Hochschule
Bonn-Rhein-Sieg
University of Applied Sciences

b-it Bonn-Aachen International Center for Information Technology

Master's Thesis

Uncertainty Estimation for Keypoint Detection and its Application in Robot Grasping

Nandhini Shree Mathivanan

Submitted to Hochschule Bonn-Rhein-Sieg,
Department of Computer Science
in partial fullfilment of the requirements for the degree
of Master of Science in Autonomous Systems

Supervised by

Prof. Dr. Nico Hochgeschwender
Prof. Dr. Teena Chakkalayil Hassan
Mr. Deebul Nair

January 2024

I, the undersigned below, declare that this work has not previously been submitted to this or any other university and that it is, unless otherwise stated, entirely my own work.

Date	Nandhini Shree Mathivanan

Abstract

In the real world robot grasping applications, understanding uncertainty is like giving the robot a sixth sense, it helps them adapt to unpredictable situations and make better choices. It helps robots to make informed decisions, ensuring robust performance even when faced with uncertainties and ultimately enhancing the reliability of the overall task. Recognizing the significance of Uncertainty Estimation in enhancing the reliability of robotic systems, our research journey began with comparison of Uncertainty Estimation in UCI regression 2D datasets. We then ventured into a more complex 3D domain using the Cornell Grasping Dataset, employing five distinct loss functions: Gaussian Negative Log-Likelihood, Laplace Negative Log-Likelihood, Cauchy Negative Log-Likelihood, Evidential loss and Generalized Gaussian Negative Log-Likelihood. Our approach involved modeling Neural Networks to produce outputs that not only include predicted values but also incorporate uncertainty. After comparison of various Uncertainty Estimation methods, we subsequently translated our findings into practical implementation on a real-time Kinova robot arm. As a result of our extensive exploration, this thesis proves that the use of Generalized Gaussian Negative Log-Likelihood excels in predicting accurate keypoints along with better uncertainty estimates, demonstrating its efficacy in real-world robotic grasping tasks.

Acknowledgements

I would like to express my deep appreciation to Prof.Dr.Nico Hochgeschwender, my primary supervisor, for his invaluable guidance, constant support, and insightful feedback during the entire research period. His expertise and commitment have been instrumental in shaping the trajectory of my work.

Acknowledgment is also extended to Prof. Teena Chakkalayil Hassan, my second supervisor, whose valuable contributions, constructive critiques, and words of encouragement have enhanced the quality of my research.

I extend special thanks to Mr.Deebul Nair, my third supervisor, for his practical insights, technical assistance, and collaborative approach, all of which have significantly contributed to the success of this project.

Gratitude is extended to the entire Department of Autonomous systems for providing an enriching research environment.

Lastly, I wish to convey my heartfelt thanks to my family and friends for their unwavering support and encouragement throughout this academic journey.

Contents

List of Figures

List of Tables

1

Introduction

Deep learning, which is a subset of Machine learning domain makes use of artificial neural network with various layers which are used for learning and at the same time extracting complex representations and patterns from data. In our case, Deep learning is very crucial in the field of uncertainty estimation as it helps in providing the model's ability to detect intricate relationships within the data, thus allowing the estimation of uncertainties. Uncertainty estimation techniques, especially in complex and uncertain real-world scenarios, can enable more robust decision-making process and also improves the reliability and safety of systems by utilizing the expressive power of deep neural networks to provide reliable measures of confidence in predictions.

In this thesis, we focus on estimating uncertainties on keypoint detection tasks where in precise, the keypoint or feature in an image can be defined as a unique meaningful structure in that image such as edges, object parts or corners in an image. The method of assessing the uncertainty or lack of confidence associated with the predicted keypoints in a particular dataset is known as uncertainty estimate in keypoint detection data. By providing information beyond the keypoints themselves, this uncertainty estimation enables a more thorough understanding of the accuracy and reliability of the detections.

In particular we are estimating uncertainty for a robot gripper application for improving the system's reliability and safety. Robotic grippers are add-ons for robotic arms that let them control things in their surroundings. Estimating uncertainty can be helpful in determining how confident the gripper is in their operations, guaranteeing robustness, and averting failures or mishaps. The robot gripper uses uncertainty estimates to evaluate the accuracy of the keypoints that have been detected and used for grasp planning. The keypoints and their corresponding uncertainties may both be taken into account by the gripper when choosing the best grasping points, which can be done by measuring the uncertainty related to keypoint detection. This in turn lowers the possibility of unsuccessful or unstable grasps and increases grasping success rates.

Therefore in this thesis work, we are working on uncertainty estimation of keypoint detection task using five different loss functions particularly focused on robot gripper application and the performance of our model will be evaluated by implementing the uncertainty estimation model on a real time Kinova robot arm.

Why is it important?

Uncertainty estimation becomes crucially important as it plays a crucial role in risk assessment and decision making process. When an inaccurate measurement results are detected it leads to increase in decision risks. In case of safety-critical applications relying on Neural Network models, a measure of uncertainty can be helpful to decide whether low-confidence decisions need further validation.

Uncertainty estimation is important for keypoint detection tasks as it addresses challenges related to ambiguity, noisy data and generalization where precise keypoint localization is crucial. They also help to estimate the uncertainties and challenges that are involved in accurately localizing keypoints in images or data domains. One may increase the robot gripper application's reliability, decision-making, and capacity to adapt to various circumstances and handle uncertainties by introducing uncertainty estimating approaches into it.

Uncertainty estimation for robot gripper applications helps to avoid the following accidents,

- Object slippage: Without estimating the uncertainties, The robot gripper might not be able to account for changes in the grip or the uncertainty in accurately identifying the object's keypoint positions. This might result in the object sliding from the gripper's hold during manipulation, potentially causing damage or accidents.

- Overexertion of force: Lack of uncertainty estimate may cause the robot gripper to grab an object too tightly while manipulating it. This may occur if the gripper lacks confidence in its keypoint detection. Uncertainty estimation allows the robot to assess the confidence level in its keypoint detection process. With a proper estimate of uncertainty, the robot can adjust its gripping force based on how sure it is about the detected keypoints. This prevents excessive force application, minimizing the risk of damaging the object or the gripper itself.

- Failure to adapt to changing conditions: Keypoint detection may be uncertain in real-time environments due to dynamic changes in the surroundings or object attributes. The robot gripper might not be able to efficiently react to these changes without uncertainty estimation. This may lead to a failure in recognizing or reacting effectively to unfamiliar or unforeseen events, which may result in accidents, poor grasp, or jeopardized safety.

The above examples illustrates the importance of uncertainty estimation by explaining the dangers and mishaps that could develop in a robot gripper application if uncertainty estimation is not performed. The robot gripper may lessen these risks, increase safety, and improve overall performance in real-time operations by implementing uncertainty estimation approaches, such as analyzing confidence levels in keypoint detection and taking uncertainties in grasping and manipulation into account.

1.0.1 Problem statement

Deep neural networks (DNNs) inherently exhibit stochastic behavior characterized by randomness and uncertainties in their outputs. In applications where risks are high, recognizing and understanding these uncertainties becomes paramount. The utilization of DNNs in safety-critical and autonomous systems has gained prominence due to their ability to not only provide predictions but also estimate and quantify uncertainty in their predictions, making them invaluable for such contexts. Accurate and calibrated uncertainty can be used to build confidence in autonomous systems decision-making. Because there is no ground truth for uncertainty, uncertainty estimation is a challenging task, especially in high-dimensional data. The existence of noisy labels or outliers in the training data makes uncertainty estimation more challenging.
It often becomes difficult or impractical to acquire ground truth uncertainty for keypoint annotations. Normally, annotators give accurate coordinates for keypoints without expressing their level of confidence or uncertainty. It is difficult to effectively model the uncertainty during training or estimate without clear uncertainty information in the annotated data.

1.0.2 Research Goals

The field of Deep Learning has witnessed significant advancements in recent years, with several studies focusing on Uncertainty estimation. However, there remains a critical research gap regarding Estimating uncertainty for Keypoint detection tasks, prompting the need for further investigation and analysis. One such research gap as described in the paper [22] is that many real-world datasets used for keypoint detection tasks often have sparse and incomplete annotations. This means that not all keypoints in an image are labeled, and some may be missing due to various reasons, such as occlusion or annotation limitations. Estimating uncertainty accurately in such cases is challenging because the model needs to understand not only the confidence in its detected keypoints but also the uncertainty linked with the annotated keypoints. Therefore, In this research we implement uncertainty estimation methods on real time robot arm by modeling the loss function using a heavy-tailed distribution and is evaluated using the complex regression tasks such as Keypoint estimation.
Furthermore, this work aims to answer the following research goals.

- **Comparison of uncertainty estimation methods for Keypoint detection using UCI Datasets**
- **Comparison of uncertainty estimation methods for Keypoint detection using Cornell grasp Dataset**
- **Implementation of uncertainty estimation methods on a real time robot**

Therefore, This research adds to the overall understanding of uncertainty estimation in Keypoint detection, paving the way for enhanced applications in complex regression tasks.

2

Related work

This chapter provides information on previous research and related works within the realm of uncertainty estimation, with a specific focus on its application in robot grasping. The chapter explores existing studies, methodologies, and findings pertinent to this domain, with the goal of offering context and laying the groundwork for the current research.

2.1 Uncertainty estimation for regression

Uncertainty estimation methods can be classified into three different categories such as: 1. Bayesian methods [14] , 2. Single-model methods [23] and 3. Sampling based methods [9]. Although sampling based methods and bayesian methods are found to be state-of-the-art uncertainty estimation methods[7] they come with a disadvantage of : high memory requirements, high prediction time and high training time. For example, when converting weights to distributions the bayesian neural learning method requires double the parameters, In MC-dropout technique [9] which has 50 Monte Carlo samples will have the requirements to have 50x forward passes and finally the Deep Ensembles [15] methods which has an ensemble size of 5 will have 5x number of weights. As a result of this, their use in real-time autonomous systems has become an operational challenge. Hence, the uncertainty estimated by the single-model based methods predicts with a single forward pass, no increase in weights and with a single network [23]. Therefore, in this research, we concentrate on single-model based methods as they are suitable for the autonomous systems application.

2.2 Robustness study Uncertainty estimation for regression

In this paper [11], a technique is presented to enhance the estimation of uncertainties in deep neural networks by integrating noise priors. By employing this method, the authors showcase enhanced uncertainty estimates across different tasks, such as regression, demonstrating its robustness. This study [25] introduces a framework for robust regression that explicitly models uncertainty and robustness. It tackles the problem as a bi-level optimization and includes both theoretical analysis and experimental results. A novel likelihood-free inference method is proposed in this research [24] for robust and scalable Bayesian deep learning. It specifically focuses on addressing uncertainty estimation and robustness in scenarios involving model misspecification and adversarial attacks. The investigation in this paper [1] centers around robust uncertainty estimation in variational autoencoders (VAEs). A technique named

IWAE-DGM (Importance-Weighted Autoencoder with Denoising Gaussian Mixture) is introduced and its effectiveness in robust uncertainty estimation is demonstrated. This research [27] proposes a deep robust Bayesian neural network (BRNN) framework for uncertainty estimation. By incorporating prior knowledge and robustness constraints into the Bayesian neural network, it showcases its effectiveness in various tasks, including regression.

2.3 Uncertainty estimation for keypoint detection tasks:

The study in this paper [8] introduces a Bayesian convolutional neural network (CNN) method for detecting keypoints, which employs Bernoulli approximate variational inference to estimate uncertainties. This approach offers a systematic method for capturing epistemic uncertainty, leading to enhanced reliability and accuracy in keypoint detection. In this paper[17] for regression tasks, they give an explanation about the maximum likelihood estimation (MLE). By benchmarking three common pose estimation datasets with a residual Log-likelihood, this paper has proposed a paradigm on novel regression methods. The paper by [4] addresses the inherently challenging issue of 3D human localization using monocular RGB images. Recognizing the limitations of neural networks providing singular point estimates, the authors tackle the ambiguity inherent in the task. They achieve this by predicting confidence intervals utilizing a loss function derived from the Laplace distribution.

2.4 Uncertainty estimation for robot gripper application

The paper [30] tackles the issue of adapting grasping detection models to different domains. It introduces a novel network that integrates uncertainty estimation to guide the process of confidence-driven semi-supervised learning. By doing so, the approach aims to enhance the performance and robustness of grasping detection models when dealing with new domains that have limited labeled data.
The paper [5] introduces a compact CNN model designed specifically for identifying graspable objects in robotic applications. It proposes the use of a Gaussian-based representation to precisely localize these objects. The main objective of this research is to improve the efficiency and accuracy of robotic grasping systems by employing a lightweight yet effective network architecture.
The paper [16] investigates the use of deep learning methods in the context of grasp detection for robotics. It offers an extensive examination of different deep learning models and their effectiveness in detecting grasps. The objective of this study is to push the boundaries of robotic grasping by harnessing the power of deep learning algorithms to achieve precise and efficient grasp detection.
The paper [6] focuses on the task of detecting multiple objects and their corresponding grasps in real-world settings. It introduces a comprehensive approach that integrates object detection and grasp detection within a unified framework. The main objective of this research is to enhance the perception and interaction capabilities of robotic systems, enabling them to accurately identify and grasp multiple objects in complex environments.
The paper [12] presents a novel approach that utilizes a unique rectangle representation for efficient grasp learning from RGBD images. The proposed method aims to enhance the precision and computational

efficiency of robotic grasping by leveraging this innovative representation. The research specifically focuses on improving grasp detection and execution in real-world environments using RGBD data.
This research [18] presents an algorithm for grasp detection that relies on keypoints. The approach involves representing objects and grasps as points and predicting various object attributes like size and direction. Empirical findings indicate that this technique achieves a precision of 74.3 percent in the VMRD dataset, which consists of multi-object grasping scenarios.

3

Methodology

In this chapter, we'll delve into the details of the experiment we conducted to compare various methods for estimating uncertainty in regression tasks. We used specific datasets to carry out this experiment and evaluate the metrics that help us understand uncertainty. We'll provide a thorough explanation of these datasets and metrics in this section.

3.1 Datasets

In this experiment, we use two different datasets for comparing uncertainty estimation methods. We start the experiment with 2D UCI regression datasets and then we are moving on to larger image datasets.

3.1.1 UCI Datasets

The UCI regression datasets [28] are relatively simple and easy to understand. They typically have a small number of features, making it easier to understand the underlying data distribution and model behavior. This simplicity allows researchers to gain insights into how different uncertainty estimation methods work in a controlled and interpretable setting. In this work we use eight different UCI regression datasets from the website [28] such as,

- Kin8nm - This dataset is all about figuring out how an 8-link robot arm moves. There are different variants of this dataset, and we're using "8nm," which is famous for being pretty tricky to work with because the relationship between the input and the robot arm's position is quite complicated, and there's some amount of random noise in the data.

- Boston - The Boston housing data is a bunch of information collected back in 1978 about homes in different Boston suburbs. It includes details about 14 different aspects of these homes, and there are 506 entries in total.

- Yacht - This dataset helps us predict how much resistance a sailing yacht will experience when it's first designed. Knowing this is super important for figuring out how the yacht will perform and how much power it'll need to move. We use inputs like the yacht's size and how fast it can go.

- - Concrete Dataset - This dataset provides information about 1,030 samples of concrete. It describes the various components used in the concrete mixture with eight different characteristics. These characteristics are believed to have an impact on the concrete's final strength. For example, they include details like the quantity of certain components used in each cubic meter of concrete.

- Wine - The wine dataset contains data from a chemical analysis of wines produced in the same region of Italy. These wines are made from three different grape varieties. The analysis measures 13 different components present in each type of wine. It helps distinguish the characteristics of the wines based on their chemical composition.

- Protein - This dataset contains information about the physiochemical properties of protein tertiary structures. It's derived from a scientific competition (CASP 5-9) that aimed to predict the accuracy of protein structure models. The dataset includes values related to the RMSD (Root Mean Square Deviation), which is a measure of how accurate the models are.

- Power plant - The power plant dataset consists of 9,568 data points collected over six years from a combined cycle power plant. The data was recorded during periods when the plant operated at full capacity. It's used to predict the power output of this type of power plant based on various environmental conditions and factors.

- Energy - This dataset contains 36,733 instances of measurements from 11 sensors, taken over one-hour intervals, from a gas turbine located in northwestern Turkey. These measurements are used to study the emissions of gases like carbon monoxide (CO) and nitrogen oxides (NOx). The dataset is valuable for predicting the energy produced by the turbine, known as turbine energy yield (TEY), using environmental variables as features.

3.1.2 Cornell Grasp Dataset

The second dataset we used in this work is an image dataset called as Cornell Grasp dataset[13] which is a multi object multi grasp RGB dataset with different real objects and also it includes corresponding keypoints in them. In each image, we have 8 keypoints, which are stored in individual text files as (x, y) coordinate values. Given our use of a Kinova arm setup, we've captured images as bag files for 10 distinct objects. We've performed manual annotations on these images and have integrated them into the Cornell Grasp Dataset. The objects that are available in the dataset can be seen in the below image.

Figure 3.1: Cornell Grasp Dataset objects[13]

3.2 Metrics

The explanation and images associated with these metrics are derived directly from my prior research and development (R&D) work, where I initially introduced and utilized the same metrics. To avoid self-plagiarism, I provide a concise and clear explanation, along with the relevant visual representations, as they were employed in my previous research.

3.2.1 Gaussian negative log likelihood

The provided loss function and its explanation are sourced from the blog by [23]. In the context of Gaussian NLL Loss, the regression dataset's targets are treated as data originating from Gaussian distributions, with the neural network predicting both variances and expectations. In the Gaussian loss scenario, the model also aids in forecasting confidence in the prediction, specifically represented by the Gaussian variance. The Gaussian distribution is formulated as per [23]

$$p(y|\mu,\sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} exp(-\frac{input-target)^2}{2\sigma^2})$$

Upon taking the natural logarithm on both sides and introducing a negative sign, the resulting loss function is expressed as follows [23]:

$$loss = \frac{1}{2}(log(max(var, eps) + \frac{(input-target)^2}{max(var, eps)} + constant)$$

Here the term "eps" in the above loss formula is employed for stability. Unless explicitly set to True, the constant term in the mentioned loss function is omitted by default.

In the context of the Gaussian loss function, the neural network model not only predicts the target values but also provides confidence in the predictions, in the form of Gaussian variance [19], as in Figure 3.5.

```
class FaceKeypointModel(nn.Module):
    def __init__(self, freeze_resnet = False):
        super(FaceKeypointModel, self).__init__()


        self.conv1 = nn.Conv2d( in_channels=1, out_channels=3, kernel_size=(3, 3), stride=1,
                                padding=1, padding_mode='zeros' )

        # Resnet Architecture
        self.resnet18 = models.resnet18(pretrained=True)
        if freeze_resnet:
            for param in self.resnet18.parameters():
                param.requires_grad = False
        # replacing last layer of resnet
        # by default requires_grad in a layer is True
        self.resnet18.fc = nn.Linear(self.resnet18.fc.in_features, 384)

        self.relu = nn.ReLU()
        self.linear1 = nn.Linear(384, 30)
        self.variance = nn.Linear(384,30)

    def forward(self, x):
        y0 = self.conv1(x)
        y1 = self.resnet18(y0)
        y_relu = self.relu(y1)
        out= self.linear1(y_relu)

        var = F.softplus(self.variance(y_relu))
        return out,var
```

Figure 3.2: Neural Network model for Gaussian Negative Log Likelihood

3.2.2 Laplace negative log likelihood

The provided loss function and its explanation are derived from the blog by [21]. In the LaplaceNLLLoss, the targets within the regression datasets are treated as samples from Laplace distributions, with the neural network predicting both the scale and expectations. The Laplace distribution is mathematically defined as follows:

$$p(y|\mu, s) = \frac{1}{2s} exp(-\frac{|y-\mu|}{s})$$

In this study, the Negative Log Likelihood (NLL) of the Laplace distribution is employed for training neural networks, resulting in the following expression:

$$l_{NLL}(x, y, \theta) = -log(p(y|f_\theta^\mu, f_\theta^s) = log(2f_\theta^s) + (\frac{|y - f_\theta^\mu|}{f_\theta^s})$$

This loss function exhibits several properties supportive to gradient-based optimization. The convex nature of the Laplace loss ensures that the loss is minimized when $|y - \mu|$ is at a minimum, and it consistently increases monotonically concerning $|y - \mu|$.

For reference, the PyTorch code implementation of Laplace Negative Log Likelihood can be found in [19].

```python
def LaplaceNLLLoss(input, target, scale, eps=1e-06, reduction='mean'):
    loss = torch.log(2*scale) + torch.abs(input - target)/scale

    # Inputs and targets much have same shape
    input = input.view(input.size(0), -1)
    target = target.view(target.size(0), -1)
    if input.size() != target.size():
        raise ValueError("input and target must have same size")

    # Second dim of scale must match that of input or be equal to 1
    scale = scale.view(input.size(0), -1)
    if scale.size(1) != input.size(1) and scale.size(1) != 1:
        raise ValueError("scale is of incorrect size")

    # Check validity of reduction mode
    if reduction != 'none' and reduction != 'mean' and reduction != 'sum':
        raise ValueError(reduction + " is not valid")

    # Entries of var must be non-negative
    if torch.any(scale < 0):
        raise ValueError("scale has negative entry/entries")

    # Clamp for stability
    scale = scale.clone()
    with torch.no_grad():
        scale.clamp_(min=eps)

    # Calculate loss (without constant)
    loss = (torch.log(2*scale) + torch.abs(input - target) / scale).view(input.size(0), -1).sum(dim=1)


    # Apply reduction
    if reduction == 'mean':
        return loss.mean()
    elif reduction == 'sum':
        return loss.sum()
    else:
        return loss
```

Figure 3.3: PyTorch code for Laplace Negative Log Likelihood [19]

3.2.3 Cauchy negative log likelihood

The provided loss function and its explanation are sourced from the blog [19]. In the CauchyNLLLoss, the targets within the regression datasets are regarded as samples from Cauchy distributions, with the neural network predicting both the scale and expectations. The Cauchy distribution is defined mathematically as follows:

$$p(y|\mu, s) = \frac{1}{\pi s \left(\frac{y-\mu}{s}\right)^2}$$

In this study, the Negative Log Likelihood (NLL) of the Cauchy distribution is employed for training neural networks resulting in the following expression:

$$c_{NLL}(x, y, \theta) = -log(p(y|f_\theta^\mu, f_\theta^s) = log(3.14 f_\theta^s) + log1 + (\left(\frac{|y - f_\theta^\mu|^2}{f_\theta^{s\,2}}\right)$$

The PyTorch code implementation for Cauchy Negative Log Likelihood is provided in [19].

3.2.4 Evidential loss

The following loss function and its elucidation are extracted from the paper by [2]. In the context of Evidential deep learning, the learning process is conceptualized as evidence acquisition. During training, all the data contributes evidence to an Evidential distribution, characterized as a higher-order distribution that is learned. Evidential techniques ensure that priors are directly placed on the likelihood function, in contrast to Bayesian neural networks where priors are typically imposed on network weights.

```python
def CauchyNLLLoss(input, target, scale, eps=1e-06, reduction='mean'):

    # Inputs and targets much have same shape
    input = input.view(input.size(0), -1)
    target = target.view(target.size(0), -1)
    if input.size() != target.size():
        raise ValueError("input and target must have same size")

    # Second dim of scale must match that of input or be equal to 1
    scale = scale.view(input.size(0), -1)
    if scale.size(1) != input.size(1) and scale.size(1) != 1:
        raise ValueError("scale is of incorrect size")

    # Check validity of reduction mode
    if reduction != 'none' and reduction != 'mean' and reduction != 'sum':
        raise ValueError(reduction + " is not valid")

    # Entries of var must be non-negative
    if torch.any(scale < 0):
        raise ValueError("scale has negative entry/entries")

    # Clamp for stability
    scale = scale.clone()
    with torch.no_grad():
        scale.clamp_(min=eps)

    # Calculate loss (without constant)
    loss = (torch.log(3.14*scale) +
            torch.log(1 + ((input - target)**2)/scale**2)) .view(input.size(0), -1).sum(dim=1)


    # Apply reduction
    if reduction == 'mean':
        return loss.mean()
    elif reduction == 'sum':
        return loss.sum()
    else:
        return loss
```

Figure 3.4: PyTorch code for Cauchy Negative Log Likelihood [19]

By bypassing the sampling method, the neural network (NN) can output the hyper parameters of the Evidential distribution, a higher-order distribution. This approach enables the learning of both aleatoric and epistemic uncertainties, representing model and data uncertainties, respectively. The Evidential loss is mathematically defined as:

$$loss = \left(\frac{\Gamma(\alpha - 0.5)}{4\Gamma(\alpha)\lambda\sqrt{\beta}}\right)(2\beta(1+\lambda) + (2\alpha - 1)(y_i - \hat{y})^2$$

This evidential model has four outputs such as $\hat{y}, \alpha, \beta, \lambda$
The PyTorch code for Evidential Negative Log Likelihood is given as [19],

3.2.5 Generalized Gaussian Negative Log-Likelihood

The below loss function and its explanation are from the blog[19].
The symmetric Generalized normal distribution, alternatively referred to as the exponential power distribution or the Generalized error distribution, is a specific class of symmetric probability distributions. Within this class, you can find both normal and Laplace distributions, and, in extreme cases, it encompasses all continuous uniform distributions over finite intervals along the real number line. The Generalised Gaussian Negative Log-Likelihood loss is defined as,

$$loss = \frac{(Input - Target)}{\alpha} * \beta + log\beta + log2\alpha + \Gamma(1/\beta)$$

```python
class EvidentialLoss(torch.nn.Module):
    def __init__(self, mu, alpha, beta, lamda, targets, weight=None, size_average=True):
        super(EvidentialLoss, self).__init__()
        self.mu = mu
        self.alpha = alpha
        self.beta = beta
        self.lamda = lamda
        self.targets = targets

    def forward(self,  mu, alpha, beta, lamda, targets,smooth=1):
        targets = targets.view(-1)
        y = self.mu.view(-1) #first column is mu,delta, predicted value
        loga = self.alpha.view(-1) #alpha
        logb = self.beta.view(-1) #beta
        logl = self.lamda.view(-1) #lamda

        a = torch.exp(loga)
        b = torch.exp(logb)
        l = torch.exp(logl)


        term1 = (torch.exp(torch.lgamma(a - 0.5)))/(4 * torch.exp(torch.lgamma(a)) * l * torch.sqrt(b))

        term2 = 2 * b *(1 + l) + (2*a - 1)*l*(y - targets)**2


        J = term1 * term2

        Kl_divergence = torch.abs(y - targets) * (2*a + l)


        loss = J + Kl_divergence


        return loss.mean()
```

Figure 3.5: PyTorch code for Evidential Negative Log Likelihood [19]

This distribution family gives normal distribution when the value of $\beta = 2$ (when it has mean value as μ and variance as $\frac{\alpha^2}{2}$) and it gives laplace distribution when the value of $\beta = 1$ The PyTorch code for Generalised Gaussian Negative Log-Likelihood is given as [19],

3.3 Uncertainty metrics

This study concentrates on assessing the uncertainty of Deep Neural Networks (DNNs) accurately. To achieve this, it is essential to evaluate both the prediction accuracy and the quality of estimated uncertainty. Employing numerical score-based metrics on both prediction and uncertainty, scoring rules emerge as a category of metrics that assess the effectiveness of predicted uncertainties. While most research on uncertainty estimation for regression commonly utilizes Root Mean Square Error (RMSE) to gauge model performance and Negative Log-Likelihood (NLL) to evaluate uncertainty performance, NLL is a valid scoring rule but lacks the ability to compare different distributions. Therefore, in this study, we employ the Interval score scoring rule to appraise the quality of uncertainty estimation [21].

3.3.1 Interval score

The following section regarding the calculation of the interval score for various distributions is sourced from the blog[20]:

Various methods have been discussed in the literature for comparing uncertainty, and in this research, we rely on the Interval Score, a specific type of proper scoring rule[10]. This metric involves utilizing a prediction interval with upper and lower endpoints determined by predictive quantiles at the levels of $1 - \alpha/2$ and $\alpha/2$. The values of α, such as 0.02, 0.05, and 0.1, correspond to nominal coverage at rates of 98%, 95%, and 90%, respectively.

```
def GeneralGaussianNLLLoss(input, target, alpha, beta, eps=1e-06, reduction='none'):

    input = input.view(input.size(0), -1)
    target = target.view(target.size(0), -1)
    if input.size() != target.size():
        raise ValueError("input and target must have same size")

    alpha = alpha.view(input.size(0), -1)
    if alpha.size(1) != input.size(1) and alpha.size(1) != 1:
        raise ValueError("alpha is of incorrect size")

# Second dim of scale must match that of input or be equal to 1
    beta = beta.view(input.size(0), -1)
    if beta.size(1) != beta.size(1) and beta.size(1) != 1:
        raise ValueError("beta is of incorrect size")


  # Check validity of reduction mode
    if reduction != 'none' and reduction != 'mean' and reduction != 'sum':
        raise ValueError(reduction + " is not valid")

  # Entries of var must be non-negative
    if torch.any(alpha < 0):
        raise ValueError("alpha has negative entry/entries")
  # Entries of var must be non-negative
    if torch.any(beta < 0):
        raise ValueError("beta has negative entry/entries")

  # Clamp for stability
    alpha = alpha.clone()
    beta = beta.clone()
    with torch.no_grad():
        alpha.clamp_(min=eps)
        beta.clamp_(min=eps)
    loss = (torch.abs(input - target)/alpha)**beta - torch.log(beta) + torch.log(2 * alpha )
            + torch.lgamma(1/beta)

  # Apply reduction
    if reduction == 'mean':
        return loss.mean()
    elif reduction == 'sum':
        return loss.sum()
    else:
        return loss
```

Figure 3.6: PyTorch code for Generalized Gaussian Negative Log-Likelihood [19]

To compute the intervals for different distributions, the built-in function *.interval*(*alpha*) from the Python library sympy is employed.

```
normal_dist = scipy.stats.norm(loc=outputs, scale=std)
l,u = normal_dist.interval(alpha=0.95)
print ("Interval Score ", interval_score(keypoints.numpy() , l, u))

laplace_dist = scipy.stats.laplace(loc=outputs, scale=scale)
l,u = laplace_dist.interval(alpha=0.95)
print ("Interval Score ", interval_score(keypoints.numpy() , l, u))

cauchy_dist = scipy.stats.cauchy(loc=outputs, scale=scale)
l,u = cauchy_dist.interval(alpha=0.95)
print ("Interval Score ", interval_score(keypoints.numpy() , l, u))

evidential_dist = scipy.stats.norm(loc=mu, scale=std)
l,u = evidential_dist.interval(alpha=0.95)
print ("Interval Score ", interval_score(keypoints.numpy() , l, u))

generalised_gaussian_dist = scipy.stats.gennorm(loc=outputs, scale=alpha, beta = beta)
l,u = generalised_gaussian_dist.interval(alpha=0.95)
print ("Interval Score ", interval_score(keypoints.numpy() , l, u))
```

Figure 3.7: Pytorch code for getting upper and lower limits in interval score metric [20]

Building upon the upper and lower limits obtained from the aforementioned code, the interval score is defined as per [21],

$$S_{\alpha}^{int}(l, u : y) = (u - l) + 2/\alpha(l - y)1\{y < l\} + 2/\alpha(y - u)1\{y > 5\}$$

The pytorch code for interval score metric is [20],

```
def interval_score(x, lower, upper, alpha=0.05):
    assert np.all(upper>=lower),
    return (upper - lower) + (2/alpha)*(lower-x)*(x<lower) + (2/alpha)*(x-upper)*(x>upper)
```

Figure 3.8: Pytorch code for Interval score[20]

TThe score is designed to favor accurate predictions while imposing penalties for predictions that fall outside the specified range. In this research, α is fixed at 95%. Initially, the computation involves determining the 95% quantile prediction interval (l, u). This interval is then utilized to calculate the Interval score considering the variance and mean predicted by the neural network. Consequently, if the actual value falls within the lower and upper boundaries, the score will be minimized. Achieving tighter bounds leads to better scores, as explained in [20].

3.3.2 Entropy

Entropy is a measure of uncertainty that expresses how unpredictable a model is. In the field of machine learning, entropy is often used to measure the uncertainty linked with the model's predictions. A model in machine learning assigns probability to various possibilities in order to generate predictions. In this case entropy reflects the range or spread of these probabilities. If the model is confident in its prediction, entropy remains low. conversely, if uncertainty exists then the value of entropy becomes high. In our thesis work, for the results of the grasping experiment, the best pose is determined for each loss function using entropy metric, where the lower value indicates the most certain pose. Entropy in this context, measures the uncertainty associated to various keypoint poses, with lower entropy signifying a more certain and best pose for successful grasping. The pytorch code for the entropy metric is given as ,

```
normal_entropy = scipy.stats.norm(loc=outputs, scale=std).entropy()

laplace_entropy = scipy.stats.laplace(loc=outputs, scale=scale).entropy()

cauchy_entropy = scipy.stats.cauchy(loc=outputs, scale=scale).entropy()

evidential_entropy = scipy.stats.norm(loc=mu, scale=std).entropy()

generalised_gaussian_entropy = scipy.stats.gennorm(loc=outputs, scale=alpha, beta = beta).entropy
```

Figure 3.9: PyTorch code for entropy metric

4

Experiments

This chapter provides a detailed and clear explanation of the experimental setup carried out to compare different uncertainty estimation methods for the keypoint regression task. It covers the use of two datasets in the experiments and describes in detail how the evaluations are performed, utilizing uncertainty metrics. Furthermore, the section offers a comprehensive explanation of the experimental setup involving the robot.

4.1 Experiment on UCI Datasets

As said in the previous section, we performed uncertainty estimation experiments on two different regression datasets. We start the experiment with 2D UCI regression datasets and then we are moving on to larger image datasets. The outputs of the Deep Neural Networks will be two different outputs such as predicted keypoint output and the predicted uncertainty. Firstly, both UCI regression dataset and Cornell Grasp Dataset was trained with five different metrics such as Gaussian Negative Log-Likelihood, Generalised Gaussian Negative Log-Likelihood, Laplace loss, Cauchy and Evidential loss. The results are being plotted in the form of scatter plots for UCI datasets box plot for Cornell Grasp dataset for both RMSE and Interval score.

Employing scatter plots in our analysis serves as a visual tool to represent the predictive performance of the Deep Neural Networks on the UCI regression datasets. These plots effectively illustrate the correlation between the predicted values, specifically the keypoint output, and the corresponding uncertainties generated by the model. Through this graphical representation, scatter plots offer a detailed insight into how well the model captures uncertainty in its predictions. Additionally, they enable the identification of potential patterns, outliers, or systematic errors in the model's behavior, providing valuable insights into the strengths and limitations of the employed uncertainty estimation metrics. The visual nature of scatter plots enhances the interpretability of intricate data patterns, facilitating a more thorough evaluation of the model's proficiency across various uncertainty quantification methodologies. Ultimately, the inclusion of scatter plots contributes to a comprehensive and intuitive assessment of the model's competence in handling uncertainty within regression tasks.

For UCI regression datasets, the box plot can be seen below,

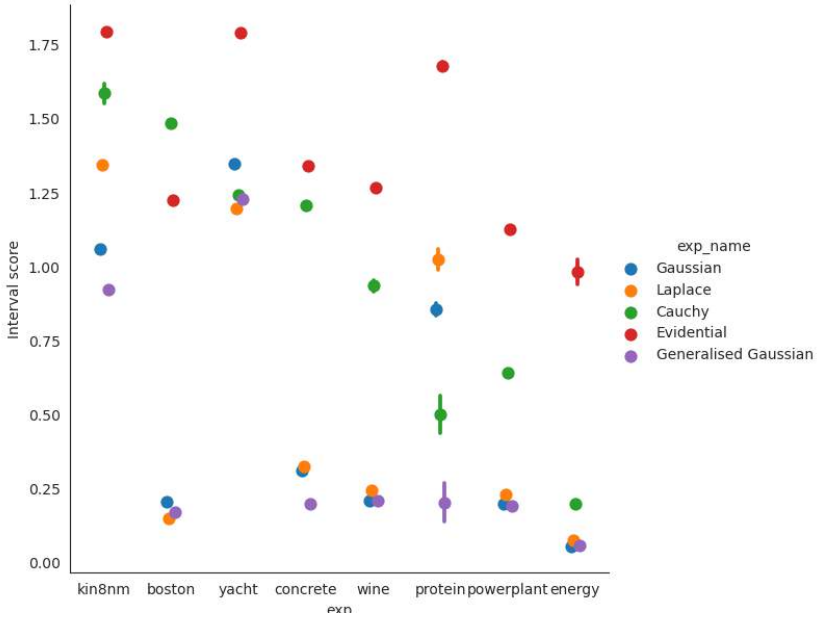


Figure 4.1: Interval score result for UCI dataset

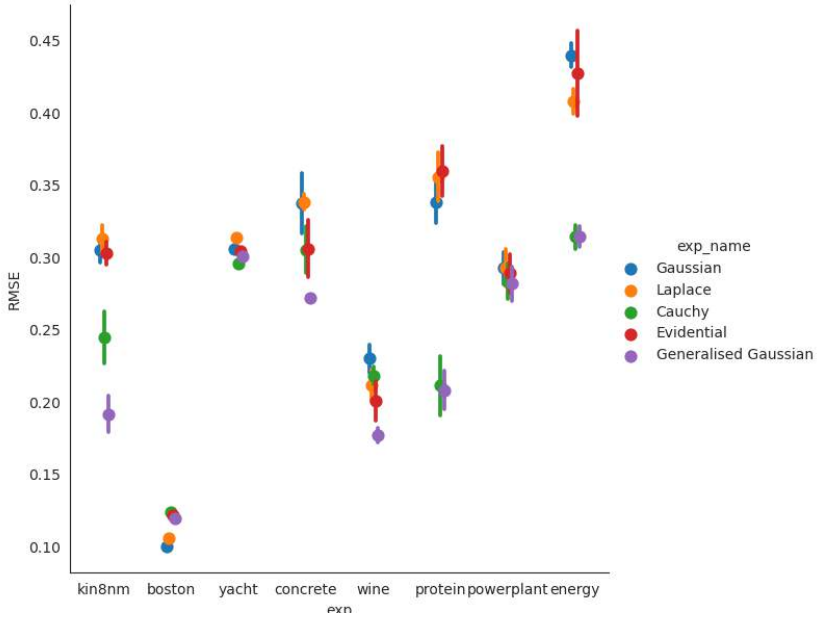


Figure 4.2: RMSE result for UCI dataset

We conducted experiments to validate the results of uncertainty estimation using real-world UCI datasets. In these experiments, we compared the Root Mean Square Error (RMSE) and the Interval score across different methods. The main objective was to assess how well our proposed method's uncertainty

estimation compared to that of various other methods when applied to real-world datasets. We utilized a fully-connected neural network with fifty hidden layers and ensured that it was trained to convergence. All models underwent training using a learning rate of 0.001 and a batch size of 128. The results depicted in Figure 3.1 and Figure 3.2, indicate that the proposed Generalised Gaussian Negative Log-Likelihood method has shown statistical comparable results in five of the eight datasets for interval score metric and six of the eight datasets for RMSE metric. In order for the clear understanding of the results, we also have attached the result table below for Interval score (Table 3.1) and RMSE (Table 3.2).

Interval score Result table					
Dataset	Gaussian	Laplace	Cauchy	Evidential	Generalised Gaussian
Kin8nm	1.062	1.346	1.586	1.796	**0.923**
Boston	0.206	0.148	1.486	**1.226**	0.169
Yacht	1.347	**1.197**	1.244	1.790	1.230
Concrete	0.310	0.323	1.207	1.341	**0.197**
Wine	0.209	0.244	0.935	1.267	**0.208**
Protein	0.856	1.026	0.500	1.679	**0.203**
Power plant	0.199	0.230	0.640	1.126	**0.191**
Energy	**0.053**	0.075	0.198	0.983	0.059

Table 4.1: Interval score Result table for UCI dataset

RMSE score Result table					
Dataset	Gaussian	Laplace	Cauchy	Evidential	Generalised Gaussian
Kin8nm	0.305	0.312	0.245	0.303	**0.191**
Boston	**0.099**	0.105	0.123	0.121	0.119
Yacht	0.305	0.314	**0.295**	0.304	0.301
Concrete	0.337	0.338	0.305	0.306	**0.271**
Wine	0.230	0.211	0.218	0.201	**0.177**
Protein	0.338	0.355	0.211	0.359	**0.208**
Power plant	0.292	0.293	0.283	0.289	**0.282**
Energy	0.440	0.408	0.317	0.427	**0.314**

Table 4.2: RMSE Result table for UCI dataset

4.2 Experiment on Cornell Dataset

In the second phase of our experimentation, we focused on uncertainty estimation within the context of a regression task specifically geared towards keypoint estimation for grasping objects. This task was carried out utilizing the Cornell Grasp Dataset, which was comprehensively introduced in Section 3. In this phase, our approach involved training a ResNet 18 model over 200 epochs. Notably, we employed five distinct loss functions during the training process, namely Gaussian Negative Log-Likelihood, Laplace Negative Log-Likelihood, Cauchy Negative Log-Likelihood, Evidential loss, and Generalised Gaussian Negative Log-Likelihood.

In this experiment, the models give an output of the predicted keypoints as well as how certain the predicted keypoints are. After the training, we saved these best models for performing the validation experiment for assessing the performance and robustness of the models and also to choose the best pose for an object for a stable grasping experiment. The results are being plotted in the form of box plots for RMSE value to assess the quality of the predictions and Interval score value to assess which metric performs uncertainty estimation better.

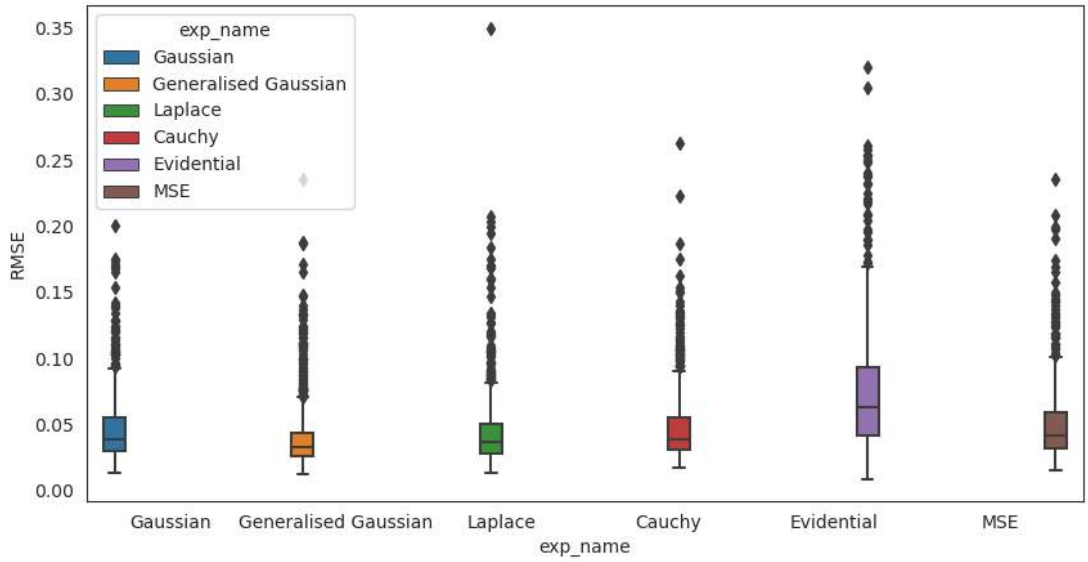


Figure 4.3: RMSE result for Cornell Grasp dataset

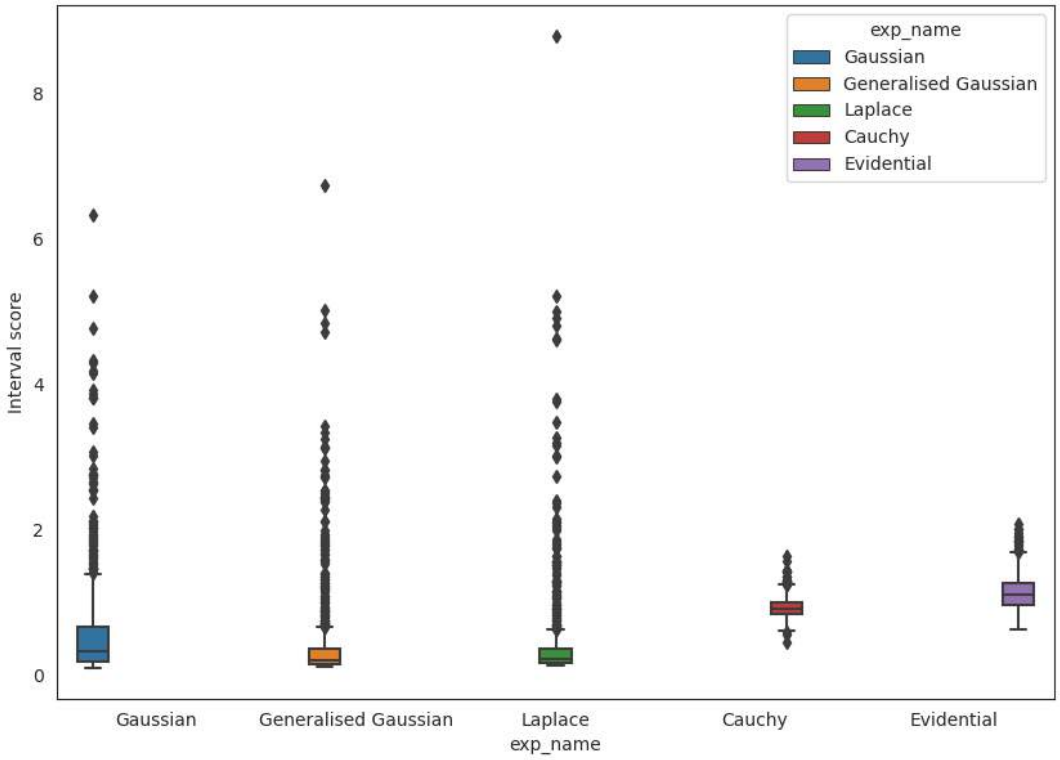


Figure 4.4: Interval score result for Cornell Grasp dataset

As observed in the depicted Figures 3.3 and 3.4, box plots are used analyze the graphs, with an emphasis on comparing the relative differences in RMSE and Interval Score for five different metrics. Lower values in both metrics signify better performance. RMSE primarily describes the quality of predicted keypoints, while Interval Score concentrates on the estimates of uncertainty.
In the box plot for comparing RMSE values, we have also included MSE for depicting the experiment without uncertainty estimation. So in brief, it compares five loss functions with uncertainty estimation and one without uncertainty estimation. This shows that Generalised Gaussian Negative Log-Likelihood has lower RMSE value and Interval score value when compared with other loss functions and hence it predicts keypoints and also estimates uncertainty in a better way. We also tabulated the results as show in the below Table 3.3,

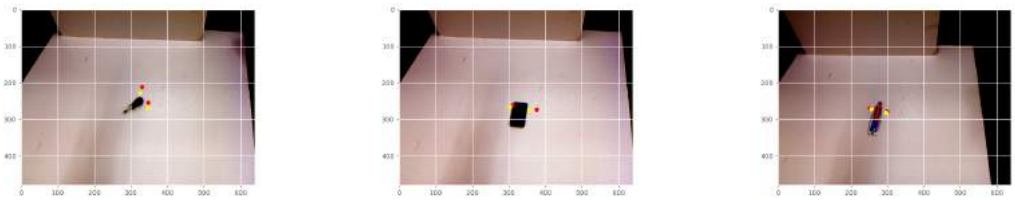
RMSE and Interval score Result table		
Dataset	RMSE	Interval Score
Gaussian	0.0469	1.7960
Laplace	0.0459	0.4348
Cauchy	0.0477	0.9202
Evidential	0.0729	1.1217
Generalized Gaussian	**0.0411**	**0.4331**

Table 4.3: RMSE and Interval Score Result table for Cornell Grasp dataset

The above table, On average the model's predictions show a difference of around 0.0411 compared to the actual values when using the Generalized Gaussian loss function. The corresponding Interval Score for Generalized Gaussian NLL is notably low at 0.4331 indicating accurate uncertainty estimation whereas on the other hand, Evidential loss function has greater RMSE and Interval score values which suggests that it gives out bad predictions of keypoints and uncertainty estimation when compared to other loss functions.

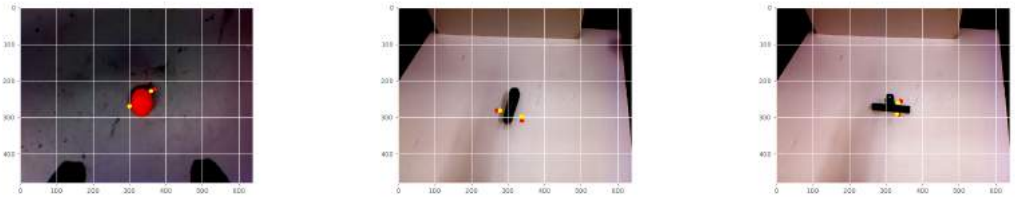
In this research we are plotting keypoints on images with the five different loss functions, and we are sorting images into groups of most and less confident images based on the entropy metric. Furthermore, we are distinguishing images by their RMSE values and categorizing them as either more error or less error images. These visualizations helps in gaining insights into the model's confidence and accuracy in predicting keypoints.

The below images are the most confident images of our five different loss functions, where yellow dots represent the actual keypoints and the red dot represents the predicted keypoints. For the most confident images where the model exhibits high certainty enables us to observe the instances where it confidently and accurately predicts keypoints. This offers valuable insights into the model's better performance in estimating the uncertainties. As concluded in the above Table 3.3 we can see that Generalized Gaussian and Laplace have less interval score values suggesting that they estimate better uncertainty estimation and hence Figure 3.8 of Generalized Gaussian and Figure 3.6 of Laplace shows us visually that the predicted keypoints align together with the actual keypoints compared to the other loss functions.



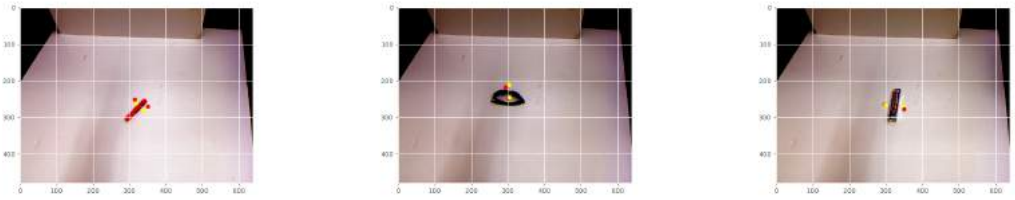
(a) (b) (c)

Figure 4.5: Most confident images in Gaussian loss



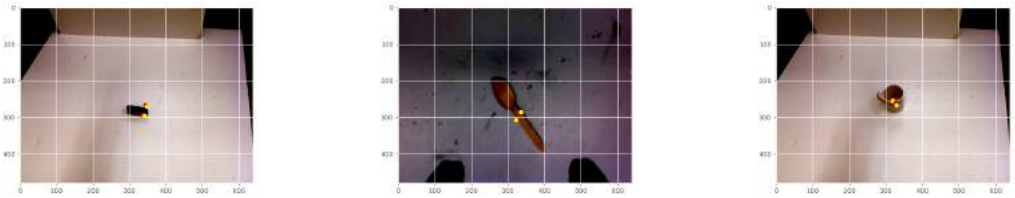
(a) (b) (c)

Figure 4.6: Most confident images in Laplace loss



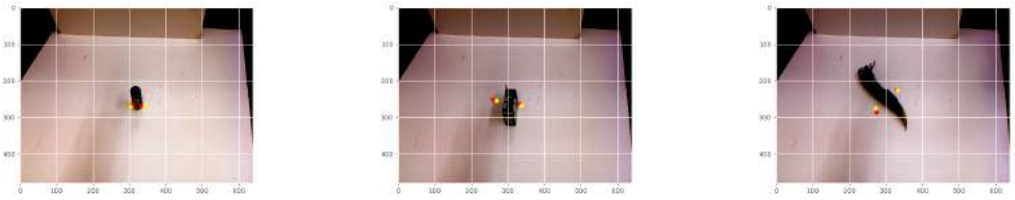
(a) (b) (c)

Figure 4.7: Most confident images in Cauchy loss



(a) (b) (c)

Figure 4.8: Most confident images in Evidential loss

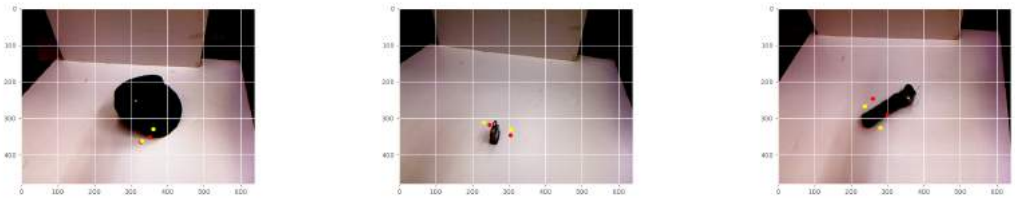


(a) (b) (c)

Figure 4.9: Most confident images in Generalized Gaussian loss

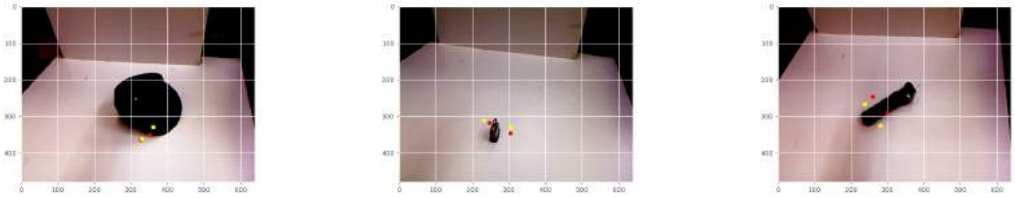
The below images are the least confident images of our five different loss functions, where yellow dots

represent the actual keypoints and the red dot represents the predicted keypoints. Analyzing less confident images provides a glimpse into situations where the model encounters difficulties or uncertainties in its predictions. This may indicate challenging or ambiguous scenarios, highlighting areas where the model faces challenges. We can observe that in all these images the predicted keypoints do not coincide with the actual keypoints. From the above Table 3.3 and Figure 3.13, Evidential loss function has high interval score and RMSE values which can be seen in the images that their two predicted keypoints are plotted in the same pose giving us inaccurate predictions compared to other loss functions.



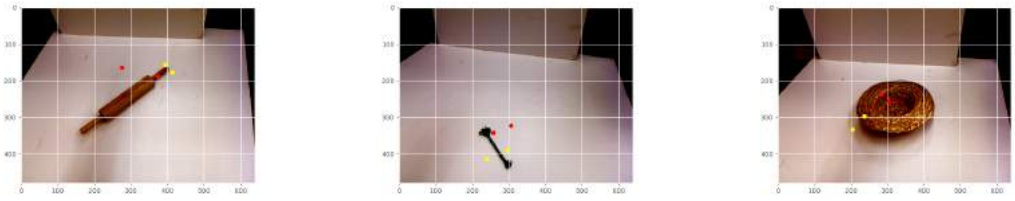
(a) (b) (c)

Figure 4.10: Least confident images in Gaussian loss



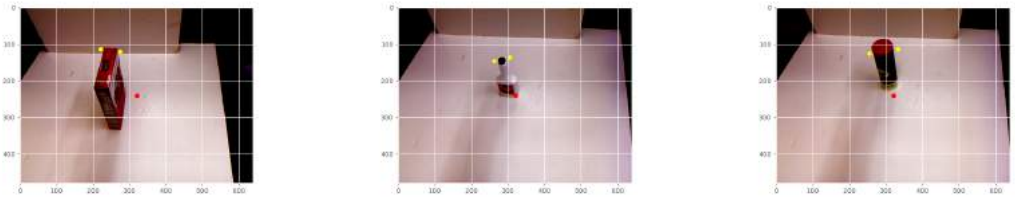
(a) (b) (c)

Figure 4.11: Least confident images in Laplace loss



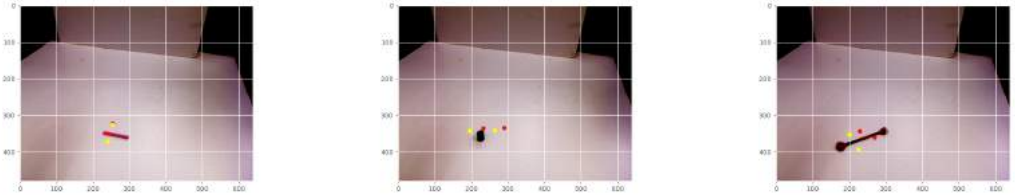
(a) (b) (c)

Figure 4.12: Least confident images in Cauchy loss



(a) (b) (c)

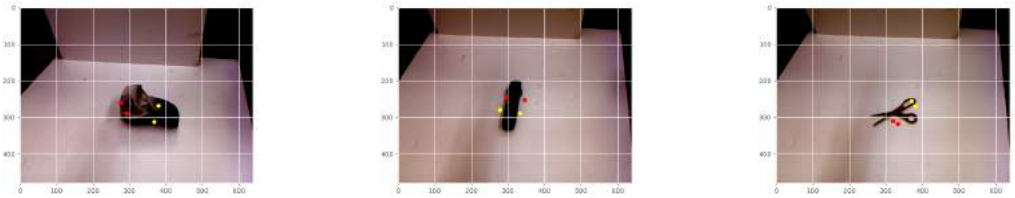
Figure 4.13: Least confident images in Evidential loss



(a) (b) (c)

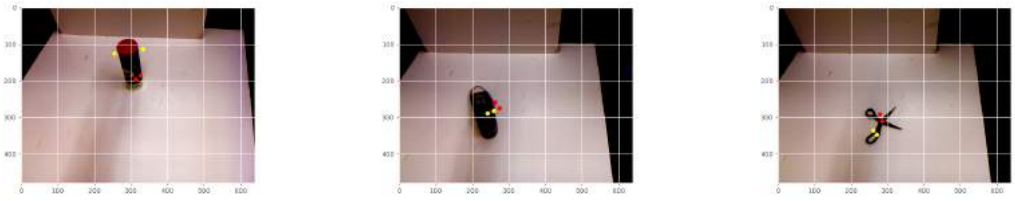
Figure 4.14: Least confident images in Generalized Gaussian loss

The below images are the most error images of our five different loss functions, where yellow dots represent the actual keypoints and the red dot represents the predicted keypoints. Most error images provides insights into areas where the model's predictions significantly deviate from the actual keypoints. We can observe that in all these images the predicted keypoints do not coincide with the actual keypoints. From the above Table 3.3 and Figure 3.18, Evidential loss function has high interval score and RMSE values which can be seen in the images that their two predicted keypoints are plotted in the same pose giving us inaccurate predictions compared to other loss functions.



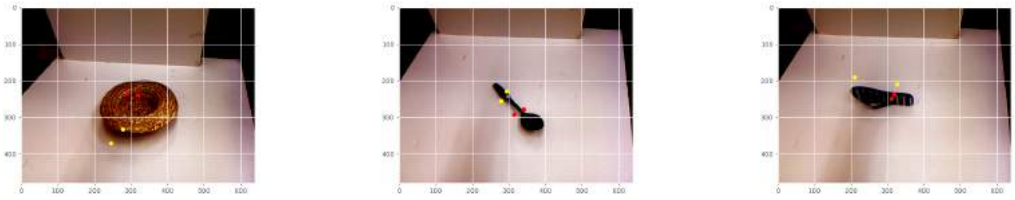
(a) (b) (c)

Figure 4.15: Most error images in Gaussian loss



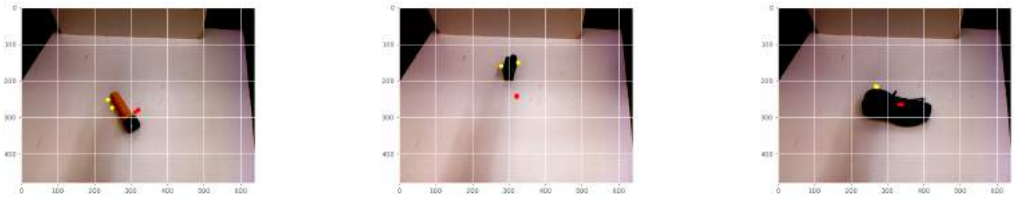
(a) (b) (c)

Figure 4.16: Most error images in Laplace loss



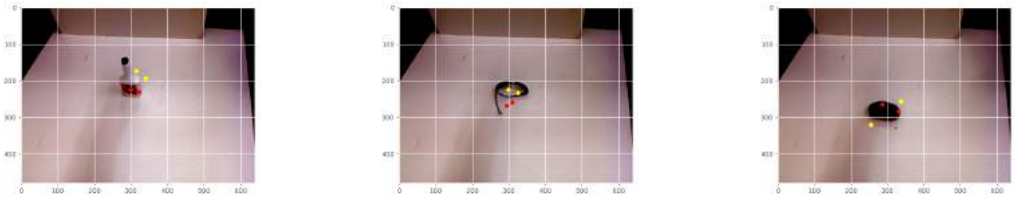
(a) (b) (c)

Figure 4.17: Most error images in Cauchy loss



(a) (b) (c)

Figure 4.18: Most error images in Evidential loss

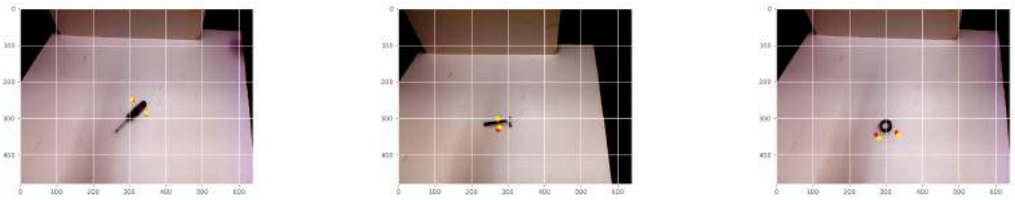


(a) (b) (c)

Figure 4.19: Most error images in Generalized Gaussian loss

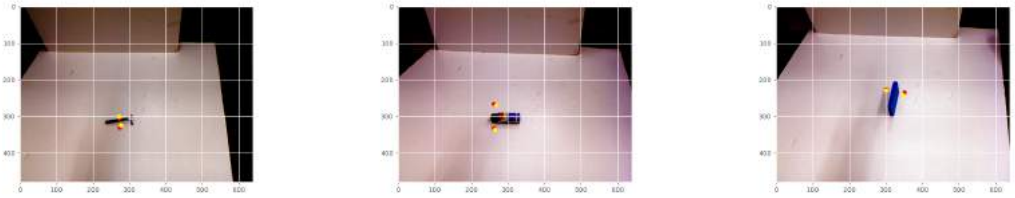
The below images are the less error images of our five different loss functions, where yellow dots

represent the actual keypoints and the red dot represents the predicted keypoints. Less error images highlight situations where the model's predictions closely align the actual keypoints. These visualizations act as a means of validation, affirming the model's effective performance in specific scenarios.. As concluded in the above Table 3.3 we can see that Generalized Gaussian and Laplace have less RMSE score values suggesting that they predict accurate keypoints and hence Figure 3.24 of Generalized Gaussian and Figure 3.21 of Laplace shows us visually that the predicted keypoints align together with the actual keypoints compared to the other loss functions.



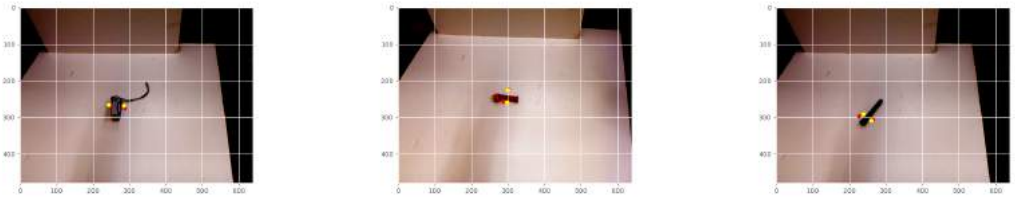
(a) (b) (c)

Figure 4.20: Less error images in Gaussian loss



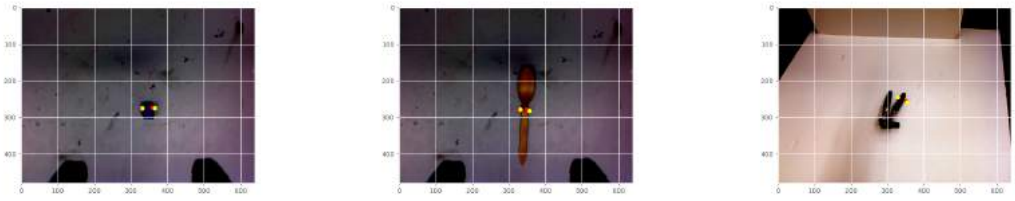
(a) (b) (c)

Figure 4.21: Less error images in Laplace loss



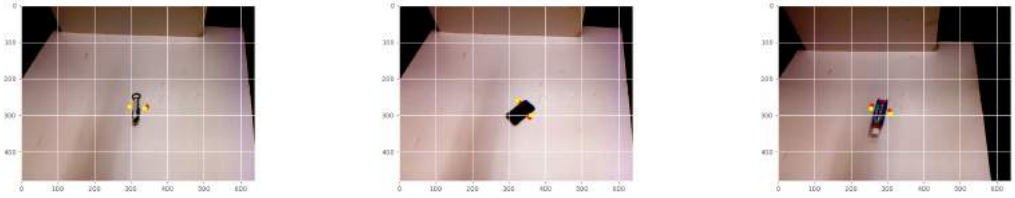
(a) (b) (c)

Figure 4.22: Less error images in Cauchy loss



(a) (b) (c)

Figure 4.23: Less error images in Evidential loss



(a) (b) (c)

Figure 4.24: Less error images in Generalized Gaussian loss

4.3 Robot Grasping experiment

As shown in the below Figure 3.5, The flowchart outlines the steps involved in developing and deploying a machine learning model for empowering a Kinova robotic arm in object grasping. The process is structured into two primary sections: Model development and evaluation and robotic arm manipulation.

4.3.1 Model Development and Evaluation

- Cornell Grasp Data: The procedure initiates with the Cornell Grasp Dataset, comprising images of objects annotated with grasp points. This dataset serves as the training data for instructing the machine learning model to identify potential grasp points on diverse objects.

- Train a model: The model undergoes training using the dataset, wherein its parameters are adjusted to accurately predict grasp points on objects.

- Loss Functions: Different loss functions are used during training to measure the difference between the model's predictions and the actual labeled data. These functions, such as Gaussian, Laplace, Cauchy, Evidential, and Generalized Gaussian helps in optimizing the model.

- Evaluation of model: The trained model is evaluated using interval Score metric which assess the confidence intervals of the model's predictions that are crucial for reliability tasks.

- Saved UE model: After evaluation, the final model, referred to as the UE (Uncertainty Estimation) model, is saved. This model accounts for uncertainty in its predictions, which is important for real-world applications.

4.3.2 Robotic Arm Manipulation

- Camera: The camera captures images of the target object to be grasped.
- Keypoint Detection: The saved UE model processes the camera input to identify keypoints on the object, representing unique points in an object for understanding its orientation and shape.
- Detection to Object pose estimation: Using the detected keypoints the model estimates the pose of the object in the real world coordinates, which includes its location and orientation in space.
- Manipulation: The robotic arm uses the pose estimation to manipulate the object adjusting its movements based on the model's predictions.
- Grasping: Finally, the robotic arm executes the grasping action, achieving the ultimate objective of the process.

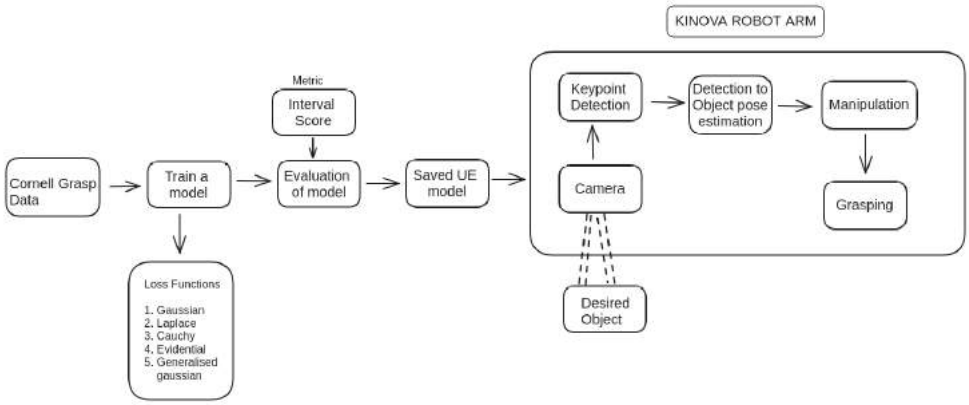


Figure 4.25: Experimental setup for robot experiment

Experimental setup

Goal: To illustrate the application of keypoints in Uncertainty estimation for robot gripping application.

- In this experiment, we are using Kinova robot arm setup which has an end effector attached to it as shown in Figure 3.6. In our case, it is a gripper used for grasping the single objects.

Figure 4.26: Kinova robot arm with single object

- Since we are focusing on cornell grasp dataset, we propose using the following single objects for grasping purposes as shown in Figure 3.7. The objects used for grasping are: markers, torch light, mouse, hair brush, stapler, tooth paste, duster, cube, box and allenset.

Figure 4.27: Grasping objects

- A camera is being mounted at the end-effector of the robot in such a way that it faces down and

has a clear vision on the objects and we project the center of the robot flange/tool onto the ground plane i.e. the tool is vertically facing down.

- The objects are placed on a horizontal surface/table with white background and also on an area with good reachability.

4.3.3 Experiment and evaluation

- Each object is placed at a known initial position and orientation with respect to the robot and these objects are placed in two different poses such as in vertical and horizontal position.
- Following the manual placement of the object as outlined previously and with access to either the observed scene (such as RGB/RGB-D data) or the known object model and its pose, we proceed to activate the grasp planner and carry out the optimal grasp, which is determined based on the best pose we got it from our uncertainty estimation method and they are performed N number of times repeatedly.
- The application of uncertainty estimation on grasping application can be described as in Figure 3.8 where in the context of having an image of an object that needs to be grasped, a deep neural network is employed thoroughly to predict various possible keypoints. This process results in the prediction of eight possible grasp keypoints in the form of x and y. Following this, we use an uncertainty metric to identify the best optimal keypoint for robot grasping application.

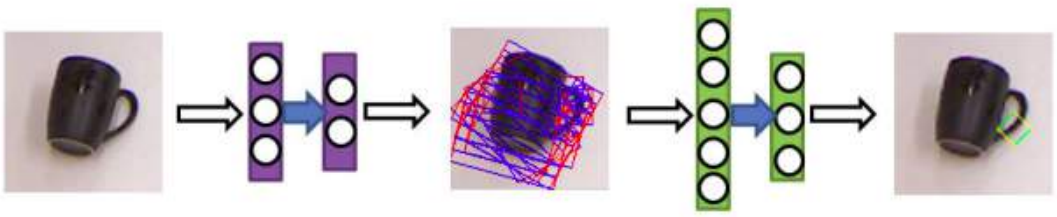


Figure 4.28: Uncertainty estimation on robot grasping [16]

- As demonstrated in the works [3], [5], [26], [29] which employ a top-down grasping approach, our experiment also adopts a similar top-view strategy to grasp single objects positioned on the table.
- For executing the grasp planning process, we have an RGB-D image of 640*480 pixels. The keypoints are in the form of (x,y) values. As discussed above, using our uncertainty estimation method we select an optimal 2D pose for grasping which is then converted into 3D real world coordinates. In order to execute the grasping in real world using grasp keypoints in image coordinates, the grasp must undergo series of known transforms,

$$G_r = t_{rc}(t_{CI}(G_i)) \tag{4.1}$$

Here, t_{CI} represents the conversion from 2D image coordinates to the 3D camera frame using established camera intrinsics, while t_{rc} symbolizes the transition from the camera frame to the world or robot frame and G_i is the the most effective grasp configuration in the form of x and y.

- From the paper [3], The results can be evaluated in the form of s/a, where s denotes the successful grasps counts and $a = 2N$ where N is the number of grasp attempts for a single object for all the objects and the reason for 2 is that in this experiment we are considering two different poses for an object for grasping such as vertical and horizontal placements.

- We consider the following circumstances as instances of trial failures: If the gripper fingers displaces the object from its position while attempting to grasp it; If the object slides or moves away during the execution of the grasp or when lifting the grasped item.

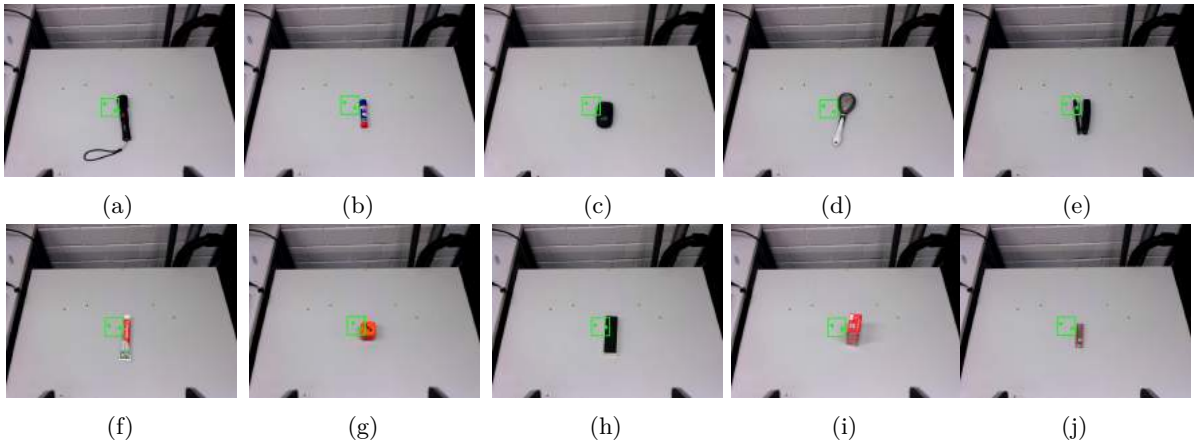
- This experiment can be used for finding the best pose for grasping an object using an uncertainty estimation methods and it can be evaluated based on the above discussion.

- A final success rate (normalized average) based on all trials and poses can be reported from these results for each object.

4.3.4 Grasping Image Results

For the results of the grasping experiment, the best pose is determined for each loss function using the metric called entropy, where the lower value indicates the most certain pose. Entropy in this context, measures the uncertainty associated to various keypoint poses, with lower entropy signifying a more certain and best pose for successful grasping. Upon loading the best model, the 2D pose is transformed into a 3D pose, and the RVIZ visualization displays the grasping region with a bounding box. A bounding box is crucial in grasping experiments as it precisely outlines the targeted region for effective object manipulation. The robot then utilizes the best pose, depicted as a keypoint, to maneuver and grasp the designated area of the object within the bounding box.

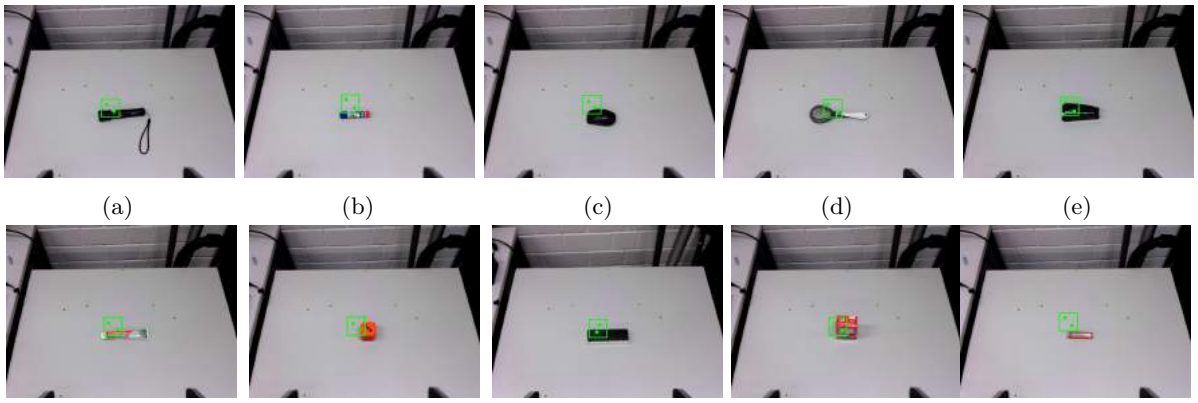
For all the five loss functions, we have attached the best pose prediction images with boudning box and the best pose keypoints of an uncertainty estimation model for both vertical and horizontal postiions. In the below Figure 3.9 and 3.10, we have attached the best pose prediction of Gaussian uncertainty estimation model. The bounding boxes are plotted to all 10 grasping objects, but certain items such as hairbrush, box, mouse, and cube do not coincide with the actual grasping region. This indicates that, during the attempts to grasp the objects, only a limited number were successful due to inaccurate predictions.



(a) (b) (c) (d) (e)

(f) (g) (h) (i) (j)

Figure 4.29: Best pose prediction using Gaussian UE model for vertical position

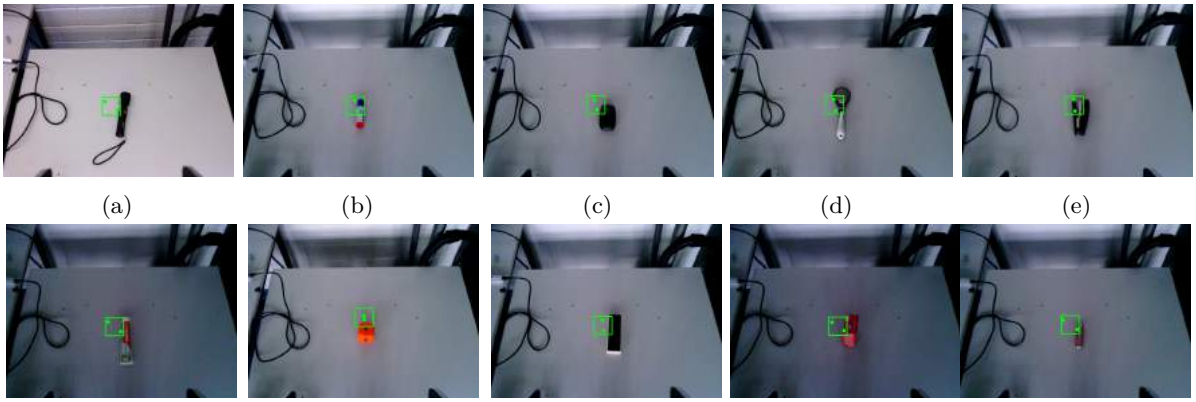


(a) (b) (c) (d) (e)

(f) (g) (h) (i) (j)

Figure 4.30: Best pose prediction using Gaussian UE model for horizontal position

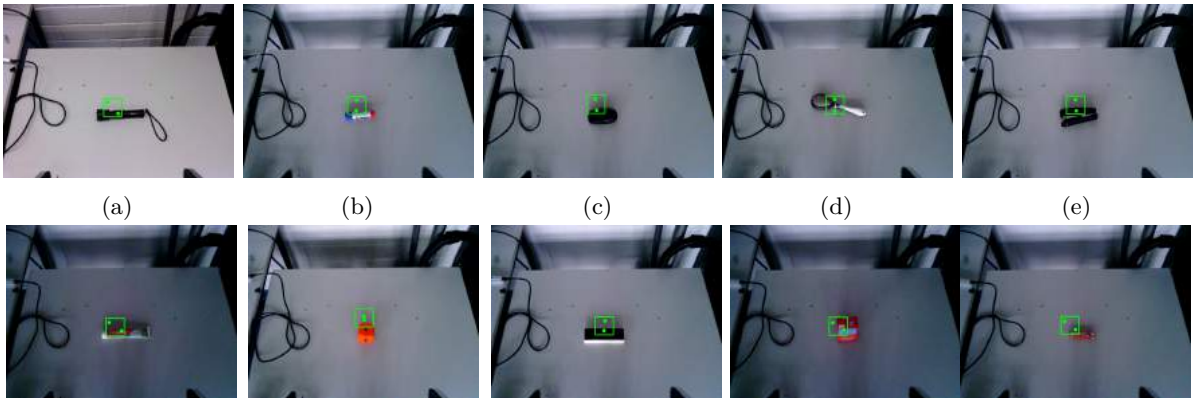
In the below Figure 3.11 and 3.12, we have attached the best pose prediction of Lapalce uncertainty estimation model. The bounding boxes are plotted to all 10 grasping objects, but most of the items in the lists coincide with the actual grasping region. This indicates that, during the attempts to grasp the objects, most of the grasping attempts were successful due to more accurate predictions. Hence, Laplace uncertainty estimation model contribute to better predictions of the best pose keypoints because of its ability to handle outliers with heavy-tailed distribution nature.



(a) (b) (c) (d) (e)

(f) (g) (h) (i) (j)

Figure 4.31: Best pose prediction using Laplace UE model for vertical position

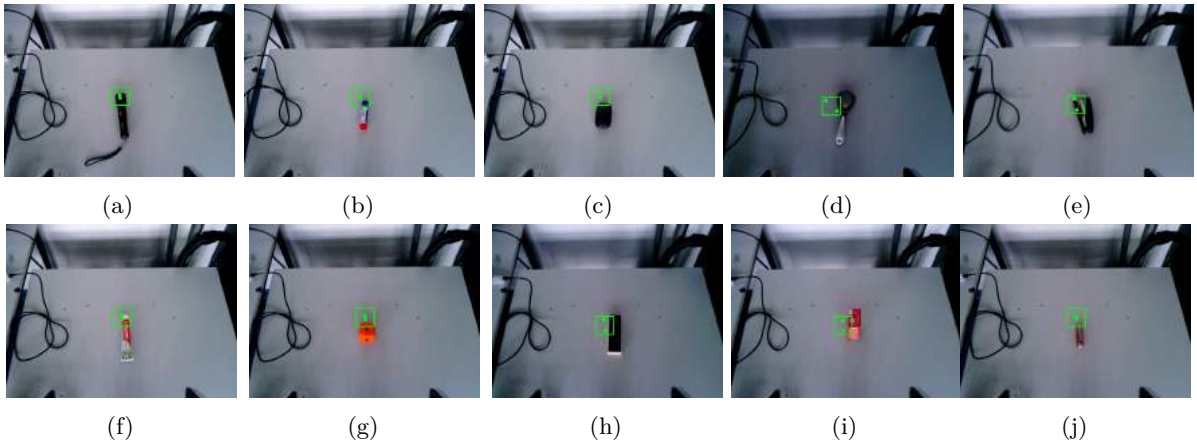


(a) (b) (c) (d) (e)

(f) (g) (h) (i) (j)

Figure 4.32: Best pose prediction using Laplace UE model for horizontal position

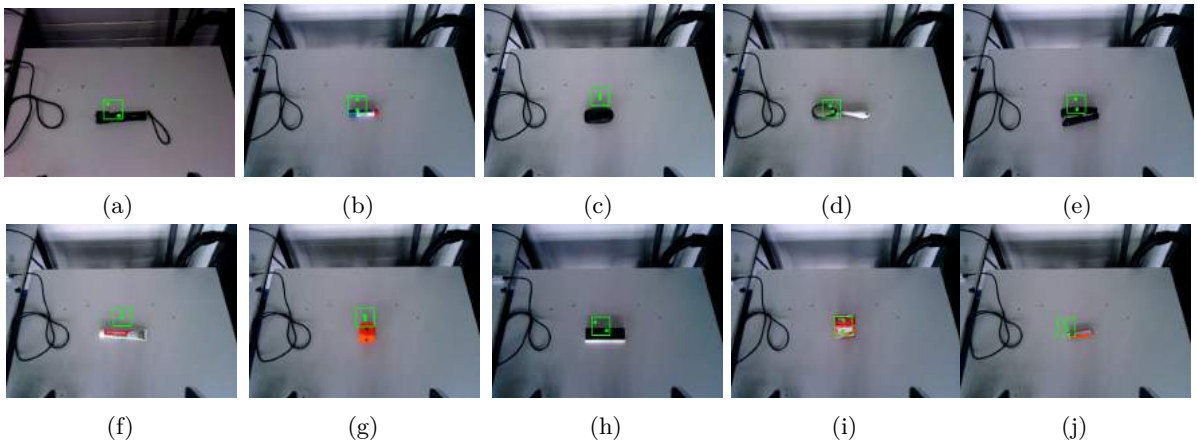
In the below Figure 3.13 and 3.14, we have attached the best pose prediction of Cauchy uncertainty estimation model. The bounding boxes are plotted to all 10 grasping objects, but certain items such as marker, stapler, mouse and box do not coincide with the actual grasping region. This indicates that, during the attempts to grasp the objects, only a limited number were successful due to inaccurate predictions. Despite the Cauchy distribution being known for its heavy tails, its performance is poorer compared to Laplace distributions. This attributed to inaccurate predictions and unsuccessful attempts to grasp various objects.



(a) (b) (c) (d) (e)

(f) (g) (h) (i) (j)

Figure 4.33: Best pose prediction using Cauchy UE model for vertical position

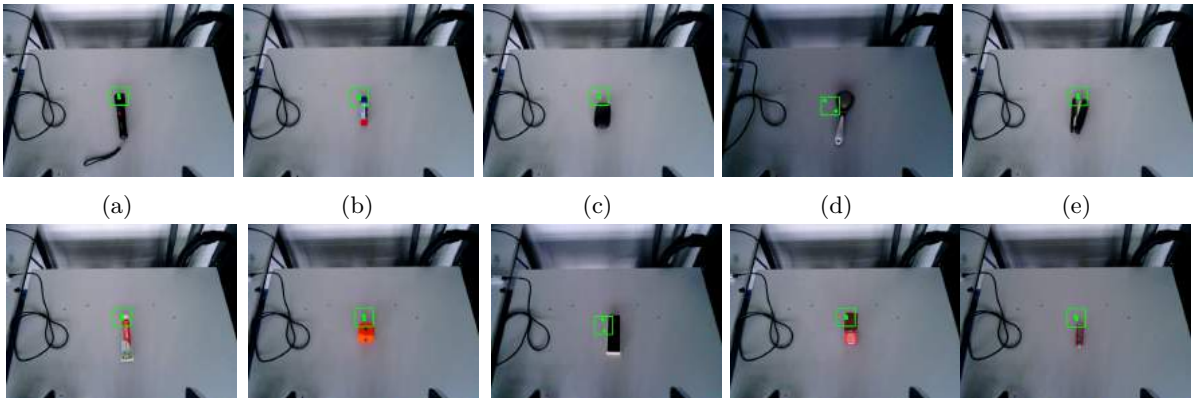


(a) (b) (c) (d) (e)

(f) (g) (h) (i) (j)

Figure 4.34: Best pose prediction using Cauchy UE model for horizontal position

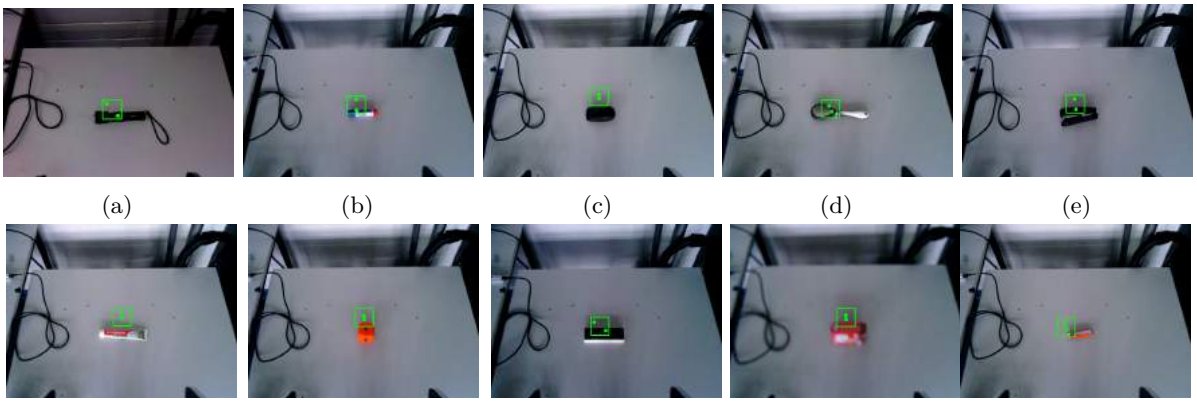
In the below Figure 3.15 and 3.16, we have attached the best pose prediction of Evidential uncertainty estimation model. The bounding boxes are plotted to all 10 grasping objects, but most of the items in the object lists do not coincide with the actual grasping region. This indicates that, during the attempts to grasp the objects, only a limited number were successful due to inaccurate predictions. The Figures 3.3 and 3.4 clearly explains that Evidential loss function has high RMSE and Interval score values which leads to the inaccurate predictions and unsuccessful attempts to grasp various objects.



(a) (b) (c) (d) (e)

(f) (g) (h) (i) (j)

Figure 4.35: Best pose prediction using Evidential UE model for vertical position

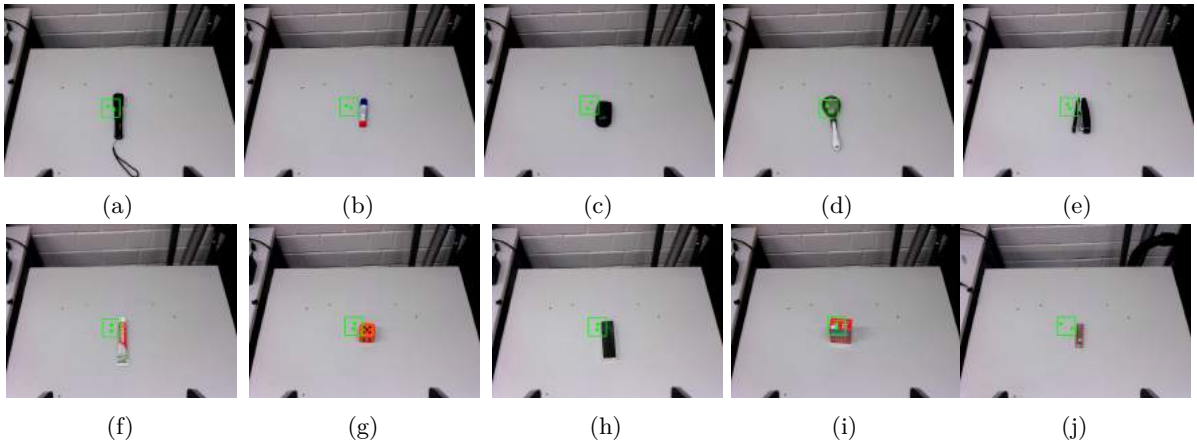


(a) (b) (c) (d) (e)

(f) (g) (h) (i) (j)

Figure 4.36: Best pose prediction using Evidential UE model for horizontal position

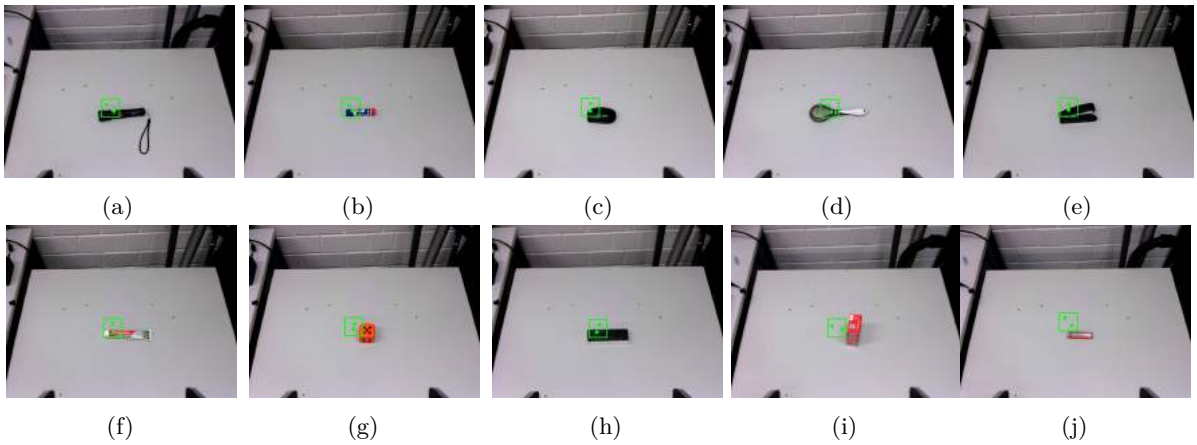
In the below Figure 3.17 and 3.18, we have attached the best pose prediction of Generalised Gaussian uncertainty estimation model. The bounding boxes are plotted to all 10 grasping objects, but most of the items in the object lists coincide with the actual grasping region. This indicates that, during the attempts to grasp the objects, most of the attempts were successful due to its accurate predictions. The Figures 3.3 and 3.4 clearly explains that Generalized Gaussian as lower RMSE value and Interval score value when compared with other loss functions and hence it predicts keypoints and also estimates uncertainty in a better way. This led to the many successful attempts to grasp objects in two different positions.



(a) (b) (c) (d) (e)

(f) (g) (h) (i) (j)

Figure 4.37: Best pose prediction using Generalized Gaussian UE model for vertical position



(a) (b) (c) (d) (e)

(f) (g) (h) (i) (j)

Figure 4.38: Best pose prediction using Generalized Gaussian UE model for horizontal position

4.3.5 Grasping experiment Evaluation ans Results

- The below Table 3.4, provides a detailed breakdown of the success rates for different grasping methods applied to various objects.
- **P1** represents the vertical position at which the object has been placed for grasping.
- **P2** represents the horizontal position at which the object has been placed for grasping.
- The entries in the table indicate the number of successful grasps corresponding to each combination of object and uncertainty model. These values represent the number of successful grasps out of a total of 10 attempts (N).

Objects	**G Gaussian**		**Gaussian**		**Laplace**		**Cauchy**		**Evidential**	
	P1	P2	P1	P2	P1	P2	P1	P2	P1	P2
Torch light	9	5	6	2	8	5	5	0	4	0
Marker	9	4	7	1	7	2	5	1	5	1
Mouse	4	3	2	2	3	2	2	3	4	3
Hair brush	5	4	4	3	6	4	3	1	3	0
Stapler	6	7	1	4	3	5	1	3	1	2
Toothpaste	8	8	4	4	8	7	5	5	4	3
Cube	6	6	5	5	6	6	5	5	5	5
Duster	6	5	4	2	7	5	2	1	3	0
Box	9	4	7	3	9	3	7	2	7	1
Allenset	10	5	4	2	9	2	8	2	8	3

Table 4.4: Result table for Grasping experiment

- From the paper [2], The results can be presented in terms of s/a, where s denotes the number of successful grasps and a = 2N where N is the number of grasp attempts for a single object for all the objects and the reason for 2 is that in this experiment we are considering two different poses for an object for grasping such as vertical and horizontal placements.
- On adding up the poses separately and applying the formula, we get the following results:

Method	Success Rate
Generalized Gaussian	6.15
Gaussian	3.6
Laplace	5.8
Cauchy	3.3
Evidential	3.1

Table 4.5: Success Rates for Different Grasping Methods

- The results indicate that Generalized Gaussian yields the highest success rate of 61.5 % for grasps whereas Evidential yields the lowest success rate of 31 %. The similarity between Generalized Gaussian results and Laplace distribution is explained by the choice of a beta value less than 2 by the Generalized Gaussian distribution function, which leads the distribution to adopt a form closely resembling the Laplace distribution.
- Therefore, based on the Kinova robot arm grasping experiment, it is inferred that using a Generalized Gaussian Negative Log-Likelihood function for uncertainty estimation outperforms other loss functions in predicting keypoints and estimating uncertainty. This resulted in the highest success rate of 61.5 % when grasping various objects.

5

Conclusions

In the realm of real-world robot grasping applications, the uncertainty estimation helps in improving the ability of the robot to adapt to uncertain environments and make more informed decisions. This ensures robust performance and elevating the overall task's reliability. On understanding the important role of Uncertainty Estimation in robotic systems, our research initiated with a comparison of Uncertainty Estimation using UCI regression 2D datasets. As we progressed, we explored into the high dimensional 3D image Dataset called Cornell Grasping Dataset. We used five distinctive loss functions—Gaussian Negative Log-Likelihood, Laplace Negative Log-Likelihood, Cauchy Negative Log-Likelihood, Evidential loss and Generalized Gaussian Negative Log-Likelihood. Our approach involved modeling Neural Networks to produce outputs that not only include predicted values but also incorporate uncertainty. After comparison of various Uncertainty Estimation methods, we subsequently translated our findings into practical implementation on a real-time Kinova robot arm. As a result of our extensive exploration, this thesis concludes that the use of Generalized Gaussian Negative Log-Likelihood excels in predicting accurate keypoints along with better uncertainty estimates, demonstrating its efficacy in real-world robotic grasping tasks.

5.1 Contributions

- Literature survey on "Uncertainty Estimation in Robot Grasping", exploring the datasets, assessing the evaluation metrics that has been used in the paper and exploring the various approaches taken in grasping experiments.
- Recording bag files for various objects and manually annotating them to identify potential keypoints for grasping the objects.
- Comparison of Uncertainty estimation methods for 2D UCI regression Dataset and 3D Cornell Grasp Dataset.
- Explored the use of a metric called entropy to identify the optimal pose for grasping an object.
- Performed experiments involving five distinct uncertainty estimation models, conducting 10 trials for each pose in each model.

5.2 Lessons learned

- It is crucial for a Neural Network to provide both predictions and distributions, as these distributions enhance the accuracy of uncertainty prediction and its likelihood.
- In comparing different uncertainty estimation methods, metrics play a significant role. Negative Log-Likelihood (NLL) is not suitable as an uncertainty metric for comparing various distributions. Therefore, this research employs proper scoring rules to effectively compare uncertainties.
- Uncertainty estimation should not be limited to model development; it should also be explored through real-time robot experiments to enhance adaptability and decision-making in practical scenarios.

5.3 Future work

- The scope of this research can be broadened to include other heavy-tail distribution loss functions, enabling an assessment of the model's performance across different uncertainty estimation methods.
- In this study, we conducted only 10 trials for each model and pose. Future work could involve increasing the number of objects in the datasets and assessing results with a larger number of objects and trials, given that we currently used only 10 objects for this research.
- Adjusting uncertainty estimation models to accommodate dynamic environments, wherein scenarios involve moving objects or changing surroundings for the robot.

Bibliography

[1] Haleh Akrami et al. “Robust variational autoencoder”. In: *arXiv preprint arXiv:1905.09961* (2019).

[2] Alexander Amini et al. “Deep evidential regression”. In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 14927–14937.

[3] Yasemin Bekiroglu et al. “Benchmarking protocol for grasp planning algorithms”. In: *IEEE Robotics and Automation Letters* 5.2 (2019), pp. 315–322.

[4] Lorenzo Bertoni, Sven Kreiss, and Alexandre Alahi. “Monoloco: Monocular 3d pedestrian localization and uncertainty estimation”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 6861–6871.

[5] Hu Cao et al. “Lightweight convolutional neural network with Gaussian-based grasping representation for robotic grasping detection”. In: *arXiv preprint arXiv:2101.10226* (2021).

[6] Fu-Jen Chu, Ruinian Xu, and Patricio A Vela. “Real-world multiobject, multigrasp detection”. In: *IEEE Robotics and Automation Letters* 3.4 (2018), pp. 3355–3362.

[7] Michael Dusenberry et al. “Efficient and scalable bayesian neural nets with rank-1 factors”. In: *International conference on machine learning*. PMLR. 2020, pp. 2782–2792.

[8] Yarin Gal and Zoubin Ghahramani. “Bayesian convolutional neural networks with Bernoulli approximate variational inference”. In: *arXiv preprint arXiv:1506.02158* (2015).

[9] Yarin Gal and Zoubin Ghahramani. “Dropout as a bayesian approximation: Representing model uncertainty in deep learning”. In: *international conference on machine learning*. PMLR. 2016, pp. 1050–1059.

[10] Tilmann Gneiting and Adrian E Raftery. “Strictly proper scoring rules, prediction, and estimation”. In: *Journal of the American statistical Association* 102.477 (2007), pp. 359–378.

[11] Danijar Hafner et al. “Reliable uncertainty estimates in deep neural networks using noise contrastive priors”. In: *stat* 1050 (2018), p. 24.

[12] Yun Jiang, Stephen Moseson, and Ashutosh Saxena. “Efficient grasping from rgbd images: Learning using a new rectangle representation”. In: *2011 IEEE International conference on robotics and automation*. IEEE. 2011, pp. 3304–3311.

[13] Kaggle. “Cornell grasp Dataset”. In: *https://www.kaggle.com/datasets/oneoneliu/cornell-grasp/data* (2018).

[14] Alex Kendall and Yarin Gal. “What uncertainties do we need in bayesian deep learning for computer vision?” In: *Advances in neural information processing systems* 30 (2017).

[15] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. “Simple and scalable predictive uncertainty estimation using deep ensembles”. In: *Advances in neural information processing systems* 30 (2017).

[16] Ian Lenz, Honglak Lee, and Ashutosh Saxena. "Deep learning for detecting robotic grasps". In: *The International Journal of Robotics Research* 34.4-5 (2015), pp. 705–724.

[17] Jiefeng Li et al. "Human pose regression with residual log-likelihood estimation". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 11025–11034.

[18] Tong Li et al. "Keypoint-based robotic grasp detection scheme in multi-object scenes". In: *Sensors* 21.6 (2021), p. 2132.

[19] Deebul Nair. In: *URL https://deebuls.github.io/devblog/categories/uncertainty* (2018).

[20] Deebul Nair. In: *URL https://deebuls.github.io/devblog/statistics/uncertainty/2022/12/15/comparing-Normal-Laplace-Cacuhy-Distirbutions-Interval-score.html* (2018).

[21] Deebul S Nair, Nico Hochgeschwender, and Miguel A Olivares-Mendez. "Maximum Likelihood Uncertainty Estimation: Robustness to Outliers". In: *arXiv preprint arXiv:2202.03870* (2022).

[22] Yusuke Niitani et al. "Sampling techniques for large-scale object detection from sparsely annotated objects". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 6510–6518.

[23] David A Nix and Andreas S Weigend. "Estimating the mean and variance of the target probability distribution". In: *Proceedings of 1994 ieee international conference on neural networks (ICNN'94)*. Vol. 1. IEEE. 1994, pp. 55–60.

[24] Abdul Sattar Safaei, Saba Farsad, and Mohammad Mahdi Paydar. "Robust bi-level optimization of relief logistics operations". In: *Applied Mathematical Modelling* 56 (2018), pp. 359–380.

[25] Qi She, James T. Kwok, and Yuan Yao. "Deep Robust Regression via Bi-level Optimization". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 33. 01. 2019, pp. XXX–XXX.

[26] Priya Shukla et al. "Generative model based robotic grasp pose prediction with limited dataset". In: *Applied Intelligence* (2022), pp. 1–15.

[27] Jost Tobias Springenberg et al. "Bayesian optimization with robust Bayesian neural networks". In: *Advances in neural information processing systems* 29 (2016).

[28] UCI. "UCI Machine Learning Repository". In: *https://archive.ics.uci.edu/* (2018).

[29] Ruinian Xu, Fu-Jen Chu, and Patricio A Vela. "Gknet: grasp keypoint network for grasp candidates detection". In: *The International Journal of Robotics Research* 41.4 (2022), pp. 361–389.

[30] Haiyue Zhu et al. "Grasping detection network with uncertainty estimation for confidence-driven semi-supervised domain adaptation". In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2020, pp. 9608–9613.