



E5 INTERNSHIP REPORT

FROM 6TH FEBRUARY 2023 TO 4TH AUGUST 2023

Uncertainty Estimation and Active Learning

Author:

Praise OKETOLA

Supervisor:

Pontus LOVIKEN
Gordana DRASKOVIC

*A report submitted in fulfilment of the requirements
for the 5th year*

August 2023

Contents

Contents	i
List of Figures	iii
List of Tables	iv
Abbreviations	v
1 Introduction	1
1.1 Context and Motivation	2
1.2 The AI Lab in SLB	2
1.3 Internship Objectives	3
1.4 Thesis Outline	3
2 Background	4
2.0.1 Uncertainty Estimation	4
2.0.2 Active Learning	6
2.0.2.1 Active Learning Query Strategies	7
3 Uncertainty Estimation Techniques	9
3.1 Monte Carlo Dropout	11
3.1.1 Effect of p on Uncertainty Estimation and Model Performance	12
3.2 BNNs(Bayesian Neural Networks)	13
3.2.1 Dropout Vs No Dropout For Monte Carlo Sampling	14
3.2.2 Comparison of Different BNN Configurations	14
3.3 Autoencoders	15
3.3.1 Comparison of Reconstruction Loss and Model Absolute Error	15
3.3.2 Using the Encoder of An Autoencoder to Train the Model	16
3.3.3 Autoencoder Reconstruction Loss with Out-of-Distribution (OOD) Data	17
3.4 Ensembles	20
3.4.1 Effect of Number of Epochs on Uncertainty Estimation	21
3.4.2 Effect of Network Complexity on Uncertainty Estimation	22
3.4.3 Effect of Number of Ensembles on Uncertainty Estimation	23
3.4.4 Effect of Ensemble Composed of Different Architectures on Uncertainty Estimation	24
3.5 Ensemble Model Pairs	26
3.5.1 Different Uncertainty Metrics And What They Mean	26
3.5.2 Many to One	27

3.5.3	Many to Many	29
3.5.4	Dual Task	31
3.6	Predicting Absolute Error vs Variance in Error Models	34
4	Uncertainty Estimation Simulator	36
4.1	Approach	36
4.1.1	True Function	36
4.1.2	Epistemic Uncertainty	37
4.1.3	Aleatoric uncertainty	37
4.1.4	Model Uncertainty	38
4.2	Calculating the True Uncertainties	40
4.2.1	Experimental Setup	42
4.2.1.1	Scenarios	43
4.2.1.2	Uncertainty Models	43
4.2.1.3	Expectation from different models	43
4.2.2	Results	44
4.2.2.1	Model Scenario Result for a set seed	44
4.2.2.2	Model Scenario Result for multiple runs	45
5	Applications to Real Projects	52
5.1	Image Classification	52
5.1.1	Dataset	52
5.1.2	Multi-label Classification Results	54
5.1.3	Binary Classification Results	54
5.1.4	Effect of Data Imbalance on Active Learning	55
5.1.5	Image Ranking Techniques for Active Learning	58
5.2	Black Box Simulator	62
5.2.1	Random vs Uniform Initialization for Active Learning	63
5.2.2	Approach	64
5.2.3	Results	64
5.2.3.1	1D Simulator Result(2 Features)	64
5.2.3.2	1D Simulator Result(3 Features)	66
6	Conclusion	68
6.1	Future Work	69

Bibliography	70
---------------------	-----------

List of Figures

2.1	Data with Epistemic Uncertainty	5
2.2	Data with Aleatoric Uncertainty	5
2.3	Reducing Aleatoric Uncertainty by Embedding in a Higher-Dimensional Space	6
2.4	Active Learning Workflow	6
3.1	Age Distribution	9
3.2	Example Images in Age Prediction Dataset	10
3.3	Monte Carlo (MC) Dropout	12
3.4	Bayesian Neural Network	13
3.5	Autoencoder	15
3.8	Autoencoder Reconstruction on Randomly Generated Noise	19
3.11	How Ensembles Are Trained	21
3.16	Ensemble Error Model Pair (Many To One)	28
3.18	Ensemble Error Model Pair (Many To Many)	30
3.20	Dual Task Ensemble Model Pair	32
4.1	True function for simulator	37
4.6	Parameters used in Calculating True Uncertainty	42
4.7	Model Scenario Result for a set seed	44
4.8	Model Scenario Comparison with True Aleatoric Uncertainty	46
4.9	Model Scenario Comparison with True Epistemic Uncertainty	48
4.10	Model Scenario Comparison with True Model Uncertainty	50
5.1	Class Distribution Of Data in Active Learning Simulation	53
5.2	Active Learning Multi-label Classification Results	54
5.3	Active Learning Binary Classification Results	54
5.4	Active Learning Binary Classification Results Using Multiple Query Strategies	55
5.6	Effect of Data Imbalance on Active Learning	57
5.7	No of acquisition images in each active learning step	58
5.8	Autoencoder Reconstruction Loss for Active Learning	59
5.9	Dino_v2 Small Active Learning Performance	61
5.10	Comparison of size variants of Dino_v2	62
5.11	Black Box Simulator Example	63
5.12	Example of Output Signal	63
5.13	Random vs Uniform Initialization	64
5.14	1D Simulator Result(2 Features)	65
5.15	Example Prediction of Output Signal for 2 Input Features	65
5.16	Example Prediction of Output Signal for 3 Input Features	66

List of Tables

3.1	MC Dropout Results	12
3.2	Dropout Vs No Dropout For Monte Carlo Sampling(Test)	14
3.3	BNN Configuration Comparison Results(Test Set)	14
3.4	Comparison of Reconstruction Loss to Model Absolute Error(Test Set)	16
3.5	Comparison of Reconstruction Loss to Model Absolute Error(Test Set)	17
3.6	Ensemble Baseline Result	27
3.7	Ensemble Error Model Pair (Many To One) Result)	28
3.8	Ensemble Error Model Pair (Many To Many) Result	30
3.9	Many to Many Connected Ensemble Error Model Result	32
5.1	Class Distribution for Data Imbalance Experiment	56
5.2	DinoV2 Variants Details	61
5.3	1D Simulator Result(3 Features)	66

Abbreviations

UE	Uncertainty Estimation
AL	Active Learning
BNNs	Bayesian Neural Networks
EE	Epistemic Error
AE	Aleatoric Error
ME	Model Error
LM	Label Model
EM	Error Model
GPs	Gaussian Processes
OOD	Out of Distribution

Chapter 1

Introduction

Uncertainty estimation and active learning are key components in the development of robust and reliable machine-learning models. Understanding and quantifying the uncertainty linked with these models is important in order to provide explainability in various applications. One of the goals of this internship is to develop techniques for assessing uncertainty caused by insufficient data, noise in data, and the model itself.

There are a few main sources of uncertainties, which we can classify into epistemic, aleatoric, and model uncertainty. Epistemic uncertainty arises when there is a lack of sufficient data necessary to train a model to its full capacity. For instance, if we are training a model to recognize and classify different breeds of dogs, but we have very limited images of certain breeds, this lack of data contributes to epistemic uncertainty. By increasing the diversity and quantity of minority breeds, we can reduce or eliminate this type of uncertainty. Aleatoric uncertainty arises from the inherent noise within the data. For example, varying lighting conditions, angles, or the presence of obstructions in the image can introduce noise. This type of uncertainty is difficult to eliminate due to the complex nature of real-world conditions. Model error refers to the limitations of the model's ability to accurately represent the true function. Assuming we are using a simple machine learning algorithm to classify dog breeds, the model may struggle to correctly classify breeds that look similar. This is a form of model error, as the model might not be complex enough to make the distinction. Ideally, given a perfect model and infinite data, we would only have aleatoric uncertainty, while minimizing all other sources of error.

The main idea is to examine epistemic, aleatoric, and model uncertainty, and look into scenarios where these can be minimized. We accomplish this by creating a simulated environment that

allows control over each uncertainty. For instance, by using some uncertainty techniques we examine if acquiring more data can eliminate epistemic uncertainty, if improving the quality of data can decrease aleatoric uncertainty, and if improving the model can reduce model uncertainty. By understanding the contributions of each component and they interact, we can create effective methods for estimating and reducing uncertainty in machine learning models. The ultimate goal is to apply uncertainty estimation to active learning, a technique that smartly selects the most informative data samples to enhance the model's performance and decrease uncertainty.

The contributions include exploring various techniques of uncertainty estimation, such as dropout, ensembles, BNNs, and error models on an image-regression task. We further compared different active learning techniques, including a custom ranking technique for multilabel/binary image classification tasks and active learning using Gaussian processes to model a black box.

1.1 Context and Motivation

The increasing role of machine learning models in various applications highlights the need for robust and reliable models. However, there is an insufficient amount of expert-labeled data and the presence of uncertainties in these models presents challenges that need to be addressed. The motivation for this internship is to explore/develop effective methods for active learning and uncertainty estimation, we would like to know when a model can be trusted and why it should be trusted. The explainability provided by this will allow for understanding the decision-making process it goes through. In addition, the uncertainty estimates will help to optimize model performance and minimize the cost of obtaining labeled data.

1.2 The AI Lab in SLB

SLB is a multinational company that provides technology and services for the Oil & Gas industry. The AI Lab at SLB is a research-focused team that brings together AI Engineers and domain experts to work on projects related to various industries, specifically the Oil & Gas sector. The lab consists of different teams with expertise in areas such as computer vision, natural language processing, and time series analysis.

1.3 Internship Objectives

The primary goals of this internship are:

- Investigate the characteristics and properties of various sources of uncertainty in machine learning models (epistemic, aleatoric, and model error) and devise techniques for estimating and mitigating these uncertainties.
- Discover, implement and compare active learning approaches that effectively minimize the number of required labels while maximizing the model's performance on unseen data.
- Develop an understanding of the relationship between active learning and uncertainty estimation, and how these can be combined to optimize data acquisition and enhance model performance.
- Apply the developed techniques to real-world projects in the AI lab.

1.4 Thesis Outline

This thesis is structured as follows:

- Chapter 2 gives a brief background/overview on uncertainty estimation, types of uncertainty explored, and a description of active learning and its usage.
- Chapter 3 details the various uncertainty estimation techniques explored, their results, comparisons, and other experiments.
- Chapter 4 investigates what kind of uncertainty different uncertainty techniques correspond to using a simulated environment where 3 types of uncertainty can be controlled.
- Chapter 5 goes over the practical application of uncertainty estimation and active learning to 2 real-world projects.
- Chapter 6 gives a conclusion on what was explored and achieved and a discussion of potential future improvements.

Chapter 2

Background

2.0.1 Uncertainty Estimation

Uncertainty estimation is a way to measure the confidence level of the predictions derived from the model and provide an understanding of the limitations of the model. With uncertainty estimation, we can help to improve robustness, handle unforeseen situations, and quantify risk.

Epistemic Uncertainty is a type of uncertainty that arises from a lack of sufficient data necessary to train a model to its full capacity. It is linked to the knowledge about the function or system we are attempting to model. When data is limited or incomplete, this reduces the effectiveness of the model and contributes to epistemic uncertainty.

For example, if we are training a machine learning model to recognize and classify different breeds of dogs. The model should ideally be trained on a dataset that includes equal representation of each dog breed. However, if we only have a very limited number of images for certain breeds in the training set, the model may not learn to recognize these breeds as effectively as the more commonly represented breeds. This absence of sufficient training data contributes to epistemic uncertainty. Epistemic uncertainty can be reduced or eliminated by gathering more data.

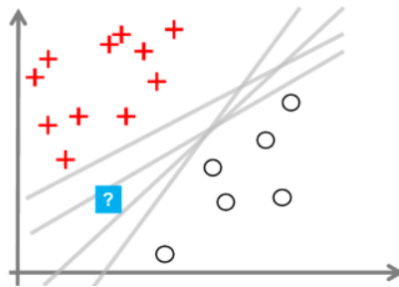


FIGURE 2.1: Data with Epistemic Uncertainty

The figure above is an example of data with epistemic uncertainty, different decision boundaries can be drawn due to insufficient data in the middle. Several methods can be used to reduce epistemic uncertainty, such as more training data and using different models. [Hüllermeier and Waegeman \(2021\)](#).

Aleatoric uncertainty is the intrinsic randomness or variability present in the data. An example of aleatoric uncertainty is the roll of a die, where each side has an equal probability of $1/6$ when the die is rolled, regardless of the amount of information about the die it is not possible to predict what side we roll. There are two categories of aleatoric uncertainty [Kendall and Gal \(2017\)](#): **homoscedastic uncertainty**, which stays constant for different inputs, an example of this could be a dog and cat classification problem where all dog and cat images is heavily obscured or distorted. This uncertainty is constant for all images, leading to a case of homoscedastic aleatoric uncertainty. **heteroscedastic uncertainty**, depends on the inputs to the model, with some inputs potentially having more noisy outputs than others. For example, some images might be obscured or distorted but this is not constant for all inputs.

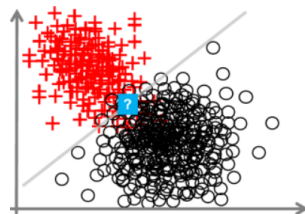


FIGURE 2.2: Data with Aleatoric Uncertainty

Figure 2.2 is an example of aleatoric uncertainty where no decision boundary will give zero error due to noise at the center of the image. Reducing aleatoric uncertainty can be done by improved data collection techniques or embedding data in a higher-dimensional space (which increases epistemic uncertainty).

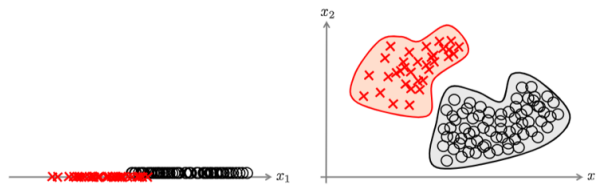


FIGURE 2.3: Reducing Aleatoric Uncertainty by Embedding in a Higher-Dimensional Space

In figure 2.3, given only x_1 there is a region where it is impossible to distinguish the two classes. However, by adding another feature x_2 along with x_1 , a perfect separation of the classes can be made.

Model uncertainty is the third type of uncertainty we will look into. It refers to the limitations in the performance of a machine learning model due to its architecture, design, or other model parameters. It arises from the model's inability to capture the true underlying relationships present in the data or generalize to new data. It can occur when a model is too simple or not complex enough to represent the complexity of the data. Predicting model uncertainty can let us know when to focus on model improvement rather than the data.

2.0.2 Active Learning

In the cases where a model is performing badly because of due to epistemic uncertainty (a lack of data), active learning provides a way. Active learning is a technique to intelligently choose the most informative samples from a large pool of unlabeled data to minimize human labeling costs and achieve high-quality labeled data. The idea of active learning is that a learning algorithm can query specific data points to be labeled by a human expert in order to train a better model using less data.

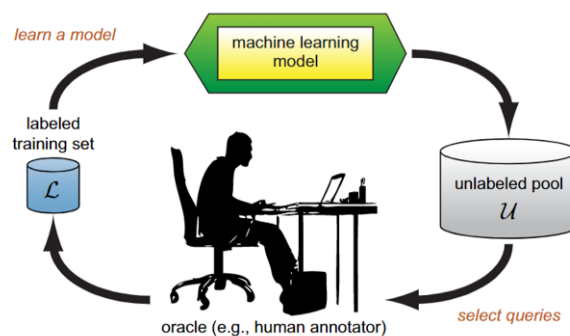


FIGURE 2.4: Active Learning Workflow

Reducing labeling costs is a reason to use active learning, in many domains, obtaining the necessary labels for a task can be costly and time-consuming, especially when experts are needed for accurate annotation. Though active learning is presented as a more efficient alternative to traditional annotation, it might not always guarantee efficiency due to factors such as the amount of unlabelled, the absence of epistemic uncertainty, the chosen query strategy, and the model itself.

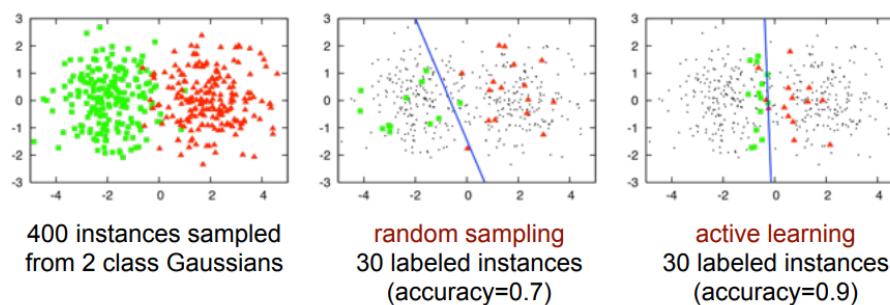


FIGURE 2.5: Active Learning vs No Active Learning from [Settles \(2009\)](#)

In figure 2.5, plot (a) represents the full dataset with the classes, (b) shows the kind of boundary we get when a few data points are picked at random. (c) Show what happens with active learning with uncertainty sampling(entropy). We are able to get a good model performance with active learning while using as little data as possible.

2.0.2.1 Active Learning Query Strategies

Active learning query strategies are techniques applied for the selection of new data points. [Kirsch and Gal \(2022\)](#) According to [Settles \(2009\)](#) active learning query strategies can be classified into:

1. **Uncertainty Sampling:** This method is connected to uncertainty estimation, the model queries the samples for which it has the highest level of uncertainty. The uncertainty can be calculated using the lowest probability, entropy, or standard deviation.
2. **Query-By-Committee:** In this method, a committee/ensemble of diverse models is created, and each member is trained on a subset of a dataset. The samples that cause the most disagreement among the committee members are queried for labeling. Two to three committees work well for this purpose, and the uncertainty can be measured using entropy or standard deviation.

3. **Expected Model Change (Expected Gradient Length):** This strategy focuses on selecting samples that maximize the expected change in the model's parameters, thereby improving generalization performance.
4. **Variance Reduction and Fisher Information Ratio:** Variance reduction approach minimizes the model's future error by reducing its variance. It estimates the variance of the model after labeling a new instance and selects the instance that results in the greatest future variance reduction.
5. **Estimated Error Reduction:** This strategy focuses on minimizing the expected error in the model by selecting the samples that are expected to reduce the model's error the most.
6. **Density-Weighted Methods:** These methods consider the distribution of the data points and their density in the feature space, selecting samples that are representative of the data. However, density-weighted methods are prone to querying outliers, since they may have a high impact on the model's performance.

For this internship, the focus was on Uncertainty Sampling and Query-By-Committee(Ensemble) techniques for active learning.

Chapter 3

Uncertainty Estimation Techniques

Dataset

The dataset used to explore and compare uncertainty techniques is an age prediction dataset, where given an image we guess the age. It is an open-source dataset from Kaggle [shahraizanwar \(2020\)](#) consisting of 23,315 grayscale images in total, images are of size (48×48) .

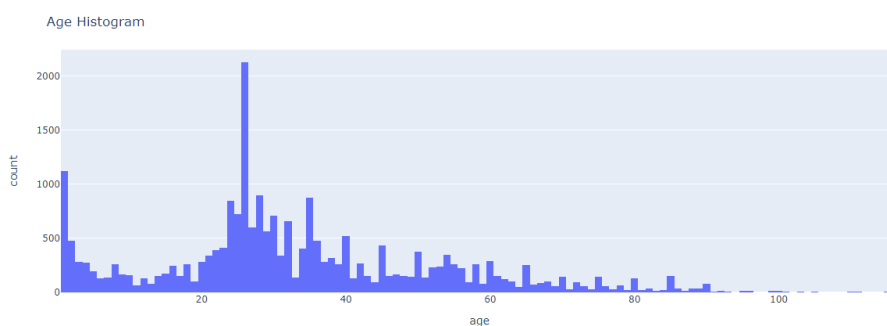
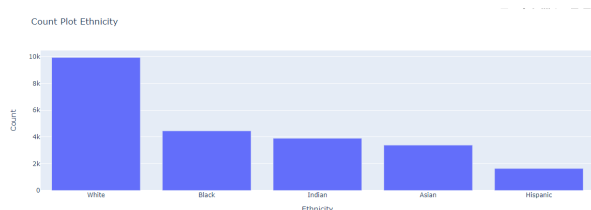
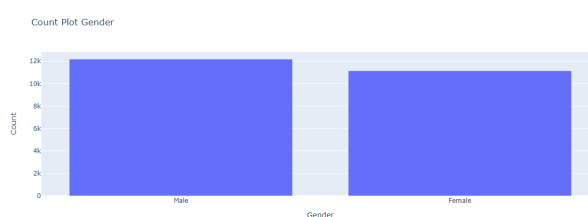


FIGURE 3.1: Age Distribution



Ethnicity Distribution



Gender Distribution

With 12,175 males and 11,140 females, the gender distribution can be considered balanced. There are 5 possible ethnicities and the classes are noticeably imbalanced. The age distribution is skewed toward ages 20 to 40 and infants, there is little data on much older people.

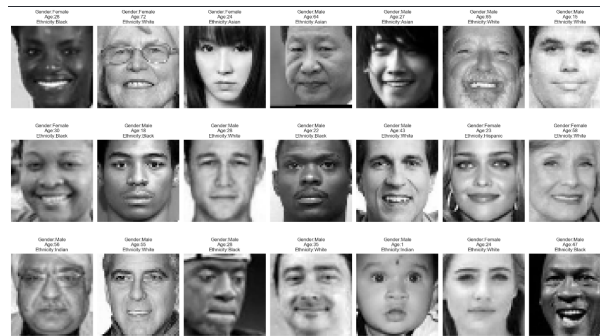


FIGURE 3.2: Example Images in Age Prediction Dataset

Estimating Uncertainty In Regression

To estimate uncertainty in a regression task, variance and standard deviation are the metrics commonly used:

1. **Variance** is a way to see how much a group of numbers is spread out from the average. It's calculated by taking the average of the squared differences from the mean. When we want to know how unsure we are about a prediction made by a model, variance can help. A high variance means the predictions have high uncertainty, while a low variance means the predictions are dependable.

Mathematically, the variance is calculated as follows:

$$\sigma^2 = \frac{\sum_{i=1}^n (x_i - \mu)^2}{n}$$

where x_i stands for each label, e.g. age in the group, μ is the average of the group, and n is how many numbers there are.

2. **Standard Deviation** is related to variance and is found by taking the square root of the variance. Since standard deviation has the same unit as the labels in the group it is easier to interpret the output. Just like variance, standard deviation helps show how much our errors made by a model are spread out.

Evaluating Uncertainty Estimates In Regression

To know how well the provided uncertainty estimate is, we need an evaluation metric. Pearson and Spearman correlation were explored for this purpose, what we are seeking to measure with these correlation coefficients is the relationship between the model's estimated uncertainty and

its actual error rates. So, if a model reliably estimates its own uncertainty, we anticipate a strong correlation between the model's estimated uncertainties and its actual errors.

1. Pearson Correlation

The Pearson correlation coefficient is a measure of the linear relationship between two variables. It ranges from -1 to 1, where -1 represents a perfect negative linear relationship, 1 represents a perfect positive linear relationship, and 0 indicates no linear relationship. In the context of uncertainty estimation, the Pearson correlation can be used to assess the strength and direction of the linear relationship between the predicted uncertainty and the absolute error of a model's predictions. However, there are some disadvantages to using Pearson correlation according to [Ramzai \(2021\)](#), it's sensitive to outliers and does not account for non-linear relationships between variables.

2. Spearman Correlation

The Spearman correlation coefficient, also known as Spearman's rank correlation, is a non-parametric measure of the monotonic relationship between two variables. The Spearman correlation is less sensitive to outliers and non-linear relationships than the Pearson correlation.

These 2 evaluation metrics were used in this section, some other ways to evaluate uncertainty estimates are using the Average Root Mean Squared error (RMSE) and Loglikelihood (LL) [Sluijterman et al. \(2021\)](#).

3.1 Monte Carlo Dropout

Dropout is a popular technique used in machine learning to estimate uncertainty in the predictions of models that are in the form of neural networks. It is also one of the simplest ([Abdar et al., 2021](#)). This process involves adding dropout layers into the neural network both during training and testing phases. During training, dropout layers randomly drops out some of the neurons for each batch of training samples, which serves to prevent overfitting [Srivastava et al. \(2014\)](#). For testing, the dropout layers are intentionally activated to produce multiple predictions for each data point, where each prediction is derived from different random dropout masks ([Gal and Ghahramani, 2016](#)). These predictions are then utilized to estimate the uncertainty

through statistical methods such as Monte Carlo sampling by considering the variability of these predictions.

Nine studies have reported MC dropout as the most effective uncertainty quantity technique [Abdar et al. \(2021\)](#). MC dropout is also one of the most popular uncertainty estimation techniques.

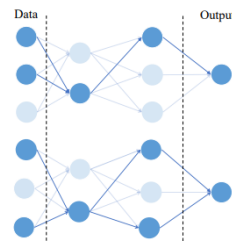


FIGURE 3.3: Monte Carlo (MC) Dropout

The figure above is an example of MC dropout, it shows 2 different calls to a model, with each randomly dropping out different neurons. The variance between the different outputs can be used as an uncertainty estimate.

3.1.1 Effect of p on Uncertainty Estimation and Model Performance

p refers to the dropout probability. The network used is a 4-layer CNN with two dropout layers, one after the convolutional layers and the other after the fully connected layer.

Parameters

$mc_dropout_samples = 100$

$epochs = 50$

p		MAE	Loss	Pearson Corr.	Spearman Corr.
0.5	Train	3.7934	1.1492	0.2274	0.2737
	Test	6.4906	2.6134	0.3685	0.4193
0.8	Train	3.7478	1.1347	0.2256	0.2734
	Test	6.5446	2.6411	0.3709	0.4239

TABLE 3.1: MC Dropout Results

The results above show that for a higher value of p we have a slightly lower correlation on the train set but a higher correlation on the test set which indicates a better uncertainty estimate

with higher dropout probabilities. In general, there is no best selection of p , and the selection of p should be influenced by the size and complexity of the model [Vergara et al. \(2019\)](#).

3.2 BNNs(Bayesian Neural Networks)

Bayesian Neural Networks (BNNs) are a class of neural networks that use Bayesian inference to estimate uncertainty in the model's predictions. Unlike traditional neural networks that have fixed weights and biases, BNNs treat these parameters as random variables and model their distributions [Hüllermeier and Waegeman \(2021\)](#). By propagating the uncertainty through the network, BNNs can provide probabilistic predictions that incorporate uncertainty. Uncertainty estimates from BNNs can be used to quantify the model's confidence in its predictions. BNNs have shown promising results in tasks where uncertainty estimation is crucial, such as medical diagnosis and autonomous driving although they require more computational resources and can be challenging to train compared to traditional neural networks.

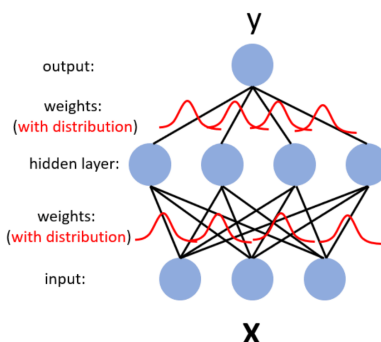


FIGURE 3.4: Bayesian Neural Network

From the literature review, [Abdar et al. \(2021\)](#), BNNs have a good record for estimating uncertainty in deep learning architectures. Bayesian CNNs have been used successfully to estimate a model's re-localization uncertainty [Kendall and Cipolla \(2016\)](#).

For exploring BNNs, the Bayesian Torch library [Krishnan et al. \(2022\)](#) was used. It provides us with a set of configurations to transform an existing deterministic neural network into a BNN.

3.2.1 Dropout Vs No Dropout For Monte Carlo Sampling

The goal of this experiment is to see if turning on dropout during bayesian inference with BNNs leads to better performance. Even though BNNs provide a distribution, enabling dropout during inference was explored to see if combining both provides a better estimate of uncertainty.

The network used is a 4-layer CNN with two dropout layers, one after the convolutional layers and the other after the fully connected layer.

Number of runs = 1

Parameters: $mc_samples = 100$, $epochs = 50$, BNN Type: *Flipout*

Dropout	MAE	Pearson Corr.	Spearman Corr.
None	6.3702	0.4158	0.4695
0.5	6.3610	0.4203	0.4697
0.8	6.3795	0.4164	0.4736

TABLE 3.2: Dropout Vs No Dropout For Monte Carlo Sampling(Test)

The result shows that turning on dropout improves the Uncertainty Correlation slightly.

3.2.2 Comparison of Different BNN Configurations

There are different configurations for BNNs such as reparameterization [Blundell et al. \(2015\)](#) and flipout [Wen et al. \(2018\)](#). Reparameterization is a technique used for variational inference that views model parameters as random variables with distributions instead of deterministic variables. Flipout is similar to reparameterization with improvement in performance and speed for large batches. A comparison was done to identify the most performant in terms of correlation with the absolute error of the model. Average of 5 runs.

Config Name	MAE	Pearson Corr.
Reparameterization	6.74	0.416
Reparameterization Large	6.65	0.392
Flipout	6.72	0.419
Flipout Large	6.51	0.396

TABLE 3.3: BNN Configuration Comparison Results(Test Set)

3.3 Autoencoders

Autoencoders are a class of neural networks that are commonly used for unsupervised learning tasks, such as data compression and denoising. However, they can also be used for estimating uncertainty in a model's predictions. The approach we explored is to use the reconstruction error of the autoencoder as a measure of uncertainty [Cruz et al. \(2022\)](#). During training, the autoencoder learns to reconstruct the input data, and during testing, the difference between the input and the reconstructed output can be used as a measure of uncertainty. The idea behind this is that if the autoencoder is well-trained, it should be able to reproduce the input data effectively. If the reconstruction is significantly different from the input when it attempts to reconstruct it during testing, this suggests that the model is uncertain about its response.

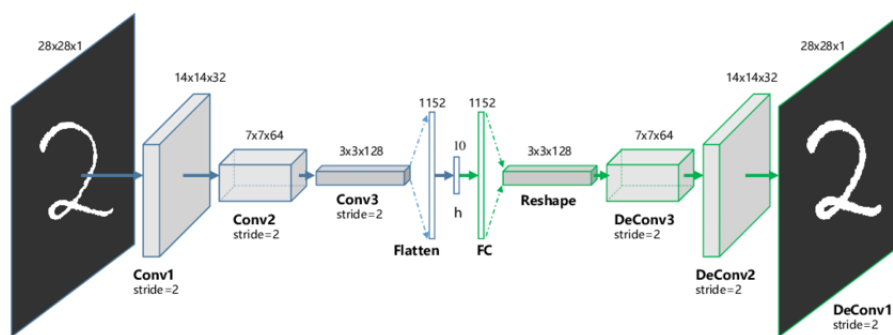


FIGURE 3.5: Autoencoder

3.3.1 Comparison of Reconstruction Loss and Model Absolute Error

For this experiment, we tested different autoencoder models and compared the reconstruction loss of the autoencoder, to the absolute error of the age prediction model but also to evaluate how differences in architecture affected these aspects. Having a variety of architectures enables us to gain more understanding of how different architectures contribute to model performance.

Autoencoder Models

1. Architecture 1

- (a) The encoder has 5 convolutional layers with batch normalization for an in-depth feature extraction and leaky ReLU activation function. The decoder has 4 transposed convolutional layers with batch normalization and ReLU activation function followed by a final sigmoid activation function.
- (b) Loss function: Binary cross-entropy loss

2. Architecture 2

- (a) A simpler architecture compared to Architecture 1
 - (b) The encoder consists of two convolutional layers with a kernel size 3 and padding size 1, followed by a max pooling layer with a kernel size of 2 and stride of 2. The decoder consists of two transposed convolutional layers. Both transposed convolutional layers have kernel size 2 and stride 2. The final layer of the decoder applies a sigmoid activation function to the output.
 - (c) Loss function: Binary cross-entropy loss
3. Architecture 3
- (a) The encoder consists of 6 convolutional layers with a kernel size of 3x3, padding of 1 and stride of 2. Between each pair of convolutional layers, there is an activation function, which is a GELU function. The last convolutional layer is flattened into a vector and fed into a fully connected layer to produce the latent representation.
 - (b) The decoder is a mirror image of the encoder, consisting of 6 convolutional layers with a kernel size of 3x3, padding of 1 and stride of 2. The activation function is again a GELU function. The last layer is a Tanh activation, which scales the output image to the range $[-1, 1]$.
 - (c) Loss function: MSE

Architecture	Latent Space	Pearson Corr.
1	32	0.0400
1	1152	-0.0010
2	2304	-0.0200
3	64	-0.0500
3	384	-0.0800

TABLE 3.4: Comparison of Reconstruction Loss to Model Absolute Error(Test Set)

From the experiments, there is no correlation between the reconstruction loss of our autoencoder to the age prediction model true error for this dataset.

3.3.2 Using the Encoder of An Autoencoder to Train the Model

We hypothesize that the poor correlation may be due to a contrasting neural network architecture between the autoencoder and the Age Prediction Model. The age prediction model now uses the output of the encoder with the addition of 2 linear layers for predicting age. We tested by freezing the encoder and using it normally. The autoencoder architecture used is *Architecture 1*.

The baseline model without using the encoder head has an MAE of 6.36(test).

Is Frozen	Latent Space	Age Ped Model MAE	Pearson Corr.
True	1152	10.40	0.08
False	1152	6.90	0.04

TABLE 3.5: Comparison of Reconstruction Loss to Model Absolute Error(Test Set)

We found no correlation between autoencoder reconstruction error and the true error of the age prediction model predictions after various tests, such as:

1. Different autoencoder architectures and hyperparameters
2. Different latent space dimensions, this refers to the compressed representation of our input data, and adjusting its size can impact the ability of the model to reconstruct.
3. Using the already trained encoder from the autoencoder to train the age prediction model

We conclude that the autoencoder might not be a good estimator due to the fact that it is trained on the same data as the age prediction model, therefore creating bias in the predictions. To confirm we would like to explore whether an autoencoder trained on a subset of the data would be effective at:

1. Reconstructing Out of Distribution data (OOD data in this case refers to the different age groups excluded). The idea behind this is that we would expect the uncertainty to increase when estimating ages that are far from the range used for training. For instance, a model trained exclusively on people over age 20 would be less accurate at predicting the age of a 3-year-old compared to a 15-year-old.
2. Estimating the uncertainty of the age prediction model predictions on OOD data

3.3.3 Autoencoder Reconstruction Loss with Out-of-Distribution (OOD) Data

The goal here is to train an autoencoder on data for people aged 20 and above and test on data for people aged less than 20 to evaluate the effect of OOD on the autoencoder's performance, as this could be used as an uncertainty estimate.



FIGURE 3.6: Autoencoder prediction on validation set age range(>20)



FIGURE 3.7: Autoencoder prediction on validation set age range(<20)

Figure 3.7 shows the autoencoder prediction on OOD ages, it tends to reconstruct people older than they are as it was trained on people aged >20(Figure 3.6). The reconstruction loss was also higher in these cases

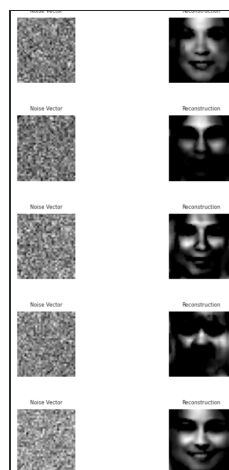


FIGURE 3.8: Autoencoder Reconstruction on Randomly Generated Noise

From figure 3.8 given a random noise, the reconstruction always resembles faces which shows that the autoencoder has learned the distribution of human faces and not copying information. In addition, the reconstructed faces from the generated noise seem to be faces over 20.



FIGURE 3.9: Autoencoder Reconstruction Loss vs Age range(<20)

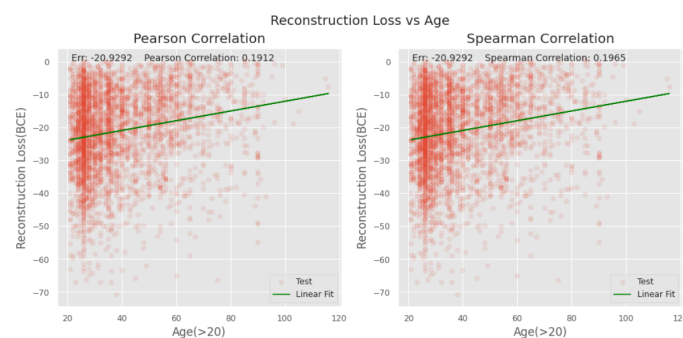


FIGURE 3.10: Autoencoder Reconstruction Loss vs Age range(>20)

The results from the plots are noisy, there isn't really a clear trend, especially when taking into account the frequency distribution of the ages.

3.4 Ensembles

Ensemble methods involve combining multiple models to improve predictive accuracy and estimate uncertainty. One approach is to use a technique called bagging, which involves training multiple models on different subsets of the training data and combining their predictions [Abdar et al. \(2021\)](#). Ensembles can be used for estimating uncertainty by combining the predictions of multiple models to obtain a distribution of predictions. The spread of this distribution can be used as a measure of uncertainty. From [Hüllermeier and Waegeman \(2021\)](#), Ensembles have shown promising results in tasks such as classification and regression. However, ensembles can be computationally expensive and require significant resources to train and deploy.

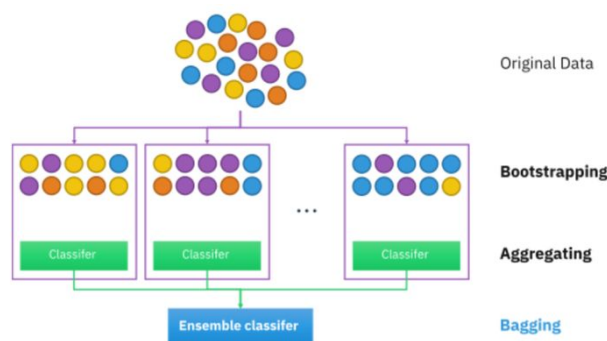


FIGURE 3.11: How Ensembles Are Trained

3.4.1 Effect of Number of Epochs on Uncertainty Estimation

We want to know if the quality of the uncertainty estimate depends on the extent of model training, for example, would an overfitted ensemble provide a better uncertainty estimate since it is trained on different subsets of data? To confirm this we explored the following scenarios:

1. Converged (Optimal Training): Optimized for best test set performance
2. Underfitted (Insufficient Training): Model training stops before being converged.
3. Overfitted (Excessive Training): Excessive number of epochs and high learning rate, optimized for training set performance.

Parameters: $n_ensembles = 5$. Number of runs = 5. Same Seed.

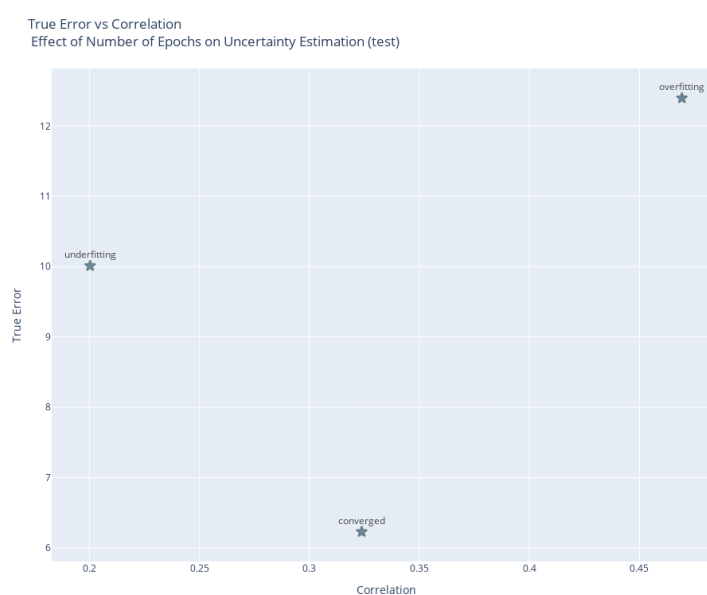


FIGURE 3.12: MAE vs Pearson Correlation of Uncertainty

The results show that the optimal training scenario, where the model is converged, resulted in the best error rate and the second-best correlation for both train and test sets. The excessive training scenario, where the model is overfitted, resulted in the worst error rate (worse than underfitting) but the best correlation for both train and test sets. The insufficient training scenario, where the model is underfitted, resulted in the second-worst error rate and the worst correlation for both train and test sets. Therefore, the number of epochs used for training can have an impact on the uncertainty estimates. Overfitting can result in a better correlation between predicted uncertainty and true errors, but it comes at the cost of reduced overall accuracy and increased risk of making incorrect predictions on new, unseen data

3.4.2 Effect of Network Complexity on Uncertainty Estimation

We want to investigate the effect of network complexity on uncertainty estimation. Specifically, we want to test the hypothesis that a more complex network will lead to better uncertainty estimates. The idea behind is this that complex networks are typically more capable of understanding complex patterns in the data. Therefore, they are likely to make accurate predictions along with better estimates of uncertainty.

To test this hypothesis, we will compare the performance of 5 networks with different levels of complexity. The details of the networks are:

1. **Standard:** A 4-layer CNN with 2 fully connected layers
2. **Simple:** A 2-layer CNN with 2 fully connected layers
3. **Resnet:** A frozen Resnet 18 network, with only the last fully connected layer updatable
4. **Resnet_unfreeze:** A Resnet 18 network, with all parameters updatable
5. **PT:** A 6-layer CNN with 2 fully connected layers with small channel sizes.

Parameters: $n_{ensembles} = 5$

Number of runs = 5.

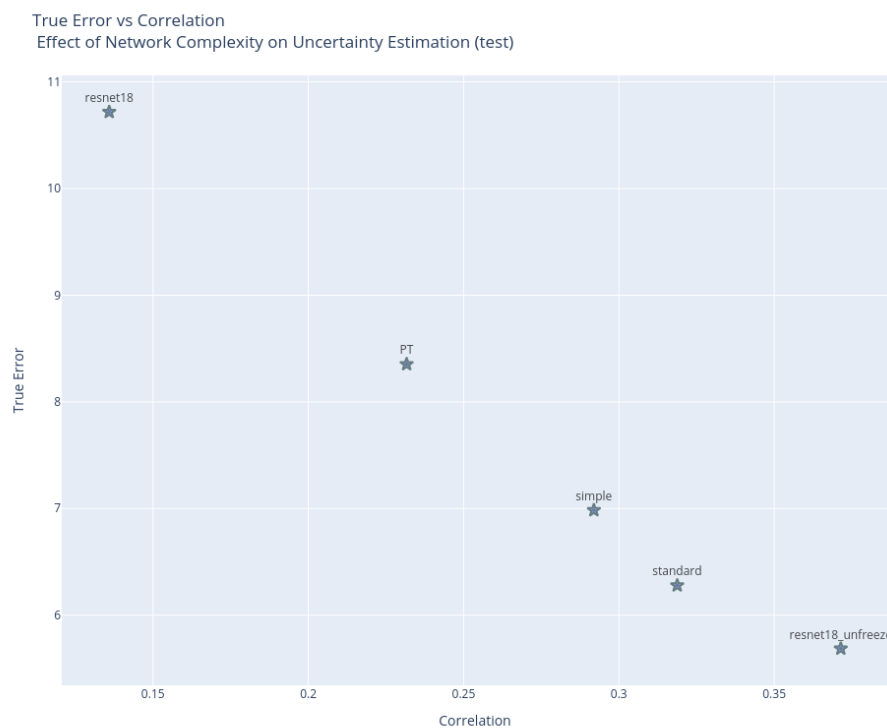


FIGURE 3.13: Mean Absolute Error vs Pearson Correlation(Test)

From the experiment, we can conclude that complex networks generally provide better uncertainty estimates due to their ability to capture complex patterns and relationships in the data. In addition, there seems to be a correlation between the models true error and uncertainty correlation, this indicates model performance correlates with uncertainty estimation, however, more exploration needs to be done to find out why worse networks cannot estimate their uncertainty well.

3.4.3 Effect of Number of Ensembles on Uncertainty Estimation

In this experiment, we investigate the effect of the number of ensembles on uncertainty estimation. Specifically, we want to test the hypothesis that increasing the number of ensembles will lead to better uncertainty estimates.

To test this hypothesis, we will compare the performance of 5 ensemble models with 2, 5, 10, 20 and 50 models.

Number of runs = 5

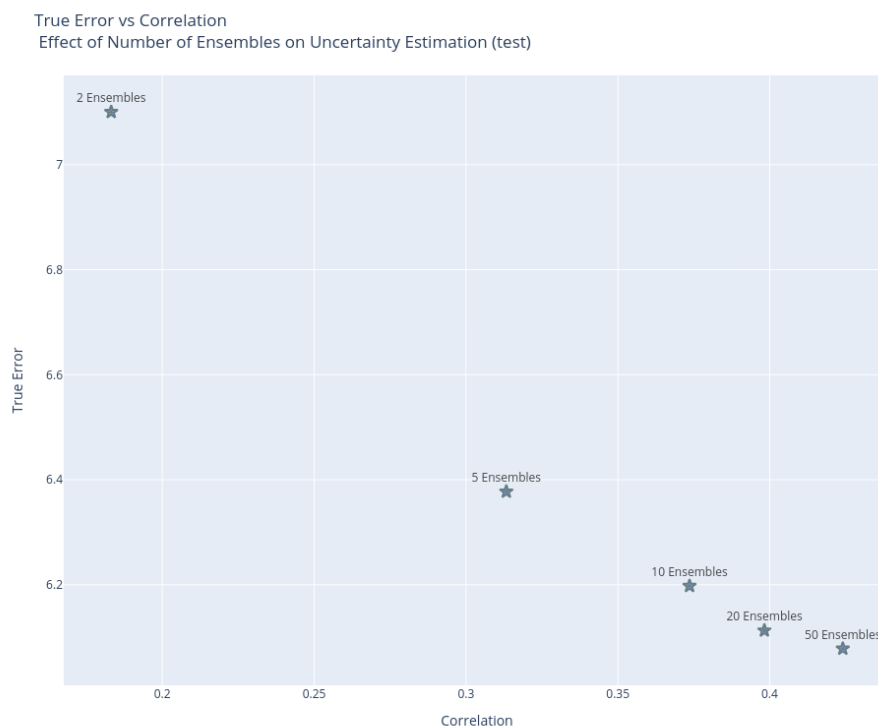


FIGURE 3.14: Mean Absolute Error vs Pearson Correlation(Test)

From the experiment, we can conclude that increasing the number of ensembles generally leads to better uncertainty estimates, as it allows for better capture of the model's uncertainty. However, the improvement is more significant when going from 2 to 5 models, and the gains are diminishing beyond 10 models. Therefore, there is a trade-off between computational cost and performance that needs to be considered when choosing the number of models in an ensemble.

3.4.4 Effect of Ensemble Composed of Different Architectures on Uncertainty Estimation

In this experiment, we investigate the effect of an ensemble composed of different architectures on uncertainty estimation. Specifically, we want to test to see that an ensemble composed of different architectures will lead to better uncertainty estimates.

To test this hypothesis, we will compare the performance of 2 ensembles with different compositions of architectures and 3 with singular architectures. The details of the ensembles are:

1. **Multi_architecture(CNN)**: An ensemble composed of 1-5 layer CNNs.

2. **Multi_architecture(mix)**: An ensemble of
 - (a) 2 Layer CNN
 - (b) 4 Layer CNN
 - (c) 6 Layer CNN
 - (d) 4 Linear Layers
 - (e) Resnet 18
3. **Single Architecture(standard)**: An ensemble composed of five 4 layer CNNs
4. **Single Architecture(pt)**: An ensemble composed of five 6 layer CNNs with small channel sizes
5. **Single Architecture(Resnet 18)**: An ensemble composed of five Resnets

Number of runs = 3.

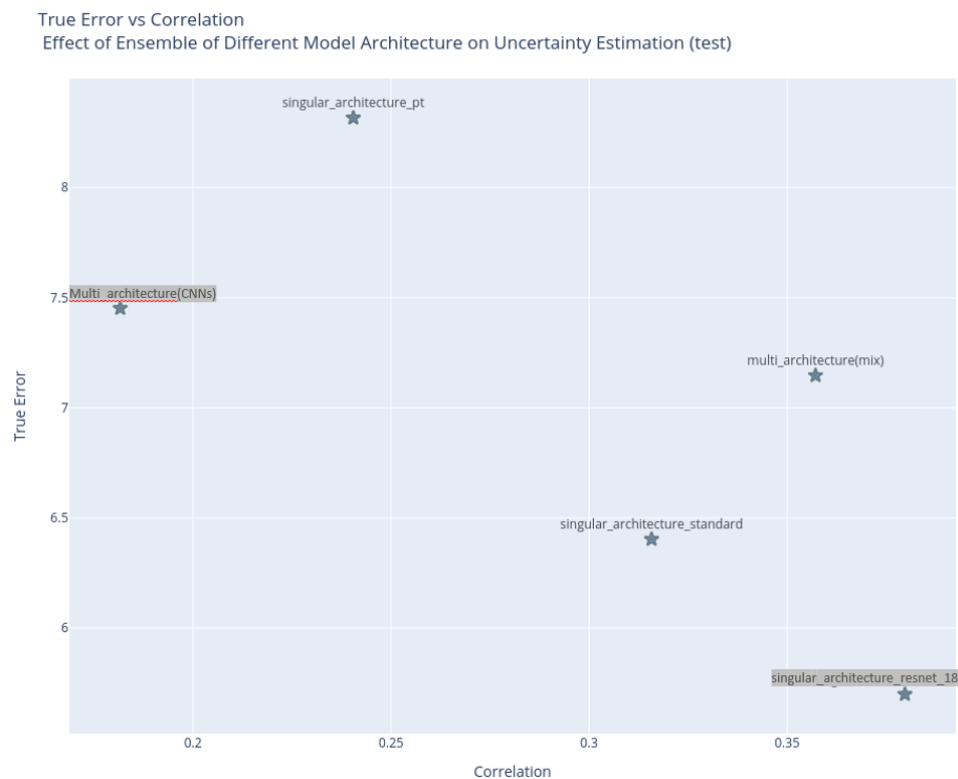


FIGURE 3.15: Mean Absolute Error vs Pearson Correlation(Test)

The reason why the *Multi_architecture(CNN)* performed the worst could be that the ensemble is composed of similar architectures that have similar weaknesses, leading to a lack of diversity

in the predictions. On the other hand, the *Multi_architecture(mix)* ensemble includes architectures with different strengths and weaknesses, providing a more diverse set of predictions that can be combined to give better uncertainty estimates which came in second. For *Singular Architecture(Resnet)* ensemble which outperformed others, it appears that network complexity may take precedence over mixing different architectures when it comes to uncertainty estimation. ResNet is a complex network with a deeper architecture, having 5 of these beats having *Multi_architecture(mix)* (a mix of different architecture where the average architecture complexity is lower). This suggests that for uncertainty estimation, having a more complex architecture may be more important than mixing different architectures. However, further experiments with a wider range of architectures would be needed to confirm this hypothesis.

3.5 Ensemble Model Pairs

Ensemble Model Pairs involve combining an ensemble model with an error model to estimate uncertainty in the ensemble's predictions. The error model can be used to predict the errors of each model in the ensemble, the error of the ensemble as a whole or merged with the individual model to predict errors. By using the error model to estimate uncertainty, Ensemble Model Pairs is expected to have a more accurate measure of uncertainty than traditional ensembles.

3.5.1 Different Uncertainty Metrics And What They Mean

LM(Label Model): Refers to prediction for label(age)

EM(Error Model): Refers to the prediction of the error of the Label Model

Error in this case refers to any error estimates such as: absolute error, variance, std.

1. Std of Label Models(std_lm): This is the standard deviation of label models, which is the same measure used for standard ensemble models. Taking the standard deviation of label models in an ensemble provides a measure of the spread or dispersion in the predictions made by the individual models. A higher standard deviation indicates a higher degree of disagreement among the models, which can be interpreted as a higher level of uncertainty in the ensemble's predictions.
2. Mean of Error Models (mean_em): The mean of error models represents the average uncertainty produced by the ensemble's individual models. This metric provides an estimate

of the overall accuracy of the ensemble. A lower mean error indicates a more accurate ensemble, while a higher mean error points to higher uncertainty in the ensemble's predictions.

3. Std of Error Models (std_em): The standard deviation of error models measures the spread or dispersion in the errors made by the individual error models in the ensemble. A high standard deviation of error models suggests a high level of disagreement among the models about the level of uncertainty, this indicates that the error models are not sure of what uncertainty the label model provides.

The several ensemble model pair techniques explored are:

1. Many to One: Using one error model to predict the error of an ensemble of label models
2. Many to Many: Uses an ensemble containing pairs of error models and label models. Where in each pair the error models predict the error of the label model.
3. Many to Many Connected: Similar to many-to-many but the error model now takes prediction from all label models instead of a single label model.
4. Dual Task: Instead of having a separate error model the prediction model predicts both mean and variance

An ensemble of 5 is used in all cases. Baseline result for ensemble:

	MAE	Pearson Corr.
Train	3.81	0.47
Test	6.28	0.32

TABLE 3.6: Ensemble Baseline Result

3.5.2 Many to One

The Many to One technique involves using one error model to predict the error of an ensemble of label models. This approach is less computationally expensive than other Ensemble Model Pair techniques since only one error model is used. However, it may not be as accurate as other techniques since the error model may not be able to capture the full range of errors present in the ensemble.

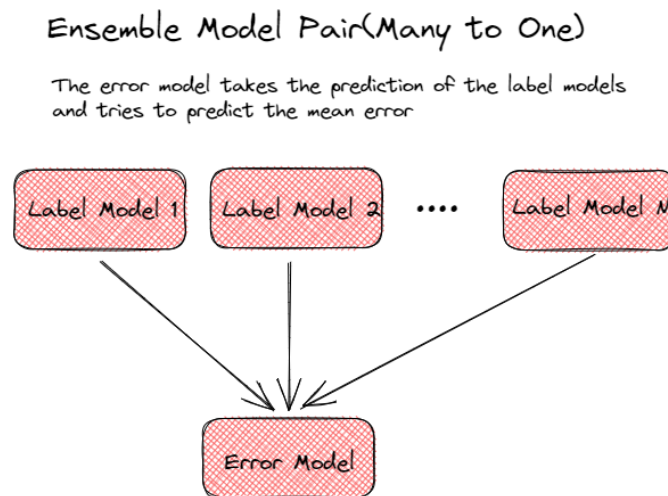


FIGURE 3.16: Ensemble Error Model Pair (Many To One)

In the Many to One technique, an error model is trained alongside an ensemble of label models. The error model has the same architecture as the label model and takes in the image as well. The error model takes the combined outputs of each label model and generates an uncertainty estimate. This uncertainty estimate can be used to provide a more accurate measure of uncertainty than traditional ensembles.

Label Model: Image $\rightarrow$ Prediction

Error Model: (Image, LM Prediction, Std of LM Prediction) $\rightarrow$ Error Estimate

Results

	LM MAE	EM MAE	Pearson Corr.
Train	3.44	2.40	0.35
Test	6.13	2.30	0.35

TABLE 3.7: Ensemble Error Model Pair (Many To One) Result)

The Pearson correlation in the table above is the correlation between the absolute error and the error model prediction.



FIGURE 3.17: Absolute Error vs Error Estimate

Compared to normal ensembles, the anomaly of the group of points close to the middle in the figure can indicate that the error model is not accurately predicting the uncertainty for samples that are close to the mean prediction. This could be due to a lack of diversity in the ensemble or an oversimplified error model. One possible solution to this problem is to increase the diversity of the ensemble or to use a more complex error model that can capture the uncertainty more accurately. Overall, there is a slightly higher correlation, however, it requires training both the label model and the error model, which can be computationally expensive and time-consuming.

3.5.3 Many to Many

The Many to Many technique involves using an ensemble of error models to predict the error of an ensemble of label models. The idea is that this approach might provide a more accurate measure of uncertainty since multiple error models can capture a wider range of errors present in the ensemble. However, it is also more computationally expensive since multiple error models are used.

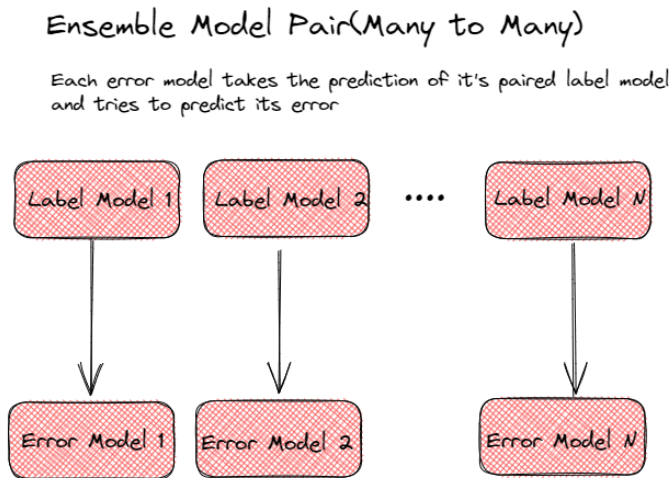


FIGURE 3.18: Ensemble Error Model Pair (Many To Many)

In the Many to Many technique, an ensemble of error models is trained alongside an ensemble of label models. Each error model takes in the image and the output of a single-label model and generates an uncertainty estimate. The outputs of all the error models are then combined to generate a final uncertainty estimate.

Label Model: Image $\rightarrow$ Prediction

Error Model: (Image, LM Prediction) $\rightarrow$ Error Estimate

Results

	LM MAE	Pearson Corr.	Spearman Corr.
Test	6.178	0.3304	0.38641
Train	3.6832	0.2580	0.2982

TABLE 3.8: Ensemble Error Model Pair (Many To Many) Result

The Pearson correlation in the table above is the correlation between the absolute error the error model prediction.

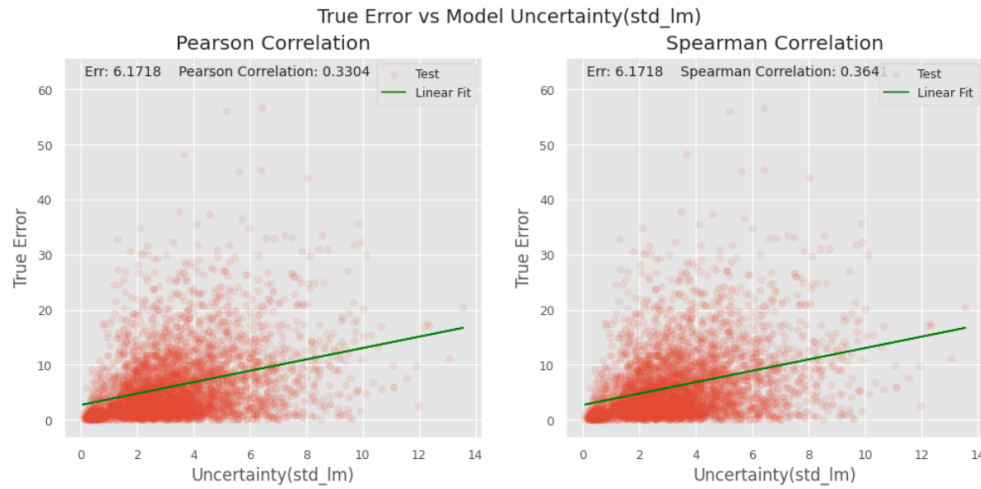


FIGURE 3.19: Mean Absolute Error vs Error Estimate(Test)

The Many to Many technique shows a lower Pearson correlation between the error estimates and the actual errors in comparison to the Many to One technique. This indicates that the Many to Many technique may not be as effective in capturing uncertainty in the model as the Many to One technique.

3.5.4 Dual Task

The Dual Task technique involves training the label model to predict both the mean and variance of the target distribution. This approach can be more accurate than other Ensemble Model Pair techniques since it directly models the target distribution instead of estimating uncertainty from errors. [Kendall and Gal \(2017\)](#) referred to the variance as heteroscedastic aleatoric uncertainty.

[Ghosh et al. \(2021\)](#) also referred to this as heteroscedastic aleatoric uncertainty. They implemented *EnsembleHeteroscedasticRegression* which is synonymous to our implementation with the exception of the way variance is calculated.

$$\text{Var}_{ensemble} = \frac{\sum(\text{Var} + \text{Mu}^2)}{\text{Num}_{models}} - \text{Mu}_{ensemble}^2$$

Where:

- $\text{Var}_{ensemble}$: Total variance of the ensemble predictions
- Var: The variance(sigma) predictions of each model in the ensemble
- Mu: The mean (mu) predictions of each model in the ensemble

- $\text{Num}_{\text{models}}$: Number of models in the ensemble
- $\text{Mu}_{\text{ensemble}}$: Mean of the ensemble predictions, calculated as the average of the mean predictions from all models.

For our implementation, we take just the mean of the combined variance of each model.

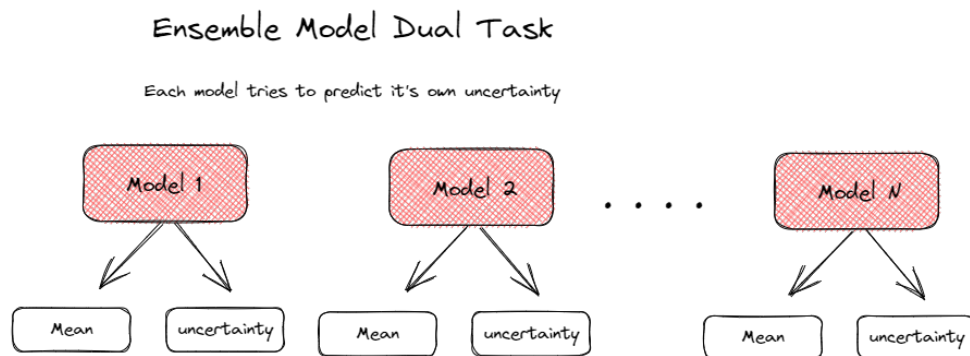


FIGURE 3.20: Dual Task Ensemble Model Pair

In the Dual Task technique, the label model is trained to predict both the mean(μ) and variance(σ) of the target distribution using a Gaussian negative likelihood loss.

Model: Image $\rightarrow$ Mean, Variance

Results

	LM MAE	Pearson Corr.(mean_em)	Spearman Corr. (mean_em)	Pearson Corr.(std_lm)	Spearman Corr. (std_lm)
Test	6.4986	0.4918	0.5201	0.3724	0.4072
Train	4.9452	0.6122	0.5997	0.5157	0.5438

TABLE 3.9: Many to Many Connected Ensemble Error Model Result

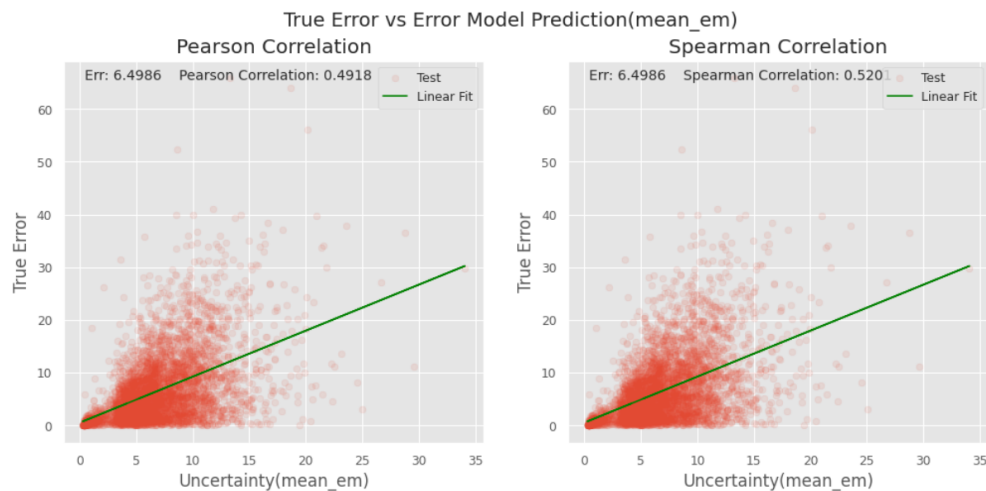


FIGURE 3.21: Dual Task: Absolute Error vs Uncertainty(mean_em)

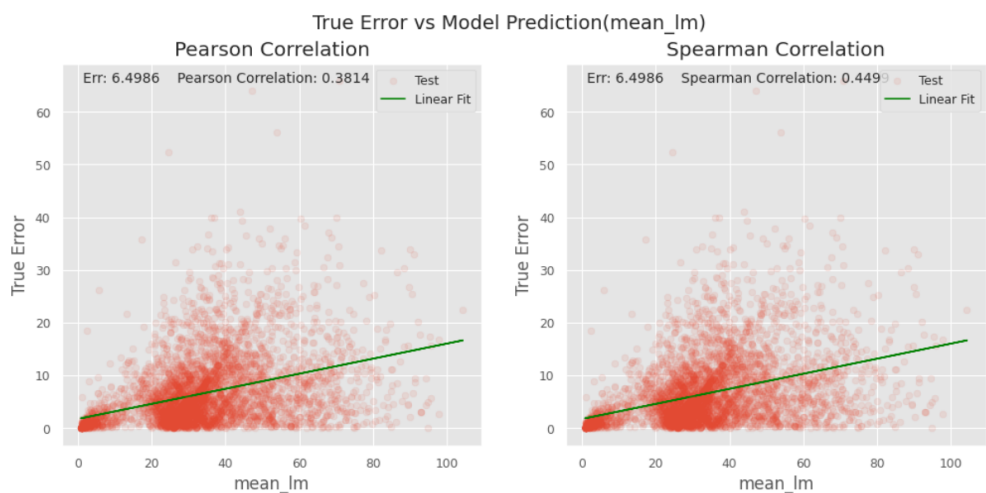


FIGURE 3.22: Dual Task: Absolute Error vs Uncertainty(mean_lm)

The Dual Task technique is able to model both the mean and std of the target distribution, resulting in a more accurate prediction. The mean prediction has a lower MAE compared to other techniques. The uncertainty produced has the highest correlation with the model's absolute error out of all explored techniques.

Potential Next Steps for Ensemble Model Pair

1. Test out technique use in [Kang and Kang \(2023\)](#), Surrogate approach to uncertainty quantification of neural networks for regression.
2. Try out the noise technique use in [Kendall and Gal \(2017\)](#) to see if that improves performance.

3. Calculate ensemble uncertainty using [Ghosh et al. \(2021\)](#) formula.

3.6 Predicting Absolute Error vs Variance in Error Models

1. Predicting the absolute error: In this variant, the model is trained to predict the mean of the target distribution and the absolute error between the true and predicted values. The main advantage of this method is that it provides a direct measure of the error, which can be useful for applications where you need to quantify the difference between the true and predicted values

Disadvantages of predicting absolute error include: .

- (a) It may be sensitive to outliers, as large errors can dominate the training process.
 - (b) The predicted errors may not accurately capture the underlying structure of the uncertainty, as they are based solely on the difference between the true and predicted values.
2. Predicting the variance: In this variant, the model is trained to predict the mean of the target distribution and the variance, which represents the spread of the target distribution. This approach is more closely aligned with the concept of heteroscedastic aleatoric uncertainty, as it directly models the target distribution's variability.

The advantages of predicting variance include

- (a) It provides a more comprehensive estimate of uncertainty by considering the entire target distribution, rather than just the difference between true and predicted values.

In conclusion, we have taken a look at dropout, ensembles, BNNs, error models, and autoencoders. Dropout is the easiest uncertainty technique to use, and best in terms of computational efficiency [3.1](#), it provides decent uncertainty estimates but gets outperformed by BNNs, ensembles, and model pairs [3.13.4](#). The uncertainty estimate of ensembles improves with the number of models, although only to a certain degree where each addition no longer improves it. Having an ensemble of complex architecture seems more important than mixing different architectures. The mean prediction of dual-task [3.4](#) ensembles provided the best correlation with the model's true error. Based on the results, there was no correlation between the autoencoder reconstruction loss and the model's error. [3.2](#).

Some of these techniques might predict different types of uncertainties and correlation with the model's error might be the best way to evaluate these techniques. The next section explores a simulated environment to identify what types of uncertainties some of these techniques provide.

Chapter 4

Uncertainty Estimation Simulator

Given the various uncertainty estimation techniques explored, we have no idea what type of uncertainty they correspond to in order to effectively compare and select the best. For active learning, the required uncertainty to detect is epistemic uncertainty which reduces the more data is added. To better understand how the different types of uncertainties from the various uncertainty estimation techniques, we created a simulated environment to emulate a real scenario. By doing so, we should have an insight into what uncertainty our explored techniques correspond to.

4.1 Approach

To do this we need a way to control the 3 types of uncertainty we are looking at: epistemic, aleatoric and model uncertainty.

4.1.1 True Function

The true function used for the simulation is:

$$y = \sin(x) + \cos(x) \quad -6 \leq x \leq 6$$

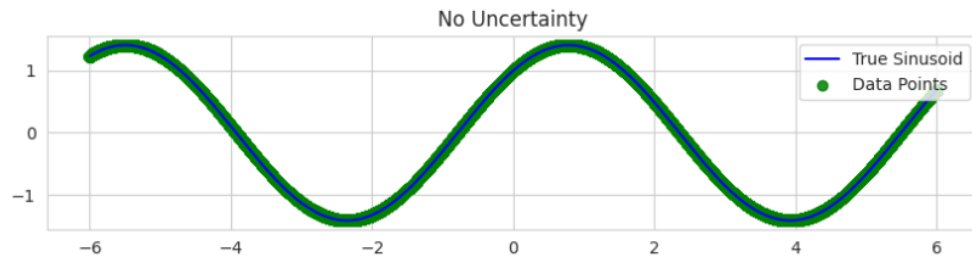


FIGURE 4.1: True function for simulator

The function and range were chosen because it adheres to our requirements which were that each additional polynomial increases the models performance.

4.1.2 Epistemic Uncertainty

Since epistemic uncertainty is related to lack of data, we should have not epistemic uncertainty with an infinite amount of data. To model epistemic uncertainty, we control the amount of data. Reducing the amount of data increases epistemic uncertainty and increasing data reduces epistemic uncertainty.

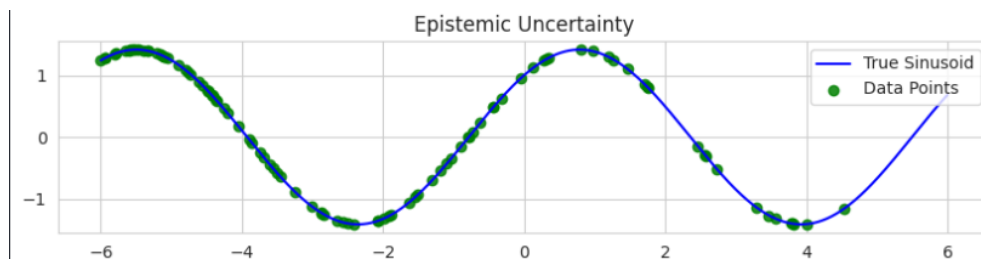


FIGURE 4.2: Example of Epistemic Uncertainty in Simulator

4.1.3 Aleatoric uncertainty

Aleatoric uncertainty is the inherent noise present in the observed data, aleatoric uncertainty can be reduced/increased by controlling the amount of noise present in the data.

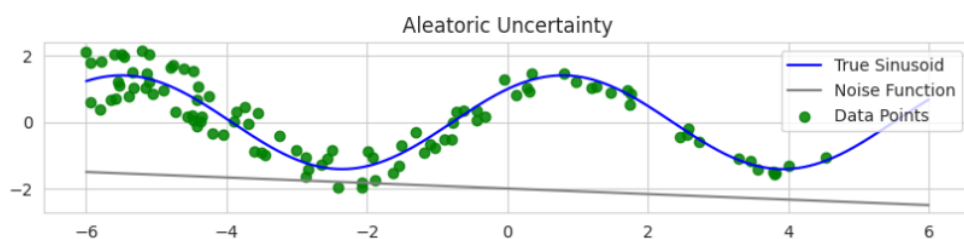


FIGURE 4.3: Example of Aleatoric Uncertainty in Simulator

To control aleatoric uncertainty we use a noise function. The noise function applies noise with a normal distribution given a corresponding x coordinate. The noise function used for the simulator is a descending straight line from 1 to 0. The noise function shown in Figure 4.3 is shifted downwards not to overlap the sinusoid. Other noise functions can be used as well.

4.1.4 Model Uncertainty

Model uncertainty is the uncertainty in the predictions generated by the model due to the limitations and assumptions of the model itself.

To simulate model uncertainty, we need a way to make sure improving the model reduces model uncertainty. In addition, we can find the best fits instantly without needing to spend time fitting. A naive approach could be to add more layers to the network, but adding more layers does not guarantee model uncertainty reduces/increases. The approach we opted for was the use of polynomial models. Polynomial models are a type of model that uses regression to understand the relationship between independent variables (input) and dependent variables (output) by fitting the best polynomial line to observed data. The degree of the polynomial determines the curves in the lines, the higher the degree, the higher the number of curves.

The general formula for a polynomial of degree n is:

$$y = c + b_1x + b_2x^2 + b_3x^3 + \dots + b_nx^n$$

Where:

- y represents the dependent variable we want to predict or model
- x represents the independent variable that we are using to make predictions
- c, b_1 to b_n are the regression coefficients. In our case our, model parameters, and they shift the line up and down or change its slope.
- n is the degree of the polynomial

Polynomial models are simple and we can use them to emulate some of the uncertainty techniques explored such as an ensemble of different models, an ensemble of different data, and error models. In theory, each additional degree of a polynomial should increase the fit of the model to the data, thereby reducing model uncertainty.

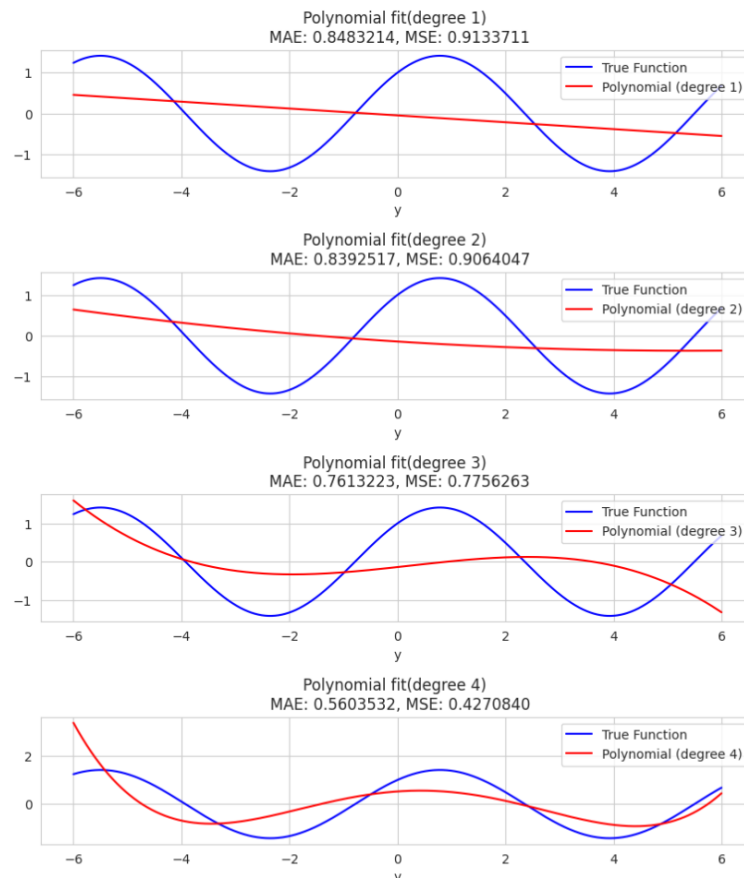


FIGURE 4.4: Example of Aleatoric Uncertainty in Simulator 1

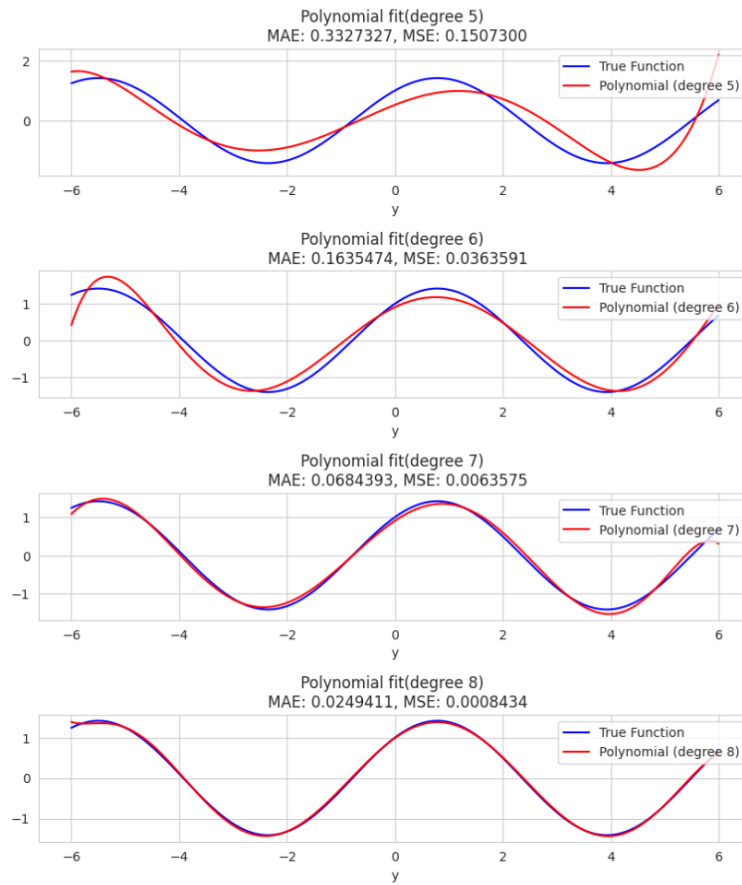


FIGURE 4.5: Example of Aleatoric Uncertainty in Simulator 2

Given these properties, we can make a simulated environment with controls over the different types of uncertainties.

4.2 Calculating the True Uncertainties

With being able to control the different types of uncertainties, we also need to be able to get the true uncertainty for every point in our true function. To do this, we take $error_i$ as a combination of Model error, Epistemic error, and Aleatoric error.

$$E(x) = ME(x) + EE(x) + AE(x)$$

Where x is a data point, $E(x)$ is the total error at x , $ME(x)$ is the model error at x , $EE(x)$ is the epistemic error at x , $AE(x)$ is the aleatoric error at x .

Definition of Uncertainty:

$$U(x) = E[x^2]$$

x in this case is the total error. The term $E[x^2]$ essentially represents the expected value of the square of the data point x . In terms of uncertainty, this expression helps quantify the spread or dispersion of possible values of x around the expected value.

We know that:

1. Model error(ME_i) is the difference in prediction between the perfect model $f(x)$ and the given model $g(x)$ when both have full data.

$$ME(x) = [f(x) - g(x)]^2$$

2. Epistemic error(EE_i) is the difference in predictions between a model $g(x)$ with full data and the same model with partial information $g(x|D)$.

$$EE(x) = [g(x) - g(x|D)]^2$$

3. Aleatoric error is the inherent uncertainty in the input data that even a perfect model $f(x)$ would make. This is the noise function defined by us.

$$AE(x_i) = E[N(f(x), h(x)) - f(x)]^2$$

Where:

- $N(f(x), h(x))$ is the noise function, $N(\mu, \sigma)$ represents the normal distribution, where $f(x)$ is the true function and $h(x)$ is the noise distribution.
- $f(x)$ is the perfect model with all data
- $g(x)$ is an imperfect model with all data
- $g(x|D)$ is an imperfect model with a subset of data

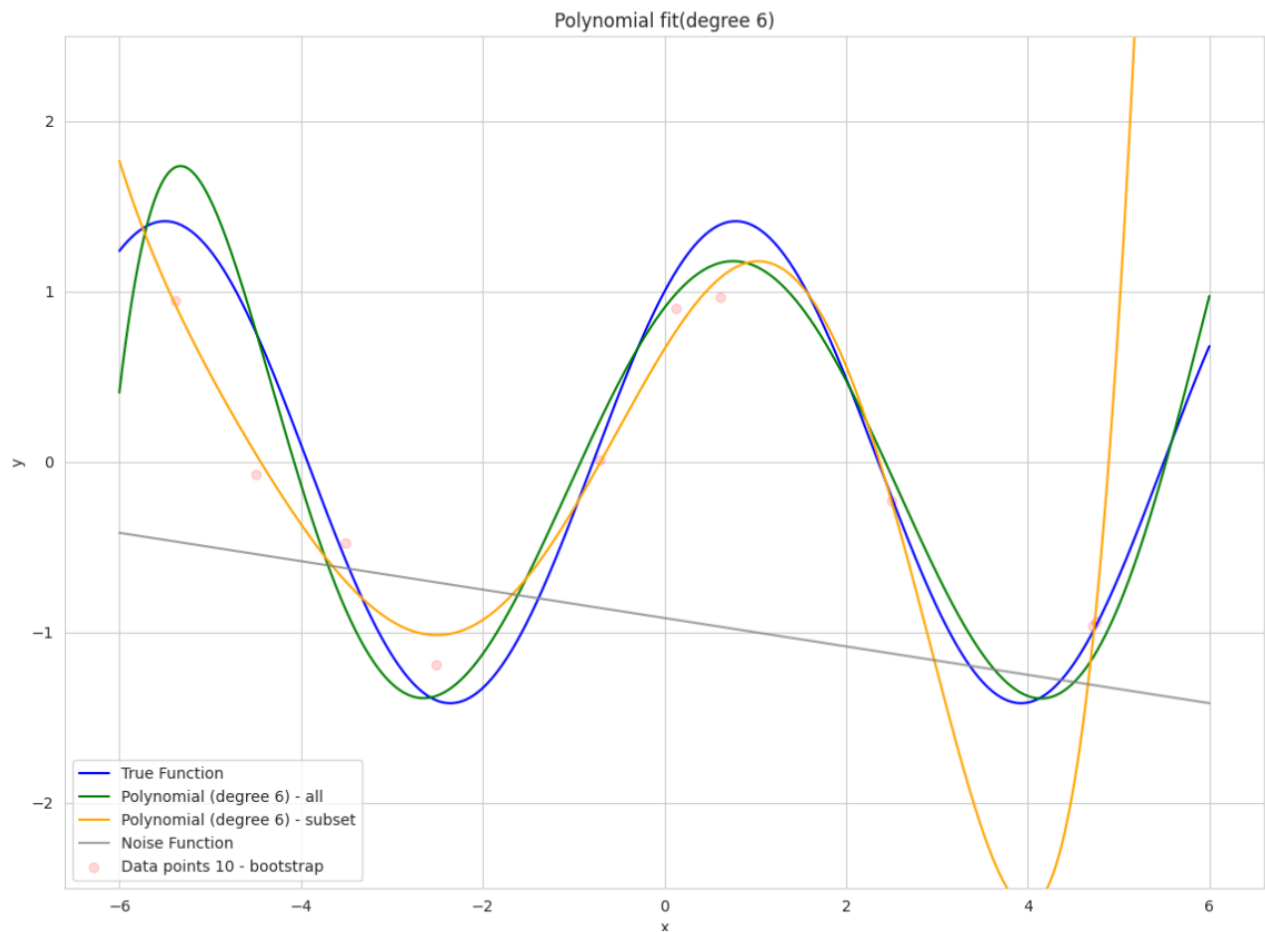


FIGURE 4.6: Parameters used in Calculating True Uncertainty

The figure above shows the different parameters visually.

- True function is $f(x)$
- The noise function refers to $N(f(x), h(x))$
- Polynomial(degree 6 - all) is the imperfect model with all data ($g(x)$)
- Polynomial(degree 6 - subset) is the imperfect model with a subset of data in this case 10 data points.

4.2.1 Experimental Setup

For the experimental setup, we use 3 scenarios and 3 uncertainty models

4.2.1.1 Scenarios

1. Scenario 1 - $n_points = 10$ $degree = 6$. We modeled this scenario to show epistemic uncertainty due the few number of data points. Our model shouldn't be great at predicting aleatoric and model uncertainties in this case.
2. Scenario 2 - $n_points = 1000$ $degree = 6$. This scenario has 1000 points which should reduce epistemic uncertainty while the other uncertainties(model and aleatoric) should be dominant.
3. Scenario 3 - $n_points = 1000$ $degree = 10$. This scenario uses a high degree, which should have a low model uncertainty, this should have mostly aleatoric uncertainty.

4.2.1.2 Uncertainty Models

1. Model 1 - Ensemble of Different Models - Here each model in the ensemble has varying degrees of complexity. Uncertainty in this model is also the standard deviation of the ensemble predictions.
2. Model 2 - Ensemble of Different Data - Here each model in the ensemble is trained on a different subset of the training data(bootstrap technique). Uncertainty in this model is the standard deviation of the ensemble predictions.
3. Model 3 - Direct Error Estimation - Here we attempt to predict the error of a polynomial model using another polynomial model. The uncertainty in this model is the prediction of the error model. This uncertainty model is representative of a type of error model already explored.

4.2.1.3 Expectation from different models

1. Ensemble of Different Data - This should have some correlation with epistemic uncertainty since each model in the ensemble is trained on a different subset of data.
2. Ensemble of Different Models - This model should be good at model uncertainty as each model in the ensemble has a different degree while seeing the same data.
3. Direct Error Estimation - This should be good at predicting aleatoric uncertainty, as it learns from the prediction of the polynomial. It should be able to predict on average what error we should have at point x.

4.2.2 Results

4.2.2.1 Model Scenario Result for a set seed

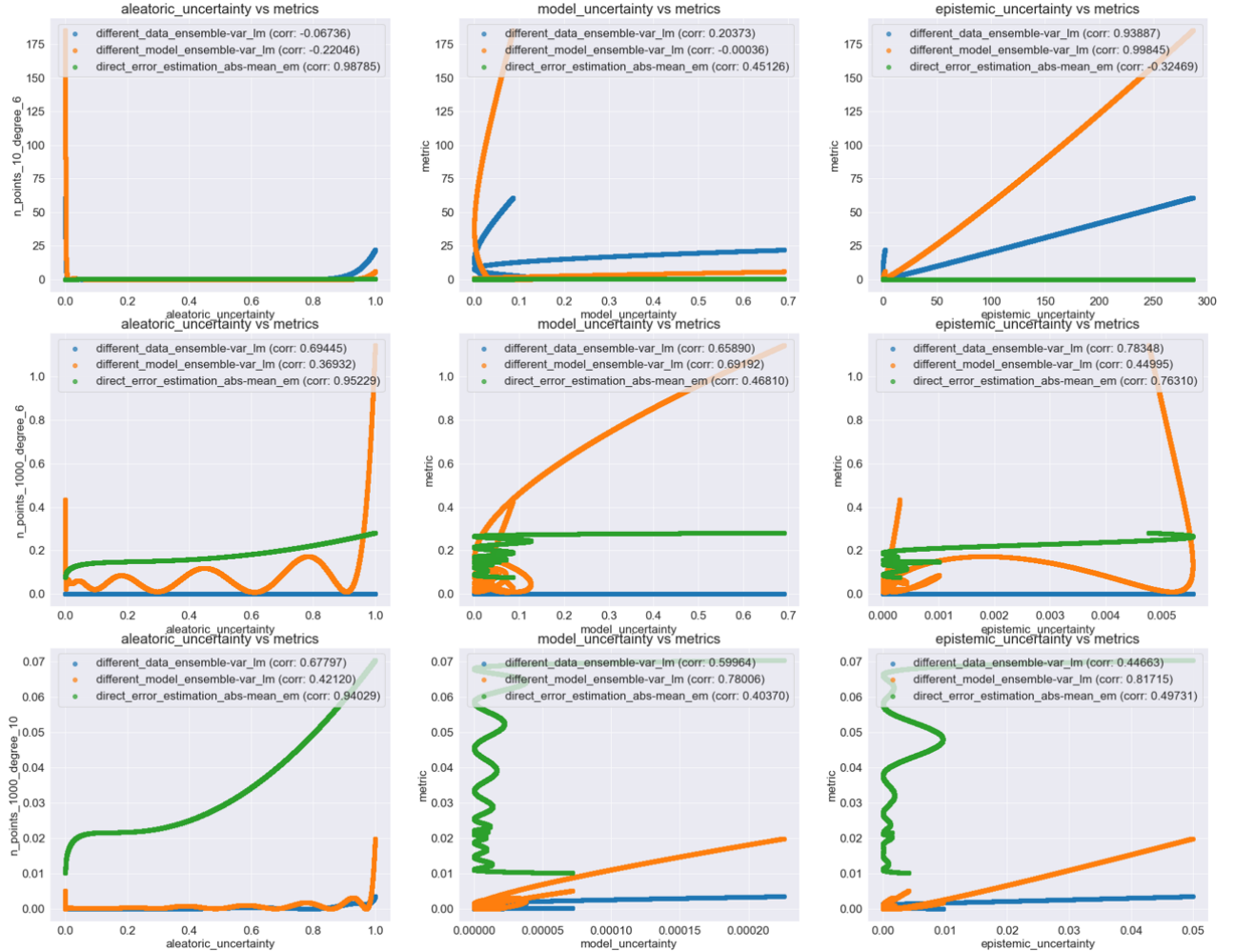


FIGURE 4.7: Model Scenario Result for a set seed

The plot shows each model metric plotted against all the true uncertainties. This is a single run using a selected seed of 3, tested on a million points. Each row represents a specific scenario and column a specific model.

- Scenario 1

- Aleatoric uncertainty: In this scenario, none of the models should be able to predict aleatoric uncertainty due to the small amount of data.
- Model Uncertainty: No model correlated with model uncertainty as there isn't enough data to predict this type of uncertainty.
- Epistemic Uncertainty: Model 1 and model 2 have a correlation with epistemic uncertainty, which is what was expected but model 1 had a slightly better correlation in this run.

- Scenario 2
 - Aleatoric uncertainty: In this case, we should be able to detect aleatoric uncertainty. Model 3 is the closest
 - Model Uncertainty: Model 1 has some correlation but struggles to predict smaller model uncertainties.
 - Epistemic uncertainty: Since epistemic uncertainty is low, it wasn't detected by any model.
- Scenario 3
 - Aleatoric Uncertainty: The dominant correlation is Model 3, Direct Error Estimation model, which was expected.
 - Model Uncertainty: Although model uncertainty is very low, Model 1 is still the dominant model for estimating model uncertainty.
 - Epistemic Uncertainty: In this case, model 1 is the dominant model with a slight correlation.

4.2.2.2 Model Scenario Result for multiple runs

10 runs were made with different seeds using and tested with 1000 points. The goal of this is to see a pattern of correlation between different scenarios and models. The results are 3 subplots with each comparing the model and scenarios on the true aleatoric, epistemic and model uncertainty. Each row in a subplot represents the scenario and column the uncertainty model, each plot in the subplot is the combined individual runs.

Aleatoric Uncertainty

Model 3 is expected to be dominant in predicting this type of uncertainty.

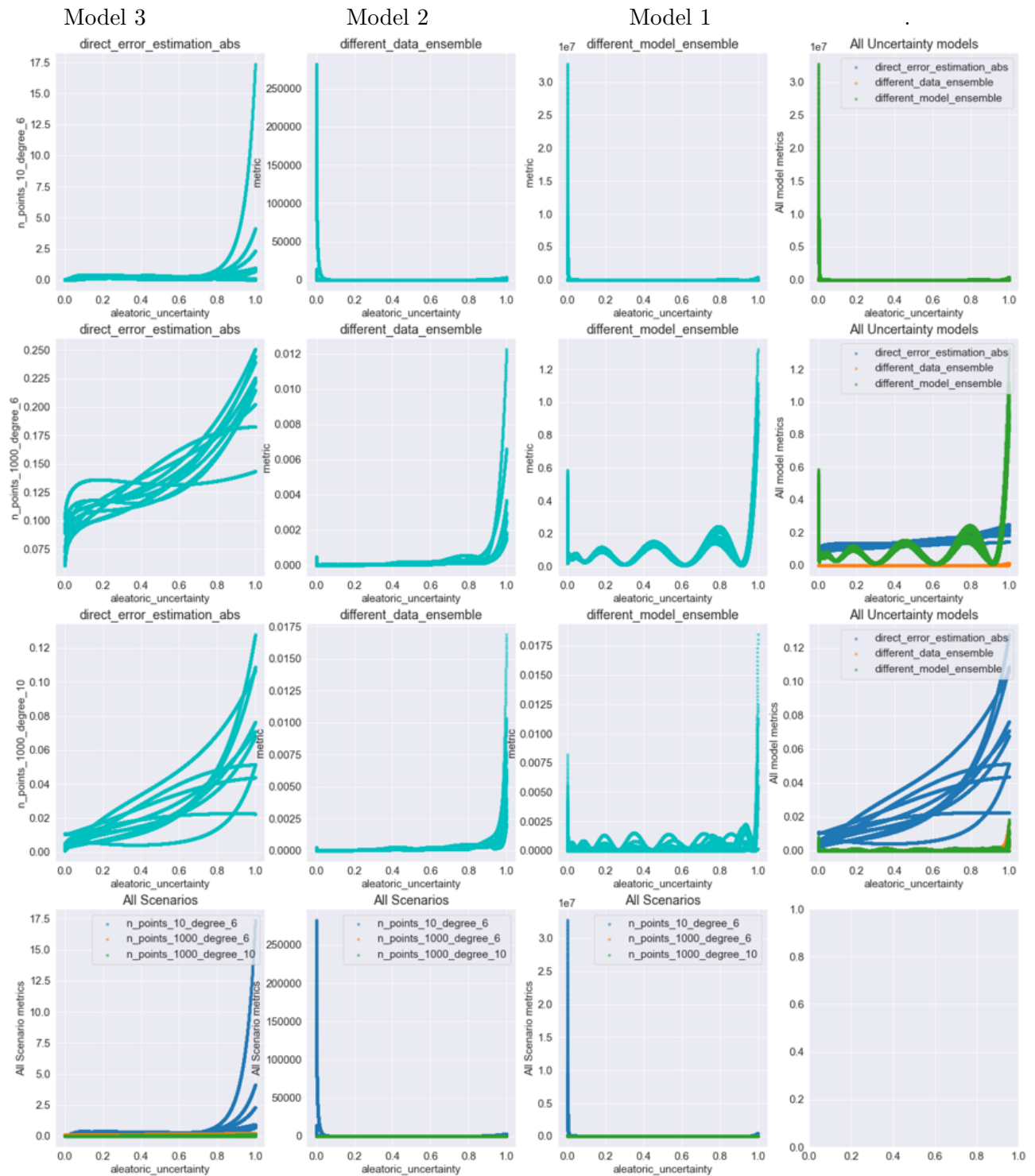


FIGURE 4.8: Model Scenario Comparison with True Aleatoric Uncertainty

- Scenario 1
 - Model 3 - Direct Error Estimation: In this scenario, the error model doesn't show any pattern with aleatoric uncertainty which is expected due to the small amount of data.
 - Model 2 - Ensemble of Different Data: No correlation with aleatoric uncertainty as this model should not predict this type of uncertainty with the given amount of data.

- Model 1 - Ensemble of Different Models: No correlation with aleatoric uncertainty as this model should not predict this type of uncertainty.
- Scenario 2
 - Model 3 - Direct Error Estimation: Given 1000 points, the model was able to show a correlation with aleatoric uncertainty.
 - Model 2 - Ensemble of Different Data: No correlation with aleatoric uncertainty as this model should not predict this type of uncertainty with the given amount of data.
 - Model 1 - Ensemble of Different Models: No correlation with aleatoric uncertainty as this model should not predict this type of uncertainty.
- Scenario 3
 - Model 3 - Direct Error Estimation: With this scenario, the model was able to show a correlation with aleatoric uncertainty although not as strong as scenario 2 which could indicate that model complexity plays a role in predicting aleatoric uncertainty.
 - Model 2 - Ensemble of Different Data: No correlation with aleatoric uncertainty as this model should not predict this type of uncertainty with the given amount of data.
 - Model 1 - Ensemble of Different Models: No correlation with aleatoric uncertainty as this model should not predict this type of uncertainty.

Epistemic Uncertainty

Model 2 is expected to be dominant in predicting this type of uncertainty.

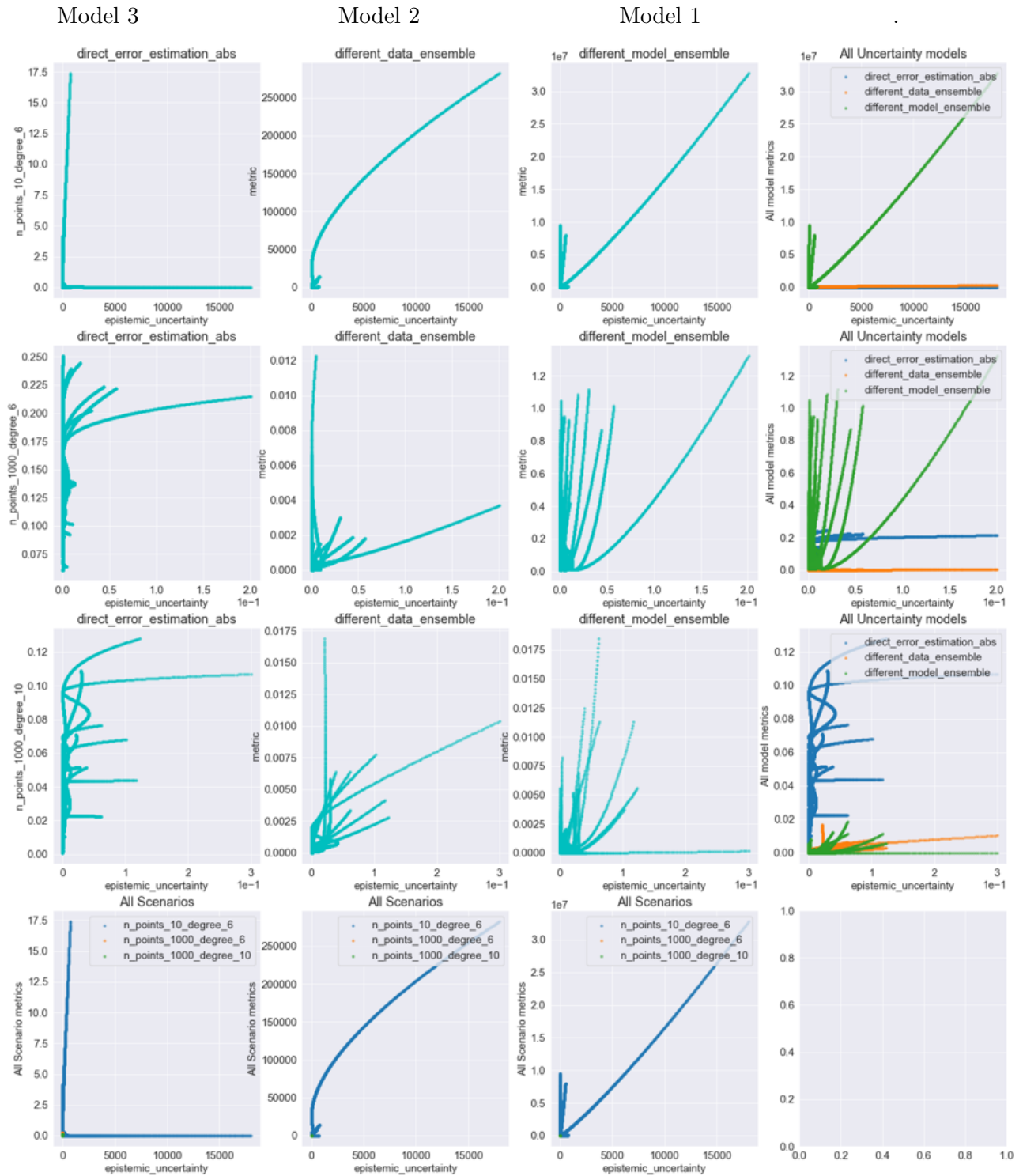


FIGURE 4.9: Model Scenario Comparison with True Epistemic Uncertainty

- Scenario 1
 - Model 3 - Direct Error Estimation: No correlation with epistemic uncertainty as this model should not predict this type of uncertainty.
 - Model 2 - Ensemble of Different Data: There is also a correlation with epistemic uncertainty with the range for the prediction reaching 250,000, a better range than Model 1 but still makes a large prediction due to the small number of points.

- Model 1 - Ensemble of Different Model: There is a correlation with epistemic uncertainty although the range for the prediction is too large, it goes up to $3 * 10^7$ with the true epistemic uncertainty at 17,500. The reason for this can be due to the lower degree polynomial models in the ensemble which causes the prediction to be large as they are poor at predicting the true function with just 10 points.
- Scenario 2
 - Model 3 - Direct Error Estimation: No correlation with true epistemic uncertainty as this model should not predict this type of uncertainty.
 - Model 2 - Ensemble of Different Data: Slight correlation after a certain threshold, although doesn't seem to be enough evidence to suggest a correlation even though it's expected, could be a result of a very low true epistemic error. Further exploration needs to be done.
 - Model 1 - Ensemble of Different Model: Slight correlation with true epistemic uncertainty after a certain threshold(0.75) with a higher estimate than Model 2.
- Scenario 3 No correlation for all models due to low epistemic and low model uncertainty

Model Uncertainty

Model 1 is expected to be dominant in predicting this type of uncertainty.

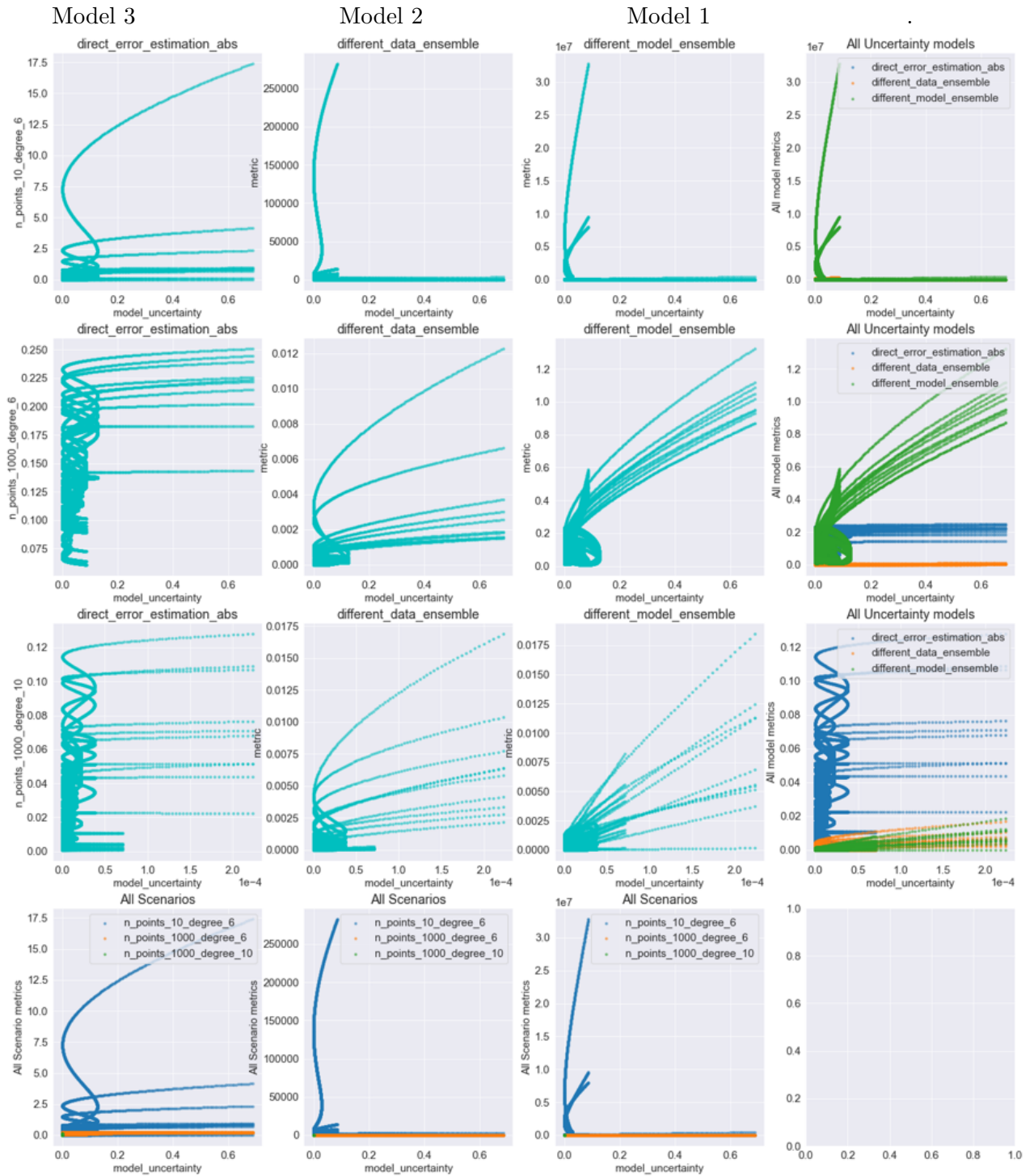


FIGURE 4.10: Model Scenario Comparison with True Model Uncertainty

- Scenario 1 There is no general trend with true model uncertainty with all 3 models, due to model uncertainty not being detected given the small data points.
- Scenario 2
 - Model 3 - Direct Error Estimation: No correlation with true model uncertainty as this model should not predict this type of uncertainty.

- Model 2 - Ensemble of Different Data: Slight correlation but too high of a variance between the runs to conclude.
- Model 1 - Ensemble of Different Model: There seems to be a correlation after a certain threshold of (0.15). Different runs show different angles of correlation, some more than others but there is a general trend.
- Scenario 3
 - Model 3 - Direct Error Estimation: No correlation with true model uncertainty as this model should not predict this type of uncertainty.
 - Model 2 - Ensemble of Different Data: Similar results to Scenario 2, no correlation.
 - Model 1 - Ensemble of Different Model: In Scenario 2, the model couldn't estimate model uncertainty of smaller values. In this scenario with degree 10, this true model uncertainty becomes much lower which results in no correlation. More work needs to be done to address the poor estimates of lower model uncertainty values.

In conclusion, there exists some correlation between Model 1(Ensemble of Different Models) and true model uncertainty, Model 2(ensemble of different data) and true epistemic uncertainty, Model 3(Direct Error Estimation) and true aleatoric error. Although the results have not shown this with high accuracy, there are problems such as the range of the estimated uncertainty not matching the true uncertainty, no one-to-one correlation only slight correlations, and some models only showing correlation after a certain threshold. Further work needs to be done to address these issues, in addition, different types of scenarios, uncertainty models, and true functions could be explored.

Chapter 5

Applications to Real Projects

5.1 Image Classification

We conducted an active learning simulation using different techniques on multi-label and binary classification data. The goal is to determine what active learning technique works best for this particular dataset.

5.1.1 Dataset

The data used is private data gathered using a drilling tool that captures the images of a well. The tool moves up and down and can capture images of different resolutions. These images can be used to check the integrity and potential faults in the well. The annotation was done as a segmentation task, where we have the segmentation of the fault we would like to detect. In our case, images can be one of the following classes:

1. Background (BG)
2. Breakout (BO)
3. Induced Fracture (IF)

Since the images are continuously taken as we drill, we split them in patches of $(180, 180)$. These patches undergo a few preprocessing steps such as the removal of NaN values, normalization, and assigning class labels.

For **Multi-label Classification**, a patch can either be a background, breakout, induced fracture or a breakout and induced fracture. For **Binary Classification**, a patch can either be a background or a breakout. Breakout in this case refers to both mature breakout and induced Fracture.

The final dataset is composed of 21 wells, 19 wells will be used for the active learning simulation and 2 will be used for testing. The 19 wells are further split to train/val with sizes 80%, and 20%. The reason for testing on 2 separate wells is to ensure generalization and prevent data leakage.

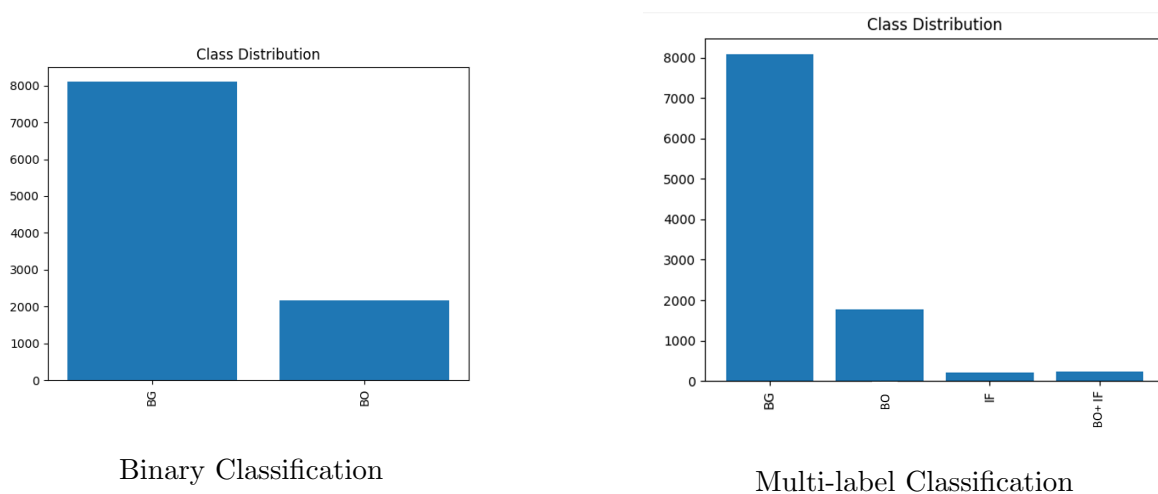


FIGURE 5.1: Class Distribution Of Data in Active Learning Simulation

Model and Hyperparameters

The model used for all simulations was a pretrained Resnet 18 model, with a learning rate of $1e - 4$, SGD Optimizer, batch_size of 32 and binary cross entropy as the loss function. Early stopping was used based on the model performance on the validation set.

Evaluation metric

For evaluating the active learning performance we use the F1 score. F1 is chosen due to the class imbalance present in the data.

5.1.2 Multi-label Classification Results



FIGURE 5.2: Active Learning Multi-label Classification Results

The figure above shows the active learning simulation, done in 10 active learning steps, with 100 initial points and a budget of 100 for each step. max_entropy superior to random sampling, which indicates doing active learning using max_entropy is a good approach to use to label our data in this multilabel setting.

5.1.3 Binary Classification Results

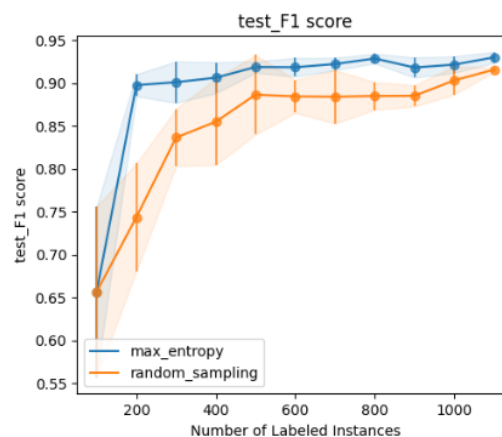


FIGURE 5.3: Active Learning Binary Classification Results

In a binary setting, entropy is still better. The simulation was done in 10 active learning steps, with 100 initial points and a budget of 100 for each step. The simulation was done 5 times with different seeds.

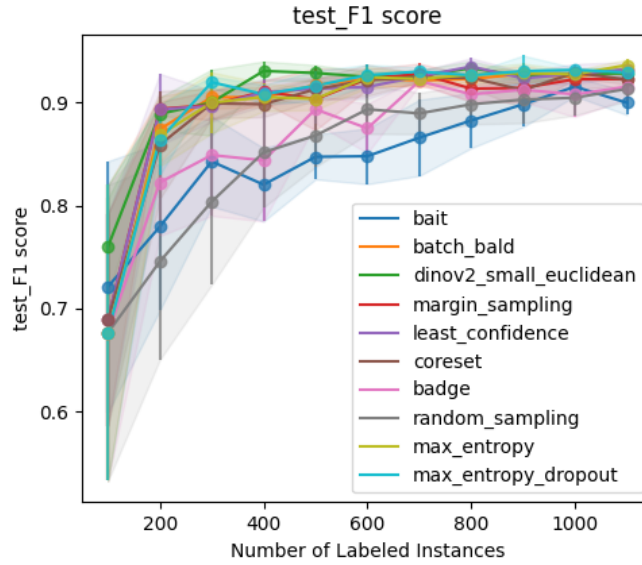


FIGURE 5.4: Active Learning Binary Classification Results Using Multiple Query Strategies

Here we compare some of the state-of-the-art active learning techniques such as Batch bald Kirsch et al. (2019), BAIT Ash et al. (2021), BADGE Ash et al. (2020), CoreSet Sener and Savarese (2018), a custom ranking technique using dinoV2 and other simpler techniques to see if entropy can be outperformed. The simulation was done in 10 active learning steps, with 100 initial points and a budget of 100 for each step. From the figure above, most techniques perform relatively the same on average outperforming random sampling, with the exception of BADGE and BAIT which performed worse. The results are representative of the dataset used, the best strategy for the dataset cannot be determined as they mostly perform relatively the same. Since entropy was not outperformed, entropy was chosen as the query strategy for active learning.

5.1.4 Effect of Data Imbalance on Active Learning

The purpose of this experiment is to know how entropy performs with different data imbalance ratios. The experiment was done in a binary classification setting in 10 active learning steps, with 100 initial points and a budget of 100 for each step. This process was done 5 times to obtain the average performance. The imbalance ratios tested are 1, 0.75, 0.5, 0.25 and 0.1. A ratio of 1 means we have a balanced dataset, 0.5 means the minority class is half as much as the majority class. Since the base dataset is not balanced, we had to programmatically create data imbalance. The distribution of classes for each ratio is given as:

<i>Ratio</i>	num_of_BG	num_of_B0	n_train	n_test
0.1	8102	810	7129	779
0.25	8102	2025	8101	779
0.5	4356	2178	5227	779
0.75	2904	2178	4065	779
1	2178	2178	3484	779

TABLE 5.1: Class Distribution for Data Imbalance Experiment

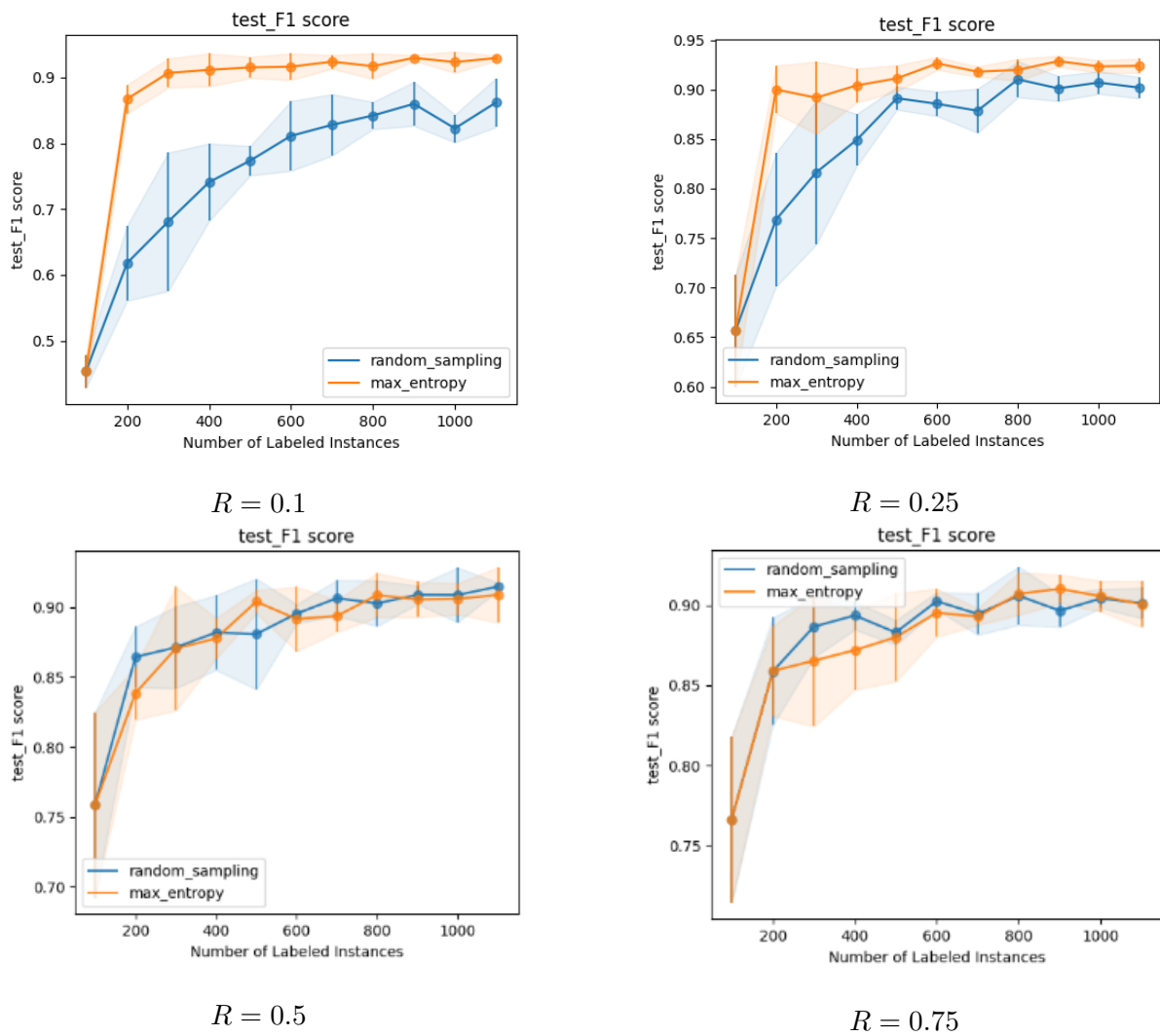
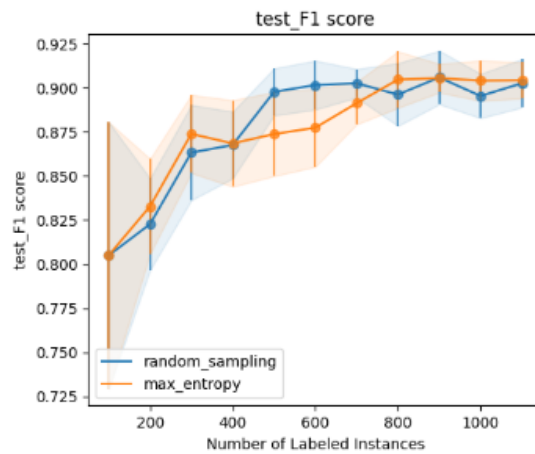


FIGURE 5.5: Effect of Data Imbalance on Active Learning



$$R = 1$$

FIGURE 5.6: Effect of Data Imbalance on Active Learning

Based on the results above, it can be noticed that as we increase the data imbalance, entropy performs better. With balanced data, entropy and random sampling seem to perform the same. To better understand the reason for this we can take a look at the class distribution of acquisition images in each active learning step.

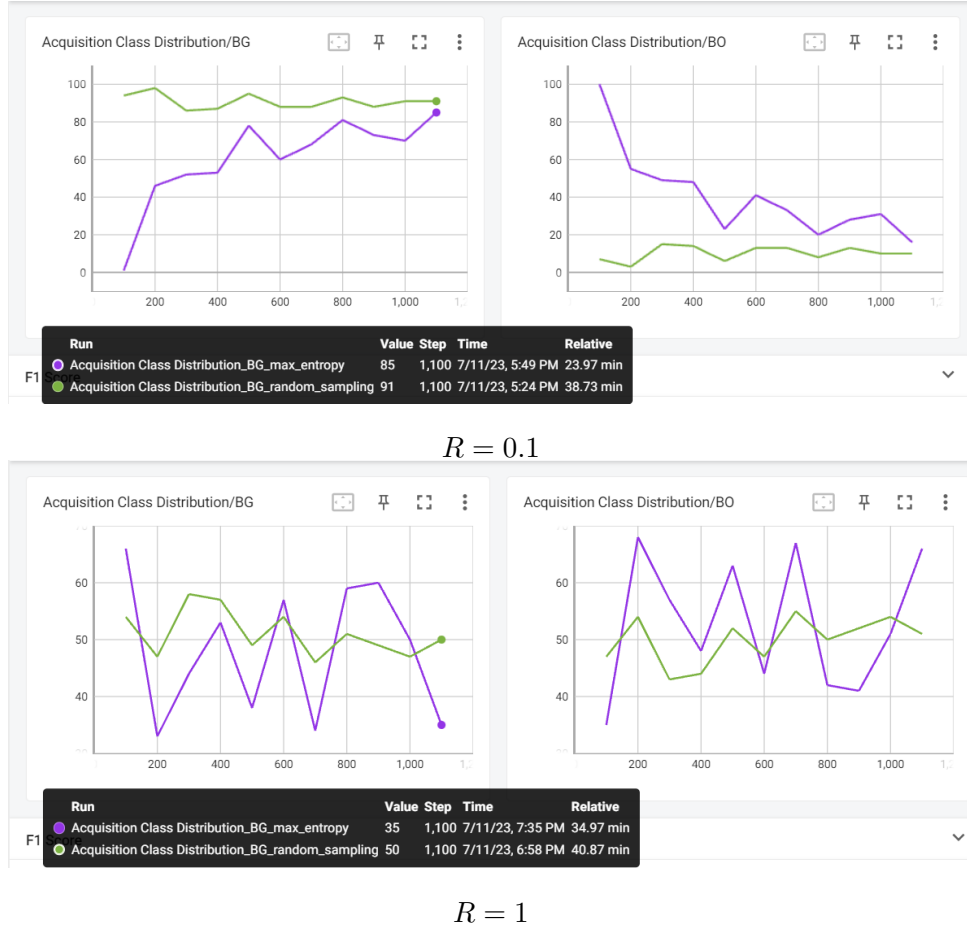


FIGURE 5.7: No of acquisition images in each active learning step

We can see that with an imbalanced ratio of 0.1 entropy tends to pick a lot of the imbalanced class(Breakout) and very few backgrounds while random sampling picks relatively the same amount of images each step based on the dataset distribution, this results in picking more background and not breakout which results in a poor F1 score. With a balanced dataset($R = 1$) they perform the same, both entropy and random sampling pick relatively the same amount of images, with entropy having a bit of noise. This result is representative of this dataset and might not apply to others.

5.1.5 Image Ranking Techniques for Active Learning

In active learning, the model needs to be retrained after each step. Depending on the active learning technique this is needed in other to get the uncertainty or latest embedding of the model on the unlabelled data. The problem with this is that retraining the model for each active learning step is time-consuming, in addition, a lot of time is spent querying the unlabelled data

to get probabilities or embeddings. The proposed solution is to use a ranking technique where the unlabelled images are ranked based on how likely they are to lead to higher performance. This eliminates the need for retraining the model each time and it only needs to be done once. Two ranking techniques, autoencoders, and dinoV2 were explored.

AutoEncoders

The idea behind this is to train an autoencoder on all the images, both labeled and unlabelled. Once trained, we run the autoencoder on all the images and calculate the reconstruction loss. Based on the reconstruction loss, we create a ranking for each image, with the expectation that images with higher reconstruction errors are more informative and should create robustness while quickly improving the model's performance with only a few images.

The autoencoder architecture is adapted from [Lippe \(2023\)](#), it consists of an encoder and a decoder. The encoder consists of several convolutional layers with batch normalization and the GELU activation function. The number of channels increases gradually across the 4 convolutional layers. The encoder then flattens the features into a single feature vector and applies a linear layer to create the latent representation. The decoder has a similar structure to the encoder but in reverse order.

Results

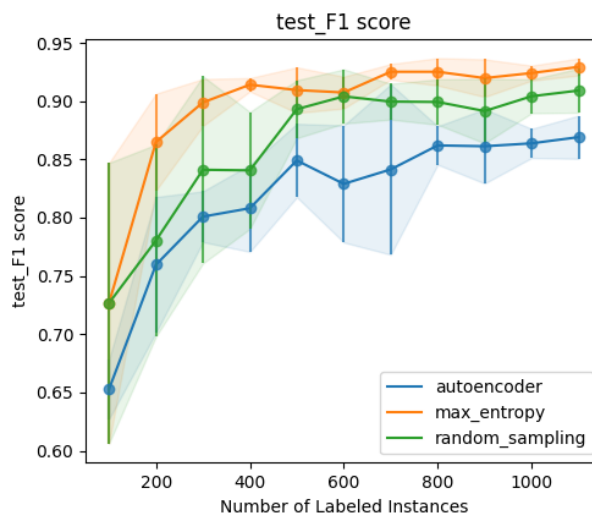


FIGURE 5.8: Autoencoder Reconstruction Loss for Active Learning

Based on the results, the autoencoder technique performed poorly. Examining the rankings, the top images with high reconstruction loss are mostly backgrounds with very rough textures. Smoother textures had low reconstruction loss so they weren't ranked highly. A potential

improvement could be to add some sort of diversity along with the reconstruction loss or explore a pre-trained autoencoder not trained specifically on our images.

DinoV2

DinoV2 [Oquab et al. \(2023\)](#) is the current state-of-the-art feature extractor from meta. It uses a distilled ViT [Dosovitskiy et al. \(2021\)](#) model trained on diverse images using self-supervision. was trained. The idea behind this technique is to run the feature extractor on all images, then using the embeddings we use a ranking technique to rank all the images based on diversity. The 3 ranking techniques explored are:

1. Kmeans++ from BADGE: Initializes the centroid with kmeans++ and ranks diverse instances based on the largest minimum distances between them. The implementation is adapted from the BADGE [Ash et al. \(2020\)](#) active learning technique.
2. Pairwise Euclidean Distance: The Euclidean distance is computed for all image embeddings to all other image embeddings. The distances for each image are summed up and the images are ranked based on the summed up distance.
3. Cosine similarity: Same as Pairwise Euclidean Distance except cosine similarity is used instead.

Results

The simulation was done in 10 active learning steps, with 100 initial points and a budget of 100 for each step. This process was done 5 times to obtain the average performance.

The small variant of dinoV2 is used here with an embedding dimension of 384.

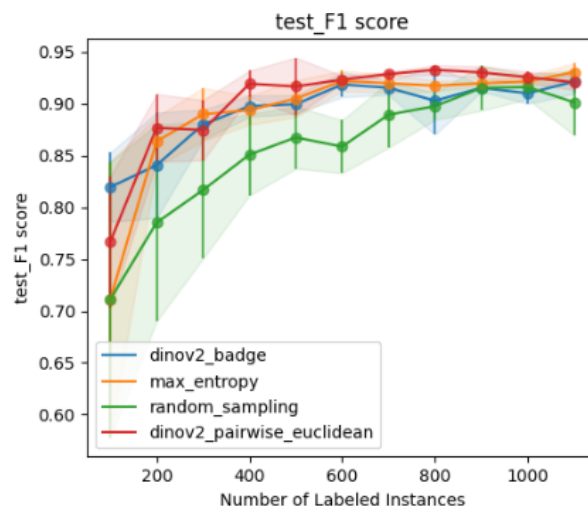


FIGURE 5.9: Dino_v2 Small Active Learning Performance

The results show that the dinoV2 ranking technique is on par with entropy sampling with the Euclidean distance variant being the better technique. The initial points for the ranking technique are not taken at random since the ranking is not dependent on the model, due to this we have a much higher F1 score during the initialization, with BADGE(Kmeans) being the best initialization.

Comparison of DinoV2 Variants The available DinoV2 Variants we explored to know if the size and embedding dimension play a role in the active learning performance.

	DINOv2-S	DINOv2-B	DINOv2-L	DINOv2-G
Embed Dim	384	768	1024	1536
# of Params	21 M	86 M	300 M	1,100 M

TABLE 5.2: DinoV2 Variants Details

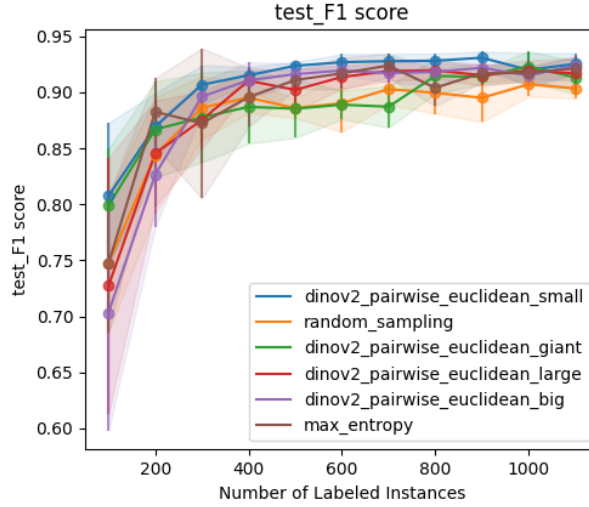


FIGURE 5.10: Comparison of size variants of Dino_v2

As the size of the feature extractor increases the performance becomes worse, this could be due to the high dimensionality of the features. It could also be due to the increasing difference between the feature extractor(dino_v2) and the model(Resnet).

To conclude, entropy remains competitive with other state-of-art query strategies when active learning was done on the data 5.1.3. Entropy performs better the more imbalanced the unlabeled pool is 5.1.4. Ranking techniques were explored to eliminate the need for retraining, using an autoencoder reconstruction loss performed worse than random, although the dinoV2's variant 5.1.5 gave promising results performing as well as entropy on this dataset.

5.2 Black Box Simulator

The problem definition for this project is that we have a black-box physics simulator and we would like to have a model to emulate this simulator. The simulator takes as input, the parameters/settings of a drill and responds with a output signal. So given the input parameters of a drill we would like to predict the emitted output signal. The problem with this is that the simulator takes a lot of time to run, thus we cannot generate sufficient data quickly to train a model. Active learning can help in this case to efficiently simulate only informative points necessary for training.

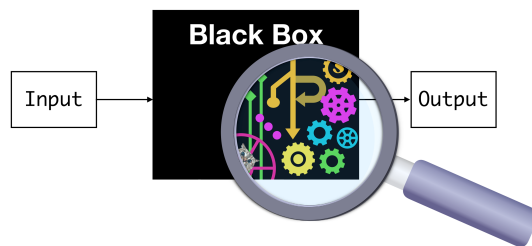


FIGURE 5.11: Black Box Simulator Example

Before testing on the real simulator we tested on a simpler 1D simulator to see if active learning works. The 1D simulator is a simplified version of the real simulator, its output signal is a time series signal of length 600.

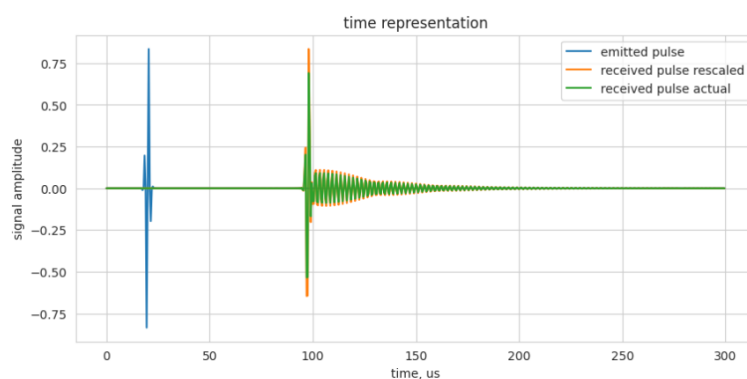


FIGURE 5.12: Example of Output Signal

We are interested in predicting the "receive pulse actual" given the simulator's input parameters. The input parameters are integers, they have a predefined domain which reduces the search space.

5.2.1 Random vs Uniform Initialization for Active Learning

Initialization in this case refers to the initial points we start with for active learning. In active learning, initial points are usually chosen at random. Given we have the domain of our input parameters we can do a uniform initialization based on the domain. We would like to know what type of initialization is better suited to perform active learning.

The black box used for this test is a damped sin wave function, where the input parameters are the decay rate and frequency. We start with 5 initial points and a budget of 1 for 50 steps. Our acquisition function is a variance reduction sampling technique, the kernel used for the Gaussian process is an RBF Kernel. This was run 10 times with different seeds.

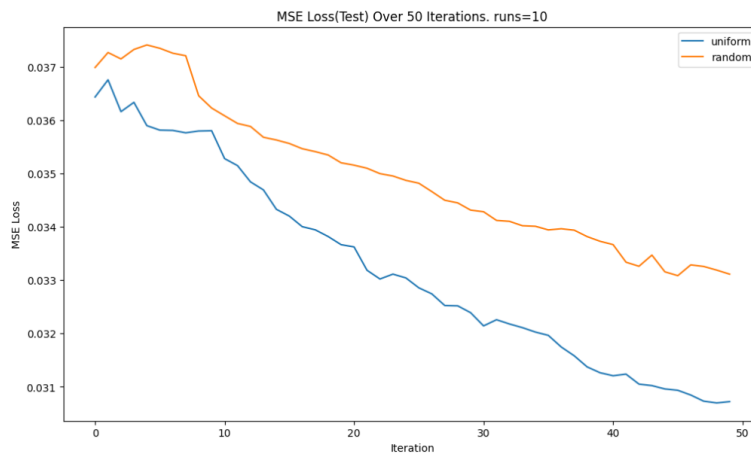


FIGURE 5.13: Random vs Uniform Initialization

Based on the result above, both tend to go down but uniform initialization goes down faster than random, thereby getting a lower MSE loss, so uniform should be preferred for initialization.

5.2.2 Approach

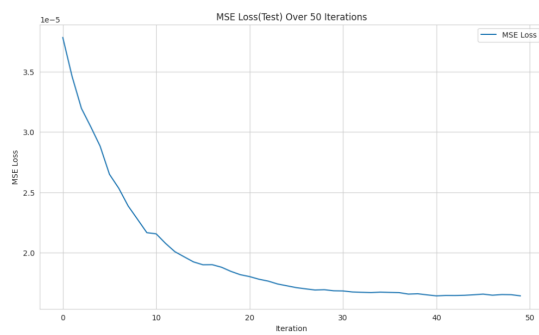
To model the signal we opted for a Gaussian process model. A Gaussian process model is a probabilistic model that provides a distribution over functions rather than on a single prediction. Using this distribution we can obtain the uncertainty of the prediction, in this case the standard deviation of the prediction, Gaussian process is also good for modeling complex functions.

The query strategy used is a *variance reduction sampling technique*. This technique works by selecting the points with the highest predictive variance according to the current GP model, as these points are considered to be the most uncertain and therefore most informative for the model. In other words, the model is expected to benefit the most from additional information at these points.

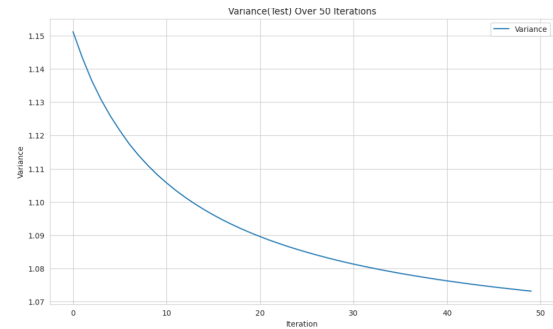
5.2.3 Results

5.2.3.1 1D Simulator Result(2 Features)

The kernel used for the Gaussian Process is a combination of a Matern32 kernel, periodic exponential, and bias kernel. The initial samples are 10, chosen uniformly, active learning is done for 50 steps with a query budget of 1.



Test MSE over Active Learning Steps



Mean Variance on Test Set Over Active Learning Steps

FIGURE 5.14: 1D Simulator Result(2 Features)

The figure above shows the MSE reduction on the test set as we do active learning. After training, we achieved an MSE of 1.67400×10^{-5} on the train set and 1.64309×10^{-5} on the test set.

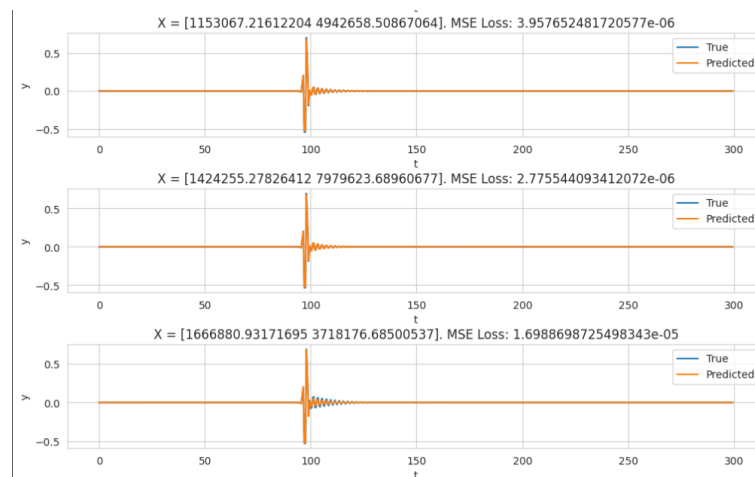


FIGURE 5.15: Example Prediction of Output Signal for 2 Input Features

The figure above shows some of the predictions made by the Gaussian process and the true values. With 60 points in total and 2 input features, we were able to model our black box function accurately.

5.2.3.2 1D Simulator Result(3 Features)

Kernel		MSE
Matern32	Train	0.00195
	Test	0.00219
MLP + PeriodicMatern52	Train	0.00175
	Test	0.00232

TABLE 5.3: 1D Simulator Result(3 Features)

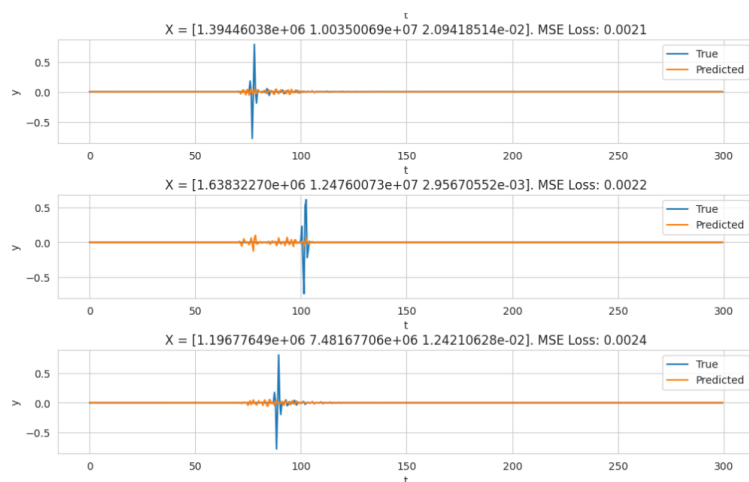


FIGURE 5.16: Example Prediction of Output Signal for 3 Input Features

With 3 features, we were not able to predict the signal accurately using the same Gaussian process model and others. The table above presents the MSE results for different kernel combinations on both the training and testing sets. As can be observed, the performance of the models deteriorated when compared to the 2-feature scenario.

There are several factors that could have contributed to the failure of accurately predicting the signal for 3 features.

1. **High Dimensionality:** One of the main challenges in working with Gaussian processes is handling high-dimensional input spaces. As the dimensionality of the input space increases, Gaussian process models become more complex, and their performance may degrade. In our case, by adding an extra feature, we increased the complexity of the problem, and the performance of the model deteriorated. This also is known as the curse of dimensionality.

2. Independence of Variables: Another possible reason for the inaccurate predictions is that in the Gaussian process model, each value in the output vector is considered independent of each other. In our case, this means that each timestep in the signal is presumed to be independent of any other timestep. However, the signal is a time series, and the previous timesteps likely influence the future timesteps. This sequential dependency between the timesteps may not be properly captured by the Gaussian process model leading to inaccurate predictions.

To address these challenges and get an improvement over the Gaussian process model for the 3-feature case, we can explore other types of models such as conditional GANs where given an input condition we produce an output signal. The conditions in this case will be our input parameters.

In summary, uniform initialization is best for active learning when modeling a black box [5.2.1](#). Active learning using Gaussian processes produced good results for 2 features but struggles with 3 due to high dimensionality and independence of variables [5.2.3](#). Other types of models can be explored to fix the limitation of Gaussian processes on this task.

Chapter 6

Conclusion

We explored different techniques for measuring uncertainty from dropout, ensembles to autoencoders in 3.0. For regression tasks dropout is the easiest way to get an uncertainty estimate, ensembles and bayesian neural networks give a better uncertainty correlation 3.1, 3.2, 3.4. For classification taking the entropy is the easy way to go 5.1.

To better understand what type of uncertainty the different techniques produce, we created an uncertainty estimation simulator 4.0 where we can control the 3 types of uncertainty: epistemic, aleatoric and model uncertainty. Based on the results, an assumption can be made that ensemble of different subsets of data corresponds the most to epistemic uncertainty, an ensemble of different models corresponds with model uncertainty, and a model trying to predict its own error corresponds to aleatoric uncertainty although this needs to be explored further since there is not a 1:1 correlation.

Active learning was applied to 2 real-world projects 5.0, the classification project 5.1 improved its annotation efficiency by adopting entropy sampling, entropy was chosen due to its simplicity and performance. The performance of entropy sampling improves the more data imbalance there is 5.1.4. To solve the problem of retraining a model for each active learning step, an exploration was done on image ranking techniques for active learning, the dinoV2 variant performed better than random sampling and showed competitive results with entropy as shown in 5.1.5. For modeling a black box using active learning 5.2, active learning was successful on the 1D simulator using 2 features but struggled with 3 due to the limitation of Gaussian Processes.

6.1 Future Work

For the uncertainty estimation simulator 4.0, there was not a one-to-one correlation with the explored scenarios and uncertainty models, further investigation is needed to understand why this is the case. In addition, further development of the simulator is needed so that if given a data point, the type of uncertainty associated with the data point can be predicted accurately.

Since we achieved good results from ranking-based techniques for active learning, further exploration can be done such as ranking based on the embedding of a pretrained autoencoder, testing other feature extractors and testing with other datasets.

Bibliography

- Abdar, M., Pourpanah, F., Hussain, S., Rezazadegan, D., Liu, L., Ghavamzadeh, M., Fieguth, P., Cao, X., Khosravi, A., Acharya, U. R., Makarenkov, V., and Nahavandi, S. (2021). A review of uncertainty quantification in deep learning: Techniques, applications and challenges. *Information Fusion*, 76:243–297.
- Ash, J. T., Goel, S., Krishnamurthy, A., and Kakade, S. (2021). Gone fishing: Neural active learning with fisher embeddings.
- Ash, J. T., Zhang, C., Krishnamurthy, A., Langford, J., and Agarwal, A. (2020). Deep batch active learning by diverse, uncertain gradient lower bounds.
- Blundell, C., Cornebise, J., Kavukcuoglu, K., and Wierstra, D. (2015). Weight uncertainty in neural networks.
- Cruz, S. D. D., Taetz, B., Stifter, T., and Stricker, D. (2022). Autoencoder attractors for uncertainty estimation.
- Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. (2021). An image is worth 16x16 words: Transformers for image recognition at scale.
- Gal, Y. and Ghahramani, Z. (2016). Dropout as a bayesian approximation: Representing model uncertainty in deep learning.
- Ghosh, S., Liao, Q. V., Ramamurthy, K. N., Navratil, J., Sattigeri, P., Varshney, K. R., and Zhang, Y. (2021). Uncertainty quantification 360: A holistic toolkit for quantifying and communicating the uncertainty of ai.
- Hüllermeier, E. and Waegeman, W. (2021). Aleatoric and epistemic uncertainty in machine learning: an introduction to concepts and methods. *Machine Learning*, 110(3):457–506.

- Kang, M. and Kang, S. (2023). Surrogate approach to uncertainty quantification of neural networks for regression. *Applied Soft Computing*, 139:110234.
- Kendall, A. and Cipolla, R. (2016). Modelling uncertainty in deep learning for camera relocalization.
- Kendall, A. and Gal, Y. (2017). What uncertainties do we need in bayesian deep learning for computer vision? *CoRR*, abs/1703.04977.
- Kirsch, A. and Gal, Y. (2022). Unifying approaches in active learning and active sampling via fisher information and information-theoretic quantities.
- Kirsch, A., van Amersfoort, J., and Gal, Y. (2019). Batchbald: Efficient and diverse batch acquisition for deep bayesian active learning.
- Krishnan, R., Esposito, P., and Subedar, M. (2022). Bayesian-torch: Bayesian neural network layers for uncertainty estimation. <https://github.com/IntelLabs/bayesian-torch>.
- Lippe, P. (2023). Tutorial 8: Deep autoencoders; pytorch lightning 2.0.5 documentation - lightning.ai. https://lightning.ai/docs/pytorch/stable/notebooks/course_UvA-DL/08-deep-autoencoders.html. [Accessed 30-07-2023].
- Oquab, M., Darcet, T., Moutakanni, T., Vo, H., Szafraniec, M., Khalidov, V., Fernandez, P., Haziza, D., Massa, F., El-Nouby, A., Assran, M., Ballas, N., Galuba, W., Howes, R., Huang, P.-Y., Li, S.-W., Misra, I., Rabbat, M., Sharma, V., Synnaeve, G., Xu, H., Jegou, H., Mairal, J., Labatut, P., Joulin, A., and Bojanowski, P. (2023). DINOv2: Learning robust visual features without supervision.
- Ramzai, J. (2021). Clearly explained: Pearson V/S Spearman Correlation Coefficient.
- Sener, O. and Savarese, S. (2018). Active learning for convolutional neural networks: A core-set approach.
- Settles, B. (2009). Active learning literature survey. *University of Wisconsin, Madison*, 52.
- shahraizanwar (2020). Age, Gender, Ethnicity Prediction — kaggle.com. <https://www.kaggle.com/code/shahraizanwar/age-gender-ethnicity-prediction>. [Accessed 28-07-2023].
- Sluijterman, L., Cator, E., and Heskes, T. (2021). How to evaluate uncertainty estimates in machine learning for regression?

- Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958.
- Vergara, D., Hernández, S., Valdenegro-Toro, M., and Jorquera, F. (2019). Improving predictive uncertainty estimation using dropout – hamiltonian monte carlo.
- Wen, Y., Vicol, P., Ba, J., Tran, D., and Grosse, R. (2018). Flipout: Efficient pseudo-independent weight perturbations on mini-batches.