Uncertainty Quantification of a Classification Network for Contact-Rich Manipulation

Peter Zachares , Negin Heravi

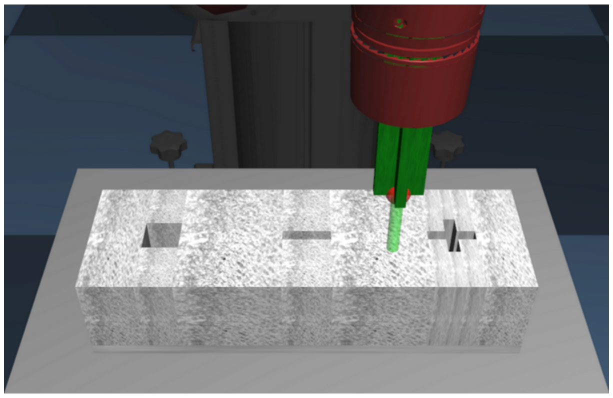


Fig. 1. Image of our Mujoco simulation

I. Project Category

Others- Robotics

II. Problem Statement

This project is an extension of a project that is being developed in the Interactive Perception and Robotic Learning lab at Stanford. Most robots in industry today can only accomplish tasks with little to no uncertainty. The aim of the original project is to develop a method which extends the set of tasks which robots can accomplish using deep learning methods to include tasks with certain forms of options uncertainty. An example of options uncertainty is selecting a tool for a task. For example, when you have to change your tire and have to choose between a set of hex wrenches. The options uncertainty is over which tool to select. For this project, we will be using a 7 DOF robotic arm in simulation to accomplish the task of peg insertion with multiple hole options for the robot to choose from. The robot will need to interact with the peg holes and quantify its uncertainty for each peg hole over the hole's shape using only force and position data. For our CS230 project we looked into different training objectives and network architectures for quantifying uncertainty of a discrete probability distribution. This should improve the performance of the method we are developing for the original research project.

III. Challenge

There are a number of challenges we face in our project. The most common approach for quantifying uncertainty of discrete probability distributions is to train a network using cross entropy loss. This is a maximum likelihood training objective which is known to learn overly conservative uncertainty estimates. In addition, neural networks tend to overfit to their training set, in which case even if the network learned an accurate uncertainty estimate for the training set, it would be conservative for the validation set where performance of the network is worse. These problems are compounded by the issue that in robotics, data collection is expensive. So datasets are relatively small in comparison with the set of inputs they may see in the wild. So our challenge is how can we accurately quantify uncertainty in the task of peg shape classification using sensor traces with a small data set.

P. Zachares and N. Heravi are with the Department of Mechanical Engineering, Stanford University, Stanford, CA 94305.[zachares,nheravi]@staford.edu

IV. Dataset

We collected the data set for this project in simulation in Mujoco. A picture of our simulation environment is shown in figure 1. This data set consists of 4500 recorded trajectories of a robotic arm interacting with three different peg hole shapes: rectangular, square, and cross. Each trajectory contains data collected over a period of 7.5 seconds. This corresponds to 75 steps in our robot's environment since the control frequency of the robotic arm is set to 10 Hz. During a trajectory, the end-effector of the robotic arm slides over one peg hole while traversing one of the diameters of a circle centered at the hole's center with a radius of 4 cm. The circle's radius is chosen such that it is large enough to contain the hole. This procedure was repeated 500 times for every combination of pegs and holes ($3x3$) resulting in the 4500 trajectories in our data set. Each step in the trajectory contains the following sensory information about the end-effector: (i) 6-d pose (ii) linear velocity (iii) angular velocity (iv) a binary value indicating the status of contact with the surface (v) force/torque readings (with a dimension of 50 since the force/torque sensor has a higher frequency (500Hz) than the robotic arm controller (10Hz)).

V. Literature Review

Several researchers have previously worked on strategies for quantifying model uncertainty [1, 2, 3]. For example, there are approaches that represent model uncertainty by proposing a theoretical framework that casts dropout as a Bayesian inference approximation in deep Gaussian processes [4]. Gal and Ghahramani [4] have shown that by simply implementing drop out layers in a network and using them both in training and testing, one can approximate uncertainty.

Furthermore, Kendall and Gal [5] have looked into the comparative importance of modeling two different type of uncertainties that are seen in Bayesian deep learning models. These uncertainties are called: (i) epistemic uncertainties which are model dependant and (ii) aleatoric uncertainties which capture the observation noise.

Lastly , Ovadia et al. [6] provide a large scale empirical comparison between different state of the art Bayesian and non-Bayesian methods on their performance for uncertainty quantification. This study can serve as a great tool for us to choose between different existing methods that we can investigate for the final report.

In our project, we compared the technique of using Monte Carlo dropout (MC dropout) originally proposed in [4] and of using an ensemble of models described in [6] with a baseline architecture to evaluate their effectiveness at quantifying uncertainty for our classification task. Both techniques have similar limitations. To quantify model uncertainty both use a Monte Carlo estimate of the integral over the weights of a network's architecture. In MC dropout, each weight in a network can randomly take on its learned value or 0 during a forward pass. This approximates the space of models that it is possible to learn, because the model changes from one forward pass to another. The limitation of this model is that the space of each weight can only take on two values. So MC dropout cannot describe the large variation in learned weights that it is possible to learn when training a classifier. An ensemble of models also uses a Monte Carlo estimate of the integral over the weights of a network's architecture. In this case, the models are fixed after the learning process finishes and so the variation in models is based on the variation in performance between the different models in the ensemble. Consequently, a very large ensemble of models must be learned to estimate model uncertainty accurately which is often too computationally expensive to be practical. So the limitation of both techniques is that they only express a small variation in the space of learned models that is possible for a specific learning task.

VI. Method

For this project we are learning a classifier network which takes in a time series of force / position readings and outputs a multinomial distribution over whether a peg fits the hole it is exploring. The output of the classifier should not only indicate whether the peg fits but also quantify the likelihood of its prediction.

A. Architecture

Figure VI shows the architecture that enables the classification of fit as well as the prediction likelihood. This architecture takes as input the change in position and contact information of the end-effector over a full trajectory (74 time steps x 13 sensor readings) as well as the corresponding force/torque readings of the sensor (74 time steps x 50 (multiple readings during 1 step) x 6 (3:Force + 3:Torque readings)). The output of this network is a classification vector that indicates the probability of fit between the robot's peg and the hole being explored.

The architecture encodes the position/contact information into a 74x80 dimensional matrix using a 3 layer ResNet [7] network where each layer has 80 neurons. Force/torque sensor readings are first processed using a fast Fourier transform which is then processed using a force encoder consisting of a 2-d convolutional neural network. The output of the force encoder is a matrix of $74x48$ dimensions.

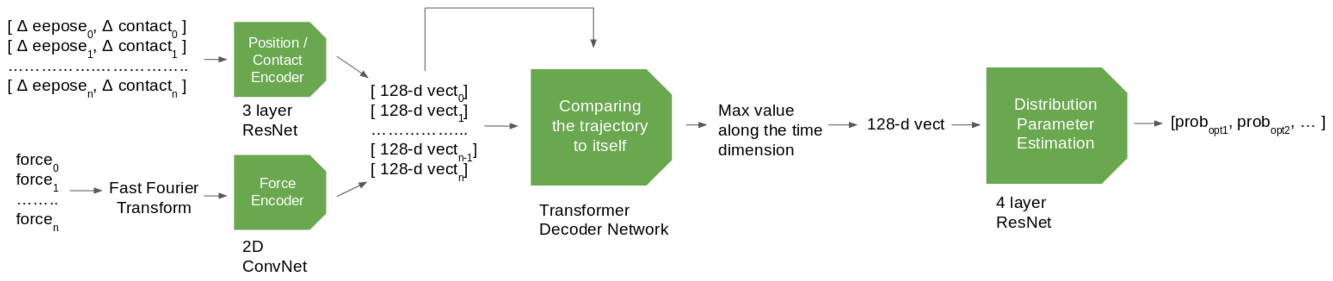
The output of these two encoders are concatenated and inputted into a transformer decoder network [8] which compares the trajectory to itself. The choice of a transformer was driven by its optimal performance on the validation set compared to an LSTM network as well as an LSTM with attention network we had explored in our previous work. The output of the transformer network has a dimension of $74x128$ where the first dimension corresponds to time steps. To reduce the time dimension, we take the maximum of each row to create a vector of 128 dimensions.

Lastly, we feed this vector into our distribution parameter estimation module which has a 4 layer ResNet architecture where each layer has 32 neurons, except for the last layer which has 2 neurons. A Softmax function is applied to the output of this network to obtain probability estimates. The choice of a ResNet architecture as opposed to fully connected layers was inspired by previous studies that have shown networks with skip connections are easier to optimize [7].

All models described in the results section where trained using the same loss function. We trained our classification networks using cross entropy loss, described in (1) below. In (1) $f(\cdot)$ is the mapping from sensor inputs to logits of the multinomial distribution over fit. If the outputs of $f(\cdot)$ are put through a softmax function, then the outputs of the softmax function will be probability masses of a multinomial distribution. In (1), y_i is the index of the logit of the correct classification for datapoint x_i and n is the number of classes. For our project n is 2, because a peg can only fit or not fit into a hole. Usually binary cross-entropy loss is used to train networks with only two possible classes, but in our case we chose to use cross entropy loss, which meant modelling the multinomial distribution over fit using two parameters instead of one.

$$Loss = -\log\left(\frac{exp(f_{y_i}(x_i))}{\sum_{j=0}^{n-1} exp(f_j(x_i))}\right) \tag{1}$$

Our baseline model used the probabilities outputted by our trained classification model after passing it through a softmax function to quantify uncertainty. In addition, we tested three different uncertainty quantification techniques which all estimated model uncertainty. The first technique proposed in [4] is called MC dropout. For MC dropout, a model is trained with dropout layers before its fully connected layers and after its convolution layers. Then at test time, the dropout layers are maintained and multiple forward passes are performed to record sample classification from different possible models. The second technique we



[Δ eepose$_0$, Δ contact$_0$]
[Δ eepose$_1$, Δ contact$_1$]
[Δ eepose$_n$, Δ contact$_n$]
Position / Contact Encoder
3 layer ResNet
force$_0$
force$_1$
force$_n$
Fast Fourier Transform
Force Encoder
2D ConvNet
[128-d vect$_0$]
[128-d vect$_1$]
[128-d vect$_{n-1}$]
[128-d vect$_n$]
Comparing the trajectory to itself
Transformer Decoder Network
Max value along the time dimension
128-d vect
Distribution Parameter Estimation
4 layer ResNet
[prob$_{opt1}$, prob$_{opt2}$, ...]

Figure 2: Schematic of Classification Network Architecture

explored was an ensemble method described in [6]. At test time, an input is used to make a forward pass through each model in the ensemble to record sample classifications from different possible models. The third technique we explored was combining MC dropout with an ensemble of models, because training a large ensemble of models is very computationally expensive and by adding dropout to a network it is possible to take multiple samples from the same network (which should improve the uncertainty quantification). So by combining the two techniques, we hoped to gain some of the effectiveness the ensemble model and the ability to record a large number of samples provided by the dropout method. For all the techniques involving an ensemble, we trained three separate models. For all techniques involving dropout, we used a dropout rate of 0.2. At test time, to quantify uncertainty for each datapoint in our batch we took a total of 180 samples. A sample was an estimate of the class label for that data point given the sensory input. Then the uncertainty estimate was calculated by using the following equation.

$$p_i = \frac{n_i}{n_{total}} \tag{2}$$

In this equation, p_i is the probability of class i, n_i is the number of samples classifying an input as class i and n_{total} is the total number of samples recorded. This is the maximum likelihood estimation of a multinomial distribution based on samples.

VII. Results

For this project, two of the three metrics we use are based on average entropy ratios. A good uncertainty quantification should be more uncertain on average of unseen data points than points on which it was trained. So a high average entropy ratio between points in the validation set and training set could indicate a good uncertainty qualification (validation average divided by training average). A good uncertainty qualification should also be less certain of points that are incorrectly classified than points that are correctly classified. As such, the ratio of average entropy between incorrectly classified points and correctly classified points in the validation set should be high (incorrect average divided by correct average). Our last performance metric is based on entropy and an assumptions about the ground truth uncertainty these techniques are trying to estimate. This metric is called entropy difference range in the results table. This metric is calculated by first calculating the entropy of every uncertainty quantification in a batch, then sorting these entropies based on magnitude. The difference is taken between consecutive points in the sorted list. The range of the resulting vector is an indication of the distribution of entropies in the batch. If there is a large range in values this indicates that the entropies are concentrated around certain values, but if there is a small value, this indicates the entropies are more uniformly spread out. We know our baseline architecture learns an overly certain uncertainty quantification and so its entropy distributions per batch are more clustered. If we assume that the ground truth uncertainty estimate is more spread out than what can be learned by our baseline architecture, the entropy difference range metric should indicate which technique is able to estimate uncertainty best. For this metric, smaller values indicate better performance.

All three uncertainty techniques outperformed our baseline model in terms of average entropy ratio between the validation and training set. This demonstrates that these techniques produce estimates which are more uncertain of points outside the training set than the baseline model. However, only the baseline with ensemble technique outperformed the baseline model in terms of average entropy ratio of incorrectly classified to classified points. This suggests that using MC dropout increases the certainty of the estimate for incorrectly classified points, possibly when it should not do so. The best performing model on our last metric of entropy range difference was the baseline with an ensemble of models and using MC dropout. This demonstrates that these techniques combined can estimate the greatest range of uncertainty quantifications, probably because the space of possible learned models that is described by combining these two techniques is greater than using one of them individually. The entropy difference range metric is also notable because of the decrease in value as more model uncertainty estimation techniques are used. In this project, the baseline model had the highest range.

These trained models were also evaluated on a downstream robotic control task where the baseline model performed very poorly. The baseline model with an ensemble and the baseline model using MC dropout did not perform much better and it was only the baseline model which used an

	Average Entropy Ratio (Val / Train)	Average Entropy Ratio (Incorrectly Classified / Correctly Classified)	Entropy Difference Range (Val)
Baseline	1.082	2.580	0.0953
Baseline with Ensemble	1.5113	3.314	0.0637
Baseline with MC Dropout	1.126	1.581	0.0645
Baseline with Ensemble + MC Dropout	1.155	1.595	0.0555

Table 1: Performance of Models on Project Metrics

ensemble of models and MC dropout which performed reasonably well on the task. This demonstrates that it is hard to find a metric for evaluating an uncertainty quantification that is not based on a downstream task performance. In our case, the entropy difference range seems to be the most useful indicator for determining which of the models will perform best in the overarching research project that inspired this class project.

VIII. Contributions

For this project, Peter developed the architecture for the classifier models and Negin implemented the uncertainty quantification techniques as well as the metrics used to evaluate their performance.

References

[1] J. Gast and S. Roth, "Lightweight probabilistic deep networks," *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3369–3378, 2018.

[2] M. Sensoy, M. Kandemir, and L. M. Kaplan, "Evidential deep learning to quantify classification uncertainty," *CoRR*, vol. abs/1806.01768, 2018. [Online]. Available: http://arxiv.org/abs/1806.01768

[3] A. Loquercio, M. Segù, and D. Scaramuzza, "A general framework for uncertainty estimation in deep learning," *IEEE Robotics and Automation Letters*, vol. 5, pp. 3153–3160, 2019.

[4] Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," in *ICML*, 2016.

[5] A. Kendall and Y. Gal, "What uncertainties do we need in bayesian deep learning for computer vision?" in *NIPS*, 2017.

[6] Y. Ovadia, E. Fertig, J. Ren, Z. Nado, D. Sculley, S. Nowozin, J. V. Dillon, B. Lakshminarayanan, and J. Snoek, "Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift," *ArXiv*, vol. abs/1906.02530, 2019.

[7] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015. [Online]. Available: http://arxiv.org/abs/1512.03385

[8] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *CoRR*, vol. abs/1706.03762, 2017. [Online]. Available: http://arxiv.org/abs/1706.03762