dependence of $Q(s_t,a_t|\omega)$ on θ, thus removes the necessity for the approximation from Degris et al. [13].

Use of a Memory Replay:

Uniformly sampling from a replay memory (as in Algorithm 1) corresponds to sampling states from the distribution of states encountered during training, i.e. a sample approximation of the aggregated state distribution $d^t_{\pi_{1:n},\mu_{1:n}}$, justifying the use of a replay memory in the aggregated state distribution case. Indeed, using states visited from training iterations 1 to n is closer to the original AggreVaTe objective [18] than AggreVaTeD [1], which uses only states from iteration n.

Details of Baseline Algorithms:

We describe the exact policy representation used in each of the baseline algorithms mentioned in the main paper below. For ease of comparison, we also provide a table of the training setting used in the various baseline approaches and our model in Table 2. Each model uses a convolutional neural network with 7 convolutional layers and 2 dense layers as a base model. The baseline approaches then represent their respective policies as follows:

Pure RL baselines: In the pure RL setting, the learner is provided with evaluations of the quality of the parses it constructs via the reward function, and does not have access to the IGM oracle agent in any form. We consider two RL baselines, where the objective is to simply maximize the cumulative reward achieved by the learner:

- *Monte-Carlo Policy Gradient (MCPG):* We consider an on-policy stochastic Monte-Carlo Policy Gradient approach, similar to REINFORCE. The policy maintains a categorical distribution over valid rules, predicted as a softmax of deep network features over the valid rules applicable at the current image segment, and a logit-normal distribution over valid split locations within the boundaries of the current image segment.
- *Deterministic Policy Gradient (DPG):* We then consider a DDPG [3] style approach, where the policy deterministically predicts split locations as scaled logistic function of the object image features. The rules here are still predicted stochastically as a softmax output of the deep policy network.

Pure IL baselines: In contrast with the RL setting, the learner in the pure IL setting has access to the actions taken by the IGM oracle, and not the reward function. Here, the learner simply tries to copy the actions executed by the expert; this corresponds to maximizing the likelihood of the rules selected by the expert under the learner's policy, and *regressing* to the split locations selected by the expert. We consider two IL baselines:

Table 2: Training Setting Ablation of Proposed model and various baselines.

Model	Training Setting	Policy Representation	Actor / Actor-Critic
MCPG	RL	Stochastic	Actor
DDPG	RL	Deterministic	Actor-Critic
Behavior Cloning	IL	Deterministic	Actor
DAgger	IL	Deterministic	Actor
AggreVaTeD	IL+RL	Stochastic	Actor
AC AggreVaTeD	IL+RL	Stochastic	Actor-Critic
Off-MCPG	IL+RL	Stochastic	Actor
Off-ACPG	IL+RL	Stochastic	Actor-Critic
DRAG (**Ours**)	IL+RL	Deterministic	Actor-Critic

- *Behavior Cloning:* Here, the agent minimizes the categorical cross entropy between the rules selected by the IGM oracle, and the softmax probability distribution over rules predicted by the policy network. This is equivalent to maximizing the log-likelihood of the rules selected by the IGM oracle. The split locations are predicted as a scaled logistic function of the deep network features. A L2 norm loss between the predicted and IGM oracle split is used to train the agent's split location policy.
- *DAgger:* Following the interactive learning paradigm DAgger [17], objects are parsed according to a *mixture* of the expert and the learner's current policy. The policy representation is identical to that used in the Behavior Cloning case.

Hybrid IL+RL baselines: Of particular interest to us is the hybrid IL+RL case, where the learner has access to both the actions executed by the expert, as well as samples of the reward function for the parses it constructs.

- *AggreVaTeD:* We consider the original stochastic policy gradient training paradigm of AggreVaTeD [1]. As in the Off-MCPG case, the rules are predicted via a categorical distribution from the deep policy network features, and the split locations are predicted as a logit-normal distribution over valid splits.
- *Actor-Critic AggreVaTeD:* We consider an Actor-Critic variant of AggreVaTeD [1], where the Monte-Carlo estimate of the oracle's return is replaced by the critic's estimate of this return. The critic is trained using Eq. (6) as in DRAG .
- *Off-Policy Monte-Carlo Policy Gradient (Off-MCPG):* Treating the IGM oracle as a behavioral policy, we maximize the learner's returns via an off-policy Monte-Carlo policy gradient. As in the case of the vanilla MCPG, the policy representation is a categorical distribution over the valid rules, and the splits with a logit-normal distribution over valid splits.
- *Off-Policy Actor-Critic Policy Gradient (Off-ACPG):* We finally consider an actor-critic variant of Off-MCPG, using a critic network to estimate the oracle's return. As in the case of AC-AggreVaTeD, we use Eq. (6) to train the critic.

Training Details and Choice of Hyperparameters:

We note details regarding our training setup, as well as values of hyperparameters used during training the various baseline approaches and our model:

- *Convergence:* We ensure each approach is trained till convergence by evaluating the model on the test set after the validation accuracy saturated (is no longer improving significantly).
- *3-fold Cross-Validation:* Our image based problem allows us to maintain distinct training and testing image sets, we follow 3-fold cross-validation, maintaining 3 different train-test sets, and reporting average train and test accuracies across these sets.
- *Learning Rate:* For all models, we use the Adam Optimizer available in TensorFlow [29]. All our models are trained with an learning rate of 10^{-4}.
- *Mixing Coefficient β:* For the DAgger baseline, as well as the hybrid IL+RL baselines and DRAG, we utilize an initial mixing coefficent of 1 (i.e. start off using the expert policy alone), annealed to a final value of 0.5 (a 50% chance to use either the learner's policy or the expert policy), over 100 training epochs.
- *Gradient Clipping:* For all models, we apply a gradient clipping to a maximum value of 10.