Hochschule
Bonn-Rhein-Sieg
University of Applied Sciences

b-it Bonn-Aachen International Center for Information Technology

R&D Project

Uncertainty Estimation for Keypoint Detection Task : Robustness Study

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
Mr.Deebul Nair

January 2023

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 everyday scenarios, we deal with uncertainties in numerous applications where Deep Neural Networks have the ability to estimate uncertainty along with the network prediction. However the output of the Deep Neural Networks is stochastic in nature which involves randomness and uncertainties. This research aims to focus on robustness of deterministic uncertainty estimation methods to augmented data for regression tasks using uncertainty metrics. The datasets used for this research are facial keypoint dataset [12] and biwi dataset[11]. The networks show degrading performance and incorrect uncertainty estimation when we use augmented data. This research focuses on using different metrics to improve robustness and is evaluated using high dimensional regression task of keypoint detection. In order to evaluate the uncertainty of the methods, we use Interval score metric [20]. In particular, this research proposes that the use of Cauchy Negative Log Likelihood metric provides better uncertainty estimation along with the robustness experiments.

Acknowledgements

I would like to sincerely thank my first supervisor Prof.Dr.Nico Hochgeschwender and my second supervisor Mr.Deebul Nair for providing me this great opportunity to work in this deep learning project. I would like to acknowledge and thank Mr.Deebul Nair for his continous insights and support throughout the project which helped me to gain more knowledge in this field.
Finally, I would like to give special thanks to my parents and friends for their support.

Contents

List of Figures

List of Tables

1

Introduction

Deep learning is an artificial intelligence (AI) function that mimics the human brain's processing and pattern-making processes to aid decision-making. Key point detection is one such pattern-making process used by the brain to make decisions. A keypoint or feature in an image which can be defined as an unique meaningful structure in that image, however it is semantically ill-defined, in the sense that it is unclear what keypoints are relevant for any given input image. More advanced computer vision tasks such as structure from motion, object recognition and SLAM which is popularly known as simultaneous localization and mapping, content-based retrieval and image matching depends on keypoint detection and description methods. New keypoint detection and description approaches have arisen as a result of the advent of deep learning methods, particularly convolutional models,as they are claiming to outperform traditional algorithms on benchmarks. Deep learning models are expected to learn abstract image features from large datasets whereas Convolutional models learn about features by supervision rather than handcrafting them, hence their performance is strongly reliant on ground truth data.
In everyday scenarios, we deal with uncertainties in numerous fields, from invest opportunities and medical diagnosis to sporting games and weather forecast, with an objective to make decision based on collected observations and uncertain domain knowledge [18]. Understanding the uncertainty of a neural network's (NN) predictions is essential for many applications. Epistemic uncertainty, which depends on the model and caused by a shortage of training data, and aleatoric uncertainty, which is generated by noise and uncertainty in the data and is thus irreducible, are the two forms of uncertainty that are frequently of importance.[6].

1.1 Motivation

In safety-critical and autonomous systems, the Deep Neural Network's ability to estimate predicted output of the network along with the uncertainty has been considered as the most relevant feature for DNN adoption [4]. Estimating accurate uncertainties plays a crucial role in decision making processes[9]. All of the uncertainty estimating techniques have only ever been tested on clean datasets, however obtaining clean annotated datasets for use in practical applications is challenging. As a result, even after human annotation, real-world datasets are still affected by significant label noise. The performance of all these methods for the regression work in the presence of augmented data has not been well explored, despite the fact that they have all been tested for their prediction and uncertainty estimations for the

classification task. The performance of all the uncertainty estimation methods for the regression work in the presence of augmented data has not been well explored, despite the fact that they have all been tested for their prediction and uncertainty estimations for the classification task. In this research, we focus on the challenging problem of accurately learning uncertainty in the presence of augmented data while maintaining robustness.

1.2 Problem Statement

The output of DNNs are stochastic in nature which involves randomness and uncertainties. In high-risk applications, identifying such uncertainties plays a very crucial role. Deep neural networks (DNN) have become popular in safety-critical and autonomous systems because of their capacity to estimate uncertainty along with network prediction. Accurate and calibrated uncertainty can be used to build confidence in autonomous systems decision-making. In safety-critical and autonomous systems, the Deep Neural Network's ability to estimate predicted output of the network along with the uncertainty has been considered as the most relevant feature for DNN adoption [4]. Estimating accurate uncertainties plays a crucial role in decision making processes[9]. Due to the lack of ground truth, uncertainty estimation is a difficult task, especially in high dimensional data. The challenge of estimating uncertainty increases when there are noisy labels or outliers in the training data. The ability of learning algorithms to properly learn uncertainty by ignoring the outliers is known as robust uncertainty estimation.
So in this R&D, we look into the problem of the robustness capacity of uncertainty estimation methods for high dimensional regression task.

1.2.1 Research Questions

There are several methods which can do uncertainty estimation, In this research we focus on estimating uncertainty based on three uncertainty estimation methods that includes Gaussian, Laplace and Cauchy distribution methods which is evaluated using the complex regression task such as Facial Keypoint estimation.
Furthermore, this work aims to answer the following research questions.

- Comparison of uncertainty estimation methods for keypoint detection task?
- Comparison of uncertainty estimation methods for keypoint detection task for a new loss function called Cauchy Negative Log Likelihood [21]?
- Robustness study on uncertainty estimation methods for keypoint detection task?
- Out-Of-Distribution on uncertainty estimation methods for keypoint detection task?

1.2.2 Contributions

In this research, the robustness capacity of uncertainty estimation methods with four different loss functions are trained and evaluated using different uncertainty metric and Out-Of-Distribution data. Hence, the contributions of this work are:

- The robustness capacity of the uncertainty estimation method is improved by modeling different loss functions.
- The robustness capacity of the uncertainty estimation is evaluated using complex regression task such as Keypoint detection.
- The predicted uncertainty is evaluated for Out-Of-Distribution (OOD) detection to check the reliability of the model.

2

State of the Art

This chapter focuses on state of the art methods for different uncertainty estimation methods, keypoint detection datasets, keypoint detection tasks and also on different types of metrics that has been used for estimating uncertainties in a regression task.

2.1 Uncertainty estimation for regression

In deep learning one of the most preferred approaches for uncertainty estimation is Bayesian approach and ensembles. In Bayesian approach the weights of the neural networks are replaced by the parametric distributions and it had prohibitively high computational cost. In deep ensembles instead of getting single output we get a set of distributional parameters. Uncertainty Estimation methods also differ based on number of forward passes required by an approach. Methods such as [16] require only one forward pass whereas methods like [1] require several passes of an input through a network to estimate uncertainty. Most approaches for estimating uncertainty in deep learning based on en- sembling or Monte Carlo sampling. A new method is introduced to estimate uncertainty in one forward pass[29]. As a result, even after human annotation, real-world datasets contain significant label noise

2.2 Robustness study Uncertainty estimation for regression:

For regression tasks, this paper [22] proposes the use of a heavy-tailed distribution (Laplace distribution) to improve the robustness to outliers. This paper [1] describes about well-calibrated learning measures of different benchmarks of uncertainty on challenging tasks of computer vision domain and also work on thwir robustness criteria. A new model [3] for estimating uncertainty in one forward pass that works on both regression and classification problems. Combining bi-Lipschitz feature extractor with point approximate Gaussian process this results in good robustness and principled uncertainty estimation. Proposed method can allow changes in Deep kernel Learning(DKL) to match with softmax neural networks accuracy. This method overcomes previous work's limitations by addressing determining uncertainty quantification. They demonstrate the performance of DUE on regression on personalized healthcare.

2.3 Keypoint detection datasets

The process of locating key object elements is known as KeyPoint detection. The eyebrows, nose tips and eye corners, for example, are important features of our faces. These components aid in the

representation of the underlying object in a feature-rich way. Pose estimation, face detection, and more appli- cations of KeyPoint detection exists in the KeyPoint detection datasets. Some of the common keypoint datasets are explained in this section. The whole survey of kepoint detection datasets are tabulated in the below table 2.1.

KEY POINT DATASETS		
S.No	NAME OF THE DATASET	DESCRIPTION
1	KeypointNet	human annotations, based on ShapeNet models
2	Facial Keypoint Detction	Unique Facial features detection
3	COCO	object detection
4	SynthHands	hand pose estimation
5	UMD datasets	face dataset
6	Biwi	Kinect Head Pose dataset
7	CASIA-face-africa	A Large-scale African Face Image Database
8	SVIRO	Synthetic Vehicle Interior Rear Seat Occupancy Dataset
9	CarFusion	Combining Point Tracking and Part Detection for Dynamic 3D Reconstruction of Vehicles
10	ApolloCar3D	A Large 3D Car Instance Understanding Benchmark for Autonomous Driving
11	Rendered Handpose Dataset	hand pose estimation
12	CrowdPose	Efficient Crowded Scenes Pose Estimation
13	3DFAW	face dataset
14	H3D (Humans in 3D)	Retrieving data. Wait a few seconds and try to cut or copy again
15	AwA Pose	animal keypoint dataset
16	ATRW	Amur Tiger Re-identification in the Wild
17	Bizarre Pose Dataset	Bizarre Pose Dataset of Illustrated Characters

Table 2.1: Survey of Datasets

2.3.1 KeypointNet:

By utilising multiple human annotations with 3D models and almost 83k keypoints from sixteen object categories, KeypointNet is a sizable and varied 3D keypoint dataset. It is based on ShapeNetmodels and includes 83,231 keypoints from sixteen object categories in 3D models.

2.3.2 COCO:

The MS COCO dataset (Microsoft Common Objects in Context) is a popular dataset for segmentation,object detection, captioning and key-point detection. There are 328K photos in the dataset.detection of keypoints: about 250,000 human instances and 200,000 photos have been tagged with keypoints (17 possible keypoints, such as nose,left eye,right ankle,right hip)

2.3.3 SynthHands:

This dataset contains genuine hand motion that has been changed to a virtual hand with real environment and interactions with various objects for hand posture estimation. The dataset includes data for both female and male hands, both with and without object engagement. The foreground and hand object were synthesized using Unity. Realistic background and textures of the images (in terms of depth and color) were also utilised. For 21 keypoints on the hand, ground truth 3D locations are presented.

2.3.4 UMD datasets:

The face dataset UMD Faces is divided into two parts: Facial annotations for 8,277 people in still images

Over three million video frames which are annotated from over 22,000 videos featuring subjects.

Part 1-Still images:

The dataset is broken into three batches and contains face annotations for almost eight thousand subjects. Human-curated bounding boxes for estimated position (yaw, pitch, and roll) and faces are included in the annotations, as are the gender information and twenty-one keypoints locations which are built by a pre-trained neural network.

Part 2-Video frames:

The second section contains annotated video frames for almost 3000 subjects, which were retrieved from a total of 22,000 frames. The positions of twenty-one keypoints, estimated pose (pitch, yaw and roll)and gender information that are created by a trained neural network are all included in the annotations.

2.3.5 CASIA-face-africa:

Face recognition is a well-studied and popular topic with numerous applications in contemporary culture. However, most State Of The Art (SOTA) face recognition systems have been shown to have racial prejudice. Many studies on facial recognition algorithms have found that African people have a

higher percentage of false positives than other subjects.
To this goal, they assemble the CASIA-Face-Africa face picture library, which contains 38,546 photos of 1,183 African people. Face photos are captured using multispectral cameras in a variety of lighting conditions. The individuals' demographic characteristics and facial expressions are also meticulously recorded. Each face image in the database is individually tagged with 68 facial keypoints for landmark detection. Different applications, tasks, partitions, and scenarios are used to create a set of evaluation procedures.

2.3.6 SVIRO:

SVIRO is a synthetic dataset for scenes in the passenger chamber of 10 different automobiles that was created to test machine learning-based techniques for their generalization and dependability when trained on a small number of variants. In contrast, most used benchmark datasets, which helps on enhancing the state-of-the-art of common tasks, have a significant level of inherent variability. For each synthetic scenery, this dataset includes bounding boxes for keypoints for posture, depth photos estimation,instance segmentation masks and object detection.

2.3.7 CarFusion:

Despite much research, reconstructing multiple dynamic rigid objects (such as cars) from wide-baseline, uncalibrated, and unsynchronized cameras remains difficult. In this paper[27], the authors provide a framework for fusing multi-view feature tracks and single-view part locations to be detected for improving moving vehicle localization, identification and reconstruction, even in the presence of heavy occlusions. We show our approach by reconstructing over 62 vehicles going through a congested traffic crossroads in a 3-minute interval. We assess the various components of our system and compare them to alternative approaches such as tracking-by-detection reconstruction.

2.3.8 ApolloCar3D:

Both industry and academia are paying close attention to autonomous driving. Estimating 3D features of a moving or parked car on the road (e.g. translation, shape and rotation) is a crucial problem. Each automobile is fitted with an 3D CAD model industry-grade with absolute model size and semantically marked keypointshe collection contains 5,277 driving photos and over 60K car instances. This dataset is more than 20 times larger than the existing state-of-the-art datasets PASCAL3D+ and KITTI.

2.3.9 Rendered Handpose Dataset:

There are 41258 training samples and 2728 testing examples in the Rendered Handpose Dataset. Each sample contains the following information:
Image in RGB and Depths map (320x320 pixels)
Background, person,each finger has three of each classes, and each hand's palm segmentation masks with same pixels. The number of keypoints in the image is 21 which has both xyz frame and uv coordinates with intrinsic camera matrix of k.

2.3.10 CrowdPose:

About 20,000 photos and 80,000 human poses with 14 identified keypoints make up the CrowdPose dataset. There are 8,000 photos in the test collection. MSCOCO, MPII, and AI Challenger are used to extract congested photos containing homes.

2.3.11 3DFAW :

The computer vision and machine learning communities have shown a growing interest in automated facial alignment during the last 15 years. Face alignment is one of the major problem in automatically tracking down precise facial landmarks across a variety of subjects, viewpoints and illuminations which is critical to all face analysis applications, including identification, action unit analysis, facial expression and as well as many multimedia applications and human-computer interactions.
There are 23 thousand photos in 3DFAW with 66 3D face keypoint annotations.

2.3.12 H3D (Humans in 3D):

H3D (Humans in 3D) is a human-annotated dataset. The joints and other keypoints are among the annotations (eyes, ears, nose, shoulders ans so on.)
The keypoints were used to infer a 3D posture. Each keypoint has a visibility boolean. Annotations to the regions (upper clothes, sunglasses, lower clothes,hat, gloves, dress, shoes, socks, hands, neck, face, hair, occluder, bag) Type of body (male, female or child)

2.3.13 AwA Pose :

The AwA Pose dataset [2] is a large-scale keypoint dataset with annotation particularly with their ground truth of animal dataset for keypoint detection of quadruped animals from photos.

2.3.14 ATRW:

This dataset has videos of 8000 amur tigers with their annotation of animals for particularly identified as tigers, their keypoints of their locations along with the bounding box.

2.3.15 Bizarre Pose Dataset:

Character illustrations in anime/manga style with a human keypoint dataset. The AnimeDrawings-Dataset has been extended to include the following features:
All 17 human keypoints that are COCO compatible
Bounding boxes for characters
Danbooru provided an extra 2000 examples (4000 total) with tough tags.

2.4 Keypoint detection tasks

Many pattern recognition applications, including 3D modeling and 3D object identification, depend on the computation of similarity across three-dimensional (3D) surfaces [17]. In order to create a comprehensive 3D model of an item, 3D modeling compares 3D surfaces taken from various angles, aligns them, and merges them. The objective of 3D object identification [5], on the other hand, is to accurately identify the nature and position of objects in a scene[30]. Both of these tasks have numerous uses in industries including robotics [28],scene comprehension[30], reverse engineering[31] and biometric systems[13]. Human pose estimation (HPE) also known for human keypoint detection means the location of keypoints that are in the human body can be detected and their categories can be recognized for a given image. It is very helpful in various applications such as recognition of activities[15], human–robot interaction [19] and video surveillance[8].

2.5 Uncertainty estimation for keypoint detection

In this paper[14] 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. In the paper[3] they tackle the fundamentally ill-posed problem of 3D human localization from monocular RGB images. Driven by the limitation of neural networks outputting point estimates, they address the ambiguity in the task by predicting confidence intervals through a loss function based on the Laplace distribution.

From all the above survey, we are planning to use the Facial keypoint detection and Biwi dataset for our experiment on four different loss functions such as Gaussian Negative Log Likelihood, Laplace Negative Log Likelihood, Cauchy Negative Log Likelihood and Evidential loss. The reason for choosing this loss functions is that these functions make Neural Network output not only the predicted output but also the distributions. These distributions makes one to understand better uncertainty and their

likelihood and effects. These methods are made to train and validate in Resnet-18 model as it helps in improving the performance.

3

Methodology

This chapter explains the experimental setup that has been carried out for comparing different uncertainty estimation using for regression task. The datasets that are used for performing this experiment and for evaluating those experiments uncertainty metrics are used in this work which are explained in detail in this section.

3.1 Datasets

3.1.1 Face Keypoint Detection Dataset

Face-related applications constitute a significant part in computer vision. Deep learning is indirectly driving the face-related applications as well because it is at the forefront of many computer vision-based applications. Face recognition, facial keypoint detection, and even security and surveillance applications are just a few examples. Therefore, having some familiarity with using deep learning and computer vision to face-related applications is a good idea.
It's interesting to note that the dataset doesn't include any raw image files. All the dataset is included in 2 CSV files. There are 31 columns in total. The face keypoint coordinates are in the first 30 columns. Now observe that the columns are arranged according to each face part's x and y coordinates. After that, it goes to the right eye. The Image column, which contains all the pixel values for a 96x96 resolution in the format of gray-scale image, is the last column. The final field of the data files contains the input image, which is a list of pixels (sorted by row), as numbers in (0,255). We'll need to eventually reshape these values in order to use it properly.
Some of the images in the dataset from training.csv file are:

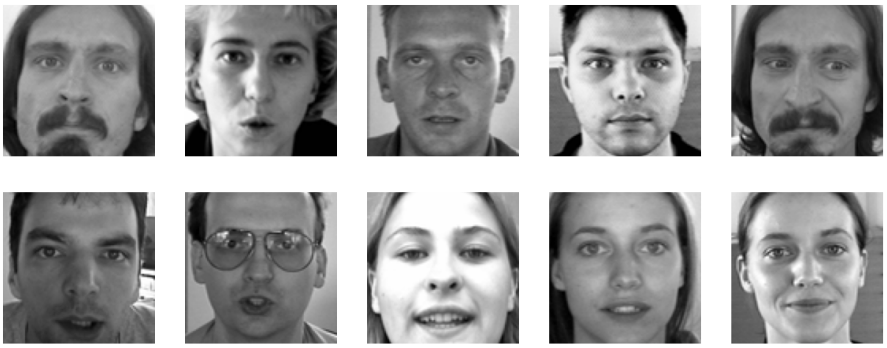


Figure 3.1: Face images from the dataset [26].

In the above figure, all the images are entirely in gray-scale..
The following figure images with corresponding keypoints.

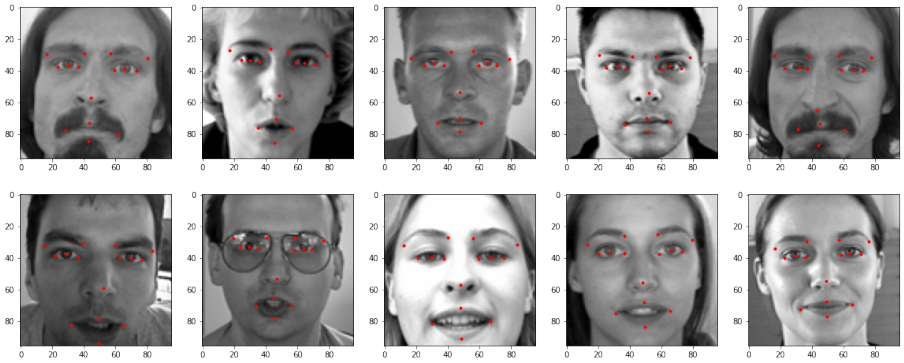


Figure 3.2: Images from the keypoint detection dataset with keypoints[26].

As seen in the above figure, each face has 15 keypoints plotted in them.

3.1.2 BIWI Dataset

Eye and mouth detection and tracking are the foundation of many facial feature detection methods. But unlike the other prominent facial features , the nose is less susceptible to changes in appearance. For example, When a person is wearing glasses or when his eyes/mouth is open or in the closed conditions. Prior to face recognition, nose detection may be crucial for detecting the face or aligning the face. Even if the face is partly occluded, the nose can be highly helpful for face tracking of the disabled. As real-time

working is frequently a constraint in the global application, nose detection in a complex computer vision tasks emphasize the importance of a low processing time. It is captured using a structured IR light device, the Microsoft Kinect sensor. With RGB (640 × 480) and depth maps 640 × 480), it has roughly 15k frames. The recordings included twenty subjects, and four of them were captured twice for a total of 24 sequences. Along with the head center and the calibration matrix, the ground truth of the yaw, pitch, and roll angles is provided.

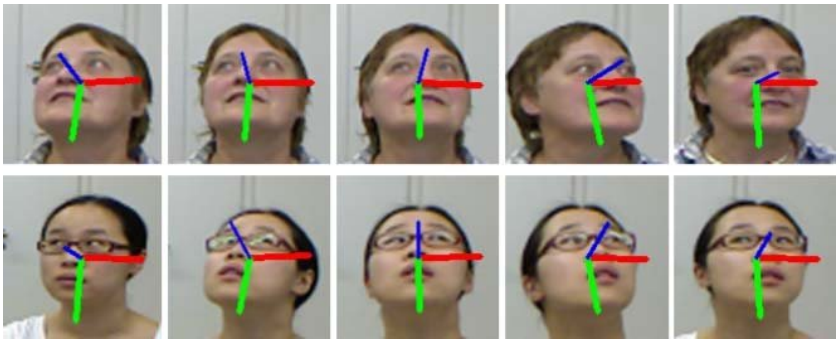


Figure 3.3: BIWI dataset with their annotations at different angles[24].

3.2 Model

Mostly in order to handle a challenging problem, additional layers are stacked in the Deep Neural Networks to improve accuracy and performance. The idea behind layering is that as more layers are added, they will eventually learn features that are more complicated. By enabling the shortcut for the gradient to flow through,the skip connection in ResNet models helps to solve the vanishing gradient problem in Deep Neural Networks. It has been observed that residual blocks in this model makes layers learn identity functions exceptionally easy. In this research, a Convolutional Neural Network(CNN) with 18 layers called ResNet-18 is used for improving the performance.

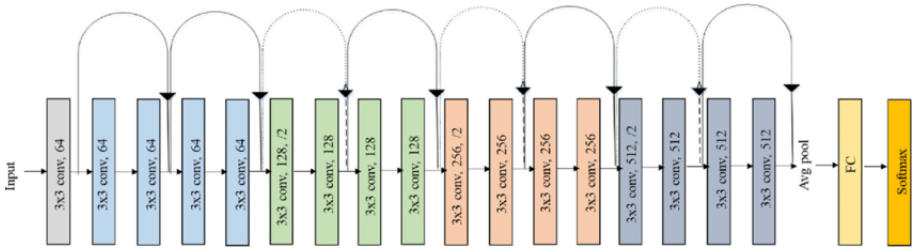


Figure 3.4: Architecture of ResNet-18 model[25].

3.3 Metrics

The performance of the model is evaluated using following metrices:

- Gaussian negative log likelihood

- Laplace negative log likelihood
- Cauchy negative log likelihood
- Evidential loss

3.3.1 Gaussian negative log likelihood

The below loss function and its explanation are from the blog[23]. In Gaussian NLL Loss, the targets in the regression datasets are considered as data from Gaussian distributions, with the NN predicting the variances and expectations. In the case of Gaussian loss, additionally it also helps to predict the confidence in the prediction particularly in the form of Gaussian variance. The Gaussian distribution is in the form of [23],

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

On taking natural log on both sides and then adding a negative sign, we get the following loss function[23]

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

where eps in the above mentioned loss formula is used for stability. Unless full is True, the constant term which is found in the above loss function is removed by default.
For Gaussian loss function, the neural network model predicts the prediction along with the confidence in the prediction which is in the form of Gaussian variance [21] 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 3.5: Neural Network model for Gaussian Negative Log Likelihood

3.3.2 Laplace negative log likelihood

The below loss function and its explanation are from the blog[22]. In LaplaceNLLLoss, the targets in the regression datasets are considered as samples from Laplace distributions, with the neural network predicting the scale and expectations. The Laplace distribution is defined as follows,

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

In this research, Negative Log Likelihood(NLL) of Laplace distribution is used for training the Neural Networks. So,

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

The loss function has various properties for gradient based optimization- Since the Laplace loss is a convex function, the convexity makes sure that the loss is minimum when $|y - \mu|$ is minimum and also it is found to increase monotonically with respect to $|y - \mu|$.
The PyTorch code for Laplace Negative Log Likelihood is given as [21],

```
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 3.6: PyTorch code for Laplace Negative Log Likelihood [21]

3.3.3 Cauchy negative log likelihood

The below loss function and its explanation are from the blog[21]. In CauchyNLLLoss, the targets in the regression datasets are considered as samples from Cauchy distributions, with the neural network

predicting the scale and expectations. The Cauchy distribution is defined as follows,

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

In this research, Negative Log Likelihood(NLL) of Cauchy distribution is used for training the Neural Networks. So,

$$c_{NLL}(x, y, \theta) = -log(p(y|f_\theta^\mu, f_\theta^s) = log(3.14 f_\theta^s) + log1 + (\left(\frac{|y - f_\theta^\mu|^2}{f_\theta^{s2}}\right)$$

The PyTorch code for Cauchy Negative Log Likelihood is given as [21],

```
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 3.7: PyTorch code for Laplace Negative Log Likelihood [21]

3.3.4 Evidential loss

The below loss function and its explanation are from the paper[1].
Evidential deep learning conceptualizes learning as a process of evidence acquisition. All of the data in training adds evidence to a evidential distribution which is known as higher form of distribution has been learned. Evidential techniques makes sure that over likelihood function it directly places priors on contrast to Bayesian NNs which are meant to place them as network weights. By not considering the sampling method, the NN can be made to give output the evidential distribution's hyper parameter which is a higher order distribution and the ground truth of both aleatoric and the model uncertainty known as

epistemic uncertainty can be learned. It can be able to model the model and data uncertainties if we constitute the model's output with a higher order data distribution. The evidential loss is 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 [21],

```
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 3.8: PyTorch code for Evidential Negative Log Likelihood [21]

3.4 Uncertainty metrics

As this research focuses on the Uncertainty estimation of DNNs to correctly estimate the uncertainty, there is a need to access the quality of prediction along with the quality of estimated uncertainty. By applying a numerical score based metrics on both the uncertainty and prediction, scoring rules are a class of metrics which evaluates the how good these predicted uncertainties are. The majority of the work on uncertainty estimation for regression uses Root Mean Square Error (RMSE) to compare model performance and uses Negative Log-Likelihood(NLL) to compare uncertainty performance. NLL is a valid scoring rule but it does not have the means to compare different distributions. Hence, we use Interval score scoring rule in this research to evaluate the quality of uncertainty estimation[22].

3.4.1 Interval score

The below section about finding interval score for different distributions is from the blog[20]:
Different methods have been mentioned in the literature for comparing uncertainty. Interval Score, a type

of proper scoring rule[7] is the one we're going to use in this research work. This metric uses prediction interval with upper and lower endpoints which are represented by predictive quantiles that are in the levels of $1 - \alpha/2$ and $\alpha/2$. The values of αsuch as 0.02, 0.05 and 0.1 represent the nominal converges at the rate of 98%, 95% and 90%.
The intervals of different distribution can be calculated using the python sympy inbuilt function $.interval(alpha)function$,

```
l, u = normal_dist.interval(alpha=0.95)
print ("Normal Distirbution :")
print ("95% interval ", normal_dist.interval(alpha=0.95))
print ("Interval Score ", interval_score(10 , l, u))


print ("Laplace Distirbution")
l, u = laplace_dist.interval(alpha=0.95)
print ("95% interval ", laplace_dist.interval(alpha=0.95))
print ("Interval Score ", interval_score(10 , l, u))



print ("Cauchy Distirbution 95%")
l, u = cauchy_dist.interval(alpha=0.95)
print ("95% interval ", cauchy_dist.interval(alpha=0.95))
print ("Interval Score ",interval_score(10 , l, u))
```

Figure 3.9: Pytorch code for getting upper and lower limits in interval score metric [20]

Based on the upper and lower limits we get from the above code, interval score can be defined as [22],

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

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 3.10: Pytorch code for Interval score[20]

The score favors precise predictions while penalizing those that lie outside the range. For this research, α is fixes to be 95%.Initially, compute the 95% quantile prediction interval (l, u), which is then used to compute the Interval score, based on the mean and variance predicted by the neural network. Hence if the true value fits between the lower and upper boundaries, the score will be at the minimum. Tighter bounds will result in better scores [20].

4

Solution

Your main contributions go here

4.1 Experiments

In this experiment, we estimate uncertainty for complex regression tasks, Here the output of the DNN will be predicted output and the predicted uncertainty. The regression task considered in this research are Facial Keypoint detection and Biwi-Head pose dataset. These two datasets are trained on Resnet-18 model with four different metrics such as Gaussian Negative Log-Likelihood, Laplce loss, Cauchy and Evidential loss.

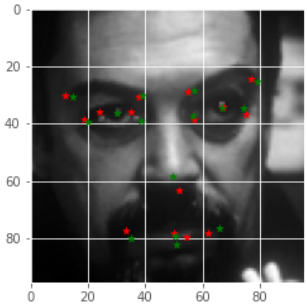
The hyper parameters used for training these models are:

Hyper parameters	
Model	Resnet-18
Number of epochs	300
Batch size	256
Learning rate	0.0001
Optimizer	Adam
Early stopping	Patience - 5

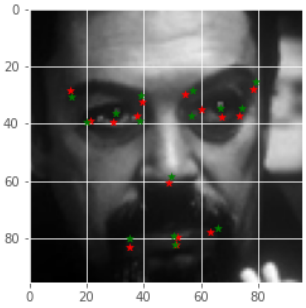
- The saved models are then loaded to finally validate once in order to evaluate how the best saved model has performed.

- The performance of the model is then evaluated by plotting box plot for separate keypoints.

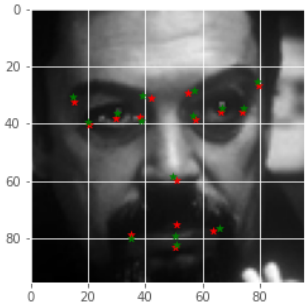
- For both Facial Keypoint detection task and biwi dataset, the predicted outputs and the actual outputs for individual keypoints has been plotted as in Figure 4.1 and Figure 4.2.,



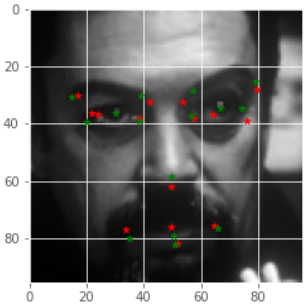
(a) Prediction-Gaussian loss



(b) Prediction-Laplace loss

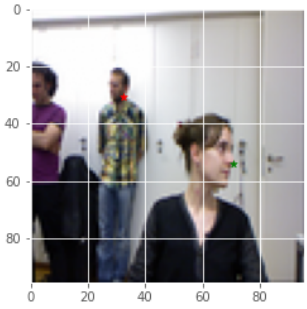


(c) Prediction-Cauchy loss

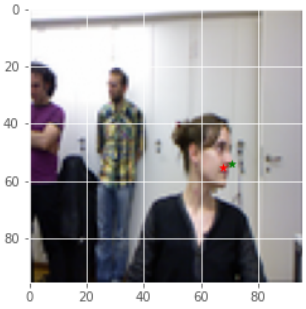


(d) Prediction-Evidential loss

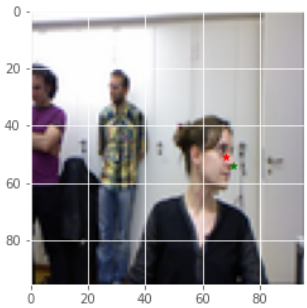
Figure 4.1: Performance of predicted keypoints on different loss functions



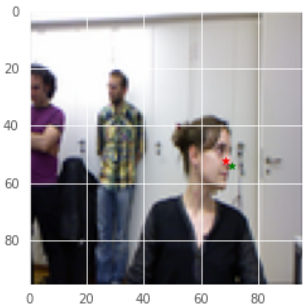
(a) Prediction-Gaussian loss



(b) Prediction-Laplace loss



(c) Prediction-Cauchy loss



(d) Prediction-Evidential loss

Figure 4.2: Performance of predicted keypoints on different loss functions

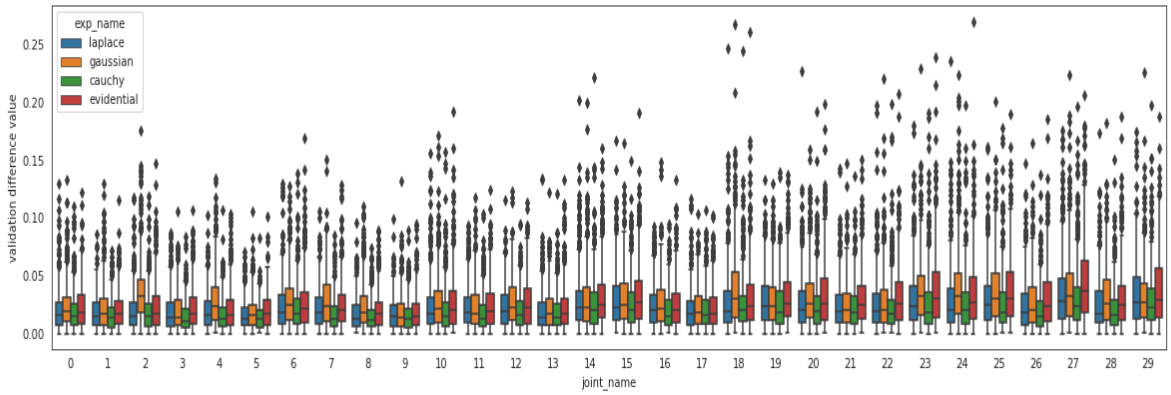


Figure 4.3: Facial Keypoint data: Overall performance of individual keypoints on validation difference value

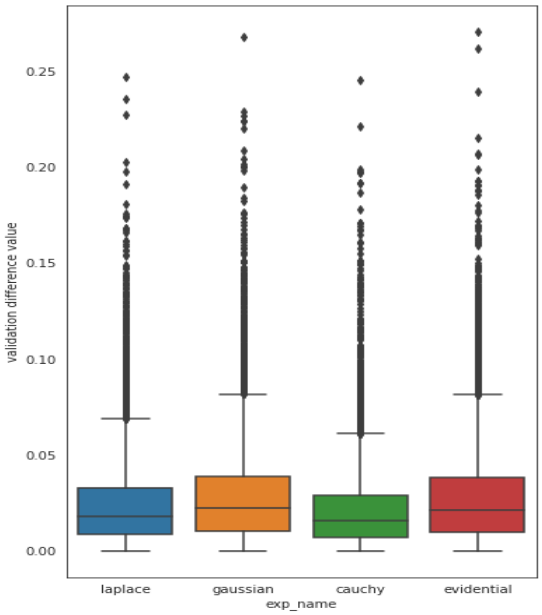


Figure 4.4: Facial Keypoint data: Comparison plot on performance of loss functions on validation difference value

For Biwi Keypoint detection,

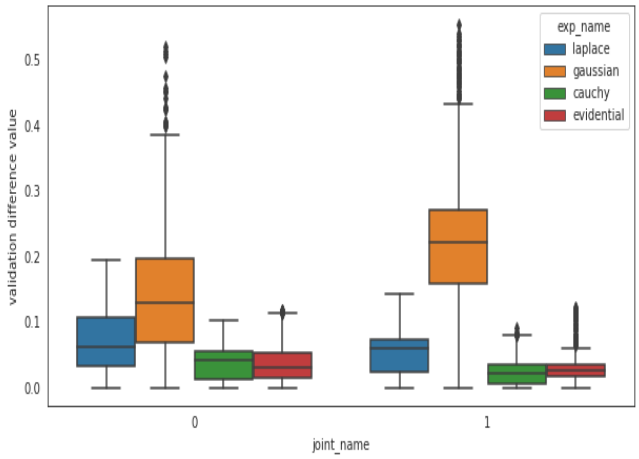


Figure 4.5: Biwi: Overall performance of individual keypoints on validation difference value

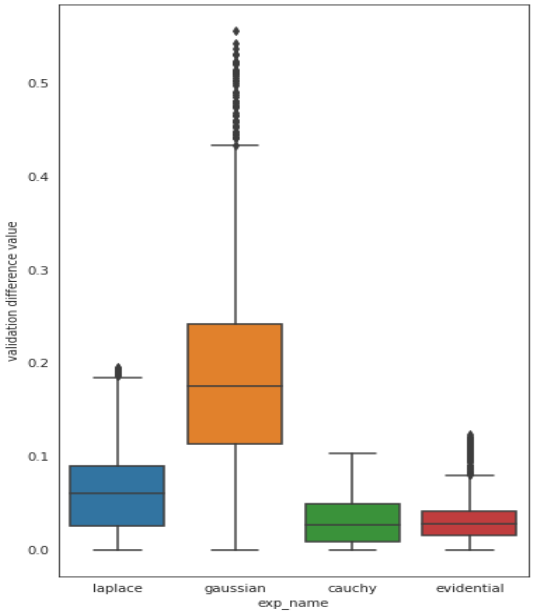


Figure 4.6: Biwi: Comparison plot on performance of loss functions on validation difference value

Interquartile box plots are being plotted for both Facial detection and Biwi datasets. In the above graph,it can be clearly seen that Cauchy loss metric is observed to perform better compared to other three metrices as it has less prediction error and it is also not widely spread like Gaussian negative log likelihood.
The estimated uncertainty is evaluated using Interval score metric and also the performance of the DNN using four different metrics are evaluated and tabulated as below,

Facial keypoint dataset				
Metrics	Gaussian	Laplace	Cauchy	Evidential
Interval score	0.111	0.042	**0.037**	0.274
RMSE	0.038	0.033	**0.029**	0.038

Table 4.1: Result table for face keypoint data

Biwi dataset				
Metrics	Gaussian	Laplace	Cauchy	Evidential
Interval score	0.864	0.223	**0.133**	0.433
RMSE	0.217	0.035	**0.030**	0.433

Table 4.2: Result table for Biwi data

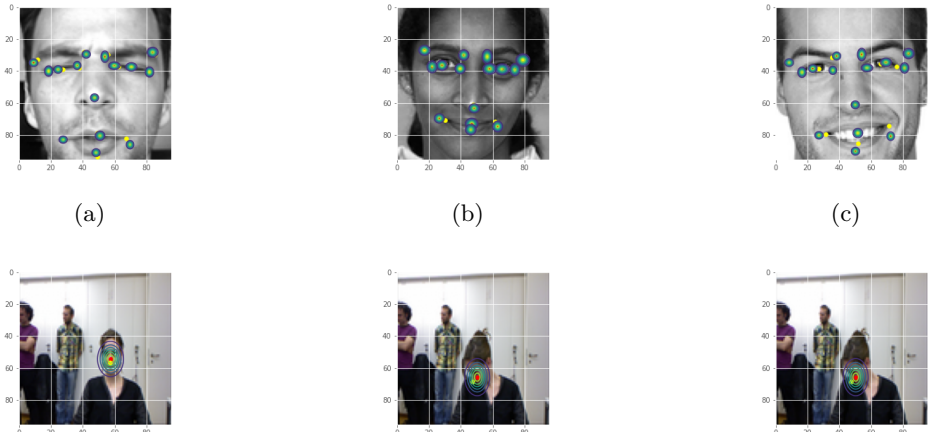
The performance of the model in estimating uncertainty is evaluated using Interval score and RMSE metric. The metrics are compared for both Facial keypoint detection dataset and Biwi dataset. Four different loss functions such as Gaussian Negative Log-Likelihood, Laplace loss, Cauchy loss and Evidential loss functions are being compared and evaluated using the uncertainty metrics. From the above tables, the interval score and Root Mean Square Error of Cauchy loss is minimum. This shows that this experiment with low prediction error is found to have test data close to where the model predicted it would be. Hence the Cauchy loss is found to estimate uncertainty and predict outputs better compared to other metrics.

4.1.1 Experimental image results

Besides plotting predicted and actual keypoints in an image, in this research we are also plotting different distributions on an image based on the type of metrics used in an experiment.In an attempt for visualizing underlying probability distribution of the data, these distributions plots are plotted. These distribution plots are one of the best methods for determining whether your data follow a specific distribution. If the distribution of the data is known then one may actually make a better prediction of uncertainty, such as the likelihood that events will occur and their corresponding effects. By establishing categories, one might also get to make some important decisions.
Since entropy is a method for determining uncertainty, In this research we have plotted most/low confidence images based on the entropy values and the most/least error images based on the error values for different

distributions.

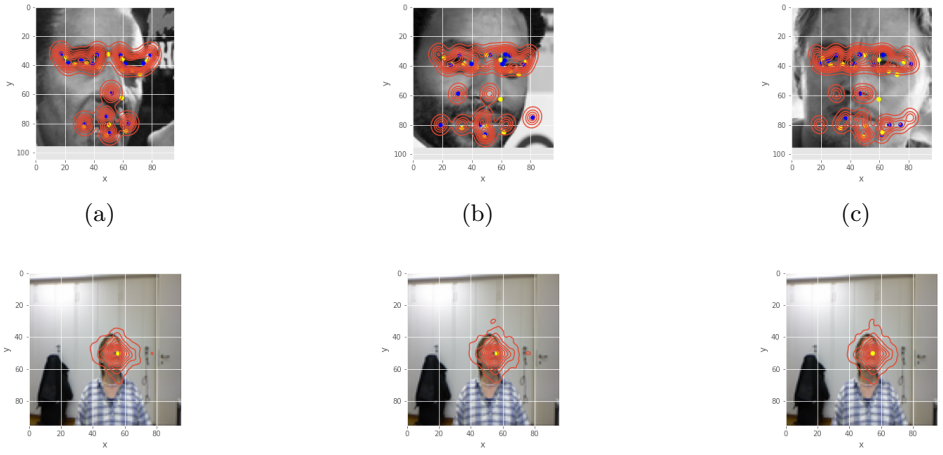
In the below most confident images the distributions are plotted for the predicted keypoints and the yellow markers are plotted to denote the actual keypoints. In the below images we can observe that the yellow keypoints are not much visible in the images which suggests the fact that these are the most confident images for different loss functions we have used in this experiment.



(a) (b) (c)

(d) (e) (f)

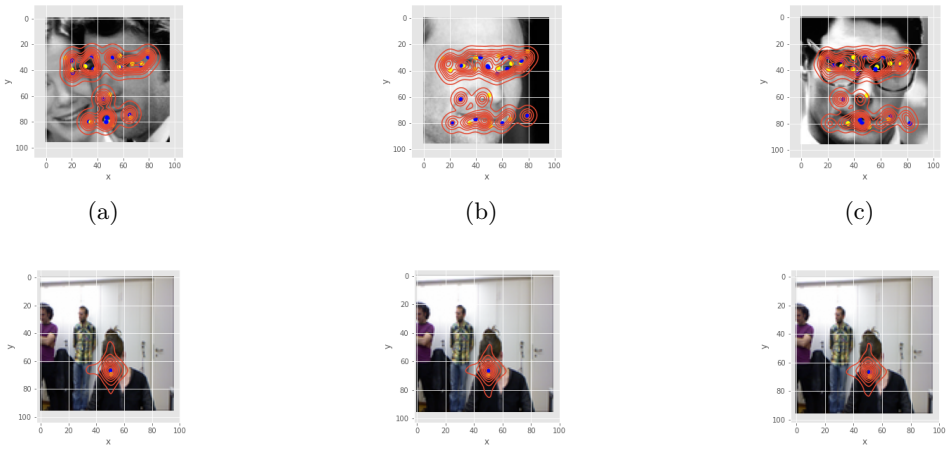
Figure 4.7: Most confident images in Gaussian loss



(a) (b) (c)

(d) (e) (f)

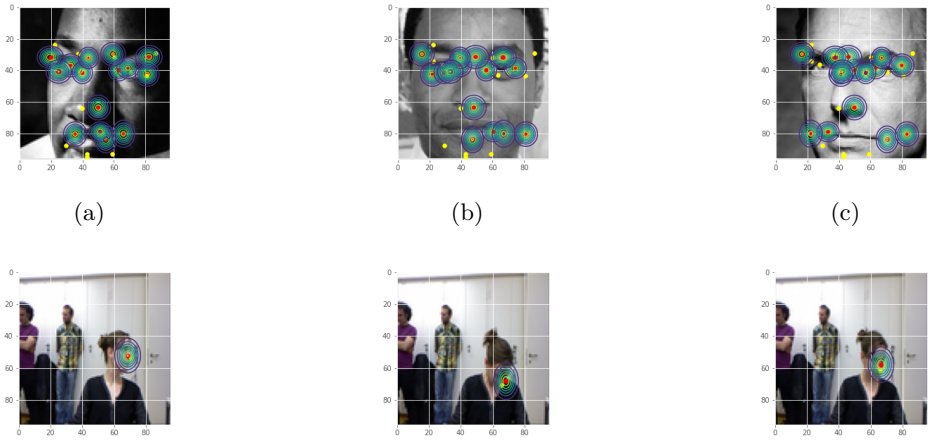
Figure 4.8: Most confident images in Laplace loss



(a) (b) (c)

(d) (e) (f)

Figure 4.9: Most confident images in Cauchy loss



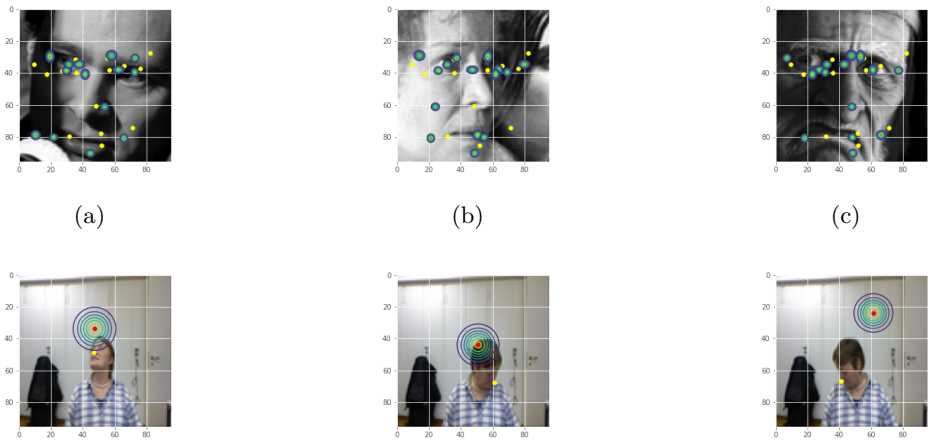
(a) (b) (c)

(d) (e) (f)

Figure 4.10: Most confident images in Evidential loss

Most/Least confident images and High/Low error images have been plotted for Gaussian loss. We can find that both the true keypoints and the predicted keypoints are also plotted in the images along with the respective Gaussian distribution. The true keypoints are plotted as yellow markers and the predicted keypoints are plotted as blue markers. In the above figures the Gaussian distributions are plotted for the predicted keypoint's μ and standard deviation values such that it is easy to observe whether the data has followed a particular distribution and to check the better prediction of uncertainties. In Figure 4.7 and Figure 4.19 where most confident and least error images are plotted, we can observe that the Gaussian distributions which is plotted using μ and standard deviation values of predicted keypoints are more close

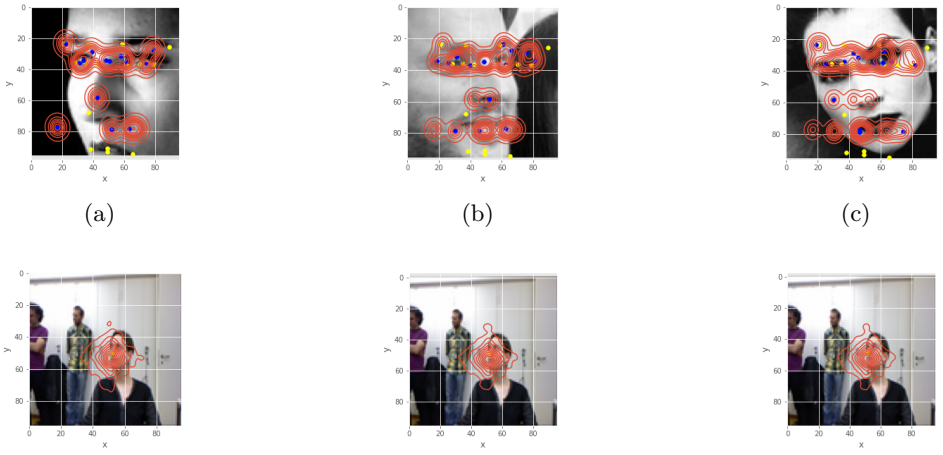
to true keypoints compared to the least confident and High error images in Figure 4.11 and Figure 4.15 where more true keypoints are visible which means the keypoints are predicted slightly away from the true ones with high uncertainty and error values.
In the below least confident images the distributions are plotted for the predicted keypoints and the yellow markers are plotted to denote the actual keypoints. In the below images we can observe that the yellow keypoints are much visible in the images which suggests the fact that these are the least confident images for different loss functions we have used in this experiment.



(a) (b) (c)

(d) (e) (f)

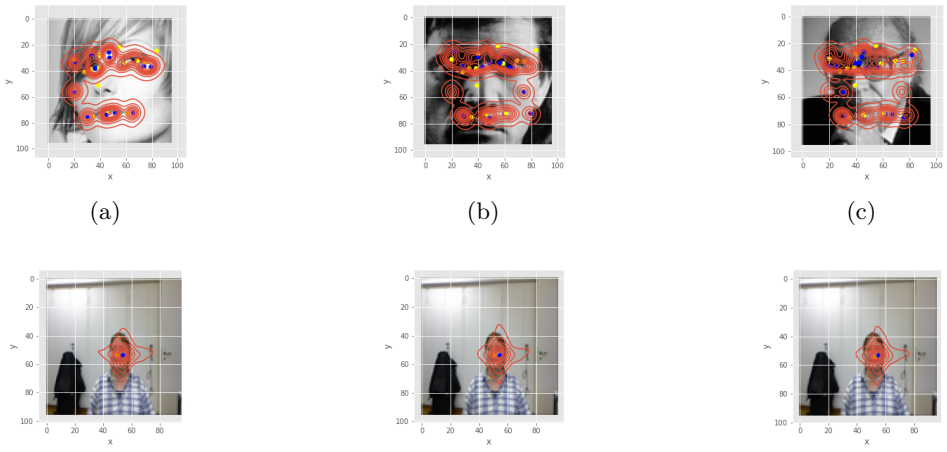
Figure 4.11: Least confident images in Gaussian loss



(a) (b) (c)

(d) (e) (f)

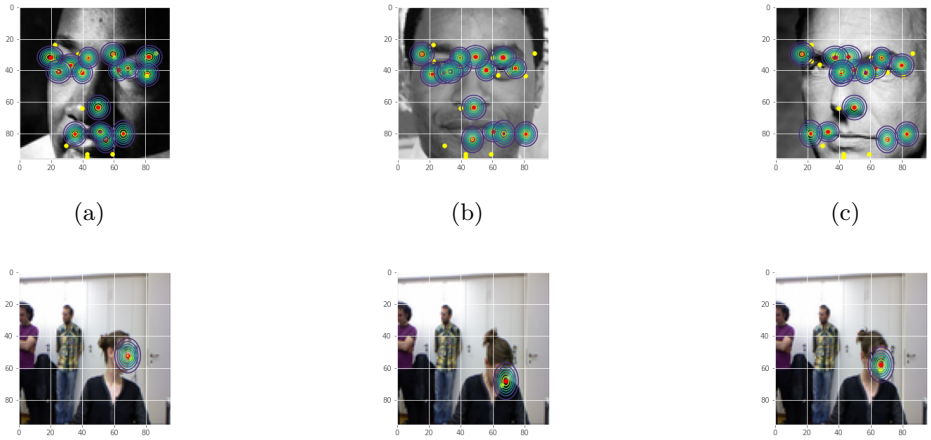
Figure 4.12: Least confident images in Laplace loss



(a) (b) (c)

(d) (e) (f)

Figure 4.13: Least confident images in Cauchy loss



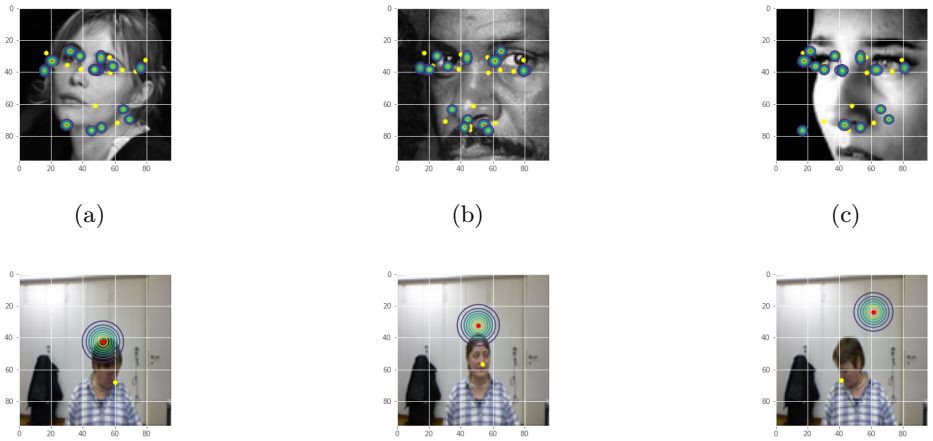
(a) (b) (c)

(d) (e) (f)

Figure 4.14: Least confident images in Evidential loss

For experimental result images using Laplace loss, we can find that both the true keypoints and the predicted keypoints are also plotted in the images along with the respective Laplace distribution. The true keypoints are plotted as yellow markers and the predicted keypoints are plotted as blue markers. Similar to the Gaussian/normal distribution, the Laplace distribution features fatter tails and a sharper peak. In this experiments where Laplace loss is being used for training and validating the models, samples are drawn from the Laplace distribution using a specified location (μ) and scale(decay) values. The data is then given as an input to the Kernel Density estimate(KDE) plot which is a method for visualizing the distribution of data. In this work, contour plots are used in order to visualize how the data progresses from low to high by

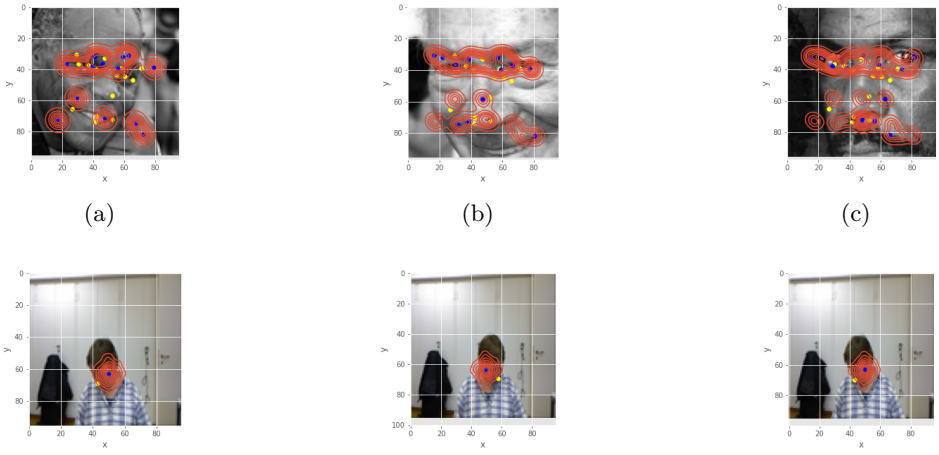
progressively increasing the saturation and darkness of the color. Darker colors by default denote higher density values. In Figure 4.8 and Figure 4.20 where most confident and least error images are plotted, we can observe that the distributions are dense near the true values and they are less dense at other places, whereas in Figure 4.12 and Figure 4.16 where least confident and high error images are plotted, the predicted distributions are away from the true ones which shows high uncertainty and high error in the data.



(a) (b) (c)

(d) (e) (f)

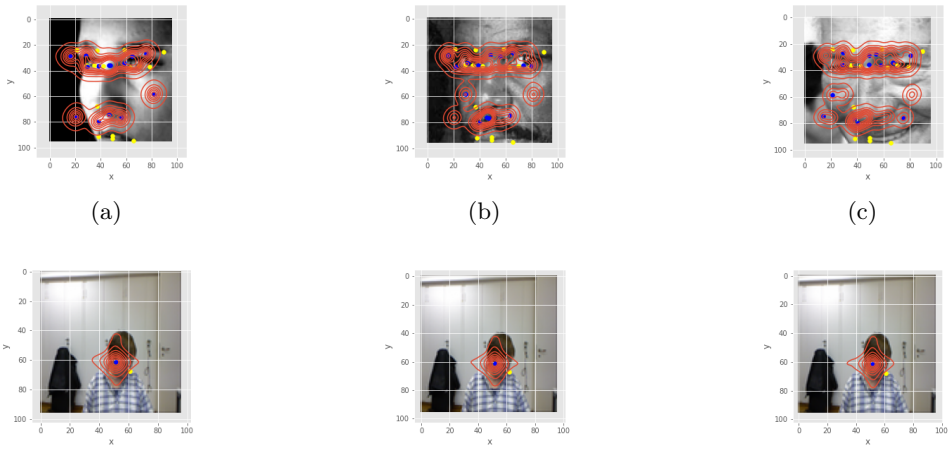
Figure 4.15: Most error images in Gaussian loss



(a) (b) (c)

(d) (e) (f)

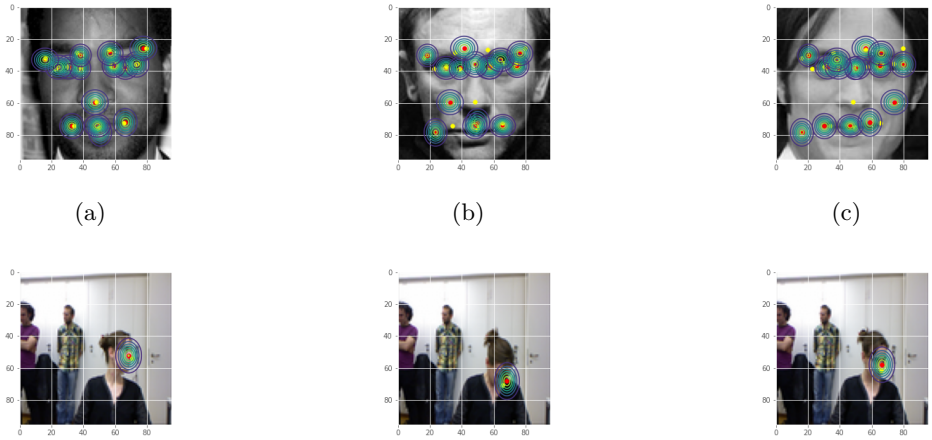
Figure 4.16: Most error images in Laplace loss



(a) (b) (c)

(d) (e) (f)

Figure 4.17: Most error images in Cauchy loss



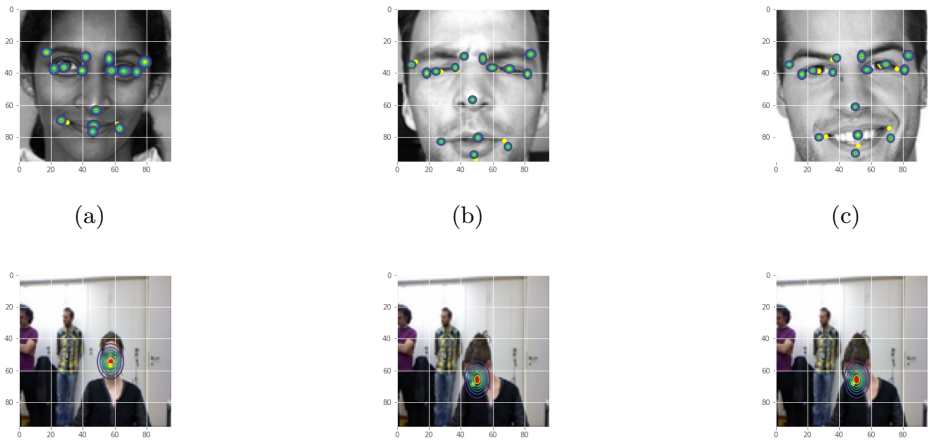
(a) (b) (c)

(d) (e) (f)

Figure 4.18: Most error images in Evidential loss

For experimental result images using Cauchy loss, we can find that both the true keypoints and the predicted keypoints are also plotted in the images along with the respective Laplace distribution. The true keypoints are plotted as yellow markers and the predicted keypoints are plotted as blue markers. Similar to the Gaussian/normal distribution, the Cauchy distribution has heavier tails. In this experiments where Cauchy loss is being used for training and validating the models, similar to Laplace distribution,samples are drawn using a specified location (μ) and scale(decay) values. The data is then given as an input to the contour plots for visualizing the data progresses from low to high density. In Figure 4.11 and Figure 4.21 where most confident and least error images are plotted, we can observe that the distributions are

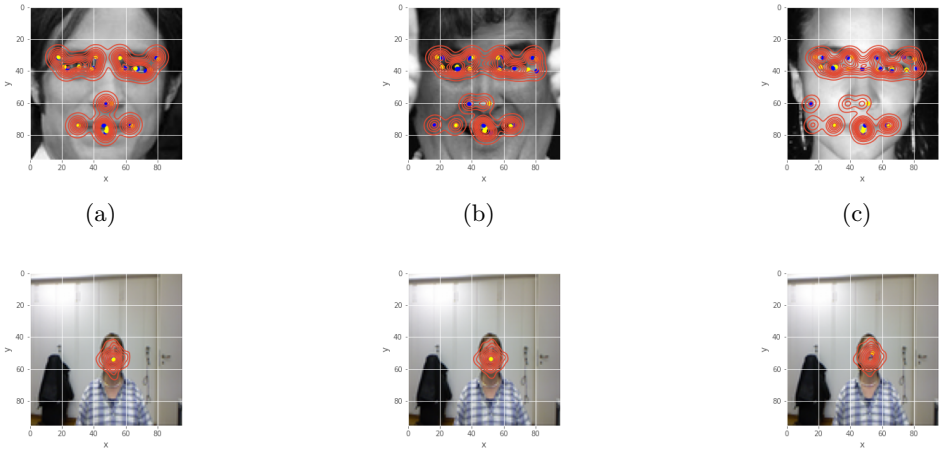
dense near the true values and they are less dense at other places, whereas in Figure 4.13 and Figure 4.17 where least confident and high error images are plotted, the predicted distributions are away from the true ones which shows high uncertainty and high error in the data.
In the below least error images the distributions are plotted for the predicted keypoints and the yellow markers are plotted to denote the actual keypoints. In the below images we can observe that the yellow keypoints are not much visible in the images which suggests the fact that these are the least error images for different loss functions we have used in this experiment.



(a) (b) (c)

(d) (e) (f)

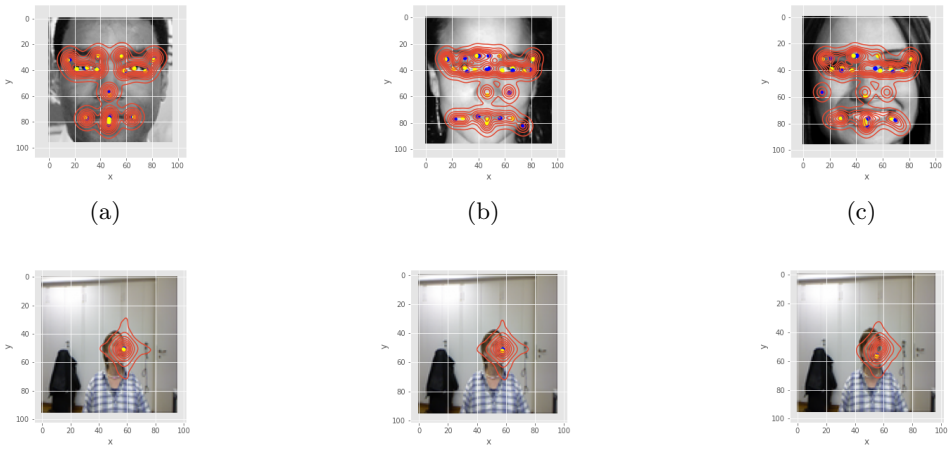
Figure 4.19: Least error images in Gaussian loss



(a) (b) (c)

(d) (e) (f)

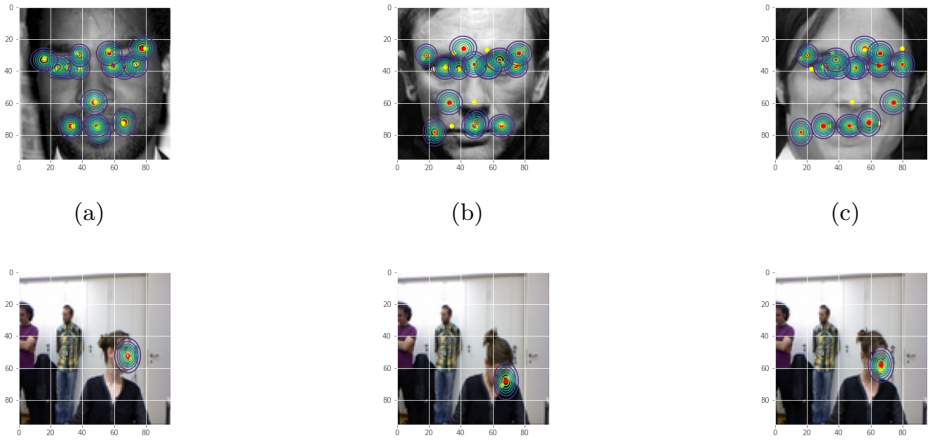
Figure 4.20: Least error images in Laplace loss



(a) (b) (c)

(d) (e) (f)

Figure 4.21: Least error images in Cauchy loss



(a) (b) (c)

(d) (e) (f)

Figure 4.22: Least error images in Evidential loss

Similar to all the above distributions, for evidential loss experimental result images predicted and true keypoints are plotted along with the distributions. In the above figures the Gaussian distributions are plotted for the predicted keypoint's μ and standard deviation values. In Figure 4.10 and Figure 4.22 where most confident and least error images are plotted, it can be observed that the Gaussian distributions which is plotted using μ and standard deviation values of predicted keypoints are more close to true keypoints compared to the least confident and High error images in Figure 4.14 and Figure 4.18 where more true keypoints are visible which means the keypoints are predicted slightly away from the true ones with high uncertainty and error values.

4.1.2 Robustness experiment

In this research we compare the uncertainty estimation methods with augmentation vs without augmentation. In this work we have used three types of data augmentation techniques such as Gaussian blur, random rotation and random noise. All three augmentation techniques are added to both the facial keypoint detection and Biwi dataset and the changes are observed to make the conclusion.

Gaussian blur

The first augmentation technique used is Gaussian blur which means adding blur to the images by means of Gaussian function. By doing this the details in the images are reduced. The pixels in an image that has to be blurred are considered independently and they are blurred depending on its own pixel value and on the surrounding pixel value which means by the kernel size. In this experiment, we have used a kernel size of 7. When the image undergoes Gaussian noise with a kernel size of 7 it looks like the example image from facial keypoint detection dataset in Figure 4.23.

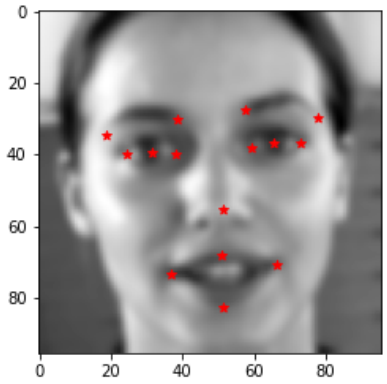


Figure 4.23: Gaussian Blur image

Random rotation

Random rotation is a the common method of data augmentation. The position of an object in the frame is altered by randomly rotating a source image a certain number of degrees either clockwise or counterclockwise.The image is made to rotate by its center based on the specified degree. In this work we have used 45 degrees. An example image from Facial keypoint detection dataset with random rotation of 45 degrees will look like the one from Figure 4.24.

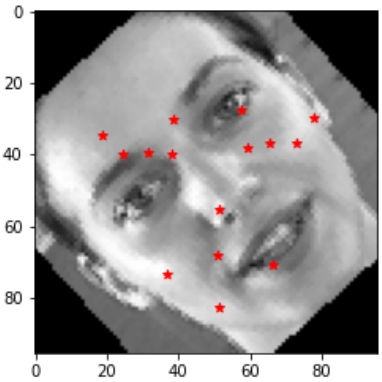


Figure 4.24: 45 degree random rotation image

Random noise

Another data augmentation technique used in this work is called random noise where the noises are added to the image randomly. The noises added to the image is a speckle noise. An example image from Facial keypoint detection dataset with speckle noise will look like the one from Figure 4.25.

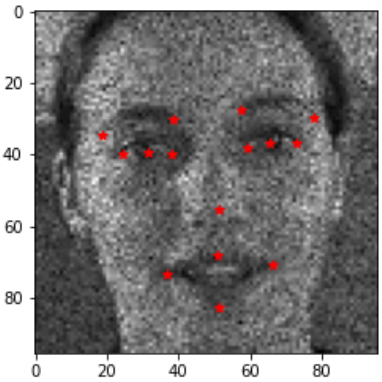
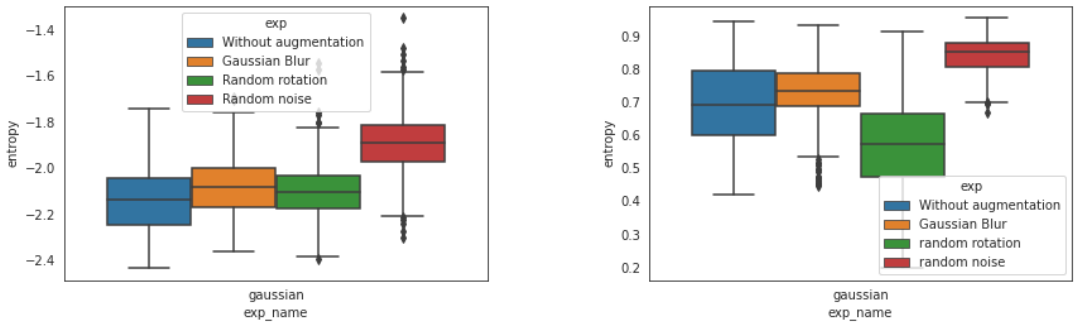


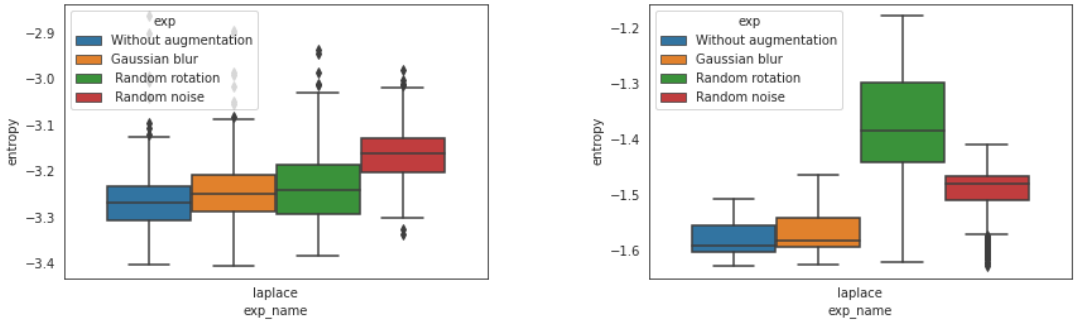
Figure 4.25: Random noise image

The entropy plot has been plotted for both the facial keypoint and Biwi dataset for both clean data and the augmented data.



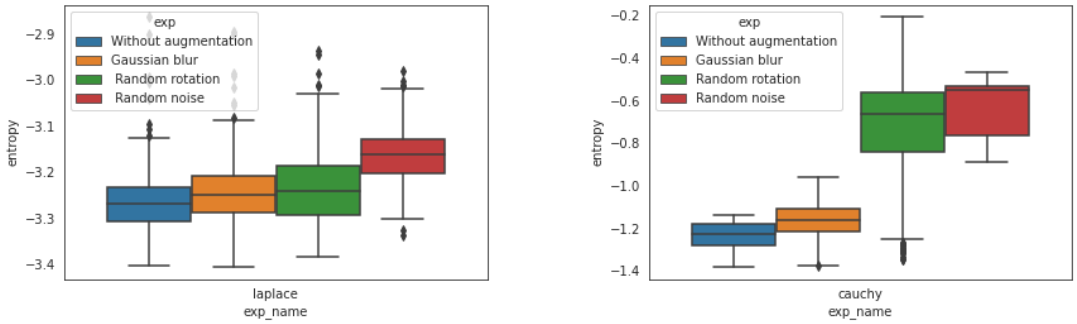
(a) Facial keypoint data for Gaussian loss (b) Biwi data for Gaussian loss

Figure 4.26: Gaussian entropy plot - clean data vs augmented data



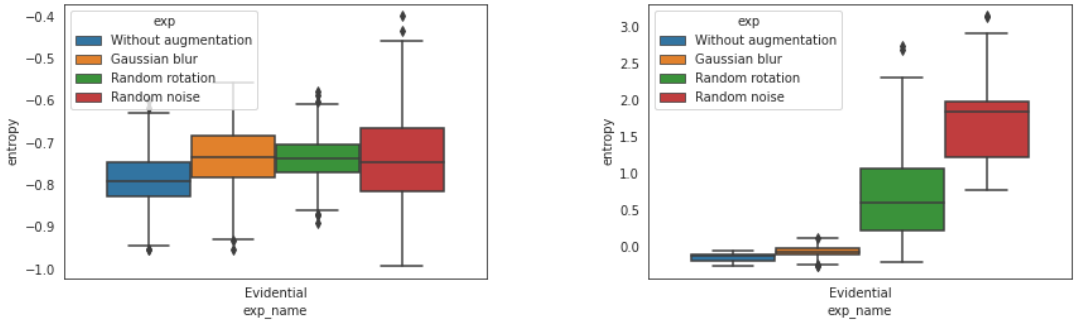
(a) Facial keypoint data for Laplace loss (b) Biwi data for Laplace loss

Figure 4.27: Laplace entropy plot - clean data vs augmented data



(a) Facial keypoint data for Cauchy loss (b) Biwi data for Cauchy loss

Figure 4.28: Cauchy entropy plot - clean data vs augmented data



(a) Facial keypoint data for evidential loss

(b) Biwi data for evidential loss

Figure 4.29: Evidential entropy plot - clean data vs augmented data

On observing all the graphs above it is found that the augmented data shows increase in uncertainty and the overall performance is degrading which is a good one. The change in entropy for clean and augmented data is minimum for all loss functions in Face keypoint dataset and is quite greater in biwi dataset. Compared to other methods, the change in entropy for clean data and the augmented data is minimum for cauchy loss function in Figure 4.28 which shows that it is robust to augmented data.

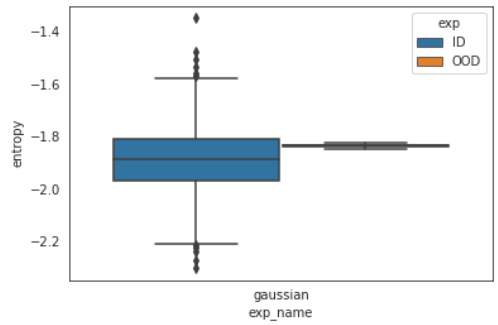
4.1.3 Out-Of-Distribution experiment

Out-of-distribution (OOD) detection is important for a machine learning model to its reliability and safety of the systems. The data distribution in real-work activities typically evolves with time, and chasing an evolving data distribution is expensive. Therefore, OOD identification is crucial for AI systems to avoid making prediction errors. There are two types of Generalizations:

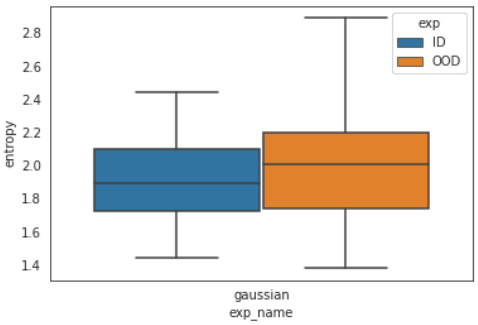
- **In-Distribution Generalization**-Generalization to new samples chosen from the similar distribution as the samples we used it for the training set.

- **Out-of-Distribution Generalization**-Generalization to new samples chosen from the different distribution as the samples we used it for the training set than the training set.

For each method with different loss functions, we take both Face keypoint detection and biwi model that was trained using ID(In-Distribution) dataset and compare them with the Out-Of-Distribution dataset. In this case, the OOD dataset is a cat dataset with its corresponding keypoints from Kaggle[10]. The data is then fed into the validation model of both datasets and the corresponding uncertainties are observed using a box plot which is plotted using the entropy metric.

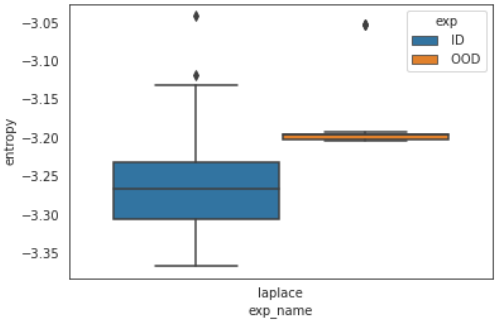


(a) Facial keypoint data for Gaussian loss

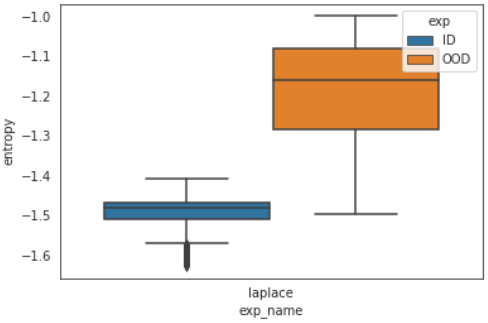


(b) Biwi data for Gaussian loss

Figure 4.30: Gaussian entropy plot - In-Distribution vs Out-of-Distribution

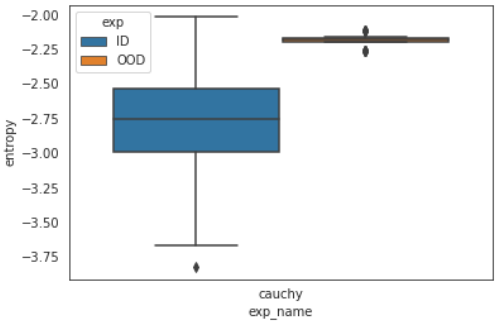


(a) Facial keypoint data for Laplace loss

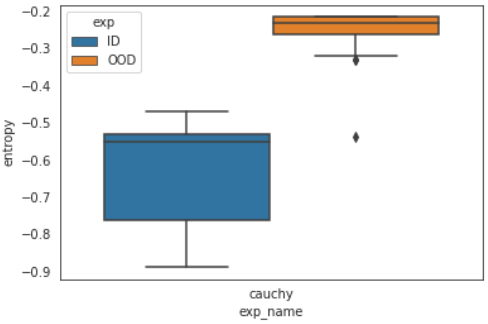


(b) Biwi data for Laplace loss

Figure 4.31: Laplace entropy plot - In-Distribution vs Out-of-Distribution

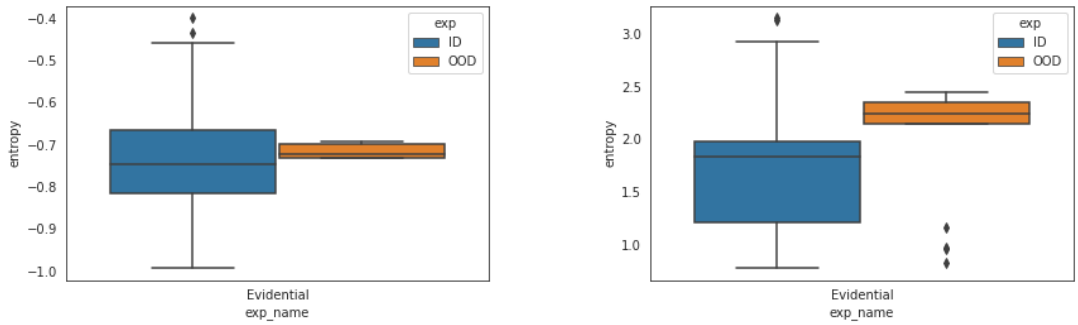


(a) Facial keypoint data for Cauchy loss



(b) Biwi data for Cauchy loss

Figure 4.32: Cauchy entropy plot - In-Distribution vs Out-of-Distribution



(a) Facial keypoint data for evidential loss

(b) Biwi data for evidential loss

Figure 4.33: Evidential entropy plot - In-Distribution vs Out-of-Distribution

From the above figures, it is observed that In-distribution (ID) predictions have less uncertainty when compared to OOD animal data predictions, which is an ideal goal of this experiment. On comparing predicted uncertainties of ID data statistically with OOD data, both of them should give a different from each other. This distinct separation aids in determining a threshold value for deciding whether to believe the predictions in the context of the evidence. In the above graphs, we can observe that the Gaussian(Figure 4.30) and Evidential(Figure 4.33) has very minimum difference between the In-distribution and Out-Of-Distribution data which suggests the fact that they are not good in handling OOD data. On comparing all the loss functions, only the Cauchy approach can statistically differentiate between ID and OOD data, as shown by the entropy box plots in Figure 4.32.

5

Conclusions

In this work we focus on robustness study of uncertainty estimation methods for complex regression tasks. In this research we particularly focus on keypoint detection tasks using facial keypoint detection and biwi dataset. We use four different loss functions in this research such as Gaussian NLL, Laplace NLL, Cauchy NLL and Evidential loss. These loss functions make the Neural Network's output not only the prediction output but also the corresponding distribution with it. If the distribution of the data is known then one may actually make a better prediction of uncertainty, such as the likelihood that events will occur and their corresponding effects. The models are then trained and validated using these loss functions with both datasets and then the quality of the model is evaluated using an uncertainty metric called interval score. The robustness study experiment is then carried by adding augmented data to the model and the performance of the model is evaluated using the entropy plot, which is also known as uncertainty measure metric. Out-Of-Distribution experiment is also performed to check the reliability of the model using animal dataset. From these experiments, it is found that the Cauchy NLL performs better compared to other loss functions by means of having low uncertainty value, robust to augmented data and also shows notable difference in performance when an OOD data is given to the model.

5.1 Contributions

- Literature survey on uncertainty estimation estimation methods, keypoint datasets and keypoint detection tasks for deep regression models.
- Survey on different metrics for comparing the performance of model and also on uncertainty metric for evaluating the quality of the model.
- Comparison of uncertainty estimation methods using face and Biwi keypoint dataset for four different loss functions.
- Robustness study on uncertainty estimation methods for keypoint detection task using data augmentation techniques.
- Out-Of-Distribution experiment was performed by animal dataset to check the reliability of the model.

5.2 Lessons learned

- It is important for Neural Network to output both predictions and distributions as these distributions helps in the better prediction of uncertainty and its likelihood.
- Metrics are more important in comparing different uncertainty estimation methods, NLL(Negative Log-Likelihood) cannot be used as an uncertainty metric for comparing different distributions and so we use a proper scoring rules in this research for comparing uncertainties.
- Out-Of-Distribution experiment has to be performed to as it plays a crucial role in evaluating the reliability of the model for safety-critical systems.
- Cauchy NLL performs better compared to other loss functions by means of having low uncertainty value, robust to augmented data and also shows notable difference in performance when an OOD data is given to the model.

5.3 Future work

- This work can be extended to other heavy tail distribution loss functions to check the performance of the model.
- The suggested robust loss function may be useful in creating software that uses uncertainty of Neural Network(NN) for safe and secure deep learning deployment in autonomous systems.
- This work can be carried out in object keypoint dataset which can be used in robotics field for gripper application.

Bibliography

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

[2] Prianka Banik, Lin Li, and Xishuang Dong. “A novel dataset for keypoint detection of quadruped animals from images”. In: *arXiv preprint arXiv:2108.13958* (2021).

[3] 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.

[4] Markus Borg et al. “Safely entering the deep: A review of verification and validation for machine learning and a challenge elicitation in the automotive industry”. In: *arXiv preprint arXiv:1812.05389* (2018).

[5] Do Hyun Chung, Il Dong Yun, and Sang Uk Lee. “Registration of multiple-range views using the reverse-calibration technique”. In: *Pattern Recognition* 31.4 (1998), pp. 457–464.

[6] Armen Der Kiureghian and Ove Ditlevsen. “Aleatory or epistemic? Does it matter?” In: *Structural safety* 31.2 (2009), pp. 105–112.

[7] 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.

[8] Hironori Hattori et al. “Synthesizing a scene-specific pedestrian detector and pose estimator for static video surveillance”. In: *International Journal of Computer Vision* 126.9 (2018), pp. 1027–1044.

[9] Susmit Jha et al. “Safe autonomy under perception uncertainty using chance-constrained temporal logic”. In: *Journal of Automated Reasoning* 60.1 (2018), pp. 43–62.

[10] kaggle. “Animal-Kaggle”. In: *URL https://www.kaggle.com/datasets/alessiocorrado99/animals10* (2018).

[11] kaggle. “Biwi-Kaggle”. In: *URL https://www.kaggle.com/datasets/kmader/biwi-kinect-head-pose-database* (2018).

[12] kaggle. “Facial keypoint detection-Kaggle”. In: *URL https://www.kaggle.com/c/facial-keypoints-detection/data* (2018).

[13] Salman H Khan et al. “Secure biometric template generation for multi-factor authentication”. In: *Pattern Recognition* 48.2 (2015), pp. 458–472.

[14] 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.

[15] Li Liu et al. “Deep learning for generic object detection: A survey”. In: *International journal of computer vision* 128.2 (2020), pp. 261–318.

[16] Antonio Loquercio, Mattia Segu, and Davide Scaramuzza. "A general framework for uncertainty estimation in deep learning". In: *IEEE Robotics and Automation Letters* 5.2 (2020), pp. 3153–3160.

[17] Athanasios Mademlis et al. "3D object retrieval using the 3D shape impact descriptor". In: *Pattern Recognition* 42.11 (2009), pp. 2447–2459.

[18] Andrey Malinin. "Uncertainty estimation in deep learning with application to spoken language assessment". PhD thesis. University of Cambridge, 2019.

[19] Osama Mazhar et al. "Towards real-time physical human-robot interaction using skeleton information and hand gestures". In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2018, pp. 1–6.

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

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

[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] A. Rakova and Bilous Nataliya. "REFERENCE POINTS METHOD FOR HUMAN HEAD MOVEMENTS TRACKING". In: *Radio Electronics, Computer Science, Control* (Nov. 2020), pp. 121–128. DOI: 10.15588/1607-3274-2020-3-11.

[25] Farheen Ramzan et al. "A Deep Learning Approach for Automated Diagnosis and Multi-Class Classification of Alzheimer's Disease Stages Using Resting-State fMRI and Residual Neural Networks". In: *Journal of Medical Systems* 44 (Dec. 2019). DOI: 10.1007/s10916-019-1475-2.

[26] Sovit Ranjan Rath. "Facial keypoint detection-Kaggle". In: *URL https://debuggercafe.com/getting-started-with-facial-keypoint-detection-using-pytorch/* (2018).

[27] N Dinesh Reddy, Minh Vo, and Srinivasa G Narasimhan. "Carfusion: Combining point tracking and part detection for dynamic 3d reconstruction of vehicles". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 1906–1915.

[28] Syed Afaq Ali Shah, Mohammed Bennamoun, and Farid Boussaid. "Iterative deep learning for image set based face and object recognition". In: *Neurocomputing* 174 (2016), pp. 866–874.

[29] Joost Van Amersfoort et al. "Uncertainty estimation using a single deep deterministic neural network". In: *International conference on machine learning*. PMLR. 2020, pp. 9690–9700.

[30] Kai Wang, Boris Babenko, and Serge Belongie. "End-to-end scene text recognition". In: *2011 International conference on computer vision*. IEEE. 2011, pp. 1457–1464.

[31] John Williams and Mohammed Bennamoun. “A multiple view 3D registration algorithm with statistical error modeling”. In: *IEICE TRANSACTIONS on Information and Systems* 83.8 (2000), pp. 1662–1670.