Removing the i's from i.i.d : Testing generalization on hard datasets

Swaminathan Gurumurthy

CMU-RI-TR-19-81

December 2019

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:
Katia Sycara, Chair
David Held
Adithya Murli
Wenhao Luo

Submitted in partial fulfillment of the requirements for the degree of Master of Science in Robotics.

This work has been funded by AFOSR award FA9550-15-1-0442 and AFOSR/AFRL award FA9550-18-1-0251

For my family

Abstract

The last few years have seen widespread success of over-parameterized deep learning models on various applications with massive datasets. However, these models are often critiqued for assuming access to perfect data, that is, a large amount of clean, i.i.d sampled data. In real-world scenarios, neither of these assumptions is entirely true. We consider four arbitrary domains as examples of some of these scenarios, namely, point cloud completion (with distribution shift), visual dialog (dataset size/bias issues), meta-rl for control (noisy, high variance and sparse training signal) and poaching prediction task (unstructured dataset with skew, noise and distribution shift). Using these datasets, we show that data and priors are meant to complement each other in machine learning models and it's important to think of them jointly on a task to task basis for better generalization.

Acknowledgments

I would like to thank my current advisor, Prof. Katia Sycara for her unwavering support over these one and a half years. She has been a solid mentor to me, and her enthusiasm for good research, served as an inspiration to me. Her ability to place trust and responsibility in the hands of her students, allowing them to follow through on their own research ideas while providing support, is rare and very valuable.

I would also like to thank my previous advisors, Prof. Fei Fang and Prof. Martial Herbert for mentoring me through the first year of my masters and helping me develop a solid foundation and teaching me how to steer my own projects.

I would also like to thank Akshat Agarwal, Shubham Agrawal, Sumit Kumar, Anirudh Goyal and Bhairav Mehta who collaborated with me on various projects throughout my Masters. Working with them reminded me that research can be fun (which admittedly I often forget) and can't imagine doing any of the projects I did without their support, help and criticisms. Apart from that, I would like to thank my current set of collaborators Akshay Sharma, Siddharth Agarwal, Rohit Jena, Vidhi Jain, Ankur Deka, Tejus Gupta, Siddharth Ghiya, Tun Jian Tan, Abhijeet Ghawade and Dana Hughes who have aptly stepped onto the shoes of my previous friends/collaborators and helped keep research fun for me. I would also like to thank some other people who I have had the good fortune of working with on various other projects, including Prof. Michael Lewis, Prof. Yoshua Bengio, Prof. Changliu Liu and Maruan Al-Shediavat. I would like to thank Adithya Murli, Wenhao Luo and David Held for their help and support, and other members of my lab for stress-busting conversations.

Finally, I am deeply grateful to my family and friends for always believing in me, even when I did not myself.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Over the last few years, we have seen the rise of end-to-end deep learning models for various problems. These models are making major advances in solving problems that have resisted the best attempts of the artificial intelligence community for many years. It has turned out to be very good at discovering intricate structures in high-dimensional data and is therefore applicable to many domains of science, business and government. In addition to beating records in image recognition [38],[16],[80],[79]], natural language understanding[51] and speech recognition, [51],[29],[64] it has beaten other machine-learning techniques at predicting the activity of potential drug molecules[49], analysing particle accelerator data [9], [2], reconstructing brain circuits[28], and predicting the effects of mutations in non-coding DNA on gene expression and disease, [42],[86].

This success has driven a lot of excitement in the field. Drawing from the success, many have started advocating for throwing in more data and reducing the priors built into the system. This thought process emanates from the observation that conventional machine-learning techniques required careful engineering and considerable domain expertise to design and add prior knowledge at various parts of the inference and training. However, oftentimes, the prior knowledge was imperfect or didn't span the entire distribution. As a result, the machine learning models which used these priors also struggled to perform well in parts of the distribution where the prior was inaccurate. Citing these issues with conventional machine learning models, more of the ML community is advocating for more data and fewer priors.

However, these models to ignore the fact that in a lot of scenarios it's not possible to get ideal datasets. Just as the conventional ML techniques assumed access to ideal priors, the deep learning community of today seems to assume ideal datasets (large amounts of i.i.d, clean, unbiased data). And in most cases, we observe that the world isn't ideal/perfect. Most problems we deal with aren't going to have i.i.d sampled, clean, unbiased datasets. Neither are we going to have exact knowledge of the priors. Hence, there has to be a middle-ground where we use as much information as we can from either sources. In fact, throughout this thesis we'll go through various problems (point cloud completion, goal directed visual dialog, poaching threat prediction and exploration in meta-reinforcement learning) and show that a structured pipeline combining priors and existing/new data helps boost generalization and task performance.

We will start by considering the point cloud completion problem. Here we will consider the distribution mismatch/shift problem between training and the testing distributions. We will show

that standard deep learning methods need to be complemented with this prior knowledge about the type of distribution shift in order to perform well at this task.

Second, we will consider the visual dialog task as an example of dataset size/bias problem. We will show that models trained using standard supervised learning on the dataset would not produce diverse enough samples on the dataset due to the lack of diversity in the dataset. We then show that, taking inspiration from humans, as we let multiple copies of the agents interact, the agents end up generating more diverse and coherent dialog.

Third, we will consider an unstructured dataset for poaching threat prediction. This dataset is skewed, biased, noisy and faces distribution shifts. To further complicate matters, the unstructured nature of the dataset also makes it hard for us to come up with any sophisticated priors. Thus, we show that in such desperate situations, we have to resort to collecting more data from domain experts but can instead use some priors to collect targeted information from the experts as in active learning.

Finally, we will consider a set of meta-reinforcement learning tasks and show that adding priors in these settings is critical as the training is otherwise very sample inefficient and unstable. This is especially true in rl/meta-rl settings since the high variance and the lack of proper signal makes learning in these scenarios already very challenging. Hence, the absence v/s presence of a prior can essentially dictate the feasibility of the solution.

Through these various examples, we wish to show that data and priors are meant to complement each other in any machine learning model.

Chapter 2

Point Cloud Completion

2.1 Problem Statement

In this chapter, we would be discussing the problem of point cloud completion. Over the last few years, the increasing availability of 3D sensors like LIDAR, Radar etc., has put a lot of focus back on processing point clouds accurately. One such task of importance is point cloud completion, where, a partial point cloud is given as input and the full point cloud has to be reconstructed. This task is tricky, because, in the real world we do not have access to the ground truth complete point cloud. However, we do have access to simulation models of the objects which could be used to obtain the full point cloud of objects. However, it's hard to get a mapping between some incomplete point cloud from the real world with its complete point cloud. Hence, training with the complete point clouds from the simulation models by adding some additional simulated incompletions seems like the most feasible option. However, this introduces several challenges that aren't typically considered hard for data driven models:

2.1.1 Challenges

As described above, in this task we create a training set using the simulation models by sampling points from the simulation models with artificially induced artifacts. The test set however could contain other types of incompletions or could even come from the real world.

Distributional Shift

In typical machine learning problems, the distribution of examples in the training and test dataset are assumed fixed. This is a strong assumption to take in many real world scenarios. The problem we are tackling here is one such example. In fact the primary problem in this task arises from the fact that the training time incompletions could be completely different from the test time incompletions as described above.

2.1.2 Assumptions

In order to tackle the increased complexity of the task at hand, we need to make additional assumptions and add prior knowledge in order to generalize well at the task. The assumptions can infact be derived from the task definition itself.

- The output distribution, i.e, the distribution of complete point clouds does not change. This is an obvious, albeit important assumption to consider as it places a strong constraint on the class of functions possible.
- The partial input given in the incomplete point cloud would remain consistent even in the output point cloud. This could be used to place a constraint on the output space in order to validate the output of the inference procedure.

2.2 Previous Literature

Neural network based models for point cloud processing are a recent phenomenon. Qi [60, 61]first introduced a deep learning network, PointNet, for point cloud classification and segmentation. The network handles the arbitrary input size of point clouds by using an element-wise symmetric operation, such as max pool, to encode any input into a fixed size feature vector.

Achlioptas [1] proposed coupling a PointNet-style encoder with a decoder of fully connected layers, along with loss metric like Earth Mover's distance (EMD) to learn representations of point clouds. They further showed that Gaussian Mixture Models or Generative Adversarial Networks could be trained to directly generate the latent representations, which can be used for point cloud generation. The authors of [1] also show that their autoencoder architecture may also be trained for completing point clouds. More robust formulations of the autoencoder architecture have been proposed in [69, 88] although these works don't address shape completion. Moreover, these improvements could be integrated into our framework as well.

Yu [91] adapt [61] for the task of upsampling point clouds. As noted by the authors in the paper, their approach is not suited for point clouds with large gaps or missing regions.

Although there hasn't been much work on point cloud completion using neural nets, some papers have tried shape completion on voxel representations due to the ease of generalizing convolution operations to 3D using 3DCNNs [10, 10, 27, 77, 81].

However, all these point cloud based methods mentioned above are based on architectural changes that improve the processing for the unique data structure that point clouds represent. Moreover, specifically for point cloud completion, most of the methods mentioned above (including voxel based approaches) adopt a straight-forward supervised learning pipeline and don't account for any distributional shifts or out-of-distribution samples. Hence, we look back at some of the earlier point cloud literature to understand the priors used for the task.

More traditional geometric methods for point cloud completion such as [34, 54, 74], used task specific priors to accomplish the task. However, since they didn't take into account data driven priors, they could only fill in small holes in surfaces where the geometric priors worked. A lot of classical approaches also relied on exemplar-based completion, where a CAD database was used to fetch similar models to reconstruct the object, which may then be deformed to match the partial input [45, 46, 70]. This was again using the idea that the output distribution/distribution

of complete point clouds remains consistent. As discussed earlier this is one of the priors that we would be exploiting in our models.

2.3 Methods

2.3.1 Generative models for point clouds

We build upon a recent model for point cloud generation proposed by Achlioptas [1], which extends [60] to learn an autoencoder for point clouds. The encoder consists of multiple layers of 1D convolutions followed by a symmetric pooling layer (max pool in our case) resulting in a single global feature vector (GFV) for the entire point cloud. The decoder consists of a set of stacked fully connected layers. The last layer of the decoder outputs an $N \times 3$ dimensional vector which corresponds to the N points of the point cloud. The Earth Mover's distance (EMD), which is a permutation invariant metric, is used as the loss function. The EMD between two point clouds S_1, S_2 is given by:

$$d_{EMD}(S_1, S_2) = \min_{\phi: S_1 \to S_2} \sum_{x \in S_1} ||x - \phi(x)||_2 \tag{2.1}$$

where $\phi : S_1 \to S_2$ is a bijection. The optimal bijection is unique and invariant under infinitesimal movement of the points. The autoencoder is trained on the ground truth complete point clouds in the training set. The trained encoder (E) is then used to extract the global feature vector encoding for each point cloud in the training set. As in [1], we then train a GAN on the extracted global feature vectors. New feature vectors generated from the generator (G) can be passed through the decoder (H) to generate point clouds. The GAN has the advantage of being a differentiable network - it is possible to take gradients through it from the output to the input distribution space. This is key for our latent-space optimization algorithm.

Generative Adversarial Network (GAN). GANs are a popular category of generative models which have been recently shown to produce state of the art results in image generation. GANs learn a mapping from an easy to sample distribution (say, a unit normal distribution) to the data generating distribution using a function approximator like a neural network (generator). The generator(G) is trained in a game theoretic set up, where the objective of the generator is to generate samples that look indistinguishable (to another network, called the discriminator) from the data. The discriminator (D) is trained to distinguish between the real data and the samples generated by G. We introduce a third network, the Initializing Encoder (IE), that learns a mapping from the output of the generator to the latent space z. But the traditional GAN training scheme is known to be unstable. Recent advances in GANs [4, 23] have tried to address this issue by modifying the loss or the training procedure itself. We use the loss modification proposed in [4] for more stable training. We train the GAN on the set of global feature vectors(GFVs) produced by the encoder. We use fully connected layers for the generator, initializing encoder and the discriminator. The training procedure for the AE and GAN is given in Algorithm 1. The losses

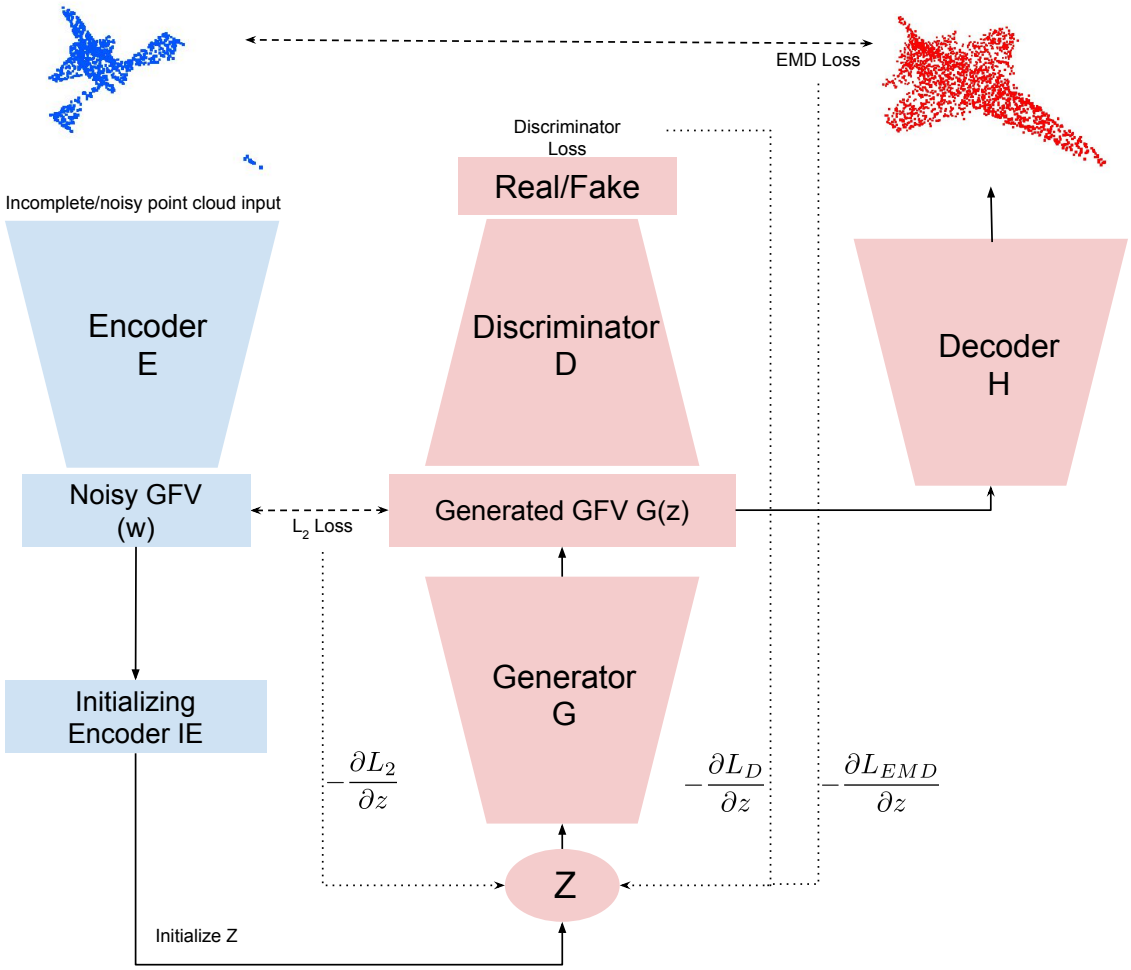


Figure 2.1: The proposed Point cloud completion algorithm. Loss terms of the LDO for an incomplete input point cloud are visualized with dotted lines. Blocks in blue are only used once for initialization. The losses are used to find the correct point in the latent space of the generator. No network weights are changed.

optimized for training the GAN alongside the IE have been described below :

$$J^{(D)} =_{z\sim p_z} D(G(z)) -_{x\sim p_{data}} D(x) \tag{2.2}$$

$$\begin{aligned} J^{(G)} =_{z\sim p_z} & [||IE(G(z)) - z||_2 - D(G(z))] \\ & +_{x\sim p_{data}} ||G(IE(s)) - x||_2 \end{aligned} \tag{2.3}$$

$$\begin{aligned} J^{(IE)} =_{z\sim p_z} & ||IE(G(z)) - z||_2 \\ & +_{x\sim p_{data}} ||G(IE(s)) - x||_2 \end{aligned} \tag{2.4}$$

where, p_z is the unit normal distribution centered at the origin, p_{data} is the distribution of GFVs, z and x are samples from these distributions. Note that the IE and generator are jointly optimized and in alternation with the discriminator.

2.3.2 Point Cloud Completion using LDO

Consider an incomplete point cloud at test time, such as one generated via SfM. The point cloud may also be noisy and have uneven density. If this point cloud is passed through the encoder E, a "noisy" GFV is obtained, i.e. one that doesn't lie on the manifold of representations learnt by the autoencoder. We model the task of completing the point cloud as obtaining a clean GFV corresponding to the noisy one, through an optimization procedure. The cleaned GFV can then be passed through the decoder (H) to obtain a completed point cloud.

Thus the task is reduced to projecting the noisy GFV onto the manifold of clean GFVs. This is not trivial, since we don't have an analytical expression to represent the clean GFV manifold. Thus we use a GAN to represent the clean GFV manifold. As described in the previous section the GAN is trained on clean GFVs, extracted from the training set of complete point clouds. Projecting a noisy data point onto the manifold of clean GFVs can be reduced to finding the closest GFV to the noisy GFV, that is also classified as real by the discriminator. However, directly optimizing over the space of GFVs would result in adversarial examples. Thus we choose to perform the optimization procedure in the latent space of the generator, represented by the latent vector z. First, we produce an initialization for z by passing the noisy GFV through the Initializing Encoder, $z_{init} = IE(GFV)$. Note that the initialization is a very important step. Starting the z from a randomly initialized point makes the optimization much more difficult and requires imposing a lot of additional constraints. From this initial value, z is optimized so as to produce a clean GFV through the generator, $G(z)$. Specifically, the objective of our Latent Denoising Optimization (LDO) algorithm can be decomposed into three parts:

Discriminator Loss: This term ensures that the generated GFV is from the data manifold. We optimize to maximize the score given by the discriminator to the generated GFV:

$$L_D(z) = -D(G(z)) \tag{2.5}$$

Latent Least Squares Loss: This term ensures that the generated GFV, $G(z)$ is close to the noisy GFV, w_i during the optimization and thus remains semantically similar to the input point cloud. The noisy GFV was obtained using the encoder E, $w_i = E(S_i)$. We simply minimize the L2 distance between the generated GFV and the noisy GFV:

$$L_2(z; w) = ||G(z) - w_i||_2^2 \tag{2.6}$$

Decoder EMD Loss: This term ensures that the generated GFV maps to a point cloud which is close to the input point cloud where it exists. Here, we minimize the Earth Mover's distance

between the input point cloud and the point cloud decoded from the generated GFV:

$$L_{EMD}(z; S_i) = d_{EMD}(S_i, H(G(z))) \tag{2.7}$$

Thus our final loss becomes a weighted combination of these losses:

$$Loss(z) = L_{EMD}(z; S_i) + \lambda L_D(z) + \beta L_2(z; w_i) \tag{2.8}$$

We perform this optimization using the ADAM optimizer. Note that we use an exponential decay to reduce the value of λ and β, starting with an initial value of 0.001 each. This helps us ensure that in the initial stages of the optimization, the emphasis on obtaining a semantically consistent and real looking point cloud and in the latter stages, the emphasis is on reconstructing fine details of the point cloud. The optimization is stopped as soon as the loss $L_D(z)$ starts increasing. This ensures that the optimization does not lead to 'unreal' looking point clouds in order to get the details right. It is important to note here that we only minimize the loss with respect to z (which is the input to the Generator). We do not update the generator or discriminator parameters when performing this optimization. At the end of the optimization, we obtain the optimal latent vector, z^*. We simply pass this through the generator to get the clean GFV, $G(z)$, which is passed through the decoder to obtain the completed point cloud, $H(G(z))$. The entire Latent Denoising Optimization (LDO) algorithm has been given in Algorithm 2 and visualized in fig 2.1. Note that the hyperparameter values stay the same for all our experiments. Thus no experiment specific tuning is required.

Algorithm 1 Training a generative model for use in LDO algorithm

A training set of clean, complete point clouds S. Train an autoencoder, with encoder E and decoder H, on the training set S, using EMD as the loss metric. For our experiments we use the implementation in [1], but our algorithm is completely transferable to other PointNet-style architectures such those presented in [61, 88]. Using the trained Encoder, extract the global feature vectors (GFVs) for all examples in the training set. Train a GAN jointly with the IE to fit on the distribution of extracted GFVs from training set using Eq 2.2-2.4.

Algorithm 2 Point cloud completion using LDO algorithm

Extract the GFV w_i for the partial cloud S_i by passing it through Encoder E. $w_i = E(S_i)$Initialize the latent vector z_i using the initializing encoder IE, $z_i = IE(w_i)$ Set $prevLD = L_D(z_i), \lambda = 0.001, \beta = 0.001$ k=1,2..N Compute $\nabla_z Loss(z_i) = \nabla_z L_{EMD}(z_i; S_i) + \lambda \nabla_z L_D(z_i) + \beta \nabla_z L_2(z_i; w_i)$Update z_i using ADAM Update $\lambda = \lambda * 0.9998$ Update $\beta = \beta * 0.9998$ $L_D(z_i) > prevLD$ Exit Loop Update $prevLD = L_D(z_i)$ Pass the cleaned GFV $G(z_i)$ through the previously trained autoencoder's decoder D to obtain the semantically completed point cloud $H(G(z_i))$

2.4 Evaluation

In this section, we demonstrate the salient features of our LDO algorithm by evaluating its quantitative and qualitative performance in multiple scenarios. We compare our model to 2 baseline

models of point cloud completion, and show the improvement in the reconstruction by applying LDO in conjunction with these baseline models. We show quantitative and qualitative results of the improvements gained by LDO on the tasks of point cloud completion and upsampling. Finally, we show experiments on a real world scenario (SfM) where we demonstrate that our approach particularly shines in generalizing to real world data, while only having been trained on synthetic data. To summarize, we'll be comparing the following models:

1. Autoencoder (**AE**): An autoencoder trained only with complete point clouds. See Appendix for full implementation details. This baseline is used to demonstrate cases where no prior information is available about the deformities in the true data.
2. Denoising Autoencoder (**DAE**): Another intuitive baseline is an autoencoder with the same architecture as AE, trained with an augmented dataset of incomplete point clouds. While working on our experiments, we became aware that the authors of [1] incidentally updated their work to suggest a similar DAE based completion method. Our initial experiments found that DAEs have a tendency to overfit, and don't generalize well to different amounts and kinds (small holes, large missing region, low-resolution, SfM point clouds) of incompletion (see appendix). Hence, we train different DAEs with different amounts of incompletion and test them on the same amounts of incompletion to get a competitive baseline for comparison. Although one would never have this luxury in the real world, this baseline is used to demonstrate the superiority of our method even when the exact kind and amount of deformity is known beforehand.
3. Latent Denoising Optimization with AE (**AE+LDO**): Our algorithm applied using AE and a GAN learnt on GFVs of the clean training data generated by the AE. We show that *despite never having been trained on noisy/incomplete point clouds,* AE+LDO is very effective at point cloud completion and achieves a huge boost over just the AE's performance.
4. Latent Denoising Optimization with DAE (**DAE+LDO**): To show the transferability of LDO, we also apply it on DAE, with a GAN trained on GFVs produced by the DAE on clean training data. We show that LDO is able to capitalize on the more robust representations learnt by DAE to improve performance even further than AE+LDO.

Dataset. We use ShapeNetCore, a subset of the full ShapeNet[7] dataset with manually verified category and alignment annotations. It covers 55 common object categories with about 51,300 unique 3D models. For the purposes of our experiments, we use 4 classes with the most available data from the dataset, namely: airplane, car, chair and table. For each class, we split the models into 85/5/10 train-validation-test sets for our experiments and results. We use the models without any pose or scale augmentations. We uniformly sample the point clouds (2048 points each) from these models, which serve as the ground truth for our training. In section 4.3, we experiment on a real-world data case we take sequences of images of faces and pass them through an SfM pipeline to get noisy point clouds of faces. We use the CMU Multi-PIE [25] dataset as the source of these face images and the Basel face model [59] to obtain a synthetic dataset of faces. More details on this are provided in section 4.3

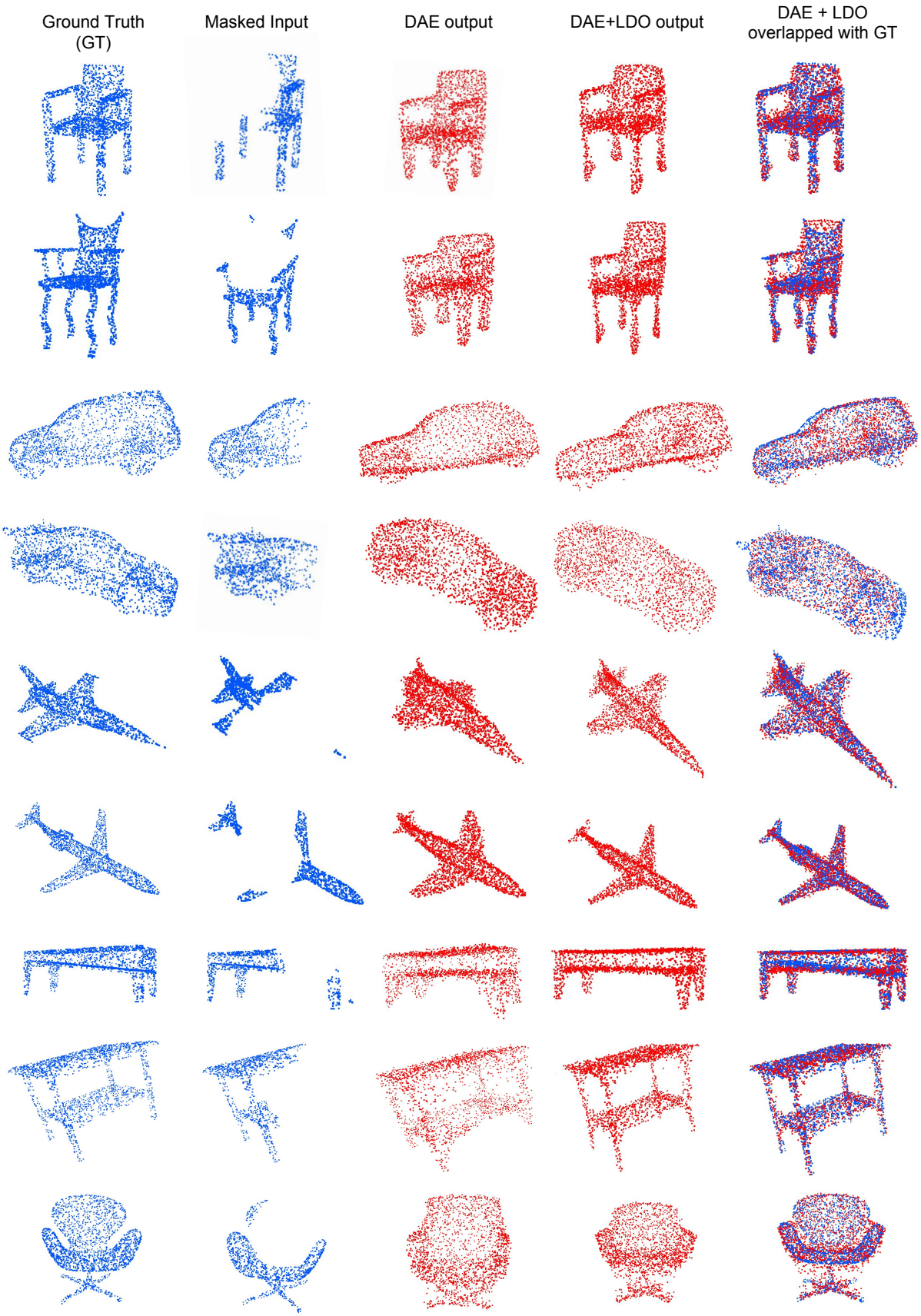


Figure 2.2: Visualizations of shape completions of LDO on a test set containing all 4 classes. The outputs under "DAE" are from single denoising autoencoder trained on objects of all 4 classes, with 50% missing data. Outputs of DAE+LDO are of the single DAE and a single GAN trained on global feature vectors of the DAE. DAE+LDO leads to much sharper outputs with more details of the partial shape captured. Last column shows the ground truth and our results overlaid for ease of comparison.

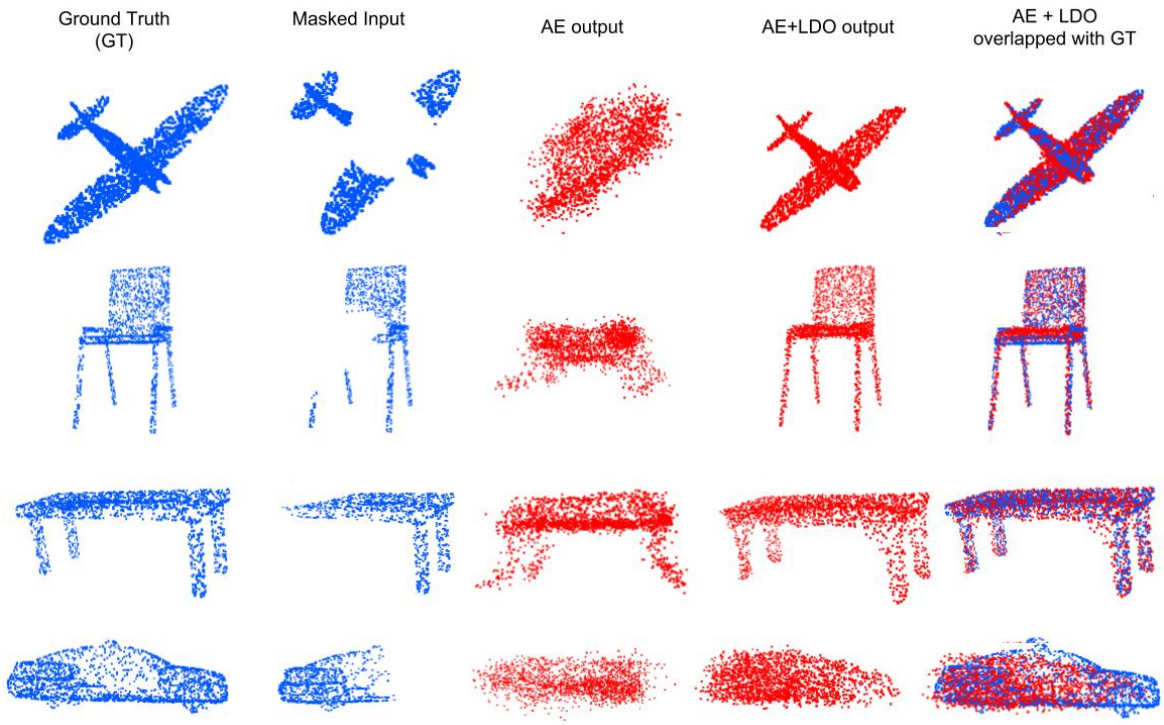


Figure 2.3: Visualizations of shape completion results of AE and AE+LDO on a test set containing all 4 classes. Random 50% chunks of the inputs are masked at test time. The outputs under "AE" are from a single autoencoder trained on objects of all 4 classes. The rightmost column shows the AE+LDO outputs overlapped with the ground truth for direct comparison. A massive improvement is seen in reconstruction quality with our method.

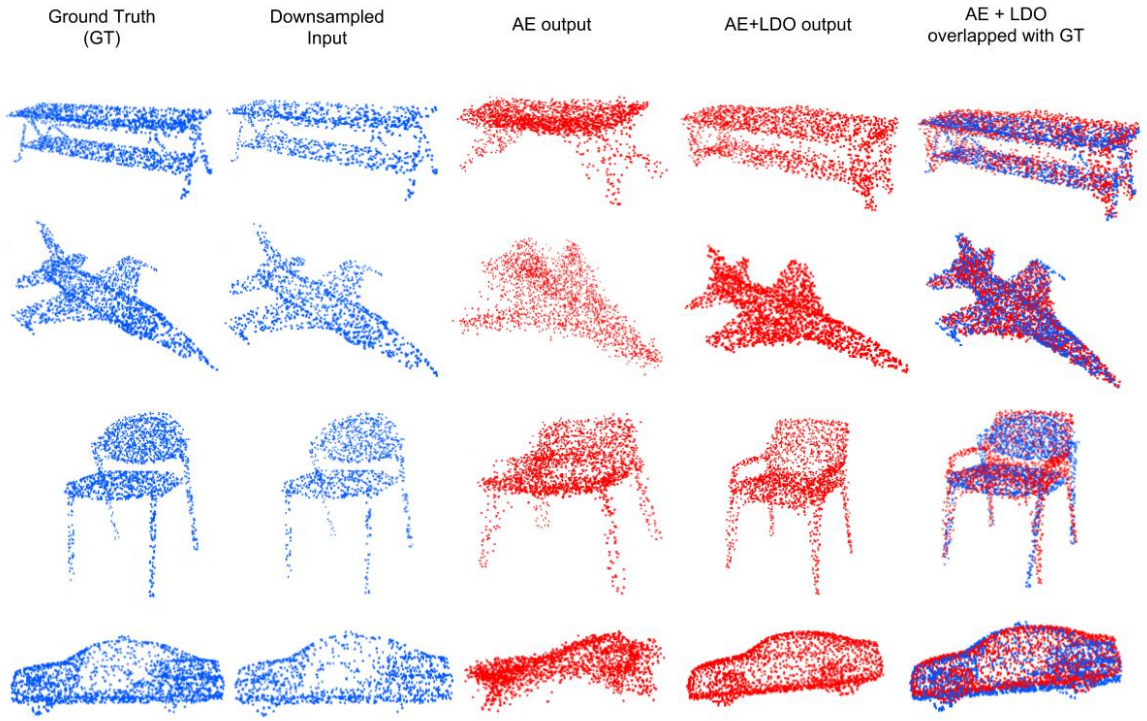


Figure 2.4: Visualizations of upsampling results of AE and AE+LDO on a test set containing all 4 classes. The inputs at test time are downsampled to 1/5th of the original points. The outputs under "AE" are from a single autoencoder trained on objects of all 4 classes. The rightmost column shows the AE+LDO outputs overlapped with the ground truth for direct comparison.

2.4.1 Masking Experiments

For the first set of experiments, we choose a synthetic masking scheme to demonstrate the benefits of using LDO in point cloud completion tasks. In order to perform masking on a point cloud, we first choose a random point from the point cloud, and remove its 2048*(X/100) nearest neighbors of the point to obtain an X% masking. To ease batch processing of point clouds with unequal number of points, we replicate one of the non-masked points so that each point cloud is the same size. The PointNet architecture by its nature ignores replicated points. In later sections, we look into more realistic scenarios where this would become important.

Varying levels of Incompletion

We train a vanilla autoencoder (AE) using the training set in the Airplane class. We then train a GAN on the GFVs of the AE. At test time, we test the AE and AE + LDO (Latent Denoising Optimization) with varying levels of masking ($20\%, 30\%, 40\%$ and 50%) in the input point cloud. The corresponding scores have been reported in Table 2.1. Note that we didn't have to retrain our model for the different levels of masking. We observe a clear improvement in performance by using our method. We also note that the performance of AE decreases with increasing levels of masking whereas AE+LDO remains more or less robust. We also train multiple denoising autoencoders (DAE) along with the corresponding GANs with varying levels of masking in the input ($20\%, 30\%, 40\%$ and 50%). The DAEs are trained to reconstruct the ground truth given the masked input. In this case, we test DAE and DAE + LDO with masking amounts that the DAE and the corresponding GAN was trained on. So a DAE and GAN trained with 40% masking are tested on 40% masking. This is done to demonstrate the benefit of LDO even when the underlying model (DAE) already has prior knowledge about the incompletion (since it was trained on the specific kind incompletion). The corresponding scores have been reported in Table 2.1. We observe that AE + LDO perform on par with a DAE despite not having any prior knowledge about the kind or amount of incompletion during training. Moreover, performing LDO on DAE provides further improvement as seen from the scores in Table 2.1. This shows that our model can integrate with any AE architecture and capitalize on the robust representations learnt by the models. A visualization of how reconstruction quality varies with increasing percentage of missing data can be found in the appendix.

Different classes

To show the robustness of our methods, we test our model on other classes, namely Chair, Car and Table, both in single-class and multi-class setups. We train a separate AE and the corresponding GAN on the training set of each category. We also train separate DAEs and corresponding GANs with 30% and 50% masking for each category. We also train a multi-class AE, GAN pair on a training set with a combination of the four classes (Table, Chair, Car and Airplane). Correspondingly we train two DAE, GAN pairs with 30% and 50% masking respectively (referred to as Multi-Class in results tables). We test these models on 30% and 50% masking using the corresponding test sets and report the results in Table 2.1. As in the previous section, the DAEs trained with specific masking amounts are tested with the same masking amounts. We observe a similar pattern as was observed in Airplane in all classes except Cars. AE+LDO performs on par or better than DAE in most of these cases. Moreover, incorporating LDO with DAE further

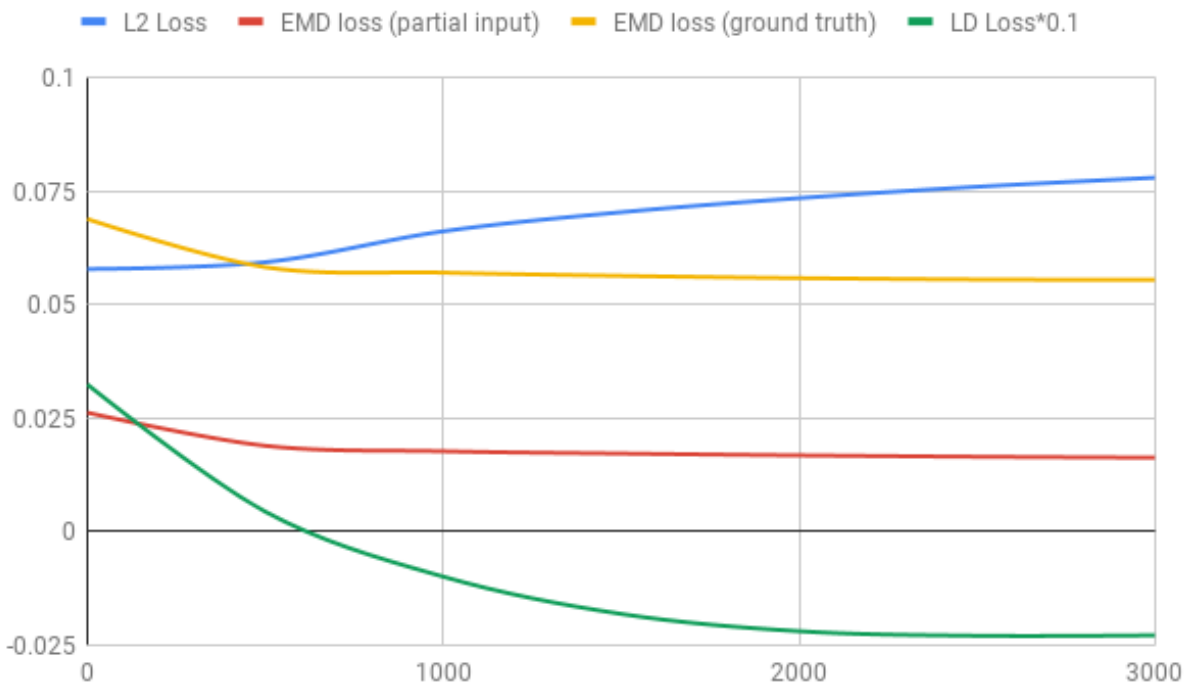


Figure 2.5: Plot of losses of a typical LDO optimization. EMD loss against ground truth is used for evaluation, not for optimization. LD loss is scaled by 0.1 for ease of visualization

improves the results and provides better scores than all other models in most cases. Interestingly, DAE trained on masked Cars performs better or on par with our models. On further inspection, we find that this is because there is very little variety in the dataset of cars. Thus DAE is able to easily transfer from the Car training set to the Car test set by simply producing the nearest neighbors from the training set. We visualize completion results with 50% masking using our best performing multi-class model (DAE + LDO) and compare it against its baseline (DAE) in Figure 5.2(An enlarged version may be found in the appendix). It is seen that our multi-loss optimization ensures that both, a sharp, valid object is reconstructed, that also fits the available partial scan as best as possible. Figure 2.3 compares the point cloud completion results of AE and AE+LDO with 50% masking. We observe that AE produces meaningless point clouds when the inputs are very highly masked/distorted. Yet, just the addition of our algorithm drastically boosts the quality of results as shown in the figure, despite the models never having been trained on incomplete point clouds.

2.4.2 Upsampling Experiments

Upsampling is another important task that comes up in processing 3D data. This is especially important for SfM methods, that often rely on sparse feature points. We investigate the performance of LDO for upsampling point clouds that had been downsampled to 20% points of the original, using just a regular AE without any special training, and see impressive results. We

Category	% Points Missing	AE	DAE	AE + LDO(ours)	DAE + LDO(ours)
Airplane	20%	0.061	0.033	0.030	**0.028**
Airplane	30%	0.079	0.036	0.037	**0.033**
Airplane	40%	0.083	0.039	0.041	**0.034**
Airplane	50%	0.097	0.039	0.038	**0.037**
Chair	30%	0.107	0.061	0.052	**0.050**
Chair	50%	0.120	0.064	0.069	**0.055**
Car	30%	0.096	0.0427	0.054	**0.041**
Car	50%	0.118	**0.046**	0.060	0.051
Table	30%	0.142	0.055	0.052	**0.047**
Table	50%	0.143	0.055	0.062	**0.050**
Multi-Class	30%	0.121	0.072	0.058	**0.044**
Multi-Class	50%	0.113	0.069	0.056	**0.046**

Table 2.1: EMD loss of completed point clouds against ground truth (lower is better). As baselines we compare against an autoencoder(AE) trained only with complete point clouds as well as a denoising AE (DAE) trained with partial point clouds. For fairness, the DAEs were trained with the same percentage of incompleteness as they were tested against. We report the performance of our LDO algorithm when used together with the AE (AE + LDO) and with the DAE(DAE+LDO). Multi-Class refers to training a single AE/DAE to reconstruct all 4 classes, as well as our own algorithm when used with these AE/DAEs

Category	Amount of downsampling at input	AE	AE + LDO
Multi-Class	80%	0.073	**0.058**

Table 2.2: EMD loss of upsampled point clouds against ground truth(lower is better). As baselines we compare against a autoencoder(AE) trained only with complete point clouds. We report the performance of our LDO algorithm when used together with the AE (AE + LDO).

show the EMD loss for plain AE and our model in Table 2.2. The upsampled visualizations are given in Figure 2.4. We see that the AE struggles to reconstruct any meaningful point clouds. Yet, just by the addition of our algorithm (AE+LDO) we observe a tremendous improvement in the upsampling quality. This shows the versatility of our approach.

2.4.3 Real-world Experiments

To evaluate the real-world applicability of LDO, we test it on the task of completing input point clouds obtained from SfM. The aim is to see whether LDO can generalize to real-world point cloud data, *while having been trained only on synthetic data*. We use COLMAP [65], a general purpose SfM pipeline to generate point clouds using sequences of images. We experiment with the following classes:

1. Shapenet type models: We use some toy car and airplane models for testing. The models were placed on a rotating surface and multiple images were clicked from different poses

around the object. These images were then processed through COLMAP. The output point cloud had incompletions due to severe lack of texture on these models. We test these incomplete point clouds on the multi-class AE and AE+LDO, as described in section 4.1.2. We choose these, since we don't have a training set of real world noisy point coulds along with their clean counterparts. But we also provide the results of DAE and DAE+LDO (trained with 50% masking on ShapeNet models) in the Appendix.

2. Faces : We first create a synthetic training set of 3000 face point clouds using the Basel 3D Morphable Model [59]. The model provides a PCA basis for faces, and different faces can be obtained by sampling the PCA coefficients from gaussians. We train an AE with similar architecture as the ShapeNet AEs, except with an input/output size of 8192x3. We then also train a GAN on the GFVs obtained by this AE, as in the regular procedure to setup LDO. Next, we use the CMU Multi-PIE [25] to obtain a sequence of images of human faces taken from different poses. These are processed through COLMAP to obtain point clouds. These are tested on the AE and AE+LDO models trained on the synthetic Basel dataset. Here again, we provide the results for DAE and DAE+LDO (trained with 50% masking on the synthetic training set) in the Appendix.

For both classes, we align and do a "rough" cleaning of the obtained point cloud by aligning it against a template point cloud of the corresponding synthetic set, and removing points beyond a threshold distance from the template. Note that is only to remove background points - the intrinsic noisiness of the points characteristic of SfM is preserved. Where needed we also downsample them to fit our model resolution. The qualitative results can be seen in Fig 2.6. It is observed that the AE by itself, having only been trained on synthetic data, fails completely on the ShapeNet-type models, and reconstructs badly fitting faces for the face models, since it fails to generalize beyond the PCA-basis constrained synthetic faces. However, AE+LDO gets better reconstructions that fit well with the partial input. We also observe DAE performs worse than AE, as expected, due to the incorrectly learnt priors. But surprisingly, DAE+LDO seems to perform better than all the other models (DAE, AE, AE+LDO). This illustrates the benefits of LDO and its ability to exploit the capacity of the underlying model even when the model was exposed to a bad prior.

2.4.4 Analysis of loss functions

We show a plot of the 3 losses used in LDO optimization and the EMD loss against ground truth (used for evaluation) during the optimization process of DAE + LDO in Fig 2.5. The x-axis shows the number of iterations and the y-axis shows the loss values as the optimization progresses. The plot shows that the initialization encoder provides a decent initialization for the optimization, as measured by the ground truth EMD Loss (EMD-GT). This shows that the initialization itself is decent enough to provide scores competitive to that of the DAE. The optimization that follows is responsible for the improvements over the baseline DAE model. We observe that throughout the optimization, the three losses, namely, Partial-EMD Loss, Discriminator loss(LD Loss) and EMD-GT Loss decay gradually until the end of optimization, whereas the L2 loss increases gradually. This indicates that the GFV is being cleaned as it moves away from the noisy GFV and moves closer to the clean GFV. The optimization highlighted here takes 324 sec to process

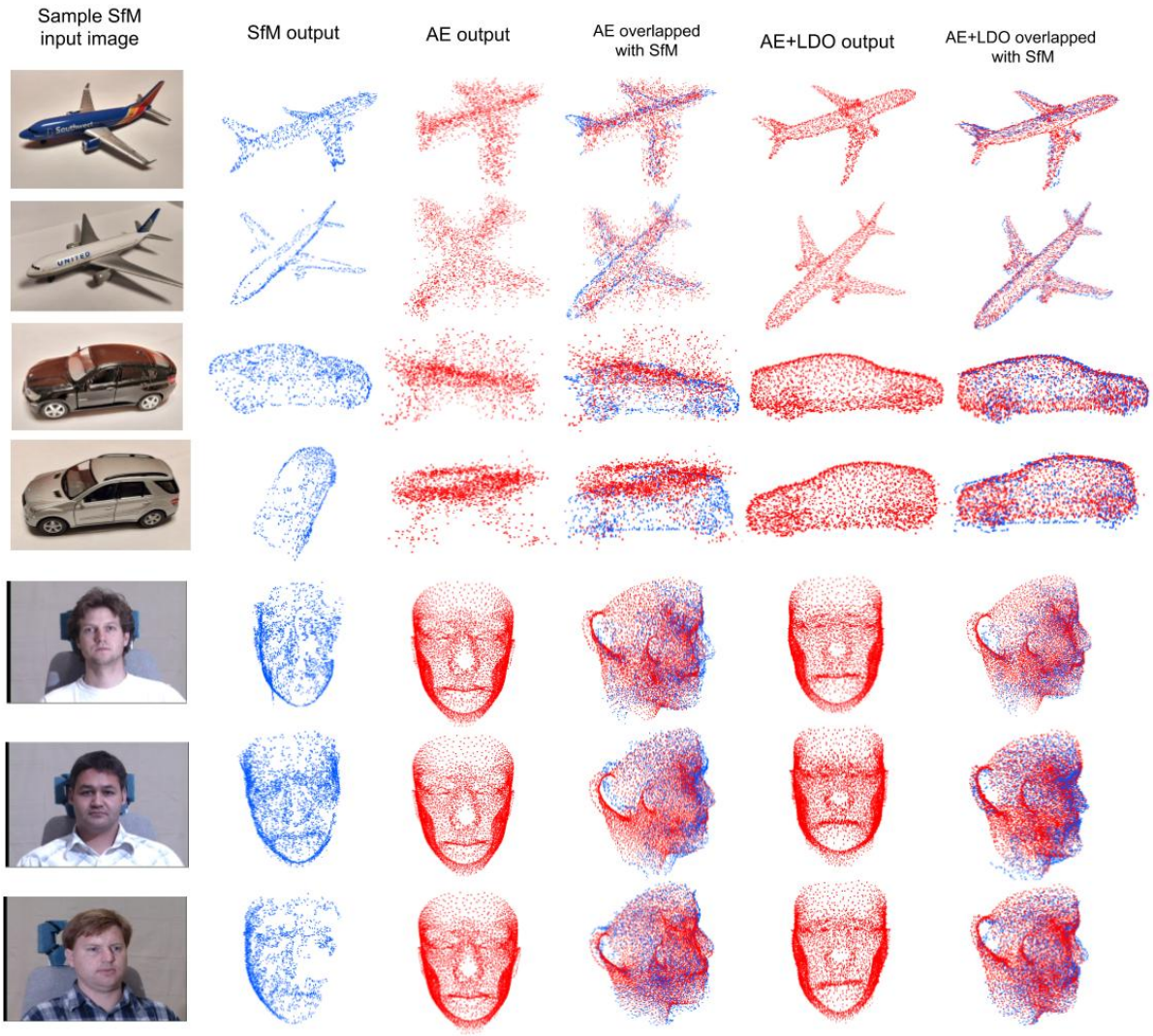


Figure 2.6: Visualizations of shape completion task on noisy and incomplete point clouds generated by a general-purpose SfM pipeline.

50 point clouds on a Titan X GPU. The batch size could be increased to achieve a lower time per point cloud.

2.5 Discussion

In this chapter, we saw that using the standard approaches for end-to-end learning using purely data-driven approaches can struggle when exposed to distribution shifts at test time. Thus, we show that the knowledge of such distributional shifts should be incorporated in the model training or inference procedure to account for the neccessary invariances. These invariances can help improve generalization performance and provide more robustness to out-of-distribution samples.

However, note that in this problem, the prior's connection to the data was very explicit, i.e, we knew that the output distribution remained constant under distributional shifts and that could be directly incorporated into the model. In the next chapted, we would be extending this thought process to a domain where the prior isn't directly linked to the data and has to be used in a much more indirect manner.

Chapter 3

Visual Dialog

3.1 Problem Statement

There has been a lot of recent progress in the machine learning community with developing models for vision and language understanding, the two most important modalities used by humans in the real world. Capitalizing on the growth in both these domains, it now seems plausible to build more advanced dialog systems capable of reasoning over multiple modalities while also learning from one another. Such systems will allow humans to have a meaningful dialog with intelligent systems containing visual as well as textual content. Use cases include assistive systems for the visually impaired, smart multimodal dialog agents (unlike current versions of Siri and Alexa which are primarily audio based and cannot make effective use of multimodal data) and even large scale visual retrieval systems. However, as these systems become more advanced, it will become increasingly common to have two agents interact with each other to achieve a particular goal [43]. We want these conversations to be interpretable to humans for the sake of transparency and ease of debugging. This motivates our work on goal-driven agents which interact in coherent language understandable to humans. This chapter presents work on Goal driven Visual Dialog Agents.

3.1.1 Challenges

Most prior work on visual dialog [[11], [13]] has approached the problem using supervised learning where the dialog model is learned using ground truth supervision from a human-human dialog dataset to simply increase the likelihood of the human-human dialog. Such supervised learning based methods have obvious issues though. First, MLE is known to result in models that generate repetitive dialogs and often produce generic responses. Second, since the agents are never allowed to interact during training, they end up encountering out-of-distribution questions and answers when made to interact during evaluation, which would further reduce the task performance. However, there has been recent work [12] trying to approach the problem using reinforcement learning by letting the agents interact using natural language with a common goal of trying to improve the Q-Bot's understanding of the image. However, as we will discuss later, the optimization problem in this interactive setting does not make the agents stick to the domain

of natural language. Thus, we observe that as we continue training the agents start generating non-grammatical and semantically meaningless sentences. As in the previous section, we observe that there is a distributional shift between the training and testing scenarios. Thus, we need to add additional biases/priors to improve the generalization ability of the model.

However, there doesn't seem to be any obvious priors on the data distribution itself that could be utilized trivially as in the last scenario. We do, however, observe that humans continue to speak in commonly spoken languages despite having plenty of opportunities to specialize, and hypothesize that this is *because they need to communicate with an entire community*, and having a private language for each person would be extremely inefficient. With this idea, we let our agents learn in a similar setting, by making them talk to (ask questions of, get answers from) multiple agents, one by one. As opposed to the last scenario, this is a much more implicit prior on the data generating process.

3.1.2 Assumptions

As described earlier, in this problem, we are going to make assumptions about the underlying data generating process and incorporate the corresponding priors. To put it more explicitly, we make the following assumptions:

- We assume that humans don't deviate from natural language because developing a consensus between a huge population of humans is hard.
- We further hypothesise that this behavior will also be observed with our trained dialog models when training with reinforcement learning.

3.2 Previous Literature

Most of the major works which combine vision and language have traditionally focused on the problem of image captioning (([36], [87], [82], [30], [47], [90]) and visual question answering ([3], [93], [24]). The problem of visual dialog is relatively new and was first introduced by Das et al. [11] who also created the VisDial dataset to advance the research on visually grounded dialog. The dataset was collected by pairing two annotators on Amazon Mechanical Turk to chat about an image. They formulated the task as a 'multi-round' VQA task and evaluated individual responses at each round in an image guessing setup. In subsequent work by Das et al. [12] they proposed a reinforcement learning based setup where they allowed the Question bot and the Answer bot to have a dialog with each other with the goal of correctly predicting the image unseen to the Question bot. However, in their work they noticed that the reinforcement learning based training quickly led the bots to diverge from natural language. In fact Kottur et al. [37] recently showed that language emerging from two agents interacting with each other might not even be interpretable or compositional. We use community regularization to alleviate this problem. Recent work has also proposed using such goal driven dialog agents for other related tasks including negotiation [44] and collaborative drawing [35]. We believe that our work can easily extend to those settings as well. Lu et al. [48] proposed a generative-discriminative framework for visual dialog where they train only an answer bot to generate informative answers for ground truth questions. These answers were then fed to a discriminator, which was trained

to rank the generated answer among a set of candidate answers. This is a major restriction of their model as it can only be trained when this additional information of candidate answers is available, which restricts it to a supervised setting. Furthermore, since they train only the answer bot and have no question bot, they cannot simulate an entire dialog which also prevents them from learning by self-play via reinforcement. Wu et al. [85] further improved upon this generative-discriminative framework by formulating the discriminator as a more traditional GAN [23], where the adversarial discriminator is tasked to distinguish between human generated and machine generated dialogs.

3.3 Method

3.3.1 Agent Architectures

We describe all the different components of the agent architectures in this section. Note that the overall architecture is mostly borrowed from Das et al. [12], Lu et al. [48] with slight modifications to individual units and an additional caption encoder. We explain these modifications in detail in this section. We would like to stress that these changes are not the main contribution of our paper. The main contribution is the Multi-agent dialog framework described in section 3.3.2.

Question Bot Architecture

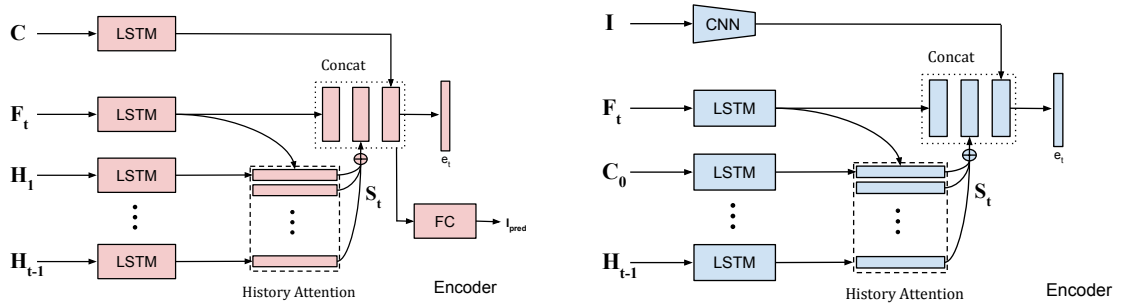
The question bot architecture we use is inspired by the answer bot architecture in Das et al. [12], Lu et al. [48] but the individual units have been modified to provide more useful representations. Similar to the original architecture, our Q-Bot, shown in Fig. 3.1, also consists of 4 parts, (a) fact encoder, (b) state-history encoder, (c) question decoder and (d) image regression network.

1. **Fact Encoder:** The fact encoder is a unidirectional LSTM which is given the previous question-answer pair (q_{t-1}, a_{t-1}) as input. The LSTM generates a fact embedding $F_t \in R^{512}$.
2. **State/History Encoder:** We modify the state-history encoder to incorporate a two-level hierarchical encoding of the dialog. The encoder first computes the fact embeddings $H_t^Q = (F_1, F_2, F_3 ... F_{t-1})$, using an LSTM akin to the fact encoder described above. We pass these embeddings and F_t computed by the Fact Encoder through a fully connected layer, generating attention weights which are used to attend over H_t^Q, producing the history embedding $S_t^Q \in R^{512}$. Notice that this results in a two-level hierarchical encoding of the dialog $(q_t, a_t) \rightarrow F_t$ and $(F_1, F_2, F_3, ... F_t) \rightarrow S_t^Q$.
3. **Caption Encoder:** This is a unidirectional LSTM which is given the image caption c as input. The LSTM generates a caption embedding $C^Q \in R^{512}$.
4. **Feature Regression Network:** $\{F_t^Q, S_t^Q, C^Q\}$ are concatenated to produce an embedding E_t^Q. This is passed through 2 fully connected layers with dropout to produce $\hat{y}_t$ from the current encoded state $\hat{y}_t = f(S_t^Q)$.
5. **Question Decoder:** The hidden state of this LSTM is initialized with the hidden state of the fact encoder. E_t^Q is passed through a fully connected layer to generate e_t^Q, which is used

to update the hidden state of the LSTM of the question decoder. The question q_t is then generated by sequentially sampling words (either via teacher forcing during supervised pretraining or via autoregressive generation during RL and evaluation).

Note that we use a dropout of 0.5 in all the LSTMs during training. All LSTM hidden layers sizes are 512, and the image embedding size is 4096. The input word embedding size is 300.

Figure 3.1: Agent Encoder Architectures (Left: Q-Bot, Right:A-Bot



Answer Bot Architecture

The architecture for A-Bot, also inspired from Lu et al. [48], shown in Fig. 3.1, is similar to that of the Q-Bot. It has 3 components: (a) question encoder, (b) state-history encoder and (c) answer decoder.

1. **Question Encoder:** The question encoder is a unidirectional LSTM which is given the current question q_t generated by the Q-Bot as input. The LSTM generates a question embedding $Q_t^A \in R^{512}$.
2. **State/History/Image Encoder:** The encoder first computes the fact embeddings $H_t^A = (F_1, F_2, F_3...F_{t-1})$, using an LSTM akin to the fact encoder described above. By passing these embeddings and the Q_t^A computed by the Question Encoder through a fully connected layer, attention weights are generated which are used to attend over H_t^A, producing the history embedding $S_t^A \in R^{512}$. Notice that this results in a two-level hierarchical encoding of the dialog $(q_t, a_t) \rightarrow F_t$ and $(F_1, F_2, F_3...F_t) \rightarrow S_t^A$. $\{Q_t^A, S_t^A, y_{gt}\}$ are then concatenated to produce an embedding E_t^A.
3. **Answer Decoder:** The hidden state of this LSTM is initialized with the hidden state of the question encoder. E_t^A is passed through a fully connected layer to generate e_t^A, which is used to update the hidden state of the LSTM of the answer decoder. The answer a_t is then generated by sequentially sampling words (either via teacher forcing during supervised pretraining or via autoregressive generation during RL and evaluation).

3.3.2 Training

We follow the training process proposed in Das et al. [12]. Two agents, a Q-Bot and an A-Bot are first trained in isolation, by supervision from the VisDial dataset. After this supervised

pretraining for 15 epochs over the data, we smoothly transition the agents to learn from each other via reinforcement learning. The individual phases of training will be described in more detail below. Note that the key novelty of the work is the multi-agent dialog framework proposed in Section 3.3.2

Supervised pre-training

In the supervised part of training, both the Q-Bot and A-Bot are trained separately, using a Maximum Likelihood Estimation (MLE) loss computed using the ground truth questions and answers, respectively, for every round of dialog. The Q-Bot simultaneously optimizes another objective: minimizing the Mean Squared Error (MSE) loss between the true (y_{gt}) and predicted (y_t) image embeddings. The ground truth dialogs and image embeddings are from the VisDial dataset.

Given the true dialog history, image features and a question from the dataset, the A-Bot generates an answer to that question. Given the true dialog history and the previous question-answer pair from the dataset, the Q-Bot is made to generate the next question to ask the A-Bot. Both agents receive only ground truth questions and answers, never what the other agent generated - so the two agents never actually interact during this phase of training. However, there are multiple problems with this training scheme. First, MLE is known to result in models that generate repetitive dialogs and often produce generic responses. Second, since the agents are never allowed to interact during training, they end up encountering out-of-distribution questions and answers when made to interact during evaluation, which reduces the task performance. This can be observed in Figure 3.2. The performance of the agents trained via supervised learning dips after each successive dialog round.

Reinforcement Learning Setup

To alleviate the issues pointed out with supervised training, we let the two bots interact with each other via self-play (no ground-truth except images and captions). This interaction is also in the form of questions asked by the Q-Bot, and answered in turn by the A-Bot, using a shared vocabulary. The state space is partially observed and asymmetric, with the Q-Bot observing $\{c, q_1, a_1 \dots q_{10}, a_{10}\}$ and the A-Bot observing the same, plus the image I. Here, c is the caption, and q_i, a_i is the i^{th} dialog pair exchanged where $i = 1 \dots 10$. The action space for both bots consists of all possible output sequences of a specified maximum length (Q-Bot: 16, A-Bot: 9) under a fixed vocabulary (size 8645). Each action involves predicting words sequentially until a stop token is predicted, or the generated statement reaches the maximum length. Additionally, Q-Bot also produces a guess of the visual representation of the input image (VGG fc-7 embedding of size 4096). Both Q-Bot and A-Bot share the same objective and get the same reward to encourage cooperation. Information gain in each round of dialog is incentivized by setting the reward as the **change in distance** of the predicted image embedding to the ground-truth image representation. This means that a QA pair is of high quality only if it helps the Q-Bot make a better prediction of the image representation. Both policies are modeled by neural networks, as discussed in Section 3.3.1.

A dialog round at time t consists of the following steps: 1) the Q-Bot, conditioned on the state

encoding, generates a question q_t, 2) A-Bot updates its state encoding with q_t and then generates an answer a_t, 3) Both Q-Bot and A-Bot encode the completed exchange as a fact embedding, 4) Q-Bot updates its state encoding to incorporate this fact and finally 5) Q-Bot predicts the image representation for the unseen image conditioned on its updated state encoding.

Similar to Das et al. [11], we use the REINFORCE [83] algorithm that updates policy parameters in response to experienced rewards. The per-round rewards that are used to calculate the discounted returns follow:

$$r_t(s_t^Q, (q_t, a_t, y_t)) = l(y_{t-1}, y^{gt}) - l(y_t, y^{gt}) \tag{3.1}$$

This reward is positive if the distance between image representation generated at time t is closer to the ground truth than the representation at time $t-1$, hence incentivizing information gain at each round of dialog. The REINFORCE update rule ensures that a (q_t, a_t) exchange that is informative has its probabilities pushed up. Do note that the image feature regression network f is trained directly via supervised gradient updates on the L-2 loss.

However, as noted above, this RL optimization problem is ill-posed, since rewarding the agents for information exchange does not motivate them to stick to the rules and conventions of the English language. Thus, we follow an elaborate curriculum scheme described in [11]. Specifically, for the first K rounds of dialog for each image, the agents are trained using supervision from the VisDial dataset. For the remaining 10-K rounds, however, they are trained via reinforcement learning. K starts at 9 and is linearly annealed to 0 over 10 epochs. Despite these modifications the bots are still observed to diverge from natural language and produce non-grammatical and incoherent dialog. Thus, we propose a multi bot architecture to help the agents interact in diverse and coherent, yet informative, dialog.

Multi-Agent Dialog Framework (MADF)

In this section we describe our proposed Multi-Agent Dialog architecture in detail. We claim that if, instead of allowing a single pair of agents to interact, we were to make the agents more social, and make them *interact and learn from multiple other agents*, they would be disincentivized to develop a private language, and would have to conform to the common language. We call this Community Regularization.

In particular, we create either multiple Q-bots to interact with a single A-bot, or multiple A-bots to interact with a single Q-bot. All these agents are initialized with the learned parameters from the supervised pretraining as described in Section 3.3.2. Then, for each batch of images from the VisDial dataset, we randomly choose a Q-bot to interact with the A-bot, or randomly choose an A-bot to interact with the Q-bot, as the case may be. The two chosen agents then have a complete dialog consisting of 10 question-answer pairs about each of those images, and update their respective weights based on the rewards received (as per Equation 3.1) during the conversation, using the REINFORCE algorithm. This process is repeated for each batch of images, over the entire VisDial dataset. It is important to note that histories are *not shared* across batches. MADF can be understood in detail using the pseudocode in Algorithm 3.

Connection to Regularization: It is interesting to note that the MADF setting can actually be seen as a regularizer for the model. To establish this more formally, we look at the total loss minimized by each agent. The Total loss (TL) being minimized during the RL phase = $A1_t$ +

$A2_t$ at time t, where $A1_t$ is negative of the RL reward as described in section 3.3.2, and $A2_t$ is the L-2 loss between predicted and true image embeddings. Consider a setting with N Abots and 1 QBot. The $A2_t$ Loss can be written as:

$$A2_t = \sum_{i=1}^{N}(y_t^{(i)} - y^{gt})^2 = (y_t^{(1)} - y_t^{gt})^2 + \sum_{i=2}^{N}(y_t^{(i)} - y^{gt})^2$$

From the equation, we observe that $A2_t$ is a sum of 2 terms, where the first term is the standard regression loss which would apply for the 1Q,1A case. The second term can be viewed as a regularization imposed by pairing the other A-bots with the Q-bot, hence we can rewrite A2 as:

$$A2_t = (y_t^{(1)} - y_t^{gt})^2 + R_{A2_t}(\theta) \tag{3.2}$$

where $R_{A2_t}(\theta)$ represents regularization imposed by the other agents. Similarly, $A1_t$ can also be broken down into a likelihood and a regularization term as follows:

$$A1_t = -G_t^{(1)} log\pi(q_t^{(1)}, a_t^{(1)}) + R_{A1_t}(\theta) \tag{3.3}$$

where $G_t^{(1)}$ is the monte-carlo return calculated using the first pair of agents at time t. Thus, both the terms in the total loss can be broken down into a loss term akin to the 1Q, 1A case and a regularization term. This regularization term comes from the regularization imposed by pairing each Q-Bot with multiple A-bots or vice versa. This clearly shows that the multi-bot framework can be seen as a form of regularization. In the experiments we show that the regularization helps with the language quality by ensuring that the bots don't deviate much from natural language.

Algorithm 3 Multi-Agent Dialog Framework (MADF)

procedure MULTIBOTTRAIN
  while train_iter ¡ max_train_iter **do** ▷ Main Training loop over batches
    $Qbot \leftarrow$ *random_select* $(Q_1, Q_2, Q_3....Q_q)$
    $Abot \leftarrow$ *random_select* $(A_1, A_2, A_3....A_a)$ ▷ Either q or a is equal to 1
    $history \leftarrow (0, 0, ...0)$ ▷ History initialized with zeros
    $fact \leftarrow (0, 0, ...0)$ ▷ Fact encoding initialized with zeros
    $\Delta image\_pred \leftarrow 0$ ▷ Tracks change in Image Embedding
    $Qz_1 \leftarrow Ques\_enc(Qbot, fact, history, caption)$
    for t in 1:10 **do** ▷ Have 10 rounds of dialog
      $ques_t \leftarrow Ques\_gen(Qbot, Qz_t)$
      $Az_t \leftarrow Ans\_enc(Abot, fact, history, image, ques_t, caption)$
      $ans_t \leftarrow Ans\_gen(Abot, Az_t)$
      $fact \leftarrow [ques_t, ans_t]$ ▷ Fact encoder stores the last dialog pair
      $history \leftarrow concat(history, fact)$ ▷ History stores all previous dialog pairs
      $Qz_t \leftarrow Ques\_enc(Qbot, fact, history, caption)$
      $image\_pred \leftarrow image\_regress(Qbot, fact, history, caption)$
      $R_t \leftarrow (target\_image - image\_pred)^2 - \Delta image\_pred$
      $\Delta image\_pred \leftarrow (target\_image - image\_pred)^2$
    $\Delta(w_{Qbot}) \leftarrow \frac{1}{10}\sum_{t=1}^{10} \nabla_{\theta_{Qbot}} [G_t \log p(ques_t, \theta_{Qbot}) - \Delta image\_pred]$
    $\Delta(w_{Abot}) \leftarrow \frac{1}{10}\sum_{t=1}^{10} G_t \nabla_{\theta_{Abot}} \log p(ans_t, \theta_{Abot})$
    $w_{Qbot} \leftarrow w_{Qbot} + \Delta(w_{Qbot})$ ▷ REINFORCE and Image Loss update for Qbot
    $w_{Abot} \leftarrow w_{Abot} + \Delta(w_{Abot})$ ▷ REINFORCE update for Abot

Table 3.1: Comparison of answer retrieval metrics with previously published work. SL has the best scores. The scores drop drastically in RL-1Q,1A, but MADF agents (RL-3Q,1A and RL-1Q,3A) are able to retain the same language quality as the SL agent.

Model	MRR	Mean Rank	R@10
Answer Prior [11]	0.3735	26.50	53.23
MN-QIH-G [11]	0.5259	17.06	68.88
HCIAE-G-DIS [48]	0.547	14.23	71.55
Frozen-Q-Multi [12]	0.437	21.13	60.48
CoAtt-GAN [85]	0.5578	14.4	71.74
SL(Ours)	**0.610**	**5.323**	**72.68**
RL - 1Q,1A(Ours)	0.459	7.097	72.34
RL - 1Q,3A(Ours)	0.601	5.495	72.48
RL - 3Q,1A(Ours)	0.590	5.56	72.61

Table 3.2: Human Evaluation Results - Mean Rank (Lower is better) : Results show that RL-3Q,1A outperforms RL-1Q,3A for A-relevance and overall coherence but otherwise SL (Supervised Learning), RL-1Q,3A, and RL-3Q,1A showed equivalent performance indicating that community regularization can effectively eliminate any losses to human intelligibility introduced through RL.

	Metric	N	SK	RL 1Q,1A	RL 1Q,3A	RL 3Q,1A
1	Question Relevance	49	**1.97**	3.57	2.20	2.24
2	Question Grammar	49	2.16	3.67	2.24	**1.91**
3	Overall Dialog Coherence: Q	49	2.08	3.73	2.34	**1.83**
4	Answer Relevance	53	2.09	3.77	2.28	**1.84**
5	Answer Grammar	53	2.20	3.75	2.05	**1.98**
6	Overall Dialog Coherence: A	53	2.09	3.64	2.35	**1.90**

3.4 Evaluation

3.4.1 Dataset description

We use the VisDial 0.9 dataset for our task introduced by Das et al. [11]. The data is collected using Amazon Mechanical Turk by pairing 2 annotators and asking them to chat about the image as a multi round VQA setup. One of the annotators acts as the questioner and has access to only the caption of the image and has to ask questions from the other annotator who acts as the 'answerer' and must answer the questions based on the visual information from the actual image. This dialog repeats for 10 rounds at the end of which the questioner has to guess what the image was. We perform our experiments on VisDial v0.9 (the latest available release) containing 83k dialogs on COCO-train and 40k on COCO-val images, for a total of 1.2M dialog question-answer pairs. We split the 83k into 82k for train, 1k for validation, and use the 40k as test, in a manner consistent with [11]. The caption is considered to be the first round in the dialog history.

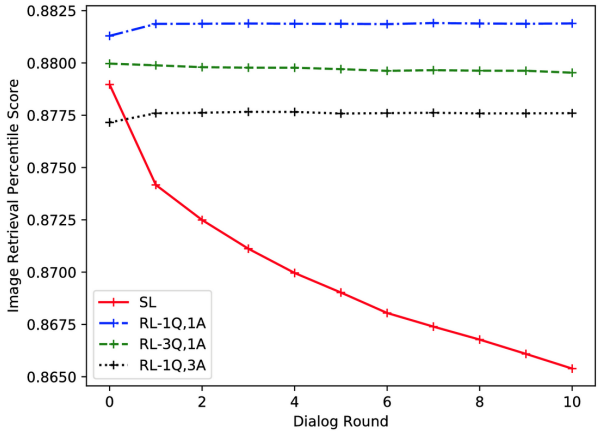


Figure 3.2: Comparison of Task Performance: Image Retrieval Percentile scores. This refers to the percentile scores of the ground truth image compared to the entire test set of 40k images, as ranked by distance from the Q-Bot's estimate of the image. The X-axis denotes the dialog round number (from 1 to 10), while the Y-axis denotes the image retrieval percentile score. The percentile score decreases monotonically for SL, but is stable for all versions using RL. This shows that the MADF agents are able to capitalize on the benefits of interactive learning.

3.4.2 Evaluation Metrics

We evaluate the performance of our model's individual responses by using 4 metrics, proposed by [12]: **1) Mean Reciprocal Rank (MRR)**, **2) Mean Rank**, **3) Recall@10** and **4) Image Retrieval Percentile**. Mean Rank and MRR compute the average rank (and its reciprocal, respectively) assigned to the ground truth answer, over a set of 100 candidate answers for each question (also averaged over all the 10 rounds). Recall@10 computes the percentage of answers with rank less (better) than 10. Intuitively all these language metrics are trying to measure similar things, i.e, how highly does the model rank the ground truth answer over a set of 100 candidate responses. Thus, if the model gives a lower rank to the ground truth answer, then we can say that the model is highly likely to produce the ground truth response to the question. But at the same time they do have some qualitative differences. For example, MRR and Rank@10 are more robust to outliers but Mean Rank is not (but it is more interpretable). The fourth metric, i.e, the Image Retrieval percentile, is different from the first 3 metrics. It is a measure of how close the image prediction generated by the Q-bot is to the ground truth. All the images in the test set are ranked according to their distance from the predicted image embedding, and the rank of the ground truth embedding is used to calculate the image retrieval percentile. This gives a measure of the quality of the information exchange. All results are reported after 15 epochs of supervised learning and 10 epochs of curriculum learning as described in Section 3.3.2. Consequently, the training time of all 3 systems are equal.

Table 3.1 compares the Mean Rank, MRR, and Recall@10 of our agent architecture and dialog framework (below the horizontal line) with previously proposed architectures (above the line). SL refers to the agents after only the isolated, supervised training of Section 3.3.2. RL-1Q,1A refers to a single, idiosyncratic pair of agents trained via reinforcement as in Section 3.3.2. RL-1Q,3A and RL-3Q,1A refer to social agents trained via our Multi-Agent Dialog framework

in Section 3.3.2, with 1Q,3A referring to 1 Q-Bot and 3 A-Bots, and 3Q,1A referring to 3 Q-Bots and 1 A-Bot.

It can be seen that our agent architectures clearly outperform all previously published results using generative architectures in MRR, Mean Rank and R@10. This indicates that our approach produces consistently good answers (as measured by MRR, Mean Rank and R@10). It is important to note that the point here is not to demonstrate the superiority of our architecture compared to other architectures. The point here is instead to show that the MADF framework (RL-3Q,1A and RL-1Q,3A) is able to maintain the same language quality as the supervised agent while improving the image retrieval scores. In fact, community regularization (in the form of the proposed MADF setup) can be integrated with any of the visual dialog algorithms in Table 3.1. Notice that SL has the best scores. The scores drop drastically in RL-1Q,1A, but RL-3Q,1A and RL-1Q,3A obtain scores comparable to SL. This shows that the agents trained by MADF are able to maintain the language quality of SL agents without sacrificing much on the task performance (image retrieval percentile). Fig. 3.2 shows the change in image retrieval percentile scores over dialog rounds. The percentile score decreases monotonically for SL, however it is stable for all versions using RL. The decrease in image retrieval score over dialog rounds for SL is because the test set questions and answers are not perfectly in-distribution (compared to the training set), and the SL system can't adapt to these samples as well as the systems trained with RL. As the dialog rounds increase, the out-of-distribution nature of dialog exchange increases, hence leading to a decrease in SL scores. Interestingly, despite having strictly more information in later rounds, the scores of RL agents do not increase - which we think is because of the nature of recurrent networks to forget.

The results in Fig. 3.2 and Table 3.1 show that the MADF setup allows the agents to achieve consistent task performance without sacrificing on language quality. We further support this claim in the next section where we show that human evaluators rank the language quality of MADF agents to be much better than the agents trained via reinforcement without community regularization.

3.4.3 Human Evaluation

There are no quantitative metrics to comprehensively evaluate dialog quality, hence we do a human evaluation of the generated dialog. There are 6 metrics we evaluate on: 1) Q-Bot Relevance, 2) Q-Bot Grammar, 3)A-Bot Relevance, 4) A-Bot Grammar, 5) Q-Bot Overall Dialog Coherence and 6) A-Bot Overall Dialog Coherence. We evaluate 4 Visual Dialog systems, trained via: 1) **Supervised Learning (SL)**, 2) **Reinforce for 1 Q-Bot, 1 A-Bot (RL-1Q,1A)**, 3) **Reinforce for 1 Q-Bot, 3 A-Bots (RL-1Q,3A)** and 4) **Reinforce for 3 Q-Bots, 1 A-Bot (RL-3Q,1A)**. We asked a total of 61 people to evaluate the 10 QA-pairs generated by each system for a total of 102 randomly chosen images, requiring them to give an ordinal ranking (from 1 to 4) for each metric. All the evaluators were provided with the caption from the dataset. Evaluators taking the perspective of the A-Bot were provided with the image and asked to evaluate answer relevance and grammar, while those taking the perspective of the Q-Bot evaluated question relevance and grammar. Both groups rated dialogs for overall coherence. Table 3.2 contains the average ranks obtained on each metric (lower is better).

The results convincingly validate our hypothesis that having multiple A-Bots/Q-Bots im-

proves the language quality as compared with single Q-Bot and A-Bot. Kruskal-Wallis tests found strong differences among rankings ($p < .0001$) across all measures. Pairwise comparisons using the Mann-Whitney U test found a consistent pattern in which RL 1Q,1A performed substantially worse than other methods across all measures: for

1. **Q-relevance**: SL: U=348, $p < .0001$; RL-1Q3A: U=2235, $p < .0001$; RL-3Q1A U=2209, $p < .0001$,
2. **Q-grammar**: SL: U=319, $p < .0001$; RL-1Q3A U=2280, $p < .0001$; RL-3Q1A U=2221, $p < .0001$;
3. **A-relevance**: SL U=256, $p < .0001$; RL-1Q3A U=2741, $p < .0001$; RL-3Q1A U-2909, $p < .0001$;
4. **A-grammar**: SL U=305, $p < .0001$; RL-1Q3A U=2857, $p < .0001$; RL-3Q1A U=2673, $p < .0001$;
5. **Overall (both groups)**: SL U=1206, $p < .0001$; RL-1Q3A U= 9458, $p < .0001$; RL-3Q1A U=10052, $p < .0001$.

Results showed that RL 3Q,1A outperformed RL 1Q,3A for A-relevance U=1889, $p < .02$ and overall coherence U=6543, $p < .006$ but otherwise SL, RL-1Q,3A, and RL-3Q,1A showed equivalent performance indicating that community regularization can effectively eliminate any losses to human intelligibility introduced through reinforcement learning. These results further support the claims made in the previous section that the MADF setup allows the agents to show consistent task performance (image retrieval percentile) while maintaining the language quality of the supervised agents.

We show a couple of randomly chosen examples from the set shown to the human evaluators in Fig. **??**. The trends observed in the scores given by human evaluators are also clearly visible in this example. MADF agents are able to model the human responses much better than RL 1Q,1A and are about as well as (if not better) than SL trained agents. It can also be seen that although the RL-1Q,1A system has greater diversity in its responses, the quality of those responses is greatly degraded, with the A-Bot's answers especially being both non-grammatical and irrelevant.

3.5 Discussion

In this chapter, we studied existing methods used for the visual dialog task and proposed the MADF framework to improve generalization. Moreover, it's important to note that, the technical problems of generalization, grounding and distribution mismatch studied in this chapter aren't specific to visual dialog. In fact, the problem and the corresponding approach suggested to tackle the problem using the MADF framework are applicable to all dialog applications. However, the core concept harks back at utilizing the priors about the dataset/problem in order to obtain a more desirable solution.

Shifting gears, in the next chapter, we will discuss a very different problem where we as data scientist don't have enough intuitions about the dataset to make reasonable assumptions and build priors into our models. In such situations, it becomes critical to intelligently collect more data to augment the limited priors available.

Chapter 4

Poaching Threat Prediction

4.1 Problem Statement

Wildlife conservation agencies try multiple solutions to protect the wildlife and their habitats from poaching and illegal trade. One of these is to send rangers to patrol in protected conservation areas [41]. However, because of limited patrolling resources, it is impossible to monitor all intrusion routes and protect the entire area. Thankfully, rangers record their findings, including animal signs and poaching activity signs, e.g., snares placed by poachers during the patrol. One can analyze these records to get insights of the poaching patterns and strategically allocate resources to detect and deter poaching activities.

In this chapter, we would be working on the problem of poaching threat prediction in a Wildlife Conservation Area and use the predicted threat to suggest routes that the rangers could take to efficiently patrol the areas. This problem statement is unique and different from the other problems in this thesis and exposes us to a variety of problems that are usually ignored or swept under the rug by the machine learning community.

4.1.1 Challenges

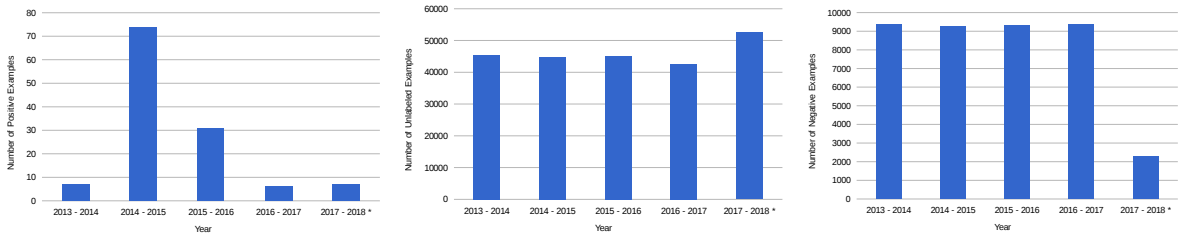
The dataset comes with several challenges and peculiarities. It suffers from significant class imbalance, sparsity and noise in negative labels. Furthermore, there aren't a lot of obvious priors about the features that we as designers had which we could exploit. So, the data was essentially unstructured. Making matters worse, we additionally had to tackle the issue of shifting distributions as the poachers would seasonally shift the areas they target depending on the previous years routes taken by the rangers.

Imbalanced Data and Data sparsity

As can be seen in Figure 4.1, the dataset is extremely skewed with very few positive examples (29.5 on average[1]) in contrast to a large number of negative examples (9310.5 on average). In

[1]The data in 2017-2018 is not complete and is excluded from the computation of average.

Figure 4.1: Yearwise number examples of each type



addition, only a small portion of the area is patrolled every year, leading to a very larger set of unlabeled data (44493.25 on average).

Noisy Data

Detecting snares requires experience and in many cases, the snares cannot be found easily, given the tree branches in the scene, especially when one is walking. As a result, the patrollers might have simply missed the snares in certain regions while patrolling and hence a lot of the negative data points might indeed be positive. Therefore, we expect noise in the negative labels in our dataset.

Unstructured Data

Based on geographical data, historical patrol records in 2013-2017, and recent records of 2017-2018 winter season patrol in HNHR, we divide the area into a 1km grid and construct a dataset where each data point corresponds to a grid cell in a patrol season. The label for each data point indicates whether or not there were any poaching activities in the grid cell in that patrol season. The features of each data point include: the distance from each area to the closest stream, village, patrol post, river, marsh, village road, provincial road, national road, highway, conservation boundary; land type, the elevation and slope of each area; patrol length (total distance patrolled in the grid) in last patrol season.

Unlike images/text/speech etc., these features are pretty hard to interpret for a data scientist. Hence, adding explicit priors on the data can be hard and requires conversing with the rangers themselves.

Dataset shift

After every season, the poachers adapt their strategy depending on which of their previous traps were found by the rangers. Thus, the poachers place the traps in different regions for the next season using their adapted strategy. So, the prediction model has to additionally take into account the distributional shift that takes place every season.

4.2 Previous Literature

There has been some previous work focusing on understanding and predicting poaching activities. [53] and [67] analyze the physical environment and find correlations between certain features and poaching incidents. More recently, machine learning based approaches have been explored to predict poaching. [55] uses a Dynamic Bayesian Network that explicitly models the dependencies between occurrence and detection of poaching activities, as well as the temporal pattern of poaching. [33] designs an ensemble of decision trees which incorporate spatial correlation of poaching to account for the undetected instances. [21] provides a hybrid model that combines decision trees and Markov Random Fields [21] to exploit the spatio-temporal correlation of poaching activities. [22] proposes to weigh the negative data points in the training set based on patrol effort so as to account for the label uncertainty. However, the challenge of having limited data is not fully resolved and human knowledge is only used in very limited ways in previous works such as to select features to be considered and to represent the dependency and correlation relationships. In addition to wildlife poaching, predicting other types of crimes based on real-world data has been studied using general principles such as "crime predicts crime" in criminology[32, 71]. However, most of these rely on a sufficiently large dataset and ignore the fact that there are undetected or unreported crime instances. Thus, new methods for exploiting expert knowledge and using these implicit information in the data is needed to efficiently handle cases with limited real-world data.

4.3 Methods

Despite the effort made towards addressing these challenges, there has been limited success towards solving the challenges mentioned above purely using machine learning techniques and hand designed priors. Thus, we use a different strategy. We recognized the fact that the conservation site managers and rangers have extensive experience in the field and have been interacting with poachers for years. Thus the information they can provide is critical to building a better understanding of the poachers. Hence, we approach these experts for more domain specific knowledge rather than trying to guess more priors by ourselves. However, extracting useful/relevant information from the experts is tricky and requires intelligently designed questions. Specifically, we test 2 approaches to elicit and exploit human knowledge described in Section **??** and **??**. However, in doing so, we need to make sure that we don't overburden the experts.

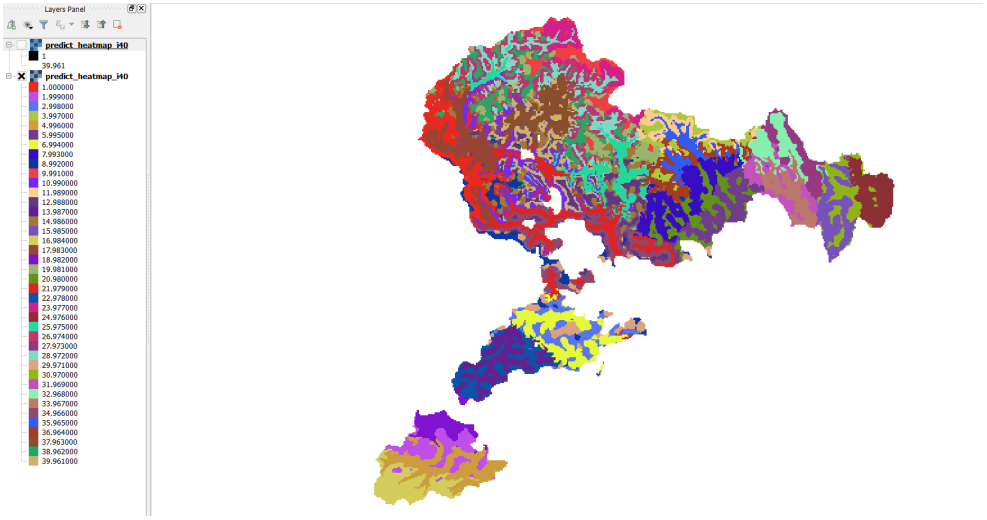
At the same time, in order to get a reasonable baseline to build upon, we also use standard machine learning techniques to balance the data and use the unlabeled data for dynamic negative sampling.

4.3.1 Eliciting Information from Domain Experts

Extracting coarse/weak supervision

Given the peculiarities of the dataset, it is very important to elicit some explicit or implicit information from domain experts so that we can make use of the unlabeled data as well as the noisy labeled data. It should be noted that several factors need to be kept in mind while collecting

Figure 4.2: Visualization of the 40 clusters



domain knowledge from the experts. Firstly, it is not possible for the experts to give very accurate and fine-grained information (e.g, the specific probabilities for every region). Second, the experts cannot be expected to label a huge amount of data given the limited amount of resources and time they have. Third, it should be expected that the information provided by the experts can be noisy. Hence, we have to settle for a limited amount of information which is coarse-grained and noisy. But at the same time, we need to ensure that we are able to extract enough information to tackle the bottlenecks in the machine learning models. With these in mind, we develop the following method to elicit information from the domain experts.

Recognizing the fact that it is difficult for the experts to provide estimates of poaching threat level for every single grid cell in the conservation area, and that fine-grained labels provided by the experts can be very noisy, we propose a method to obtain information at a coarse level from the experts. Instead of asking the experts to score each individual cell, we first group these cells into clusters using K-means clustering in the feature space. We then present these clusters to the experts and ask them to provide a score for each cluster from 1 to 10, where 1 corresponds to minimum threat level and 10 corresponds to maximum threat level. To decide the number of clusters to be used in this procedure, we asked the domain experts for feedback. They believed that $40 - 50$ clusters would be a reasonable number to ensure consistency across the labeling on a given set of clusters. Following their advice, we repeat this procedure twice, first with 40 clusters and then with 50 clusters. Doing it twice helps account for any inconsistency in the scores provided by the experts. This gives us two sets of clusters and their corresponding scores. This approach helps us extract useful information about the threat level without causing much cognitive burden on the experts. Specifically, now the experts only need to provide 90 scores rather than about 75000 scores, one for each grid cell. See Figure 4.2 for visualizations of the 40 clusters in the map.

Constructing Aggregated Score : Based on the two set of scores provided by the expert, we compute the aggregated score as an indicator of the threat level of a grid cell. The experts provide us with a score (from 1 to 10) for each cluster for each questionnaire. But we have these

Table 4.1: Feature Ranges provided by Domain experts

Snaring Threat	dist-village	dist-patrol	dist-river
low	[0,0.3]	[0,0.03]	[0.1,1]
high	[0.3,1]	[0.03,1]	[0,0.1]

confidence score for each cluster set individually. In order to combine these two sets of cluster scores, we propose a simple approach. If a grid cell k belongs to a cluster $C_{40}(k)$ with score $s_{40}(C_{40}(k))$ when 40 clusters are used and belongs to a cluster $C_{50}(k)$ with score $s_{50}(C_{50}(k))$ when 50 clusters are used, we define the aggregated score as the minimum of the two, i.e.,

$$s(k) = \min\{s_{40}(C_{40}(k)), s_{50}(C_{50}(k))\}$$

In other words, we assign a high score to a data point only if it received a high score in both cluster sets.

Feature priors

We plotted the histograms for individual features and found that in some of them there was a clear difference between positive and negative samples. The histograms for dist-village and dist-river can be seen in Figure 4.3. Hence, we asked the experts to provide specific ranges for some of the features which they think have high correlation with the labels. We also asked for their confidence on the specific ranges provided (in line with the 4-point estimates [75] to elicit distributions). However, the experts commented that providing confidence for each feature was difficult since multiple other factors need to be taken into account. Thus they provided only the ranges for 11 features. A sample of 3 of those features has been shown in Table 4.1 for illustration purposes. We further prune the features whose ranges do not correlate well with data. After the pruning, we were left with ranges for 4 features namely, dist-patrol, dist-village, elevation and slope.

Given the feature ranges for which the probability $P(Y = 1|X)$ is high, we aim to unlabeled data points which are more likely to have positive labels. We assume that, given feature X_i and corresponding feature range, F_i, the probability, $P(Y = 1|X_i \in F_i) = p$ and $P(Y = 1|X_i \notin F_i) = q$, for any the features. We can apply Bayes rule, and use the Naive Bayes assumption on the features. Simplifying it we get,

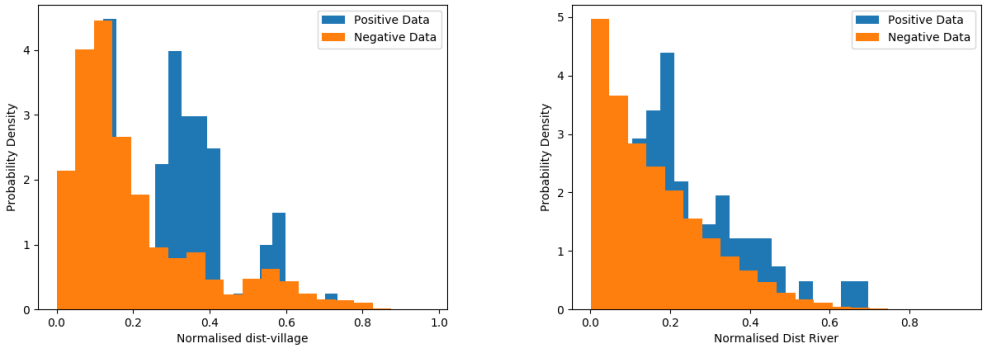
$$P(Y = 1|X^{(j)}) = \frac{\prod_{i=1}^{n} P(Y = 1|X_i^{(j)})}{P(Y = 1)^{n-1}} \tag{4.1}$$

where n is the total number of features we consider. The denominator can be replaced with a normalization constant Z. If m out of the n features fall inside the specified ranges provided, the equation above can be simplified to

$$P(Y = 1|X^{(j)}) = \frac{p^m q^{n-m}}{Z} \tag{4.2}$$

We use these probabilities to then sample from the unlabeled dataset and augment the positive examples in the dataset. This is described in more detail below.

Figure 4.3: Histograms for dist-village and dist-river



4.3.2 Data Augmentation

In order to tackle the unique properties of the dataset we use three ways to perform data augmentation. *(1) Positive Sampling (PS)* Since the number of positive examples are very low compared to the negative examples, we choose to duplicate the positive examples to balance the dataset during training. *(2) Negative Sampling (NS)* As established previously we know that most of the unlabeled examples are low threat regions, since the experts chose not to explore those regions. Hence we add a random partition of the unlabeled set into the negative set during training. It is important to note here that unlabeled data points have never been patrolled and are thus year agnostic. Thus we only add the one data point per grid when sampling unlabeled data points instead of adding a sample for each year. *(3) Score based Positive Sampling (PS)* Although the information collected from domain experts as described in Section 4.3.1 is coarse and noisy, it can be incorporated into the machine learning model in a variety of ways. In this work, we propose to use the aggregated score or the probabilities to add the unlabeled data points that are likely to have positive labels to the positive dataset. When using the cluster based scores, we add all unlabeled data points whose corresponding grid cell has aggregated scores greater than or equal to 6 to the positive dataset. In the case of feature ranges, we just sample the unlabeled examples based on the probabilities computed in 4.2 to add to the positive dataset. Similar to NS, we add only one sample per grid when sampling from unlabeled data points instead of adding an entry for each year.

4.3.3 Model Implementations

Bagging Ensemble Decision Tree We use bagging ensemble decision tree [6] with 1000 trees where each base tree is trained using only 10 percent of the total training data. We use entropy to compute the information gain at each node. We use the implementation provided scikit-learn with the above mentioned parameters to train the model.

Neural Networks We also use a three-layer feedforward neural network [40] with 8 neurons on the first layer , 4 neurons on the second layers and a single neuron in the last layer spitting out the threat probability for that data point. We use relu nonlinearity in the first and second layers

and a sigmoid at the output. To predict the final output we use an ensemble of 100 such neural networks.

4.4 Evaluation

4.4.1 Metrics

We evaluate our model performance using 4 metrics precision, recall, F1 score and the ll score. We choose to report the ll score along with the precision, recall and F1 scores because it offers better discriminability since it's not bounded between $[0, 1]$. We do not include the more commonly used AUC ROC score because we observed that it was sensitive to false positives and gave a high score even when the number of false positives were high.

$$llscore = \frac{Recall * TestSetSize}{TruePositives + FalsePositives}$$

4.4.2 Evaluation on Dataset

Given the limited number of positive samples in the dataset, it is infeasible to separate the dataset into a test set and a training set without getting a biased evaluation of our model. Hence, we perform 4-fold cross validation to train and test our model performance multiple times and average the results across all the runs. The dataset here includes the entire data collected between 2013 to 2018. Table 4.2 contains the precision, recall, F1 and the ll scores for the model and multiple baselines. In the experiments listed in the Table 4.2, DD indicates Data Duplication. We find that data duplication is very crucial. The model completely fails (predicts negative labels for every example) if we remove this component. We test a another data oversampling technique called SMOTE: Synthetic Minority Over-sampling Technique [8] to compare against data duplication. We observe that this does not help with our dataset. For DT, we also observe that the Positive Sampling (PS) when added standalone significantly deteriorates the precision since it leads to an increase in the false positive rate. Adding Negative sampling (NS) standalone does not cause much benefit either. But adding both positive and negative sampling together leads to a boost in performance. In fact, a combination of NS and PS results in performance improvement in the neural network as well. This shows that expert knowledge can boost the performance of both the machine learning models even if their relative performances are very different. We observe that the neural network has a very high false positive rate even after training with Negative sampling resulting in poor performance overall. We also include the scores computed when using a random classifier which labels any example as positive with probability 0.5 to give the readers a sense of baseline values for each of the scores.

4.4.3 Feature priors

In order to estimate the probability of being positive for each unlabeled data point, in Eq. 4.2 we use $p = 0.04$ and $q = 0.01$ as advised by the experts. We pick 1000 samples that have the highest probability computed through Equation 4.2 and add them to the Positive dataset. We refer to this data augmentation approach as Positive sampling using feature ranges (PSFR) and

Table 4.2: Scores for our model and contribution of each component

Models	LL score	Recall	Precision	F1 score
Random decisions	0.51	0.5	0.004	0.008
DT	0.0	nan	0.0	0.000
DT with DD	14.60	0.31	0.17	0.219
DT with SMOTE	11.19	0.35	0.12	0.179
DT with DD, PS	4.99	0.35	0.05	0.087
DT with DD and NS	14.05	0.27	0.19	0.223
DT with DD, NS, PS	15.42	0.31	0.18	0.227
NN	0.0	nan	0.0	0.000
NN with DD	3.26	0.72	0.016	0.031
NN with DD, NS	2.47	0.48	0.02	0.038
NN with DD, NS, PS	3.70	0.79	0.02	0.039

Table 4.3: Scores when positive sampling using feature ranges

Models	LL score	Recall	Precision	F1
DT with DD	14.60	0.31	0.17	0.219
DT with DD, PSFR	6.87	0.34	0.07	0.116
DT with DD and NS	14.05	0.27	0.19	0.223
DT with DD, NS, PSFR	10.88	0.29	0.14	0.189

report the scores in Table 4.3. We observe that using these features brings down the performance of the model even after feature pruning. This shows how sensitive the performance is to positive sampling. Hence positive sampling in such cases needs to be done with utmost care.

4.4.4 Field Tests

The predictions of poaching activities made based on *DT with DD and NS* trained on 2013-2017 dataset has been used to guide two sets of field tests. In October 2017, a two-day field test was conducted in HNHR. The rangers selected two patrol routes that covered areas which had not been frequently patrolled but were predicted to have poaching activities by our model. During the field test, 22 snares were found. During November 2017 to February 2018, a set of 34 patrol shifts were undertaken (each taking an avg of 2.85 hours). During these patrols, 7 snares were found. However, rangers mentioned that that the low number of findings during these patrols is mainly due to the reduced tolerance to poaching in China this year, as can be seen from a set of changes in policy [5]. We expect to run larger scale tests with models that incorporates human

knowledge elicited through the clustering-based questionnaires and data from conservation sites in Northeastern China in the next patrol season.

4.5 Discussion

In this chapter, we explored a weird dataset/domain that has very different properties from the ones seen in the last 2 chapter. We observed that in this dataset, it was difficult to simply rely on the data that was already provided or rely on intuitions to add priors to the machine learning model. Thus, we had to take an extra step and find innovative ways of eliciting specialized human knowledge from the experts to enhance the predictive analysis in wildlife poaching. We designed questionnaires to elicit information from domain experts. The information is then used to estimate a threat level of each data point and augment the dataset for training. However, we find that the performance is very sensitive to positive data augmentation. Hence incorporating the expert knowledge to augment the positive data needs to be done carefully. Biases in the expert supervision can affect the performance adversely in certain cases as shown in the Alternative Domain Knowledge section. We demonstrate a more principled method of collecting this prior knowledge by clustering the data points and getting weak labels over the clusters. It is important to note that our approach is fairly generic and can be used in a variety of other settings, where the expert knowledge is costly and a large portion of the data is unlabeled. All results in this section have already been published in COMPASS 2018 conference.

In the next chapter, we'll look at another very different domain of reinforcement learning but continue on this idea of data augmentation. In the RL setting, an agent is tasked with exploring and collecting its own data to perform a specific task, i.e, based on the past experience the agent itself chooses the new data to train on.

Chapter 5

Exploration in Meta-RL

5.1 Problem Statement

Reinforcement learning (RL) approaches have seen many successes in recent years, from mastering the complex game of Go [72] to even discovering molecules [57]. However, a common limitation of these methods is their propensity to overfitting on a single task and inability to adapt to even slightly perturbed configuration [92]. Meta reinforcement learning addresses these shortcomings by learning the inductive biases and heuristics required (to quickly adapt to the new task) from the data itself. These inductive biases or heuristics can be induced in the model in various ways like optimization algorithm, policy, hyperparameters, network architecture, loss function, exploration strategies [68], *etc*. Recently, a class of parameter initialization based meta-learning approaches have gained attention like Model Agnostic Meta-Learning (MAML) [18]. MAML finds a good initialization for a model or a policy that can be adapted to a new task by fine-tuning with policy gradient updates from a few samples of that task. However, we will see in this chapter, that even meta-learning algorithms benefit from the addition of priors.

5.1.1 Challenges

Since the objective of meta-RL algorithms is to adapt to a new task from a few examples, efficient exploration strategies are crucial for quickly finding the optimal policy in a new environment. Some recent works have tried to address this problem by improving the credit assignment of the meta learning objective to the pre-update trajectory distribution [63, 76]. However, that requires transitioning the base policy from exploration behavior to exploitation behavior using one or few policy gradient updates. This limits the applicability of these methods to cases where the post-update (exploitation) policy is similar to the pre-update (exploration) policy and can be obtained with only a few updates. Additionally, for cases where pre- and post-update policies are expected to exhibit different behaviors, large gradient updates may result in training instabilities and poor performance at convergence.

5.1.2 Assumption

To address the above mentioned problem, we propose to explicitly model a separate exploration policy for the task distribution. This means that we would add a strong assumption that the exploration and exploitation policies are completely uncorrelated. However, we would show through our experiments, that this assumption is very useful when it is true and doesn't hurt the performance even when it is not true.

5.2 Previous Literature

Meta-learning algorithms proposed in the RL community include approaches based on recurrent models [15, 17], metric learning [73, 78], and learning optimizers [56]. Recently, Finn et al. [18] proposed Model Agnostic Meta-Learning (MAML) which aims to learn a policy that can generalize to a distribution of tasks. Specifically, it aims to find a good initialization for a policy that can be adapted to any task sampled from the distribution by fine-tuning with policy gradient updates from a few samples of that task.

Efficient exploration strategies are crucial for finding trajectories that can lead to quick adaptation of the policy in a new environment. Recent works [26, 62] have proposed structured exploration strategies using latent variables to perform efficient exploration across successive episodes, however, they did not explicitly incentivize exploration in pre-update episodes. E-MAML [76] made the first attempt at assigning credit for the final expected returns to the pre-update distribution in order to incentivize exploration in each of the pre-update episodes. Rothfuss et al. [63] proposed Proximal Meta-Policy search (ProMP) where they incorporated the causal structure for more efficient credit assignment and proposed a low variance curvature surrogate objective to control the variance of the corresponding policy gradient update. However, these methods make use of a single base policy for both exploration and exploitation while relying on one or few gradient updates to transition from the exploration behavior to exploitation behavior. Over the next few sections, we illustrate that this approach is problematic and insufficient when the exploration and exploitation behaviors are quite different from each other.

A number of prior works have tried to utilize self-supervised objectives [19, 31, 50, 58, 84] to ease learning especially when the reward signal itself is insufficient to provide the required level of feedback. Drawing inspiration from these approaches, we modify the inner loop update/adaptation step in MAML using a self-supervised objective to allow more stable and faster updates. Concurrent to our work, Yang et al. [89] also decoupled exploration and adaptation policies where the latter is initialized as a learnable offset to the exploration policy.

5.3 Background

5.3.1 Meta-Reinforcement Learning

Unlike RL which tries to find an optimal policy for a single task, meta-RL aims to find a policy that can generalize to a distribution of tasks. Each task $\mathcal{T}$ sampled from the distribution $\rho(\mathcal{T})$ corresponds to a different Markov Decision Process (MDP) defined by the tuple $(\mathcal{S}, \mathcal{A}, \boldsymbol{P}, r, \gamma, H)$ with state space $\mathcal{S}$, action space $\mathcal{A}$, transition dynamics $\boldsymbol{P}$, reward function r, discount factor γ,

and time horizon H. These MDPs are assumed to have similar state and action space but might differ in the reward function r or the environment dynamics $\boldsymbol{P}$. The goal of meta RL is to quickly adapt the policy to any task $\mathcal{T} \sim \rho(\mathcal{T})$ with the help of few examples from that task.

5.3.2 Credit Assignment in Meta-RL

MAML is a gradient-based meta-RL algorithm that tries to find a good initialization for a policy which can be adapted to any task $\mathcal{T} \sim \rho(\mathcal{T})$ by fine-tuning with one or more gradient updates using the sampled trajectories of that task. MAML maximizes the following objective function:

$$J(\theta) = \mathcal{T}\sim\rho(\mathcal{T}) \left[{}_{'\sim P_{\mathcal{T}}('|\theta')} \left[R(') \right] \right]$$
$$\text{with} \quad \theta' := U(\theta, \mathcal{T}) = \theta + \alpha \nabla_{\theta \sim P_{\mathcal{T}}(|\theta)} \left[R() \right] \tag{5.1}$$

where U is the update function that performs one policy gradient ascent step to maximize the expected reward $R()$ obtained on the trajectories sampled from task $\mathcal{T}$.

Rothfuss et al. [63] showed that the gradient of the objective function $J(\theta)$ in Eq. 5.1 can be written as:

$$\nabla_\theta J(\theta) = \mathcal{T}\sim\rho(\mathcal{T}) \left[{}_{\sim P_{\mathcal{T}}(|\theta)' \sim P_{\mathcal{T}}('|\theta')} \left[\nabla_\theta J_{\text{post}}(,') + \nabla_\theta J_{\text{pre}}(,') \right] \right]$$

where,

$$\nabla_\theta J_{\text{post}}(,') = \underbrace{\nabla_{\theta'} \log \pi_\theta(') R(')}_{\nabla_{\theta'} J^{\text{outer}}} \underbrace{\left(I + \alpha R() \nabla^2_\theta \log \pi_\theta() \right)}_{\text{transformation from } \theta' \text{ to } \theta} \tag{5.2}$$

$$\nabla_\theta J_{\text{pre}}(,') = \alpha \nabla_\theta \log \pi_\theta() \left(\underbrace{(\nabla_\theta \log \pi_\theta() R())^\top}_{\nabla_\theta J^{\text{inner}}} \underbrace{(\nabla_{\theta'} \log \pi_{\theta'}(') R('))}_{\nabla_{\theta'} J^{\text{outer}}} \right) \tag{5.3}$$

The first term $\nabla_\theta J_{\text{post}}(,')$ corresponds to a policy gradient step on the post-update policy $\pi_{\theta'}$ w.r.t.. the post-update parameters θ' which is then followed by a linear transformation from θ' to θ (pre-update parameters). Note that $\nabla_\theta J_{\text{post}}(,')$ optimizes θ to increase the likelihood of the trajectories $'$ that lead to higher returns given some trajectories . However, this term does not optimize θ to yield trajectories that lead to good adaptation steps. That is, infact, done by the second term $\nabla_\theta J_{\text{pre}}(,')$. It optimizes for the pre-update trajectory distribution, $P_{\mathcal{T}}(|\theta)$, i.e, increases the likelihood of trajectories that lead to good adaptation steps.

During optimization, MAML only considers $J_{\text{post}}(,')$ and ignores $J_{\text{pre}}(,')$. Thus MAML finds a policy that adapts quickly to a task given relevant experiences, however, the policy is not optimized to gather useful experiences from the environment that can lead to fast adaptation.

Rothfuss et al. [63] proposed ProMP where they analyze this issue with MAML and incorporate $\nabla_\theta J_{\text{pre}}(,')$ term in the update as well. They used The Infinitely Differentiable Monte-Carlo Estimator (DICE) [20] to allow causal credit assignment on the pre-update trajectory distribution, however, the gradients computed by DICE still have high variance. To remedy this, they proposed a low variance (and slightly biased) approximation of the DICE based loss that leads to stable updates.

5.4 Method

The pre-update and post-update policies are often expected to exhibit very different behaviors, i.e, exploration and exploitation behaviors respectively. For instance, consider a 2D environment where a task corresponds to reaching a goal location sampled randomly from a semi-circular region (example shown in appendix). The agent receives a reward only if it lies in some vicinity of the goal location. The optimal pre-update or exploration policy is to move around in the semi-circular region whereas the ideal post-update or exploitation policy will be to reach the goal state as fast as possible once the goal region is discovered. Clearly, the two policies are expected to behave very differently. In such cases, transitioning a single policy from pure exploration phase to pure exploitation phase via policy gradient updates will require multiple steps. Unfortunately, this significantly increases the computational and memory complexities of the algorithm. Furthermore, it may not even be possible to achieve this transition via few gradient updates. This raises an important question: DO WE REALLY NEED TO USE THE PRE-UPDATE POLICY FOR EXPLORATION AS WELL? CAN WE USE A SEPARATE POLICY FOR EXPLORATION?

Using separate policies for pre-update and post-update sampling: The straightforward solution to the above problem is to use a separate exploration policy μ_ϕ responsible for collecting trajectories for the inner loop updates to get θ'. Following that, the post-update policy $\pi_{\theta'}$ can be used to collect trajectories for performing the outer loop updates. Unfortunately, this is not as simple as it sounds. To understand this, let's look at the inner loop updates:

$$U(\theta, \mathcal{T}) = \theta + \alpha \nabla_{\theta \sim P_{\mathcal{T}}(|\theta)} \left[R() \right]$$

When the exploration policy is used for sampling trajectories, we need to perform importance sampling. The update would thus become:

$$U(\theta, \mathcal{T}) = \theta + \alpha \nabla_{\theta \sim Q_{\mathcal{T}}(|\phi)} \left[\frac{P_{\mathcal{T}}(|\theta)}{Q_{\mathcal{T}}(|\phi)} R() \right]$$

where $P_{\mathcal{T}}(|\theta)$ and $Q_{\mathcal{T}}(|\phi)$ represent the trajectory distribution sampled by π_θ and μ_ϕ respectively. Note that the above update is an off-policy update which results in high variance estimates when the two trajectory distributions are quite different from each other. This makes it infeasible to use the importance sampling update in the current form. In fact, this is a more general problem that arises even in the on-policy regime. The policy gradient updates in the inner loop results in both $\nabla_\theta J_{\text{pre}}$ and $\nabla_\theta J_{\text{post}}$ terms being high variance. This stems from the mis-alignment of the outer gradients ($\nabla_{\theta'} J^{outer}$) and the inner gradient, hessian $\left(\nabla_\theta J^{\text{inner}}, \nabla^2_\theta \log \pi_\theta()\right)$ terms appearing in Eq. 5.2 and 5.3. This motivates our second question: DO WE REALLY NEED THE PRE-UPDATE GRADIENTS TO BE POLICY GRADIENTS? CAN WE USE A DIFFERENT OBJECTIVE IN THE INNER LOOP TO GET MORE STABLE UPDATES?

Using a self-supervised/supervised objective for the inner loop update step: The instability in the inner loop updates arises due to the high variance nature of the policy gradient update. Note that the objective of inner loop update is to provide some task specific information to the agent with the help of which it can adapt its behavior in the new environment. We believe that this could be achieved using some form of self-supervised or supervised learning objective in place of policy gradient in the inner loop to ensure that the updates are more stable. We propose to use a network for predicting some task (or MDP) specific property like reward function, expected return or value. During the inner loop update, the network updates its parameters by minimizing its prediction error on the given task. Unlike prior meta-RL works where the task

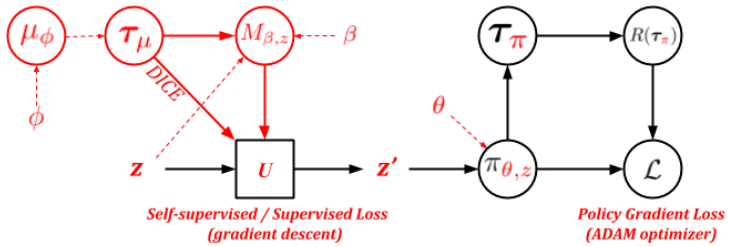


Figure 5.1: Model Flowchart: Black structures are those consistent with E-MAML/ProMP. Red structures are the key differences with E-MAML/ProMP (The thin-dotted arrow means the parameters related to that node.). Specifically, the pre-update trajectories are now collected using a separate exploration policy μ_ϕ. The corresponding adaptation update is performed using a self-supervised/supervised objective, $\left(M_{\beta,z}(s,a) - \overline{M}(s,a)\right)^2$, on z to give z' and the policy $\pi_{\theta,z'}$ is parameterized using the task specific parameters z' and the task agnostic parameters θ

adaptation in the inner loop is done by policy gradient updates, here, we update some parameters shared with the exploitation policy using a supervised loss objective function which leads to stable updates during the adaptation phase. However, note that the variance and usefulness of the update depends heavily on the choice of the self-supervision/supervision objective. We delve into this in more detail in Section 5.4.1.

5.4.1 Model

Our proposed model comprises of three modules, the exploration policy $\mu_\phi(s)$, the exploitation policy $\pi_{\theta,z}(s)$, and the self-supervision network $M_{\beta,z}(s,a)$. Note that $M_{\beta,z}$ and $\pi_{\theta,z}$ share a set of parameters z while containing their own set of parameters β and θ respectively. We describe our proposed model in Fig. 5.1. Our model differs from E-MAML/ProMP because of the separate exploration policy, the separation of task-specific parameters z and task agnostic parameters θ, and the self-supervised update as shown in Fig. 5.1.

The agent first collects a set of trajectories using its exploration policy μ_ϕ for each task $\mathcal{T} \sim \rho(\mathcal{T})$. It then updates the shared parameter z by minimizing the regression loss $\left(M_{\beta,z}(s,a) - \overline{M}(s,a)\right)^2$ on the sampled trajectories :

$$z' = U(z, \mathcal{T}) = z - \alpha \nabla_{z \sim Q_\mathcal{T}(|\phi)} \left[\sum_{t=0}^{H-1} \left(M_{\beta,z}(s_t,a_t) - \overline{M}(s_t,a_t)\right)^2 \right] \tag{5.4}$$

where, $\overline{M}(s,a)$ is the target, which can be any task specific quantity like reward, return, value, next state etc. After obtaining the updated parameters z' for each task $\mathcal{T}$, the agent samples the (validation) trajectories $'$ using its updated exploitation policy $\pi_{\theta,z'}$. Effectively, z' encodes the necessary information regarding the task that helps an agent in adapting its behavior to maximize its expected return whereas θ remain task agnostic. A similar approach was proposed by Zintgraf et al. [94] to learn task-specific behavior using context variable with MAML.

The collected trajectories are then used to perform a policy gradient update to all parameters z, θ, ϕ and β using the following objective:

$$J(z', \theta) = _{\mathcal{T} \sim \rho(\mathcal{T})} \left[_{\frac{\mathcal{T}}{\pi} \sim P_\mathcal{T}(\frac{\mathcal{T}}{\pi}|\theta,z')} \left[R(\frac{\mathcal{T}}{\pi}) \right] \right] \tag{5.5}$$

In order to allow multiple outer-loop updates, we use the PPO [66] objective instead of the vanilla policy gradient objective to maximize Eq. 5.5. Furthermore, we don't perform any outer loop updates on z and treat it as a shared latent variable with fixed initial values of 0 as proposed in [94]. The reason being, that the bias term in the layers connecting z to the respective networks would learn to compensate for the initialization. We only update z to z' in the inner loop to obtain a task specific latent variable.

Note that in all the prior meta reinforcement learning algorithms, both the inner-loop update and the outer-loop update are policy gradient updates. In contrast, in this work, the inner-loop update is a supervised learning gradient descent update whereas the outer-loop update remains a policy gradient update.

The outer loop gradients w.r.t. ϕ, $\nabla_\phi J(z', \theta)$ can be simplified by multiplying the DICE [20] operator inside the expectation in Eq. 5.4 as proposed by Rothfuss et al. [63]. This allows the gradients w.r.t. ϕ to be computed in a straightforward manner with back-propagation. This also eliminates the need to apply the policy gradient trick to expand Eq. 5.4 for gradient computation. The inner loop update then becomes:

$$z' = U(z, \mathcal{T}) = z - \alpha \nabla_{z \sim Q_\mathcal{T}(|\phi)} \left[\sum_{t=0}^{H-1} \left(\prod_{t'=0}^{t} \frac{\mu_\phi(a_{t'}|s_{t'})}{\perp(\mu_\phi(a_{t'}|s_{t'}))} \right) \left(M_{\beta,z}(s_t, a_t) - \overline{M}(s_t, a_t) \right)^2 \right]$$

where $\perp$ is the stop gradient operator as introduced in [20].

The pseudo-code of our algorithm is shown in appendix (see algorithm 3). However, we found that implementing algorithm 3 as it is, using DICE leads to high variance gradients for ϕ, resulting in instability during training and poor performance of the learned model. To understand this, let's look at the vanilla DICE gradients for the exploration parameters ϕ, which can be written as follows:

$$\nabla_\phi J(z', \theta) =_{\mathcal{T} \sim \rho(\mathcal{T})} \left[\sum_{\sim Q_\mathcal{T}(|\phi)}^{H-1} \alpha \nabla_\phi \log \mu_\phi(s_t) \left[\sum_{t'=t}^{H-1} \left(\sum_{' \sim P_\mathcal{T}(' | \theta, z')} \left(\nabla_{z'} \log \pi_{\theta, z'}(') R(') \right)^\top \right) \left(\nabla_z \left(M_{\beta,z}(s_t, a_t) - \overline{M}(s_t, a_t) \right)^2 \right) \right] \right]$$

The above expression can be viewed as a policy gradient update:

$$\nabla_\phi J(z', \theta) =_{\mathcal{T} \sim \rho(\mathcal{T})} \left[\sum_{\sim Q_\mathcal{T}(|\phi)}^{H-1} \alpha \nabla_\phi \log \mu_\phi(s_t) R_t^\mu \right] \tag{5.6}$$

with returns

$$R_t^\mu = \left[\sum_{t'=t}^{H-1} \left(\sum_{' \sim P_\mathcal{T}(' | \theta, z')} \left(\nabla_{z'} \log \pi_{\theta, z'}(') R(') \right)^\top \right) \left(\nabla_z \left(M_{\beta,z}(s_t, a_t) - \overline{M}(s_t, a_t) \right)^2 \right) \right] \tag{5.7}$$

Note that the variance depends on the policy gradient terms computed in the outer-loop and the choice of self-supervision. We'll explain the latter in Sec 5.4.1. However, irrespective of the choice, we can use value function based variance reduction ([52]) by substituting the above computed returns with advantages, i.e, we replace the return R_t^μ in Eq. 5.7 with an advantage estimate A_t^μ and use a PPO ([66]) objective to allow multiple outer loop updates: :

$$\nabla_\phi \hat{J}(z', \theta) =_{\mathcal{T}\sim\rho(\mathcal{T})} \Bigg[_{\sim Q_\mathcal{T}(|\phi_o)} \Bigg[\sum_{t=0}^{H-1} \alpha \nabla_\phi \min\left(\frac{\mu_\phi(s_t)}{\mu_{\phi_o}(s_t)} A_t^\mu \ , \ \text{clip}_{1-\epsilon}^{1+\epsilon}\left(\frac{\mu_\phi(s_t)}{\mu_{\phi_o}(s_t)}\right) A_t^\mu\right)\Bigg]$$

where,

$$A_t^\mu = r_t^\mu + V_{t+1}^\mu - V_t^\mu \tag{5.8}$$

where V_t^μ is computed using a neural network or a linear feature baseline [14] fitted on the returns R_t^μ. where r_t^μ is given by:

$$r_t^\mu =\Big(_{'\sim P_\mathcal{T}('|\theta,z')} \left(\nabla_{z'} \log \pi_{\theta,z'}(')R(')\right)^\top\Big) \left(\nabla_z \left(M_{\beta,z}(s_t, a_t) - \overline{M}(s_t, a_t)\right)^2\right) \tag{5.9}$$

Self-Supervised/Supervised Objective

It is important to note that the self-supervised/supervised learning objective not only guides the adaptation step but also influences the exploration policy update as seen in Eq. 5.6 and 5.7. We mentioned above that the self-supervised/supervised learning objective could be as simple as a value/reward/return/next state prediction for each state (state-action pair). However, the exact choice of the objective can be critical to the final performance and stability. From the perspective of the adaptation step, the only criterion is that the self-supervised objective should contain enough task specific information to allow a useful adaptation step. For example, it would not be a good idea to use the rewards self-supervision in sparse/noisy reward scenarios or the next state predictions as self-supervision when the dynamics model does not change much over tasks since the self-supervision updates in such cases will not carry enough task specific information. From the perspective of the exploration policy updates, an additional requirement would be to ensure that the returns computed in Eq. 5.7 are low variance and unbiased, which further translates to saying $\nabla_z \left(M_{\beta,z}(s_t, a_t) - \overline{M}(s_t, a_t)\right)^2$ should ideally be low variance and unbiased. For example, using the cumulative future returns as self-supervision might lead to a very high variance update in certain environments. Thus, finding a generalizable self-supervision/supervision objective which satisfies both properties mentioned above across all scenarios is a challenging task.

In our experiments, we found that, using N-step return prediction for supervision works reasonably well across all the environments. This acts as a trade-off between predicting the full return (which was high variance but also more task-specific info) and the reward (which was lower variance but lower task-specific info). Hence, $\overline{M}(s_t, a_t)$ becomes $\overline{M}(s_t, a_t, s_{t+1}, a_{t+1}, ...s_{t+N-1}, a_{t+N-1}) = \sum_{t'=t}^{t+N-1} r(s'_t, a'_t)$. However, using $M_{\beta,z}$ to directly predict $\overline{M}$ would still lead to high variance in $\nabla_z \left(M_{\beta,z}(s_t, a_t) - \overline{M}\right)^2$. Thus, we use the truncated N-step successor representations [39] (similar to N-step returns) $m_\beta(s_t, a_t)$ and a linear layer on top of that to compute $M_{\beta,z}(s_t, a_t) = w_\beta^T m_\beta(s_t, a_t)$. Using the successor representations can effectively be seen as using a more accurate/powerful baseline than directly predicting the N-step returns using the (s_t, a_t) pair.

5.5 Experiments

We evaluate our proposed model on a set of 6 benchmark continuous control environments, `Ant-Fwd-Bwd`, `Half-Cheetah-Fwd-Bwd`, `Half-Cheetah-Vel`, `Walker2D-Fwd-Bwd`, `Walker2D-Rand-Params` and `Hopper-Rand-Params` used in [63]. We also compare our

method with 3 baseline approaches: MAML, EMAML and ProMP. Furthermore, we also perform ablation experiments to analyze different components and design choices of our model on a toy 2D point environment proposed by [63].

The details of the network architecture and the hyperparameters used for learning have been mentioned in the appendix. We would like to state that we have not performed much hyperparameter tuning due to computational constraints and we expect the results of our method to show further improvements with further tuning. Also, we restrict ourselves to a single adaptation step in all environments for the baselines as well as our method, but it can be easily extended to multiple gradient steps as well by conditioning the exploration policy on the latent parameters z.

The results of the baselines for the benchmark environments have been borrowed directly from the the official ProMP website [1]. For the point environments, we have used their official implementation[2].

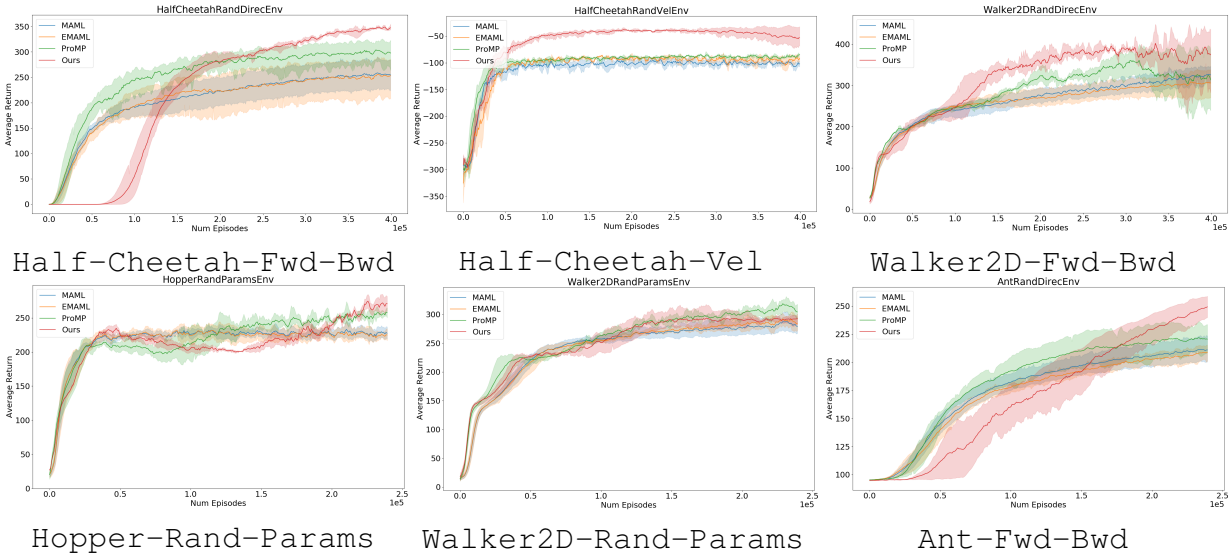


Figure 5.2: Comparison of our method with 3 baseline methods on the Meta-RL Benchmark tasks.

5.5.1 Meta RL Benchmark Continuous Control Tasks.

The continuous control tasks require adaptation either across reward functions (`Ant-Fwd-Bwd`, `Half-Cheetah-Fwd-Bwd`, `Half-Cheetah-Vel`, `Walker2D-Fwd-Bwd`) or across dynamics (`Walker2D-Rand-Params` and `Hopper-Rand-Params`). We set the horizon length to be 100 in `Ant-Fwd-Bwd` and `Half-Cheetah-Fwd-Bwd` environments and 200 in others in accordance with the practice in [63]. The performance plots for all the 4 algorithms are shown in Fig. 5.2. In all the environments, our proposed method outperforms or achieves similar performance to other method in terms of asymptotic performance.

Our algorithm performs particularly well in `Half-Cheetah-Fwd-Bwd`, `Half-Cheetah-Vel`, `Walker2D-Fwd-Bwd` and `Ant-Fwd-Bwd` environments where the N-step returns are informative. In `Ant-Fwd-Bwd` and `Half-Cheetah-Fwd-Bwd` environments, although we reach

[1] `https://sites.google.com/view/pro-mp/experiments`
[2] `https://github.com/jonasrothfuss/ProMP`

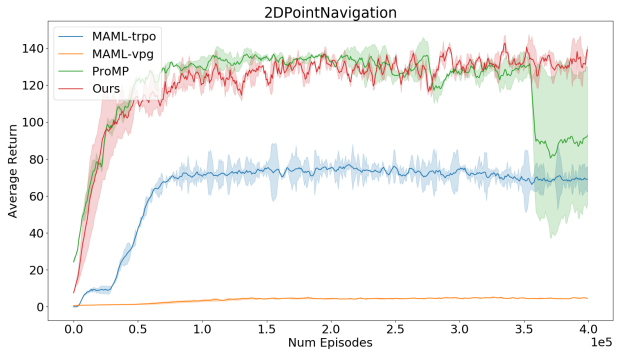


Figure 5.3: 2D Point Navigation

similar asymptotic performance as ProMP, the convergence is slower in the initial stages of training. This is because training multiple networks together can make training slower especially in the initial stages of training especially when the training signal isn't strong enough. Note that in `Walker2D-Rand-Params` and `Hopper-Rand-Params` environments, although our method converges as well as the baselines, it doesn't do much better in terms of peak performance. This could be attributed to the selection of the self-supervision signal. A more appropriate self-supervision signal for these environments would be the next state or successor state prediction since the task distribution in these environments corresponds to the variation in model dynamics and not just reward function. This shows that the choice of the self-supervision signal plays an important role in the model's performance. To further understand these design choices we perform some ablations on a toy environment in section 5.5.2.

5.5.2 2D Point Navigation.

We show the performance plots for ProMP, MAML-TRPO, MAML-VPG and our algorithm in the sparse reward `2DPointEnvCorner` environment (proposed in [63]) in Fig. 5.2. Each task in this environment corresponds to reaching a randomly sampled goal location (one of the four corners) in a 2D environment. This is a sparse reward task where the agent receives a reward only if it is sufficiently close to the goal location. In this environment, the agent needs to perform efficient exploration and use the sparse reward trajectories to perform stable updates, both of which are salient aspects of our algorithm.

Our method is able to achieve this and reaches peak performance while showing stable behavior. ProMP, on the other hand, also reaches the peak performance but shows more unstable behavior than in the dense reward scenarios, although it manages to reach similar peak performance to our method. The other baselines struggle to do much in this environment since they do not explicitly incentivize exploration for the pre-update policy.

Ablation Study

We perform several ablation experiments to analyze the impact of different components of our algorithm on 2D point navigation task. Fig. 5.4 shows the performance plots for the following 5 different variants of our algorithm:

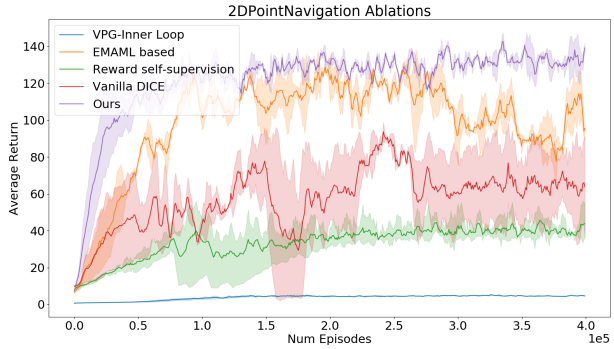


Figure 5.4: Ablation results

VPG-Inner loop: The semi-supervised/supervised loss in the inner loop is replaced with the vanilla policy gradient loss as in MAML while using the exploration policy to sample the pre-update trajectories. This variant illustrates our claim of unstable inner loop updates when naively using an exploration policy. As expected, this model performs poorly due to the high variance off-policy updates in the inner loop.

Reward Self-Supervision : A reward based self-supervised objective is used instead of return based self-supervision, i.e, the self-supervision network M now predicts the reward instead of the N-step return at each time step. This variant is stable but struggles to reach peak performance since the task is sparse reward. This shows that the choice of self-supervision objective is also important and needs to be chosen carefully.

Vanilla DiCE: In this variant, we directly use the DICE gradients to perform updates on ϕ instead of using the low variance gradient estimator. The leads to higher variance updates and unstable training as can be seen from the plots. This shows that the low variance gradient estimate has a major contribution to the stability during training.

E-MAML Based : Here, we used an E-MAML [76] type objective to compute the gradients w.r.t. ϕ instead of using DICE, i.e, directly used policy gradient updates on μ_ϕ but instead with returns computed on post-update trajectories. This variant ignores the causal credit assignment from output to inputs. Thus, the updates are of higher variance, leading to more unstable updates, although it manages to reach good performance.

Ours : The low variance estimate of the DICE gradients is used to compute updates for ϕ along with N-step return based self-supervision for inner loop updates. Our model reaches peak performance and exhibits stable training due to low variance updates.

5.6 Discussion

In this chapter, we discussed about a specific meta-RL algorithm called MAML and demonstrated some shortcomings it has. We then showed that introducing some task specific priors by splitting the exploration and exploitation policies or specifying the self-supervision objective helps improve the learning dynamics of the model and leads to better overall performance.

Chapter 6

Discussion and Conclusion

This thesis presented work on various scenarios that demonstrated that both data and the priors play an important role towards achieving good generalization. In fact, they play a complementary role in most scenarios. In the first example of point cloud completion we saw that the data helped us learn the distribution of full point clouds and the prior knowledge about the task, i.e, incompletion only affects the input distribution, helped us project any incomplete point cloud to the generative model learnt on the distribution of full point clouds. In the second example of visual dialog, we again found that data helped us learn the distribution of natural human dialog but the prior of community regularization and RL based finetuning further helped the model understand how to stay within the distribution of stable interactions. In the third example of poaching threat prediction, we found that sometimes, when the algorithm designers lack enough intuition about the data, it's very hard to design good priors. In such situations, it helps to device intelligent ways to collect more data or prior knowledge from the experts to augment and complement the knowledge in the existing data and the known priors. Finally, in the last example, we showed another example in the meta-RL space, where we found that using distribution specific priors like using specific self-supervision objectives in the inner loop and using separate exploration and exploitation policies is very helpful in obtaining more stable training dynamics and better task performance. Overall, throughout the 4 examples, we saw different ways of adding task specific priors and knowledge into the model to obtain better generalization and final performance.

Future Work

We identify the following areas of interest for future exploration:

1. Although, the meta-learning problem was posed as a framework for learning rich priors for the task distribution, we observed that it struggled as the task distributions became harder. Adding priors to make the framework more generalizable to a more diverse distribution of tasks would be a useful direction of future work. The hope finally is to also handle some distributional shifts in the meta-learning setting.
2. While working on the problems we felt that strong theoretical frameworks for formalizing distributional shift and putting the assumptions into the math was lacking. Thinking of various kinds of distribution shifts and formalizing the corresponding priors and assumptions is an important direction of future work.

Bibliography

[1] Panos Achlioptas, Olga Diamanti, Ioannis Mitliagkas, and Leonidas Guibas. Learning representations and generative models for 3d point clouds. *arXiv preprint arXiv:1707.02392*, 2017. 2.2, 2.3.1, 2.3.1, 0, 2

[2] Claire Adam-Bourdarios, Glen Cowan, Cécile Germain, Isabelle Guyon, Balázs Kégl, and David Rousseau. The higgs boson machine learning challenge. In *NIPS 2014 Workshop on High-energy Physics and Machine Learning*, pages 19–55, 2015. 1

[3] Aishwarya Agrawal, Jiasen Lu, Stanislaw Antol, Margaret Mitchell, C. Lawrence Zitnick, Devi Parikh, and Dhruv Batra. Vqa: Visual question answering. *Int. J. Comput. Vision*, 123(1):4–31, May 2017. ISSN 0920-5691. URL https://doi.org/10.1007/s11263-016-0966-6. 3.2

[4] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *International conference on machine learning*, pages 214–223, 2017. 2.3.1

[5] Rachael Bale. China shuts down its legal ivory trade. https://news.nationalgeographic.com/2017/12/wildlife-watch-china-ivory-ban-goes-into-effect/, 2017. 4.4.4

[6] Leo Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996. 4.3.3

[7] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015. 2.4

[8] N.V. Chawla, K.W. Bowyer, L.O. Hall, and W.P. Kegelmeyer. Smote: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 2002. 4.4.2

[9] T Ciodaro, D Deva, JM De Seixas, and D Damazio. Online particle detection with neural networks based on topological calorimetry information. In *Journal of physics: conference series*, volume 368, page 012030. IOP Publishing, 2012. 1

[10] Angela Dai, Daniel Ritchie, Martin Bokeloh, Scott Reed, Jürgen Sturm, and Matthias Nießner. Scancomplete: Large-scale scene completion and semantic segmentation for 3d scans. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4578–4587, 2018. 2.2

[11] Abhishek Das, Satwik Kottur, Khushi Gupta, Avi Singh, Deshraj Yadav, José M. F. Moura, Devi Parikh, and Dhruv Batra. Visual dialog. *CoRR*, abs/1611.08669, 2016. URL http://arxiv.org/abs/1611.08669. 3.1.1, 3.2, 3.3.2, 3.3.2, 3.1, 3.4.1

[12] Abhishek Das, Satwik Kottur, José M. F. Moura, Stefan Lee, and Dhruv Batra. Learning cooperative visual dialog agents with deep reinforcement learning. *CoRR*, abs/1703.06585, 2017. URL `http://arxiv.org/abs/1703.06585`. 3.1.1, 3.2, 3.3.1, 3.3.1, 3.3.2, 3.1, 3.4.2

[13] Harm de Vries, Florian Strub, Sarath Chandar, Olivier Pietquin, Hugo Larochelle, and Aaron C. Courville. Guesswhat?! visual object discovery through multi-modal dialogue. *CoRR*, abs/1611.08481, 2016. URL `http://arxiv.org/abs/1611.08481`. 3.1.1

[14] Yan Duan, Xi Chen, Rein Houthooft, John Schulman, and Pieter Abbeel. Benchmarking deep reinforcement learning for continuous control. In *International Conference on Machine Learning*, pages 1329–1338, 2016. 5.4.1

[15] Yan Duan, John Schulman, Xi Chen, Peter L Bartlett, Ilya Sutskever, and Pieter Abbeel. Rl^2: Fast reinforcement learning via slow reinforcement learning. *arXiv preprint arXiv:1611.02779*, 2016. 5.2

[16] Clement Farabet, Camille Couprie, Laurent Najman, and Yann LeCun. Learning hierarchical features for scene labeling. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1915–1929, 2012. 1

[17] Chelsea Finn and Sergey Levine. Meta-learning and universality: Deep representations and gradient descent can approximate any learning algorithm. *arXiv preprint arXiv:1710.11622*, 2017. 5.2

[18] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1126–1135. JMLR. org, 2017. 5.1, 5.2

[19] Carlos Florensa, Jonas Degrave, Nicolas Heess, Jost Tobias Springenberg, and Martin Riedmiller. Self-supervised learning of image embedding for continuous control. *arXiv preprint arXiv:1901.00943*, 2019. 5.2

[20] Jakob Foerster, Gregory Farquhar, Maruan Al-Shedivat, Tim Rocktäschel, Eric P Xing, and Shimon Whiteson. Dice: The infinitely differentiable monte-carlo estimator. *arXiv preprint arXiv:1802.05098*, 2018. 5.3.2, 5.4.1

[21] Shahrzad Gholami, Benjamin Ford, Fei Fang, Andrew Plumptre, Milind Tambe, Margaret Driciru, Fred Wanyama, Aggrey Rwetsiba, Mustapha Nsubaga, and Joshua Mabonga. Taking it for a test drive: a hybrid spatio-temporal model for wildlife poaching prediction evaluated through a controlled field test. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 292–304. Springer, 2017. 4.2

[22] Shahrzad Gholami, Sara Mc Carthy, Bistra Dilkina, Andrew Plumptre, Milind Tambe, Margaret Driciru, Fred Wanyama, Aggrey Rwetsiba, Mustapha Nsubaga, Joshua Mabonga, et al. Adversary models account for imperfect crime data: Forecasting and planning against real-world poachers. 2018. 4.2

[23] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Wein-

berger, editors, *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc., 2014. URL http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf. 2.3.1, 3.2

[24] Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. Making the V in VQA matter: Elevating the role of image understanding in Visual Question Answering. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 3.2

[25] Ralph Gross, Iain Matthews, Jeffrey Cohn, Takeo Kanade, and Simon Baker. Multi-pie. *Image and Vision Computing*, 28(5):807–813, 2010. 2.4, 2

[26] Abhishek Gupta, Russell Mendonca, YuXuan Liu, Pieter Abbeel, and Sergey Levine. Meta-reinforcement learning of structured exploration strategies. In *Advances in Neural Information Processing Systems*, pages 5307–5316, 2018. 5.2

[27] Xiaoguang Han, Zhen Li, Haibin Huang, Evangelos Kalogerakis, and Yizhou Yu. High-resolution shape completion using deep neural networks for global structure and local geometry inference. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 85–93, 2017. 2.2

[28] Moritz Helmstaedter, Kevin L Briggman, Srinivas C Turaga, Viren Jain, H Sebastian Seung, and Winfried Denk. Connectomic reconstruction of the inner plexiform layer in the mouse retina. *Nature*, 500(7461):168, 2013. 1

[29] Geoffrey Hinton, Li Deng, Dong Yu, George Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Brian Kingsbury, et al. Deep neural networks for acoustic modeling in speech recognition. *IEEE Signal processing magazine*, 29, 2012. 1

[30] Justin Johnson, Andrej Karpathy, and Fei-Fei Li. Densecap: Fully convolutional localization networks for dense captioning. *CoRR*, abs/1511.07571, 2015. URL http://arxiv.org/abs/1511.07571. 3.2

[31] Gregory Kahn, Adam Villaflor, Bosen Ding, Pieter Abbeel, and Sergey Levine. Self-supervised deep reinforcement learning with generalized computation graphs for robot navigation. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–8. IEEE, 2018. 5.2

[32] Hyeon-Woo Kang and Hang-Bong Kang. Prediction of crime occurrence from multi-modal data using deep learning. *PloS one*, 12(4):e0176244, 2017. 4.2

[33] Debarun Kar, Benjamin Ford, Shahrzad Gholami, Fei Fang, Andrew Plumptre, Milind Tambe, Margaret Driciru, Fred Wanyama, Aggrey Rwetsiba, Mustapha Nsubaga, et al. Cloudy with a chance of poaching: adversary behavior modeling and forecasting with real-world poaching data. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*, pages 159–167. International Foundation for Autonomous Agents and Multiagent Systems, 2017. 4.2

[34] Michael Kazhdan and Hugues Hoppe. Screened poisson surface reconstruction. *ACM Transactions on Graphics (ToG)*, 32(3):29, 2013. 2.2

[35] Jin-Hwa Kim, Devi Parikh, Dhruv Batra, Byoung-Tak Zhang, and Yuandong Tian. Codraw: Visual dialog for collaborative drawing. *arXiv preprint arXiv:1712.05558*, 2017. 3.2

[36] Ryan Kiros, Ruslan Salakhutdinov, and Richard S. Zemel. Unifying visual-semantic embeddings with multimodal neural language models. *CoRR*, abs/1411.2539, 2014. URL http://arxiv.org/abs/1411.2539. 3.2

[37] Satwik Kottur, José M. F. Moura, Stefan Lee, and Dhruv Batra. Natural language does not emerge 'naturally' in multi-agent dialog. *CoRR*, abs/1706.08502, 2017. URL http://arxiv.org/abs/1706.08502. 3.2

[38] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012. 1

[39] Tejas D Kulkarni, Ardavan Saeedi, Simanta Gautam, and Samuel J Gershman. Deep successor reinforcement learning. *arXiv preprint arXiv:1606.02396*, 2016. 5.4.1

[40] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436, 2015. 4.3.3

[41] Andrew M Lemieux. *Situational prevention of poaching*, volume 15. Routledge, 2014. 4.1

[42] Michael KK Leung, Hui Yuan Xiong, Leo J Lee, and Brendan J Frey. Deep learning of the tissue-regulated splicing code. *Bioinformatics*, 30(12):i121–i129, 2014. 1

[43] Yaniv Leviathan. Google duplex: An ai system for accomplishing real-world tasks over the phone, May 2018. URL https://ai.googleblog.com/2018/05/duplex-ai-system-for-natural-conversation.html. 3.1

[44] Mike Lewis, Denis Yarats, Yann N Dauphin, Devi Parikh, and Dhruv Batra. Deal or no deal? end-to-end learning for negotiation dialogues. *arXiv preprint arXiv:1706.05125*, 2017. 3.2

[45] Dongping Li, Tianjia Shao, Hongzhi Wu, and Kun Zhou. Shape completion from a single rgbd image. *IEEE transactions on visualization and computer graphics*, 23(7):1809–1822, 2016. 2.2

[46] Yangyan Li, Angela Dai, Leonidas Guibas, and Matthias Nießner. Database-assisted object retrieval for real-time 3d reconstruction. In *Computer Graphics Forum*, volume 34, pages 435–446. Wiley Online Library, 2015. 2.2

[47] Jiasen Lu, Caiming Xiong, Devi Parikh, and Richard Socher. Knowing when to look: Adaptive attention via A visual sentinel for image captioning. *CoRR*, abs/1612.01887, 2016. URL http://arxiv.org/abs/1612.01887. 3.2

[48] Jiasen Lu, Anitha Kannan, Jianwei Yang, Devi Parikh, and Dhruv Batra. Best of both worlds: Transferring knowledge from discriminative learning to a generative visual dialog model. *CoRR*, abs/1706.01554, 2017. URL http://arxiv.org/abs/1706.01554. 3.2, 3.3.1, 3.3.1, 3.3.1, 3.1

[49] Junshui Ma, Robert P Sheridan, Andy Liaw, George E Dahl, and Vladimir Svetnik. Deep neural nets as a method for quantitative structure–activity relationships. *Journal of chemical information and modeling*, 55(2):263–274, 2015. 1

[50] Parsa Mahmoudieh. Self-supervision for reinforcement learning. 2017. 5.2

[51] Tomáš Mikolov, Anoop Deoras, Daniel Povey, Lukáš Burget, and Jan Černockỳ. Strategies for training large scale neural network language models. In *2011 IEEE Workshop on Automatic Speech Recognition & Understanding*, pages 196–201. IEEE, 2011. 1

[52] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937, 2016. 5.4.1

[53] Jennifer F Moore, Felix Mulindahabi, Michel K Masozera, James D Nichols, James E Hines, Ezechiel Turikunkiko, and Madan K Oli. Are ranger patrols effective in reducing poaching-related threats within protected areas? *Journal of Applied Ecology*, 2017. 4.2

[54] Andrew Nealen, Takeo Igarashi, Olga Sorkine, and Marc Alexa. Laplacian mesh optimization. In *Proceedings of the 4th international conference on Computer graphics and interactive techniques in Australasia and Southeast Asia*, pages 381–389. ACM, 2006. 2.2

[55] Thanh H Nguyen, Arunesh Sinha, Shahrzad Gholami, Andrew Plumptre, Lucas Joppa, Milind Tambe, Margaret Driciru, Fred Wanyama, Aggrey Rwetsiba, Rob Critchlow, et al. Capture: A new predictive anti-poaching tool for wildlife protection. In *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems*, pages 767–775. International Foundation for Autonomous Agents and Multiagent Systems, 2016. 4.2

[56] Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms. *CoRR*, abs/1803.02999, 2018. URL `http://arxiv.org/abs/1803.02999`. 5.2

[57] Marcus Olivecrona, Thomas Blaschke, Ola Engkvist, and Hongming Chen. Molecular de-novo design through deep reinforcement learning. *Journal of cheminformatics*, 9(1):48, 2017. 5.1

[58] Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 16–17, 2017. 5.2

[59] Pascal Paysan, Reinhard Knothe, Brian Amberg, Sami Romdhani, and Thomas Vetter. A 3d face model for pose and illumination invariant face recognition. In *2009 Sixth IEEE International Conference on Advanced Video and Signal Based Surveillance*, pages 296–301. Ieee, 2009. 2.4, 2

[60] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 652–660, 2017. 2.2, 2.3.1

[61] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in neural information processing systems*, pages 5099–5108, 2017. 2.2, 0

[62] Kate Rakelly, Aurick Zhou, Deirdre Quillen, Chelsea Finn, and Sergey Levine. Efficient off-policy meta-reinforcement learning via probabilistic context variables. *arXiv preprint*

arXiv:1903.08254, 2019. 5.2

[63] Jonas Rothfuss, Dennis Lee, Ignasi Clavera, Tamim Asfour, and Pieter Abbeel. Promp: Proximal meta-policy search. *arXiv preprint arXiv:1810.06784*, 2018. 5.1.1, 5.2, 5.3.2, 5.3.2, 5.4.1, 5.5, 5.5.1, 5.5.2

[64] Tara N Sainath, Abdel-rahman Mohamed, Brian Kingsbury, and Bhuvana Ramabhadran. Deep convolutional neural networks for lvcsr. In *2013 IEEE international conference on acoustics, speech and signal processing*, pages 8614–8618. IEEE, 2013. 1

[65] Johannes L Schonberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4104–4113, 2016. 2.4.3

[66] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. 5.4.1, 5.4.1

[67] Michael J Shaffer and Joseph A Bishop. Predicting and preventing elephant poaching incidents through statistical analysis, gis-based risk analysis, and aerial surveillance flight path modeling. *Tropical Conservation Science*, 9(1):525–548, 2016. 4.2

[68] Amr Sharaf and Hal Daumé III. Meta-learning for contextual bandit exploration. *arXiv preprint arXiv:1901.08159*, 2019. 5.1

[69] Yiru Shen, Chen Feng, Yaoqing Yang, and Dong Tian. Neighbors do help: Deeply exploiting local structures of point clouds. *arXiv preprint arXiv:1712.06760*, 1(2), 2017. 2.2

[70] Yifei Shi, Pinxin Long, Kai Xu, Hui Huang, and Yueshan Xiong. Data-driven contextual modeling for 3d scene understanding. *Computers & Graphics*, 55:55–67, 2016. 2.2

[71] Somayeh Shojaee, Aida Mustapha, Fatimah Sidi, and Marzanah A Jabar. A study on classification learning algorithms to predict crime status. *International Journal of Digital Content Technology and its Applications*, 7(9):361, 2013. 4.2

[72] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354, 2017. 5.1

[73] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems*, pages 4077–4087, 2017. 5.2

[74] Olga Sorkine and Daniel Cohen-Or. Least-squares meshes. In *Proceedings Shape Modeling Applications, 2004.*, pages 191–199. IEEE, 2004. 2.2

[75] Andrew Speirs-Bridge, Fiona Fidler, Marissa McBride, Louisa Flander, Geoff Cumming, and Mark Burgman. Smote: Synthetic minority over-sampling technique. *Risk Analysis*, 16:2098, 2010. 4.3.1

[76] Bradly C Stadie, Ge Yang, Rein Houthooft, Xi Chen, Yan Duan, Yuhuai Wu, Pieter Abbeel, and Ilya Sutskever. Some considerations on learning to explore via meta-reinforcement learning. *arXiv preprint arXiv:1803.01118*, 2018. 5.1.1, 5.2, 5.5.2

[77] David Stutz and Andreas Geiger. Learning 3d shape completion from laser scan data with weak supervision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern*

Recognition, pages 1955–1964, 2018. 2.2

[78] Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip HS Torr, and Timothy M Hospedales. Learning to compare: Relation network for few-shot learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1199–1208, 2018. 5.2

[79] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015. 1

[80] Jonathan J Tompson, Arjun Jain, Yann LeCun, and Christoph Bregler. Joint training of a convolutional network and a graphical model for human pose estimation. In *Advances in neural information processing systems*, pages 1799–1807, 2014. 1

[81] Jacob Varley, Chad DeChant, Adam Richardson, Joaquín Ruales, and Peter Allen. Shape completion enabled robotic grasping. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2442–2447. IEEE, 2017. 2.2

[82] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and tell: A neural image caption generator. *CoRR*, abs/1411.4555, 2014. URL `http://arxiv.org/abs/1411.4555`. 3.2

[83] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. In *Reinforcement Learning*, pages 5–32. Springer, 1992. 3.3.2

[84] Mitchell Wortsman, Kiana Ehsani, Mohammad Rastegari, Ali Farhadi, and Roozbeh Mottaghi. Learning to learn how to learn: Self-adaptive visual navigation using meta-learning. *arXiv preprint arXiv:1812.00971*, 2018. 5.2

[85] Qi Wu, Peng Wang, Chunhua Shen, Ian D. Reid, and Anton van den Hengel. Are you talking to me? reasoned visual dialog generation through adversarial learning. *CoRR*, abs/1711.07613, 2017. URL `http://arxiv.org/abs/1711.07613`. 3.2, 3.1

[86] Hui Y Xiong, Babak Alipanahi, Leo J Lee, Hannes Bretschneider, Daniele Merico, Ryan KC Yuen, Yimin Hua, Serge Gueroussov, Hamed S Najafabadi, Timothy R Hughes, et al. The human splicing code reveals new insights into the genetic determinants of disease. *Science*, 347(6218):1254806, 2015. 1

[87] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron C. Courville, Ruslan Salakhutdinov, Richard S. Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. *CoRR*, abs/1502.03044, 2015. URL `http://arxiv.org/abs/1502.03044`. 3.2

[88] Yaoqing Yang, Chen Feng, Yiru Shen, and Dong Tian. Foldingnet: Point cloud auto-encoder via deep grid deformation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 206–215, 2018. 2.2, 0

[89] Yuxiang Yang, Ken Caluwaerts, Atil Iscen, Jie Tan, and Chelsea Finn. Norml: No-reward meta learning. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, pages 323–331. International Foundation for Autonomous Agents

and Multiagent Systems, 2019. 5.2

[90] Ting Yao, Yingwei Pan, Yehao Li, Zhaofan Qiu, and Tao Mei. Boosting image captioning with attributes. *CoRR*, abs/1611.01646, 2016. URL `http://arxiv.org/abs/1611.01646`. 3.2

[91] Lequan Yu, Xianzhi Li, Chi-Wing Fu, Daniel Cohen-Or, and Pheng-Ann Heng. Pu-net: Point cloud upsampling network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2790–2799, 2018. 2.2

[92] Chiyuan Zhang, Oriol Vinyals, Remi Munos, and Samy Bengio. A study on overfitting in deep reinforcement learning. *arXiv preprint arXiv:1804.06893*, 2018. 5.1

[93] Peng Zhang, Yash Goyal, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. Yin and Yang: Balancing and answering binary visual questions. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 3.2

[94] Luisa M Zintgraf, Kyriacos Shiarlis, Vitaly Kurin, Katja Hofmann, and Shimon Whiteson. Caml: Fast context adaptation via meta-learning. *arXiv preprint arXiv:1810.03642*, 2018. 5.4.1, 5.4.1